

Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model

Iman Attarzadeh, Siew Hock Ow

Department of Software Engineering, Faculty of Computer Science & Information Technology,
University of Malaya, 50603 Kuala Lumpur, MALAYSIA
attarzadeh@siswa.um.edu.my, show@um.edu.my

Abstract—Software development time and cost estimation are the process of estimating the most realistic use of time and cost required for developing a software. It is one of the biggest challenges in the area of software engineering and project management, in the last decades. The software estimates are difficult to obtain due to incomplete software information is available in the early phase of software development process. Insufficient software information causes inaccuracy in software attributes. Thus, the vagueness and uncertainty of the software attributes is the main reason of inaccuracy of software estimates. Software cost estimation models such as regression model, expert judgment, SLIM, and COCOMO require accurate software attributes and long term estimation process, which are not completely achievable in early phase of software development process. Soft computing techniques such as fuzzy logic can reduce the vagueness and uncertainty of software attributes. Therefore, it may consider as alternative to decrease the inaccuracy of software estimates. This research aims to utilise an adaptive fuzzy logic model to improve the accuracy of software time and cost estimation. Using advantages of fuzzy set and fuzzy logic can produce accurate software attributes which result in precise software estimates. The Two-Dimension Gaussian Membership Function (2-D GMF) was used in the fuzzy model to make software attributes smoother in terms of the range of values. The COCOMO I, NASA98 data sets; and four project data from a software company in Malaysia were used in the evaluation of the proposed Fuzzy Logic COCOMO II (FL-COCOMO II). The evaluation of the obtained results, using Mean of Magnitude of Relative Error (MMRE) and PRED(25%) evaluation techniques, showed that the FL-COCOMO II produced the MMRE less than the original COCOMO and the value of PRED(25%) in the Fuzzy-COCOMO II is higher than the original COCOMO. Furthermore, the FL-COCOMO II showed 8.03% improvement in terms of estimation accuracy using MMRE when compared with the original COCOMO. Using advantages of fuzzy logic such as accurate estimation; adaption; understandability, and etc., can improve the accuracy of software estimates.

Keyword—Software project management, software cost estimation models, COCOMO II, soft computation techniques, fuzzy logic

I. INTRODUCTION

The number of project failures and those projects completed over cost and over schedule has been a significant issue for software project managers. Poor estimates have not only led projects to exceed budget and go over schedule but also, in many cases, to be terminated entirely. Although between 30% and 40% of software projects are ultimately

completed despite going over budget or schedule, and many more projects are cancelled or failed [1]. Among the many reasons for failure, inaccuracy in software estimates has been identified as a root cause of a high percentage of failures in the software development [1, 2]. Software cost estimation is the set of techniques and procedures that organisations use to arrive at an estimate for proposal bidding, project planning, and probability estimates [3, 4]. As such, estimation accuracy is a very significant issue for executives, managers, technical staff, and, particularly, practitioners who perform or rely on cost estimation [5]. Accurate estimation of software development cost also continues to challenge software engineering researchers due to the continued lack of accurate estimates [2]. The reasons that software cost estimation is difficult and error prone include [3]:

- Software cost estimation requires a significant amount of cost to perform correctly;
- The process is often done hurriedly, without an appreciation for the cost required to perform the estimate;
- Experience is required for developing estimates, especially for large projects, and;
- Human bias

There are several software cost estimation models which can be classified as algorithmic and non-algorithmic models. The algorithmic models are based on the statistical analysis of historical data [6, 7], for example, Software Life Cycle Management (SLIM) [8] and Constructive Cost Model (COCOMO) [9]. Non-algorithmic techniques are based on new approaches such as, Parkinson, Expert Judgment, Price-to-Win, and Machine Learning approaches [10, 11]. The Machine learning approach is used to group together a set of techniques that represent some of the facts of human mind [12], for example regression trees, rule induction, fuzzy systems, genetic algorithms, artificial neural networks, Bayesian networks, and evolutionary computation. The last five of these approaches are classified as soft computing group. The importance of algorithmic and non-algorithmic estimation techniques discuss in the following sections, briefly.

A. Algorithmic models

The famous algorithmic cost estimation models are: Boehm's COCOMO I and II [10]; Albrecht's Function Point; and Putnam's SLIM [8]. These models require inputs, accurate estimate of specific attributes, such as source line of code (SLOC), number of user screen, interfaces, complexity, etc. which are not easy to obtain in the early phase of software development. The formula and calculation of these models are easy to understand, also, they provide fast estimates when compared with non-algorithmic models. Besides, attributes and relationships used to estimate software development cost could change over time, and/or differ for software development environments [9]. The limitations of the algorithmic models led to the exploration of the non-algorithmic models.

B. Non-algorithmic models

In 1990's, the non-algorithmic models was born and have been proposed to software cost estimation. Software researchers have turned their attention to the new approaches that were based on soft computing approach such as artificial neural networks, fuzzy logic, and genetic algorithms. Fuzzy Logic (FL) offers a powerful linguistic representation that able to represent imprecision in the model inputs and outputs, while providing a more knowledge base approach to establish an effective model. Research shows that using FL can result in good performance in terms of reducing imprecision of inputs and outputs parameters.

This paper proposed an effective fuzzy logic model for embedding in the COCOMO II to overcome the vagueness and uncertainty of software attributes which resulted in producing more accurate estimation results.

II. RELATED WORK

Gray and MacDonell [2] compared popular techniques in software cost estimation such as regression techniques, function point analysis (FPA), fuzzy logic, and artificial neural networks. Their results showed that fuzzy logic model produced better performance than the other models. They introduced an application of fuzzy logic to cost estimation which resulted in developing a tool, FUZZY logic Software MEasuring (FULSOME) [2], to assist software managers in decision making. In the FULSOME model, the two important variables were selected: complexity adjustment factor and unadjusted function point. Then a triangular membership functions were defined for the small, medium, and large intervals of size, complexity, and software effort. Fei and Liu attempted to apply fuzzy logic to the algorithmic cost estimation models in order to handle uncertainty and imprecision problems in such models [13]. They proposed a fuzzy software size model for the COCOMO I. They found that it is unreasonable to assign determinate values for the software size attributes in the COCOMO I, because an accurate estimate of delivered source instruction (KDSI) cannot be made before starting the project. Ryder [14] applied

fuzzy modeling technique to the COCOMO I and Function Points model. Idri et al and Huang et al [15, 16] investigated the application of fuzzy logic to the EMs in the intermediate COCOMO I. Also, the fuzzification of the COCOMO I without considering the adjustment factor, then they introduced the f-COCOMO. They assigned a new software size to the COCOMO I, also, the coefficients related to the development mode were assigned to a fuzzy set. In another research, Kumar et al [17] applied fuzzy logic to manpower buildup index (MBI) of Putnam estimation model. The MBI is based upon 64 different rules. The results showed that fuzzy logic can be effectively applied to software cost estimation.

Fuzzy logic also had been applied to the non-algorithmic models to overcome the uncertainty of the models. Molokken and Jorgensen, proposed a combination of a fuzzy logic model with the estimation by analogy technique [1]. Estimation by analogy is one of the classified techniques of expert-based estimation. It is a type of Case-based Reasoning (CBR) method. Besides, the fuzzy analogy for software cost estimation had also been applied to web-based software.

In summary, fuzzy logic has been applied to algorithmic and non-algorithmic cost estimation models in the pursuit of achieving better estimation results. Nevertheless, there is still much uncertainty as to what estimation technique suits which type of estimation problem [16]. Choosing between the different techniques is a difficult decision that requires the support of a well-defined evaluation method to show each estimation technique as it applies to any estimation problem.

III. PROBLEM STATEMENT

Inaccurate software cost estimation has plagued software projects for decades. Poor estimates have not only led projects to exceed budget and schedule but also, in many cases, be terminated entirely. The ability to accurately estimate software development time, cost, and manpower, changes as newer methodologies replace old ones. Therefore, an accurate software cost estimation model is highly required in software project management.

IV. RESEARCH METHOD

This section, first, introduces the characteristics and strength of the COCOMO II and fuzzy logic, briefly, then the new FL-COCOMO II is explained.

A. The COCOMO II

The COCOMO I model is a regression-based software cost estimation model, which was developed by Boehm in 1981 [9] and thought to be the most cited, best known, and the most plausible model among all traditional cost estimation models. The COCOMO I was a stable model on that time. One of the problems with the use of COCOMO I today is that it does not match the development environment of the late 1990's. Therefore, in 1997, Boehm was developed the COCOMO II to solve most of the COCOMO I problems [11]. Equation (1) and

Figure 1 show the formula and the process of software schedule, cost, and manpower estimation in the COCOMO II. The COCOMO II includes several software attributes such as: 17 Effort Multipliers (EMs), 5 Scale Factors (SFs), Software Size (SS), and Effort estimation that are used in the Post Architecture Model of the COCOMO II [20]. The description of the 17 EMs and 5 SFs based upon their numerical values and productivity ranges are shown in Table 1 [11].

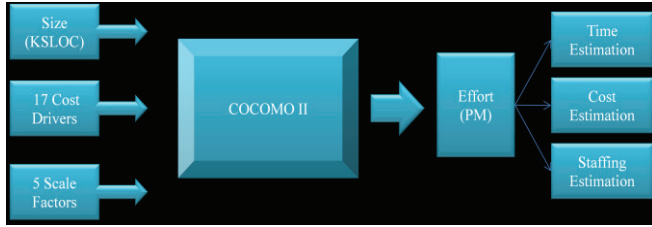


Figure 1: The process of software schedule, cost, and manpower estimation in the COCOMO II

$$\begin{cases} \text{Effort}_{PM} = A \times [\text{Size}]^{B+0.01 \times \sum_{j=1}^5 SF_j} \times \prod_{i=1}^{17} EM_i \\ \text{Schedule}_{Months} = C \times (\text{Effort})^{D+0.2 \times 0.01 \times \sum_{j=1}^5 SF_j} \\ \text{Average staffing}_{People} = \frac{\text{Effort}}{\text{Schedule}} \\ \text{COST} = \text{Effort} \times (\text{Payment}_{Month}) \\ A = 2.94; B = 0.91; C = 3.67; D = 0.28 \\ \text{Size: Software Size (SLOC)} \end{cases} \quad (1)$$

Vagueness and uncertainty of the software attributes can impact the software estimates. Thus, accurate software attributes resulted in producing accurate software estimates that is most desired for software project managers and organisations.

Table 1: The range of COCOMO II EMs

Effort Multiplier	Range
Required software reliability (RELY)	0.82-1.26
Database size (DATA)	0.90-1.28
Product complexity (CPLX)	0.73-1.74
Developed for reusability (RUSE)	0.95-1.24
Documentation match to life-cycle needs (DOCU)	0.81-1.23
Execution time constraint (TIME)	1.00-1.63
Main storage constraint (STOR)	1.00-1.46
Platform volatility (PVOL)	0.87-1.30
Analyst capability (ACAP)	1.42-0.71
Programmer capability (PCAP)	1.34-0.76
Personnel continuity (PCON)	1.29-0.81
Applications experience (APEX)	1.22-0.81
Platform experience (PLEX)	1.19-0.85
Language and tool experience (LTEX)	1.20-0.84
Use of software tools (TOOL)	1.17-0.78
Multi site development (SITE)	1.22-0.80
Required development schedule (SCED)	1.43-1.00

The value of SFs, are based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation.

B. Fuzzy Logic

In 1965, Lofti Zadeh formally developed multi-value set theory, and introduced the term fuzzy into the technical literature [18]. Fuzzy Logic (FL) starts with the concept of fuzzy set theory. It is a theory of classes with un-sharp boundaries, and considered as an extension of the classical set theory [19]. The membership $\mu_A(x)$ of an element x of a classical set A , as subset of the universe X , is defined by (2), as follows:

$$\mu_A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases} \quad (2)$$

A system based on FL has a direct relationship with fuzzy concepts (such as fuzzy sets, linguistic variables, etc.) and fuzzy logic. The popular fuzzy logic systems can be categorised into three types: Pure fuzzy logic systems, Takagi and Sugeno's fuzzy system, and fuzzy logic system with fuzzifier and defuzzifier [19]. Since most of the engineering applications produce crisp data as input and expects crisp data as output, the last type is the most widely used type of fuzzy logic systems. Fuzzy logic system with fuzzifier and defuzzifier, first, proposed by Mamdani and it has been successfully applied to a variety of industrial processes and consumer products [19]. The main three steps of applying fuzzy logic to a model are:

- Step 1:
 - Fuzzification: It converts a crisp input to a fuzzy set
- Step 2:
 - Fuzzy Rule-Based System: Fuzzy logic systems use fuzzy IF-THEN rules
 - Fuzzy Inference Engine: Once all crisp input values are fuzzified into their respective linguistic values, the inference engine accesses the fuzzy rule base to derive linguistic values for the intermediate and the output linguistic variables
- Step 3:
 - Defuzzification: It converts fuzzy output into crisp output

C. The Fuzzy Logic COCOMO II (FL-COCOMO II)

The new FL-COCOMO II is established based on the COCOMO II and FL. The COCOMO II includes a set of input software attributes: 17 EMs, 5 SFs, 1 SS and one output, Effort estimation. The architecture of the FL-COCOMO II is shown in Figure 2.

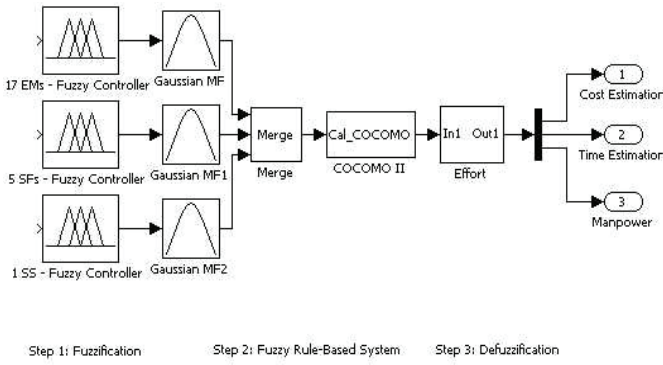


Figure 2: The architecture of the FL-COCOMO II includes: 17 EMs, 5SFs, 1 SS, and effort estimation

In the COCOMO II effort is expressed as Person Months (PM). It determines the efforts required for a project based on software project's size in Thousand Source Line of Code (KSLOC). Traditionally, the problem of software effort estimation relies on the single (numeric) values of the EMs, SFs, and software size that given as software attributes to estimate the effort. However, the size of the software may compute based on previously developed software that are similar to the current one (especially at the beginning of the project). Obviously, correctness and precision of such estimates are limited. It is essentially important to recognise this situation and come up with a technique which can evaluate the associated imprecision residing within the final results of cost estimation. Using fuzzy sets in EMs, SFs, and software size attributes can be specified by distribution of their possible values instead of using fixed values. Commonly, this form of distribution is represented in the form of a fuzzy set. It is important to clarify that vagueness and uncertainty at the input level of the COCOMO II yields uncertainty at the output level [10]. Converting the software attributes to respective fuzzy set causes improving the accuracy of the software attributes which resulted in estimation accuracy. Obviously, a certain monotonicity property holds, which is less precise estimate of inputs give rise to less detailed effort estimates. Overlapped symmetrical the Two-Dimension Gaussian Membership Function (2-D GMF) improves fuzzy model to a precise model.

Furthermore, there is a possibility when using the 2-D GMF that some attributes are assigned the maximum degree of compatibility instead of assigning to lower degrees. In order to avoid this linearity it is proposed to use more superior function, the 2-D GMF, for representing inputs of the model. The 2-D GMF is represented by (3) as follows:

$$\mu_{A_i}(x) = \text{Gaussian}(x, c_i, \sigma_i) = e^{\frac{-(x-c_i)^2}{2\sigma_i^2}} \quad (3)$$

Where c_i is the center of the i^{th} fuzzy set and σ_i is the width of the i^{th} fuzzy set.

The processes involved in applying fuzzy logic to the COCOMO II to establish the FL-COCOMO II are described as follows. The three main processes in the FL-COCOMO II are: Fuzzification, Fuzzy Rule-Based/ Fuzzy Inference Engine,

and Defuzzification. The software attributes in the COCOMO II converted to the fuzzy variables based on the fuzzification process and terms Extra Low (XL), Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH), and Extra High (XH) were defined for the 23 software attributes (17 EMs, 5 SFs, and 1 SS) were defined for each software attribute. The fuzzy sets corresponding to the various associated linguistic values for each software attribute were defined using the 2-D GMF. For example, the fuzzification of the Applications Experience (AEXP) EM based on the 2-D GMF function μ , using Fuzzy Inference System tool in the MATLAB software, is defined as shown in Table 2 and Figure 3. The Fuzzy Inference System (FIS) is a fuzzy tool in the MATLAB software which was used in the fuzzification, fuzzy calculations, fuzzy rule generation, and defuzzification process of the FL-COCOMO II. The FIS supports Mamdani and Sugeno fuzzy methods. The FL-COCOMO II was established based upon the Sugeno fuzzy interface system which is more accurate than the Mandani FIS method.

Table 2: Applications Experience (AEXP) EM Description

Effort Multiplier: Applications Experience (AEXP)						
Descriptors	2 months	6 months	1 year	3 years	6 years	-
Effort Multipliers	1.42	1.19	1	0.85	0.71	-
Rating	VL	L	N	H	VH	EX

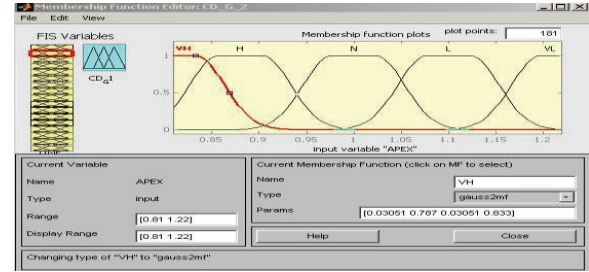


Figure 3: The fuzzification of the AEXP EM using FIS tool in the MATLAB software

At the first step, all the software attributes of the COCOMO II converted to the responding fuzzy sets and variables (Fuzzy_{EMij}) rather than using the fixed values of EM_{ij} . The Fuzzy_{EMij} were calculated using equation (4) and the original EM_{ij} values and the 2-D MGF μ was defined for the various fuzzy sets associated with the EMs, SFs, and SS. This process helps to reduce the vagueness and uncertainty of the software attributes at this level.

$$\text{Fuzzy}_{EMij} = F(\mu_{A_1}^{V_1}, \dots, \mu_{A_i}^{V_i}, EM_{i1} \dots EM_{ij}) \quad (4)$$

For ease, F is taken as a linear function, where the $\mu_{A_j}^{V_i}$ is the membership function of the fuzzy set A_j associated with the cost driver V_i as shown in equation (5).

$$\text{Fuzzy}_{EMij} = \sum_{j=1}^{k_i} \mu_{A_i}^{V_i} * EM_{ij} \quad (5)$$

The fuzzy rules for the FL-COCOMO II were defined through the linguistic variables in the fuzzification process. It

is important to take note that these fuzzy rules were adjusted as well as all pertinence level functions, in accordance with the tests and the characteristics of the project. The fuzzy rules were defined based on the connective "AND" and "OR" or combination of them between input variables, as shown as follows.

Fuzzy rules:

IF TOOL is Low THEN effort is Low
 IF PCAP is Very Low THEN effort is Very High
 IF RESUE is Nominal THEN effort is Nominal
 IF DATA is Very High THEN effort is Very High
 ...

The number of rules which were defined for the FL-COCOMO II is more than 193 based on the input variables. In the process of fuzzy rule generation of the FL-COCOMO II, the FIS tool in the MATLAB software was used as shown in Figure 4.

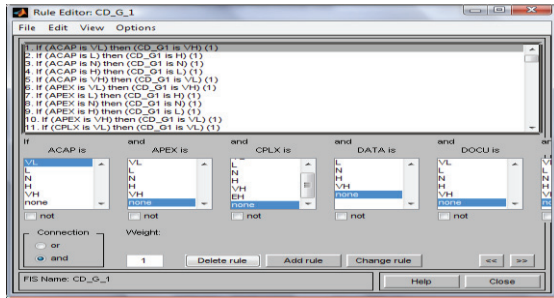


Figure 4: The fuzzy rule generation using FIS tool in the MATLAB software

The last step is defuzzification of the effort variable using defuzzification techniques such as Mean of Maximum (MOM), Center of Area (COA), and First of Maximum (FOM). The defuzzification of the output "Effort" performed using the Mean of Maximum (MOM) technique because the results obtained are more accurate when compared with the other defuzzification techniques.

V. RESULTS AND DISCUSSION

The FL-COCOMO II was evaluated through three sources: two public domain data sets (Data set 1 and 2) and 4 software project data from software companies in Malaysia (Data set 3). The descriptions of these data sets are discussed as follows.

A. Data set 1

The COCOMO I data set—Boehm [10] was the first researcher which looked at software engineering from an economic point of view then he came up with a cost estimation model from a dataset, the COCOMO I data set. The COCOMO I dataset includes 63 historical project data. This data set is in public domain and available in <http://promisedata.org>.

B. Data set 2

The NASA93 data set—the NASA93 data set includes 93 project data from NASA Manager Measurement Benchmarking Element. This data set is in public domain and available in <http://promisedata.org>.

C. Data set 3

Four real project data from two software companies A and B in Malaysia. These project data were collected through a direct data collection from software companies A and B in the COCOMO data set format.

Therefore, in this research three different data sets (160 project data) were used for evaluation of the FL-COCOMO II.

D. Evaluation Method

In the evaluation of software cost estimation models, the most widely used evaluation criterion are the Mean Magnitude of Relative Error (MMRE) and the probability of a project having a relative error of less than or equal to L (PRED(L)). The Magnitude of Relative Error (MRE) is defined in equation (6) as follows:

$$MRE_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i|}{Actual\ Effort_i} \quad (6)$$

The MRE value was calculated for each observation i that effort is estimated at that observation. The aggregation of MRE over multiple observations (N) can be achieved through the Mean MRE (MMRE) in equation (7) as follows:

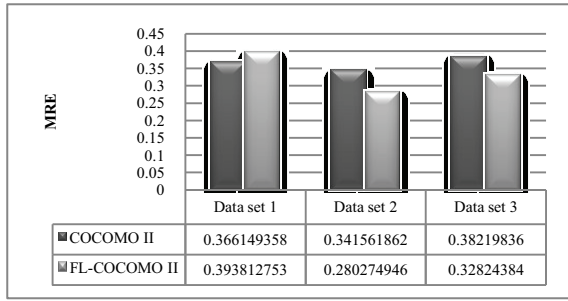
$$MMRE = \frac{1}{N} \sum_i^N MRE_i \quad (7)$$

A complementary criterion is the prediction at level L , $PRED(L) = k/N$, where k is the number of observations where MRE is less than or equal to L , and N is the total number of observations. Thus, $PRED(25\%)$ gives the percentage of projects which are predicted with a MRE less or equal than 25%. This value, 25%, is a realistic value for evaluation of models.

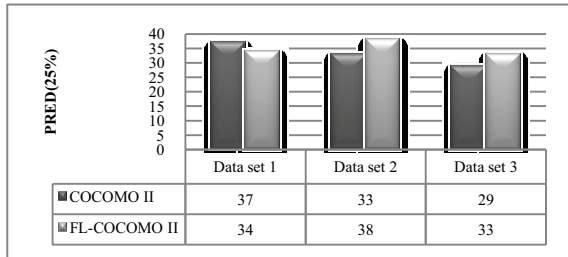
The FL-COCOMO II was evaluated through the three data sets (1, 2, and 3), separately, using the MMRE and $PRED(25\%)$. These data sets were applied to the COCOMO II and FL-COCOMO II, separately, and then the value of MRE and $PRED(25\%)$ were calculated for each data set. The comparison between obtained results from the COCOMO II and FL-COCOMO II using data set 1 to 3 are shown in Table 3 and Figure 5.

Table 3: Comparison between obtained results from the COCOMO II and FL-COCOMO II using the MRE and $PRED(25\%)$

Data set	Model	Evaluation	
		MMRE	PRED(25%)
Data set 1	COCOMO II	0.366149358	37%
	FL-COCOMO II	0.393812753	34%
Data set 2	COCOMO II	0.341561862	33%
	FL-COCOMO II	0.280274946	38%
Data set 3	COCOMO II	0.38219836	29%
	FL-COCOMO II	0.32824384	33%
Mean	COCOMO II	0.363303193	33%
	FL-COCOMO II	0.334110513	35%



(a)



(b)

Figure 5: Comparison between the COCOMO II and FL-COCOMO II using the (a) MRE and (b) PRED (25%) evaluation techniques

The row labelled in “Mean” represents the average result for the data sets 1 to 3 based upon the evaluation criterion in the models.

For the data set 1, the value of MRE and PRED(25%) evaluation criterion for the COCOMO II and FL-COCOMO II are 0.366149358 and 0.393812753; 37 and 34, respectively. These values show that in terms of MRE, the COCOMO II shows less value for MRE and higher value for PRED (25%) when compared with the FL-COCOMO II. Therefore, in this case, the COCOMO II produced more accurate results for data set 1.

For the data set 2, the value of MRE and PRED(25%) evaluation criterion for the COCOMO II and FL-COCOMO II are 0.341561862 and 0.280274946; 33 and 38, respectively. These values show that in terms of MRE, the FL-COCOMO II shows less value for MRE and higher value for PRED (25%) when compared with the COCOMO II. Therefore, in this case, the FL-COCOMO II produced more accurate results than the COCOMO II.

For the data set 3, the value of MRE and PRED(25%) evaluation criterion for the COCOMO II and FL-COCOMO II are 0.38219836 and 0.32824384; 29 and 33, respectively. These values show that in terms of MRE, the FL-COCOMO II shows less value for MRE and higher value for PRED (25%) when compared with the COCOMO II.

Finally, The values of MMRE for COCOMO II and FL-COCOMO II based upon data sets 1 to 3 are 0.363303193 and 0.334110513, respectively, and the value of PRED (25%) are 33 and 35, respectively.

By comparing these results, a software cost estimation model with a lower value of MMRE gives better estimates than a model with a higher value. The best cost estimation model shows a value 0 for MMRE evaluation criterion. Also, a

software cost estimation model with a higher value of PRED(25%) gives better estimates than a model with a lower value. The best cost estimation model shows a value 0 for MMRE and 100% for PRED(25%) evaluation criterion. Between these two estimation models (COCOMO II and FL-COCOMO II), the use of FL-COCOMO II for software development cost estimate can produce better estimates in terms of MMRE, and PRED (25%) than the COCOMO II based on the data sets 1 to 3.

The percentage of improvement of FL-COCOMO II is calculated based on the difference between the two models divided by the value of the COCOMO II. Table 4 shows the comparison of the improvement based upon estimation accuracy in COCOMO II and FL-COCOMO II.

Table 4: Comparison of improvement of the COCOMO II and FL-COCOMO II

Model	Evaluation	
	MMRE	PRED (25%)
COCOMO II vs. FL-COCOMO II		
COCOMO II	0.363303193	33
FL-COCOMO II	0.334110513	35
Improvement (%)	8.03	6.06

According to the Table 4, in terms of MMRE, the percentage of improvement based upon estimation accuracy for the FL-COCOMO II is 8.03% and based on PRED (25%) is 6.06% when compared to the COCOMO II, respectively. It shows that using FL can improve the estimation accuracy and it can be used as alternative to apply to the other software cost estimation models to improve estimation accuracy.

VI. CONCLUSION

One of the important issues in software project management is accurate and reliable estimation of software time, cost, and manpower, especially in the early phase of software development. Software attributes usually have properties of uncertainty and vagueness when they are measured by human judgment. A software cost estimation model incorporates fuzzy logic can overcome the uncertainty and vagueness of software attributes. However, determination of the suitable fuzzy rule sets for fuzzy inference system plays an important role in coming up with accurate and reliable software estimates. The objective of this research was to examine the application of applying fuzzy logic in software cost estimation that can perform more accurate result. This paper presented an adaptive software cost estimation model incorporates fuzzy logic technique to handle imprecision and uncertainty of software attributes. The results obtained from applying three different data sets 1 to 3 to the models, showed that the FL-COCOMO II produced better estimation results than the COCOMO II using evaluation criterion MMRE=0.334110513 and PRED(25%)=35. Furthermore, the percentage of improvement for the FL-COCOMO is from 6.06 to 8.03, which shows the performance of the FL-COCOMO

when compared with the COCOMO II based on the estimation accuracy. The application of fuzzy logic in other software cost estimation models can also be explored in the future.

REFERENCES

1. K. Molokken and M. Jorgensen, "A review of software surveys on software effort estimation," Proceedings of IEEE International Symposium on Empirical Software Engineering (ISESE 2003). Rome, Italy, September 30 – October 1, 2003, pp. 223 – 230.
2. S.G. MacDonell and A.R. Gray, "A comparison of modeling techniques for software development effort prediction," Proceedings of the International Conference on Neural Information Processing and Intelligent Information Systems, Dunedin, New Zealand, November 12-14, 1997, pp. 869-872.
3. R. Agarwal, M. Kumar, M. Yogesh, S. Mallick, R. M. Bharadwaj, and D. Anantwar, "Estimating software projects," SIGSOFT Software Engineering Notes, 26(4), 60-67, 2001.
4. M. Jorgensen, "Practical guidelines for expert-judgment-based software effort estimation," IEEE Software Engineering, 22(3), pp. 57-63, 2005.
5. C. F. Kemerer, "An empirical validation of software cost estimation models," Communications of the ACM, 30(5), pp. 416-429, 1987.
6. K. Strike, K. El-Emam, and N. Madhavji, "Software cost estimation with incomplete Data," IEEE Transactions on Software Engineering, 27(10), pp. 215-223, 2001.
7. A. C. Hodgkinson and P.W. Garratt, "A neurofuzzy cost estimator," Proceedings of the 3rd International Conference on Software Engineering and Applications (SAE-1999), NJ, USA, 1999, pp. 401-406.
8. L. H. Putnam, "A general empirical solution to the macro software sizing and estimating problem," IEEE Transactions on Software Engineering, 4(4), pp. 345 – 361, 1978.
9. B. W. Boehm, Software engineering economics. Englewood Cliffs, NJ, Prentice-Hall, 1981.
10. B. W. Boehm, C. Abts, and S. Chulani, "Software development cost estimation approaches – A survey", University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000-505, 2000.
11. B. W. Boehm, "Cost models for future software life cycle processes: COCOMO 2.0," Annals of Software Engineering Special Volume on Software Process and Product Measurement, Science Publisher, Amsterdam, Netherlands, 1(3) , pp. 45 – 60., 1995.
12. K. Srinivasan, and D. Fisher, "Machine learning approaches to estimating software development effort," IEEE Transactions on Software Engineering, 21(2), 1995.
13. Z. Fei, and X. Liu, "F-COCOMO: Fuzzy constructive cost model in software engineering," Proceedings of the IEEE International Conference on Fuzzy Systems, IEEE Press, New York, pp. 331–337, 1992.
14. J. Ryder, "Fuzzy modeling of software effort prediction," Proceedings of IEEE Information Technology Conference, Syracuse, NY, pp 102-114, 1998.
15. A. Idri, A. Zahi, and A. Abran, "Software cost estimation by fuzzy analogy for web hypermedia applications," Proceedings of the International Conference on Software Process and Product Measurement, Cadiz, Spain, pp. 53 – 62, 2006.
16. X. Huang, D. Ho, J. Ren, and F. Capretz, "A soft computing framework for software effort estimation," Soft Computing, A Fusion of Foundations, Methodologies and Applications Journal, 10(2), pp. 170-177, 2009.
17. S. Kumar, A. Krishna, and P. Satsangi, "Fuzzy systems and neural networks in software engineering project management," Journal of Applied Intelligence, 4(1), pp. 31-52, 1994.
18. A. Lotfi Zadeh, "Fuzzy logic, neural networks, and soft computing," Communication of ACM, 37(3), pp.77-84, 1994.
19. A. Lotfi Zadeh, "The future of soft computing," The 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada, pp. 217-228, 2001.