World Applied Sciences Journal 8 (2): 177-184, 2010
ISSN 1818-4952

Improving the Accuracy of Software Cost Estimation Model Based on a New Fuzzy Logic Model

Iman Attarzadeh and Siew Hock Ow

Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaysia (UM), 50603 Kuala Lumpur, Malaysia

Abstract: Software development effort estimation is one of the most important activities in software project management. Formal effort estimation models, for example Constructive Cost Model (COCOMO), are limited by their inability to manage uncertainties and imprecision surrounding software projects early in the development life cycle. In recent years, attention has turned to a variety of soft computing methods like fuzzy logic in particular to estimate software development effort. A software effort estimation model which adopts a fuzzy inference method provides a solution to adjust the uncertain and vague properties of software effort drivers. The present paper proposes a new fuzzy logic model to overcome the uncertainty at the input level of the COCOMO model that it causes uncertainty at the output or estimation error. The main objectives of this research are to investigate the effect of crisp inputs and fuzzification techniques on the accuracy of system's output when fuzzy logic applied to the model to derive the software effort estimates. The proposed model validated by using the 63 historical project data in the well-known COCOMO model and an artificial dataset that consists of 100 sample projects. Empirical results showed that applying fuzzy logic for software effort estimates resulted in slightly smaller mean magnitude of relative error (MMRE) and probability of a project having a relative error of less than or equal to 0.25 (Pred(0.25)) as compared with the results obtained from original model.

Key words: Software project management · Software estimation · Effort estimation · Formal estimation models · Constructive cost model · Fuzzy logic

INTRODUCTION

Software cost estimation refers to the estimations of the likely amount of effort, time and staffing levels required to build a software system. The most helpful form of effort estimation is the one made at an early stage during a project, working primarily from feasibility and requirements specifications documents. However, estimates at the early stages of the development are the most difficult to obtain and they are often the least accurate, because very little detail is known about the project and the product at its start. Accurate software cost estimates are critical to both developers and customers. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality and failure to complete on time. Most cost estimation models attempt to generate an effort estimate, which can then be converted into the project duration and cost. Although effort and cost are closely related, they are not necessarily related by a simple transformation function. Effort is often measured in person-months of the programmers, analysts and project managers. Software cost estimation techniques can be broadly classified as algorithmic and non-algorithmic models. Algorithmic models are derived from the statistical analysis of historical project data [1], for example, Constructive Cost Model (COCOMO) [1] and Software Life Cycle Management (SLIM) [2]. Non-algorithmic techniques include Price-to-Win [3], Parkinson [3], expert judgment [3] and machine learning approaches [4]. Machine learning is used to group together a set of techniques that embody some of the facets of human mind [4], for example fuzzy systems, analogy, regression trees, rule induction and neural networks. Among the machine learning approaches, fuzzy systems and neural networks are considered to belong to the soft computing group.

Corresponding Author: Iman Attarzadeh, Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya (UM), 50603 Kuala Lumpur, Malaysia

Algorithmic Models: Software effort estimation provided some of the first attempts at precise software measurement, so it is the oldest, most mature aspect of software metrics. Boehm was the first researcher to look at software engineering from an economic point of view, coming up with a cost estimation model, COCOMO-81 in 1981, after investigating a large set of data from TRW in the 1970s [3]. Putnam also developed an early model known as SLIM in 1978 [2]. COCOMO, SLIM and Albrect's function point [3] methods (i.e. measures amount of functionality in a system) were all based on linear regression techniques by collecting data from past projects. Both COCOMO and SLIM take size of lines of code (about which least is known very early in the project) as the major input to their models.

Models based on historical data have limitations, because attributes and relationships used to predict software development effort could change over time and/or differ for software development environments [4]. Existing models rely on accurate estimate of size of software in terms of line of code LOC, number of user screen, interfaces, complexity and so on, at a time when uncertainty surrounds the project the most [5]. Their inability to handle categorical data, that are specified by a range of values and most importantly lack of reasoning capabilities, the ability to draw conclusions or make judgments based on available data, contributed to the number of studies exploring non-algorithmic methods, for example neural network, fuzzy logic and etc.

Non-Algorithmic Models: Newer computation techniques to cost estimation that are non-algorithmic were sought in the 1990's. Researchers particularly have turned their attention to a set of approaches that are soft computing based. These include artificial neural networks, fuzzy logic models and genetic algorithms. Artificial neural network is able to generalize from trained data set. Over a known set of training data, a neural-network learning algorithm constructs *rules* that fit the data and fits previously unseen data in a reasonable manner as well [5]. Some of the very early works indicating that neural networks are highly applicable to cost estimation include those of Venkatachalam [6] and Krishna and Satsangi [7].

Fuzzy logic with its offerings of a powerful linguistic representation can represent imprecision in inputs and outputs, while providing a more expert knowledge based approach to model building. A study by Hodgkinson and Garratt claims that estimation by expert judgment was better than all regression based models [8].

Fuzzy Logic: The fuzzy logic model uses the fuzzy logic concepts introduced by Lofti A. Zadeh [9]. Fuzzy reasoning consists of three main components [10]: 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.

A system based on Fuzzy Logic 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 fuzzification and defuzzification [10]. 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 one fuzzy logic system with fuzzification and defuzzification was first proposed by Mamdani It has been successfully applied to a variety of industrial processes and consumer products [10].

The COCOMO Model: The Constructive Cost Model (COCOMO) is a regression based software cost estimation model developed by Barry Boehm [3]. This model allows one to estimate the cost, effort and schedule when planning a new software development activity. COCOMO II is the latest major extension to the original COCOMO (COCOMO 81) model published in 1981. COCOMO II has three models also, but they are different from those of COCOMO 81. They are [1, 3]:

- Application Composition Model - Suitable for projects built with modern GUI-builder tools. Based on new Object Points.
- Early Design Model - To get rough estimates of a project's cost and duration before have determined its entire architecture. It uses a small set of new Cost Drivers and new estimating equations. Based on Unadjusted Function Points or KSLOC.

- Post-Architecture Model - The most detailed on the three, used after the overall architecture for the project has been designed. One could use function points or LOC as size estimates with this model. It involves the actual development and maintenance of a software product.

COCOMO II describes 17 cost drivers that are used in the Post-Architecture model [3]. The cost drivers for COCOMO II are rated on a scale from Very Low to Extra High in the same way as in COCOMO 81. COCOMO II post architecture model is given as:

$$Effort = A \times [Size]^{B} \times \prod_{i=1}^{17} Effort\ Multiplier_i \quad (1)$$

$$where\ B = 1.01 + 0.01 \times \sum_{i=1}^{5} Scale\ Factor_j$$

In "1":
A: Multiplicative Constant
Size: Size of the software project measured in terms of KSLOC (thousands of Source Lines of Code, Function Points or Object Points). The selection of scale factors (SF) is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation.

LITERATURE REVIEW

In the last decades, many methods have been introduced into the area of software cost estimation to improve estimation accuracy.

Gray and MacDonell [4] compared popular techniques in software effort estimation as regression techniques, function point analysis (FPA), fuzzy logic and neural network. Their results showed that fuzzy logic model achieved good performance. They introduced an application of fuzzy logic to effort estimation. They developed a tool, FUzzy logic SOftware MEasuring (FULSOME) [4], to assist software managers in making estimation. In FULSOME model, the two most important variables were selected: complexity adjustment factor and unadjusted function point. Then a triangular membership functions were defined for the small, medium, large intervals of size, complexity and effort.

Shepperd's case-based reasoning tools [11] explore algorithmic methods for emulating expert analogical reasoning; Chulani and Boehm's Bayesian tuning method [3] for regression models allows an algorithm to carefully combine expert judgment with the available data; Chulani *et al.* [3] applied Bayesian analysis in calibrating the 1998 version of COCOMO II model to 161 data-points. When compared with the 1997 calibration done using multiple regressions, the Bayesian approach was adjudged to perform better and more robust. Bayesian analysis was also used in the calibration of the 2000 version of COCOMO II by Boehm *et al.* [3]. The result of this was a higher predictive accuracy.

Fei *et al.* have tried to fuzzify some of the existing algorithmic models in order to handle uncertainties and imprecision problems in such models [12]. They have done the first realisation of the fuzziness on COCOMO model. They found it is unreasonable to assign a determinate number for it, because an accurate estimate of delivered source instruction (KDSI) cannot be made before starting the project. Ryder [13] applied fuzzy modeling technique to COCOMO and the Function-Points models.

Idri *et al.* and Huang *et al.* [14, 15] investigated the application of fuzzy logic to the cost drivers of intermediate COCOMO model.

In another research, Kumar *et al.* [7] applied fuzzy logic in manpower buildup index (MBI) of Putnam estimation model. MBI was based upon 64 different rules. The results showed it can be effectively applied to software project management. Fuzzy logic also had been applied to the non- algorithmic models to overcome the uncertainly of the models.

Molokken et al and Idir *et al.* proposed a combination of fuzzy logic and estimation by analogy [15, 16]. Estimation by analogy is one of the classified techniques of expert-based estimation method. It is a type of Case-based Reasoning (CBR) method. The fuzzy analogy for software cost estimation had also been applied to web base software. Venkatachalam [6] applied artificial neural network to cost estimation. Neural network is able to generalise from trained data set. Over a set of training data, neural network learning algorithm constructs mappings that fit the data and fits previously unseen data in a reasonable way.

Research had also been done to combine fuzzy logic with neural network. A new system based on fuzzy logic, neural network and COCOMO II proposed [17, 18, 19]. This system Based on COCOMO II post architecture model, the input of neuro-fuzzy COCOMO consists of size and 22 cost drivers (5 scale factors plus 17 effort multipliers). In summary, fuzzy logic has been proposed to algorithmic and non-algorithmic 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 [20, 21]. 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.

Problem Statement: It is unrealistic to expect very accurate cost estimates of any software because of the inherent uncertainty in software development projects and the complex and dynamic interaction of factors that impact software development cost use. Still, it is likely that estimates can be improved because software development cost estimates are systematically overoptimistic and very inconsistent. An important objective of the software engineers has been to develop useful models that are accurately estimating the software cost. In order to address and overcome to these problems, new approaches with accurate estimation will be considerable.

RESEARCH METHOD

The new proposed model based on COCOMO II has three input's data group. Those groups are COCOMO II cost drivers; scale factors and size of software. Also one system output, effort estimation. It is shown in Figure 1.

In COCOMO effort is expressed as Person Months (PM). It determines the efforts required for a project based on software project's size in Kilo Source Line of Code (KSLOC) as well as other cost drivers known as scale factors and effort multipliers. It contains 17 effort multipliers and 5 scale factors. The standard numeric values of the cost drivers are given in Table 1.

Traditionally, the problem of software effort estimation relies on a single (numeric) value of size and scale factors values of given software project to predict the effort. However, the size of the project is, based on some previously completed projects that resemble the current one (especially at the beginning of the project). Obviously, correctness and precision of such estimates are limited. It is of principal importance to recognise this situation and come up with a technology using which we can evaluate the associated imprecision residing within the final results of cost estimation. The technology endorsed here deals with fuzzy sets. Using fuzzy sets, size of a software project can be specified by distribution of its possible values. Commonly, this form of distribution is represented in the form of a fuzzy set. It is important that uncertainty at the input level of the COCOMO model yields uncertainty at the output [10].

Table 1: COCOMO II cost drivers

Cost Driver	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

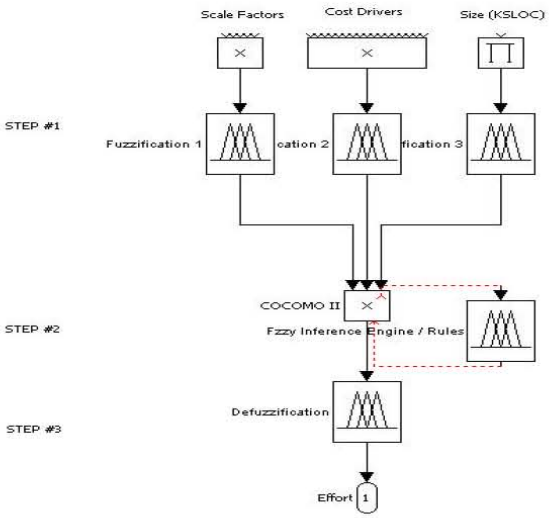

Fig. 1: The proposed model: Inputs: COCOMO II cost drivers, scale factors, size. Output: Effort estimation

This becomes obvious and, more importantly, bears a substantial significance in any practical endeavor. By changing input parameters using fuzzy set, we can model the effort that impacts the estimation accuracy. Obviously, a certain monotonicity property holds, which is less precise estimates of inputs give rise to less detailed effort estimates. Overlapped symmetrical Two-sided Gaussian function reduces fuzzy systems to precise linear systems [17]. Furthermore there is a possibility when using a Two-sided Gaussian function that some attributes are assigned the maximum degree of compatibility when they should be assigned lower degrees. In order to avoid this linearity it is proposed to use more superior function

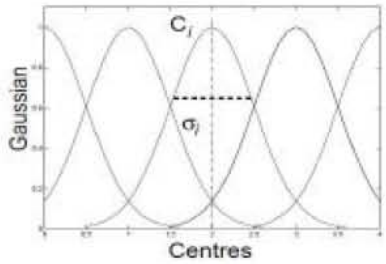

Fig. 2: Gaussian membership function

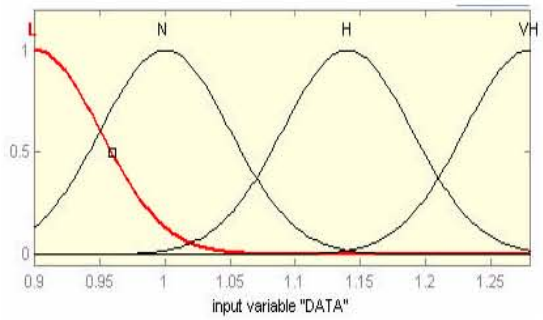

Fig. 3: Representation of DATA cost driver using Gaussian function (Input)

i.e., Two-sided Gaussian membership function for representing inputs of the project. The Gaussian Function is represented by "2". In "2" the weight of exponential is selected as 1.

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

Where C_i is the center of the i_{th} fuzzy set and σ_i is the Full Width at Half Maximum (FWHM) of the i_{th} fuzzy set. It is shown in Figure 2.

The processes involved in software effort estimation using FL are shown in Figure 1. The main processes of this system include four activities: fuzzification, fuzzy rule base, fuzzy inference engine and defuzzification. The main fours components' functions are in three steps as follows:

1. Step #1

Fuzzification: It converts a crisp input to a fuzzy set.

2. Step #2

Fuzzy Rule Base: 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.

3. Step #3

Defuzzification: It converts fuzzy output into crisp output.

All the input variables in COCOMO II model changed to the fuzzy variables based on the fuzzification process. The terms Very Low, Low, Nominal, High and Very High, Extra High were defined for the 22 variables, cost drivers and scale factors, in COCOMO II. For example, in the case of DATA cost driver, we define a fuzzy set for each linguistic value with a Two-sided Gaussian shaped membership function ì is shown in Figure 3. We have defined the fuzzy sets corresponding to the various associated linguistic values for each cost driver.

In this research, a new fuzzy effort estimation model is proposed by using Two-sided Gaussian function to deal with linguistic data and to generate fuzzy membership functions and rules for cost drivers obtained from "3". In the next step, we evaluate the COCOMO model using the "1" and cost drivers obtained from fuzzy sets (F_EMij) rather than from the classical EMij. F_EMij is calculated from "4" the classical EMij and the membership functions ì defined for the various fuzzy sets associated with the cost drivers.

$$Fuzzy_{EM_{ij}} = F(\mu^{v_2}{}_{Ai}, \ldots \mu^{v_1}{}_{Ai}, EM_{ij} \ldots EM_{ij}) \quad (3)$$

For ease, F is taken as a linear function, where the μVi Ai is the membership function of the fuzzy set Ai associated with the cost driver Vi is shown in "4".

$$Fuzzy_{EM_{ij}} = \sum_{i=1}^{kt} \mu_{At}{}^{v_i} * EM_{ij} \quad (4)$$

The new fuzzy model rules contain the linguistic variables related to the project. It is important to note that those rules were adjusted or calibrated, as well as all pertinence level functions, in accordance with the tests and the characteristics of the project. In rules use the connective "and" and "or" or combination of them between input variables, as indicated in the example below.

Fuzzy Rules:

IF DATA is Low TEHN effort is Low
IF ACAP is Very_Low THEN effort is Very_High
IF CPLX is Nominal THEN effort is Nominal
IF RUSE is Very_High THEN effort is Very_High

The number of rules that have used in proposed model is more than 193 rules for all input variables.

Table 2: The artificial dataset generated for system validation consists of 100 data samples

No.	Mode	Size	Effort
1	1.1200	51.2500	246.5900
2	1.2000	12.5500	58.2800
3	1.0500	81.5200	550.4000
…	...	…	…
97	1.2000	56.5300	354.7300
98	1.0500	16.0400	67.1400
100	1.1200	54.1700	262.3800

DATASET DESCRIPTION

B.W. Boehm [1] is the first researcher to look at software engineering from an economic point of view and he came up with cost estimation models from two datasets, COCOMO and COCOMO II. The COCOMO [1] dataset includes 63 historical projects with 17 effort drivers and one dependent variable of the software development effort. So, the first used dataset for evaluating the proposed model is based on COCOMO model. The second attempt was to create an artificial dataset based on COCOMO model. The algorithm for fuzzy set learning in a Mamdani-type fuzzy system is following this four-step procedure:

- Choose a training sample and propagate the input vector across the network to get the output.
- Determine the error in output and the error gradient in all the other layers.
- Determine the parameter changes for the fuzzy weights and update the fuzzy weights.
- Repeat until the fuzzy error is sufficiently small after an epoch is complete.

The dataset randomly generated following the procedure discussed in above and shows in Table 2.

Therefore, in this work has used two datasets for evaluation of the proposed model. Finally, by aggregate the accuracy across all testing datasets as the mean result.

EXPRIMENTAL RESULTS

For evaluating the different software effort estimation models, the most widely accepted evaluation criteria are the mean magnitude of relative error (*MMRE*) and probability of a project having a relative error of less than or equal to 0.25 (*Pred*(*l*)). The Magnitude of Relative Error (MRE) is defined in "5" as follows:

$$MRE_i = \frac{|Actual\ Effort_t - Predicted\ Effort_t|}{Actual\ Effort_t} \quad (5)$$

The MRE value is calculated for each observation *i* whose effort is predicted. The aggregation of MRE over multiple observations (N) can be achieved through the Mean MRE (MMRE) in "6" as follows:

$$MMRE = \frac{1}{N}\sum_{i}^{N} MRE_i \quad (6)$$

Another measure similar to MRE, the Magnitude of error Relative to the Estimate (MER), has been proposed. Intuitively, it seems preferable to MRE since it measures the error relative to the estimate. MER uses *Predicted Efforti* as denominator in "5". The notation MMER is used to the mean MER in "6". However, the MMRE and MMER are sensitive to individual predictions with excessively large MREs or MERs. Therefore, an aggregate measure less sensitive to extreme values is also considered, namely the median of MRE and MER values for the N observations (MdMRE and MdMER respectively) . A complementary criterion is the prediction at level *l*, Pred(*l*) = k/N, where *k* is the number of observations where MRE (or MER) is less than or equal to *l* and *N* is the total number of observations. Thus, Pred(25) gives the percentage of projects which were predicted with a MRE (or MER) less or equal than 0.25.

The proposed fuzzy model was validated by two approaches. In the first approach, has used the COCOMO dataset that consists of 63 projects (Dataset #1). In the second approach, has used the artificial dataset that consists of 100 sample projects (Dataset #2). Then both datasets are applied to the new fuzzy model and COCOMO II model. The validation of the new fuzzy model to building trained fuzzy model for effort estimation has been done using artificial dataset and standard dataset in COCOMO II. The comparison between the results of standard dataset and artificial dataset that applied on the new fuzzy model and COCOMO II model shows more accuracy in case of effort estimation by the new fuzzy model. The comparisons between results are shown in Tables 3 and 4.

In this research, each dataset separately applied to the COCOMO II model and proposed model. Then for each model, the MMRE and Pred were calculated. Finally mean of those calculations are used to compare both models. The result for 163 applied projects shows the MMRE for COOCMO II model is 0.406713037 and

Table 3: Comparison between performance of the proposed model and COCOMO II

Data set	Model	Evaluation MMRE	Pred (25%)
Data set #1	COCOMO II	0.413812453	30%
	Proposed Model	0.366545456	50%
Data set #2	COCOMO II	0.39961362	40%
	Proposed Model	0.37272956	45%
Mean	COCOMO II	0.406713037	35%
	Proposed Model	0.369637508	47.5%

Table 4: Accuracy of the proposed model

Model	Evaluation	MMRE
Proposed Model vs. COCOMOII	COCOMO II	0.406713037
	Proposed Model	0.369637508
	Improvement %	12.63

for proposed model the value equals to 0.369637508. It shows the proposed model has MMRE less than COCOMO II model, so it means the accuracy of proposed model is better than COCOMO II. In case of Pred, the final result shows the proposed model value is 47.5% in Pred (25%) and COCOMO II value is 35% in same Pred. As it mentioned above, Pred shows the number of projects that they have MMRE lass than 25%. According to this definition, the proposed model shows better accuracy. Table 4 shows how much the proposed model is accurate than COCOMO II model.

For comparing proposed model with COCOMO model, the improvement is 12.63% based on the *MMRE* 0.40 and 0.36. The experimental results show that the proposed software effort estimation model shows better estimation accuracy than the other two models, i.e., COCOMO. In summary, an output with more terms or fuzzy sets provided a better performance due to the high granularity demanded from the results. Most of the sample data in the dataset with the proposed fuzzy model resulted in a more accurate estimation when compared to the COCOMO II model.

CONCLUSION

An essential issue for project managers is the accurate and reliable estimates of the required software development effort, especially in the early stages of the software development life cycle. Software effort drivers usually have properties of uncertainty and vagueness when they are measured by human judgment. A software effort estimation model utilising fuzzy inference system can overcome these characteristics of uncertainty and vagueness exist in software effort drivers. However, the determination of the suitable fuzzy rule sets for fuzzy inference plays an important role in coming up with accurate and reliable effort estimates. Software effort estimation using fuzzy logic is an attempt in the area of software project estimation. The objective of this work is to provide a technique for software cost estimation that performs better than other techniques on the accuracy of effort estimation and a given set of test cases. This paper presented a new model for handling imprecision and uncertainty by using the fuzzy logic systems. This work has shown by applying fuzzy logic on the algorithmic and non-algorithmic software effort estimation models accurate estimation is achievable. The proposed fuzzy logic model showed better software effort estimates in view of the *MMRE*, *Pred*(0.25) evaluation criteria as compared to the traditional COCOMO. The above-mentioned results demonstrate that applying fuzzy logic method to the software effort estimation is a feasible approach to addressing the problem of uncertainty and vagueness existed in software effort drivers. Furthermore, the fuzzy logic model presents better estimation accuracy as compared to the original COCOMO dataset. The utilisation of fuzzy logic for other applications in the software engineering field can also be explored in the future.

REFERENCES

1. Boehm, B.W., 1981. Software Engineering Economics, *Englewood Cliffs*, NJ, Prentice-Hall.
2. Putnam, L.H., 1978. A General Empirical Solution to the Macro Software Sizing and Estimating Problem, *IEEE Transactions on Software Engineering*, 4(4): 345-361.
3. Boehm B., C. Abts and S. Chulani, 2000. Software Development Cost Estimation Approaches - A Survey, University of Southern California Center for Software Engineering, Technical Reports, USC-CSE-2000-505.
4. MacDonell, S.G. and A.R. Gray, 1997. A Comparison of Modeling Techniques for Software evelopment Effort Prediction, in Proceedings of the 1997 International Conf. on Neural Information Processing and Intelligent Information Systems, Dunedin, New Zealand, Springer-Verlag, pp: 869-872.
5. Schofield C., 1998. Non-Algorithmic Effort Estimation Techniques, Technical Reports, Department of Computing, Bournemouth University, England, March.

6. Venkatachalam, A.R., 1993. Software cost estimation using artificial neural networks. Proceedings of the 1993 International Joint Conference on Neural Networks, pp: 987-990.
7. Kumar, S. A. Krishna and P. Satsangi, 1994. Fuzzy systems and neural networks in software engineering project management, J. Applied Intelligence, 4: 31-52.
8. Hodgkinson, A.C. and P.W. Garratt, 1999. A NeuroFuzzy Cost Estimator, in Proceedings of the 3rd International Conference on Software Engineering and Applications - SAE., pp: 401-406.
9. Lotfi Zadeh, A., 1994. Fuzzy Logic, Neural Networks and Soft Computing, Communication of ACM., 37(3): 77-84.
10. Lotfi Zadeh, A., 2001. The Future of Soft Computing, In Joint 9th IFSA World Congress and 20th NAFIPS International Conference, Vancouver, Canada.
11. Shepperd, M.J. and C. Schofield, 1997. Estimating software project effort using analogies. *IEEE Transactions on Software Engineering,* 23(11): 736-743.
12. Fei, Z. and X. Liu, 1997. f-COCOMO: Fuzzy Constructive Cost Model in Software Engineering, in Proceedings of the IEEE International Conference on *Fuzzy Systems*, IEEE Press, New York, pp: 331-337.
13. Ryder, J., 1998. Fuzzy modeling of software effort prediction, in Proceedings of IEEE Information Technology Conference, Syracuse, NY.
14. Jingzhou, L. and R. Guenther, 2008. Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+, in Proceedings of Empirical Software Engineering J. 13(1): 63-96.
15. Idri, A., A. Zahi and A. Abran, 2006. Software Cost Estimation by Fuzzy Analogy for Web Hypermedia Applications, in Proceedings of the International Conference on Software Process and Product Measurement, Cadiz, Spain, pp: 53-62.
16. Molokken, K. and M. Jorgensen, 2003. A review of software surveys on software effort estimation, in Proceedings of IEEE International Symposium on Empirical Software Engineering, ISESE., pp: 223-230.
17. Huang, S. and N. Chiu, 2009. Applying fuzzy neural network to estimate software development effort , in Proceedings of Applied Intelligence J., 30(2): 73-83.
18. Boehm, B., 1995. 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: 45-60.
19. Boetticher, G.D., 1995. An Assessment of Metric Contribution in the Construction of a Neural Network-Based Effort Estimator, in Proceedings of 2nd International Workshop on Soft Computing Applied to Software Engineering.
20. Liu, H. and L. Yu, 2005. Toward Integrating Feature Selection Algorithms for Classification and Clustering, *IEEE Transactions on Knowledge and* Data Engineering, 17(4): 491-502.
21. MacDonell, S.G., A.R. Gray and M.J. Calvert, 1999. FULSOME: A Fuzzy Logic Modeling Tool for Software Metricians, in Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society - NAFIPS, IEEE., pp: 263-267.