Software Cost Estimation using Fuzzy Logic

Ravishankar. S
PG student,
Department of CSE,
Government College of Engineering,
Tirunelveli.

P. Latha
Associate Professor,
Department of CSE,
Government College of Engineering,
Tirunelveli.

ABSTRACT

The process of estimating time and cost required for developing software is called software cost estimation. It is one of the steps to be carried out in project planning. Early software estimation models are based on regression analysis or mathematical derivations. Today's models are based on simulation, neural network, genetic algorithm, soft computing, fuzzy logic modeling etc. This paper 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. 63 Historic projects of NASA dataset having COCOMO format is used in the evaluation of the proposed Fuzzy Logic COCOMO II. Eight membership functions available in fuzzy logic are used and a comparison is made to find out which membership function yields better result in terms of Mean Magnitude of Relative Error (MMRE) and PRED (25%).

Keywords

software cost estimation models, COCOMO II, soft computation techniques, fuzzy logic, Membership Function, Mean Relative Error, PRED (25%).

1. INTRODUCTION

It is the responsibility of the project manager to make accurate estimations of effort and cost. This is particularly true for projects subject to competitive bidding where a bid too high compared with competitors would result in losing the contract or a bid too low could result in a loss to the organization. Industry has a need for accurate estimates of effort and size at a very early stage in a project. However, when software cost estimates are done early in the software development process the estimate can be based on wrong or incomplete requirements. A software cost estimate process is the set of techniques and procedures that organizations use to arrive at an estimate. An important aspect of software projects is to know the cost and the major contributing factor is effort. Software cost estimation is applied in various government and non-government organizations, defense organizations, aeronautics, etc.

Various estimation models used are given by :

1.1 Sel Model

The Software Engineering Laboratory (SEL) of the University of Maryland has established a model i.e. SEL Model [2] for estimation. Estimation of effort according to SEL model is as follows:

$$E_{SEL} = 1.4 * (L)^{0.93}$$

Effort (E in Person-Months) and lines of code (L in thousands of lines of code i.e. KLOC) are used as predictors.

1.2 Walston-Felix Model

The model developed by Walston and Felix at IBM provides a relationship between delivered lines of source code (L in thousands of lines) and effort E (E in person-month). This model constitutes various aspects of the software development environment such as user participation, customer-oriented changes, memory constraints etc. According to Walston and Felix model [2], effort is computed by:-

$$E_{W\text{-}F} = 5.2 * (L)^{0.91}$$

1.3 Basic Cocomo Model

Basic COCOMO computes software development effort (and cost) as a function of program size. Program size is expressed in estimated thousands of source lines of code (SLOC). COCOMO [2] applies to three classes of software projects:

- Organic projects - "small" teams with "good" experience working with "less than rigid" requirements
- Semi-detached projects - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- Embedded projects - developed within a set of "tight" constraints .it is also combination of organic and semi-detached projects (hardware, software, operational, etc.

The basic COCOMO equations take the form

Effort Applied,

$E = a_b * (SLOC)^{b}{}_{b}$ [man-months]

Development Time,

$D = c_b * (\text{Effort Applied})^{d}{}_{b}$ [months]

People required ,

P= Effort Applied / Development Time [count]

where, SLOC is the estimated number of delivered lines (expressed in thousands) of code for project.

1.4 Intermediate Cocomo Model

COCOMO computes software development effort as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel and project attributes. This extension considers a set of four "cost drivers" which are Product attributes, Hardware attributes, Personnel attributes and Project attributes.

1.5 Detailed Cocomo Model

Detailed COCOMO [2] incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step (analysis, design, etc.) of the software engineering process. The detailed model uses different efforts multipliers for each cost drivers attribute these Phase Sensitive effort multipliers are each to determine the amount of effort required to complete each phase. In detailed COCOMO, the effort is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle.

The five phases of detailed COCOMO are:-

- Plan and requirement.
- System design.
- Detailed design.
- Module code and test.
- Integration and test.

1.6 Putnam Model

The Putnam model [5] is an empirical software effort estimation model. As a group, empirical models work by collecting software project data (for example, effort and size) and fitting a curve to the data. Future effort estimates are made by providing size and calculating the associated effort using the equation which fit the original data. Putnam model describes the time and effort required to finish a software project of specified size. SLIM (Software Life cycle Management) is the name given by Putnam to the proprietary suite of tools his company QSM, Inc. has developed based on his model. It is one of the earliest of these types of models developed, and is among the most widely used.

2. PROPOSED SYSTEM

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.

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

2.1 Input Data set Description

The COCOMO II effort estimation model was introduced in equation given below:

$$\text{Effort}_{PM} = A * [\text{Size}]^{E} * \prod_{i=1}^{17} EM_i$$

The inputs are the Size of software development, a constant, A, an exponent, E and a number of values called effort multipliers (EM) [1]. The number of effort multipliers depends on the model. The Size is KSLOC. This is derived from estimating the size of software modules that will constitute the application program. It can also be estimated from unadjusted function points (UFP) [3] , converted to SLOC, then divided by one thousand.

Cost drivers are used to capture characteristics of the software development that affect the effort to complete the project. A cost driver is a model factor that drives the cost (in this case Person-Months) estimated by the model. All COCOMO II cost drivers have qualitative rating levels that express the impact of the driver on development effort. These ratings can range from Extra Low to Extra High. Each rating level of every multiplicative cost driver has a value, called an effort multiplier (EM) associated with it. This scheme translates a cost driver's qualitative rating into a quantitative one for use in the model. The EM value assigned to a multiplicative cost driver's nominal rating is 1.00.

The table1 shows the the range of Scale Factors (SFs) used.

Table 1: The range of COCOMO II SFs

No.	Scale Factor	Range
1	Precedentedness (PREC)	0.00-6.20
2	Development Flexibility (FLEX)	0.00-5.07
3	Architecture/Risk Resolution (RESL)	0.00-7.07
4	Team Cohesion (TEAM)	0.00-5.48
5	Process Maturity (PMAT)	0.00-7.80

The table 2 shows the range of Effort Multipliers (EMs) used.

Table 2: The range of COCOMO II EMs

No.	Effort Multiplier	Range
1	Required software reliability (RELY)	0.82-1.26
2	Database size (DATA)	0.90-1.28
3	Product complexity (CPLX)	0.73-1.74
4	Developed for reusability (RUSE)	0.95-1.24
5	Documentation match to life cycle needs (DOCU)	0.81-1.23
6	Execution time constraint (TIME) 1	1.00-1.63
7	Main storage constraint (STOR)	1.00-1.46
8	Platform volatility (PVOL)	0.87-1.30
9	Analyst capability (ACAP)	1.42-0.71
10	Programmer capability (PCAP)	1.34-0.76
11	Personnel continuity (PCON)	1.29-0.81
12	Applications experience (APEX)	1.22-0.81
13	Platform experience (PLEX)	1.19-0.85
14	Language and tool experience (LTEX)	1.20-0.84
15	Use of software tools (TOOL)	1.17-0.78
16	Multi site development (SITE)	1.22-0.80
17	Required development schedule (SCED)	1.43-1.00

2.2 The COCOMO II

The COCOMO I [1] model is a regression-based software cost estimation model, which was developed by Boehm in 1981 and thought to be the most cited 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.

Figure 1 shows 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.

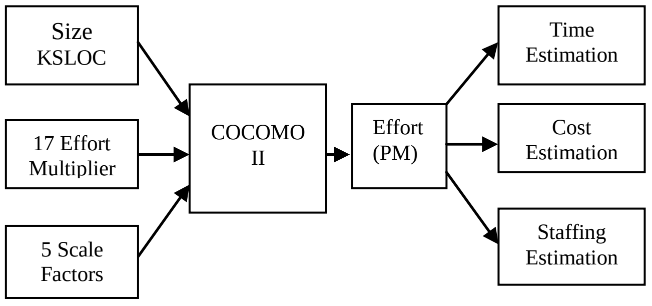


Figure 1: The process of software schedule, cost, and manpower estimation in COCOMO II

The formula for the process is given by:

$$\text{Effort}_{PM} = A * [\text{Size}]^{B+0.01*\sum_{j=1}^{5} SF_j} * \prod_{i=1}^{17} EM_i$$

$$\text{Schedule}_{Months} = C * (\text{Effort})^{D+0.02*0.01*\sum_{j=1}^{5} SF_j}$$

$$\text{Average Staffing}_{People} = \text{Effort}/\text{Schedule}$$

$$\text{Cost} = \text{Effort} * (\text{Payment}_{Month})$$

A=2.94; B=0.91; C=3.67; D=0.28

Size : Software Size(SLOC)

2.3 Fuzzy Logic

In 1965, Lofti Zadeh formally developed multi-value set theory, and introduced the term fuzzy logic.[4]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. The membership!(") of an element x of a classical set A, as subset of the universe X, is defined by (2), as follows:

$$\mu_A(x) = 1 \text{ if } x \in A$$
$$\mu_A(x) = 0 \text{ if } x \notin A$$

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 . 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. 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.

2.4 The FuzzyLogic 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 Figure2.

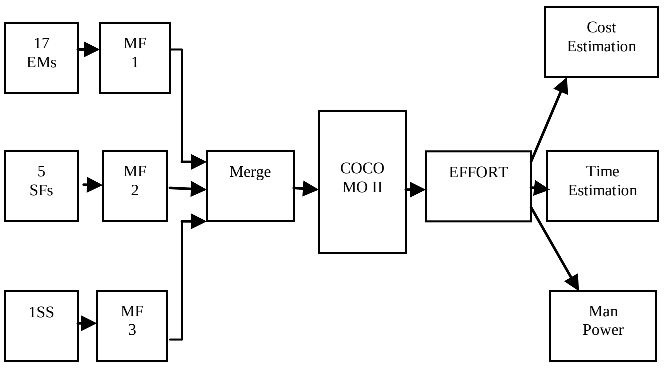
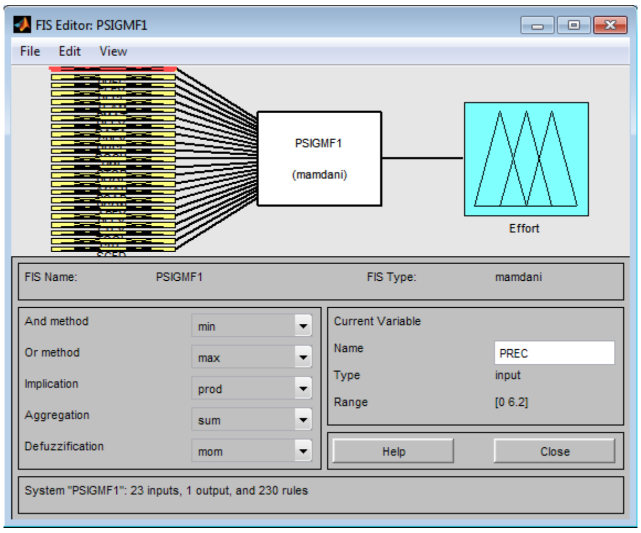


Fig 2: The architecture of the FL-COCOMO II

The figure3 shows the Fuzzy Inference System (FIS) editor for the proposed model using PSIGMF.



FIS Name: PSIGMF1 FIS Type: mamdani

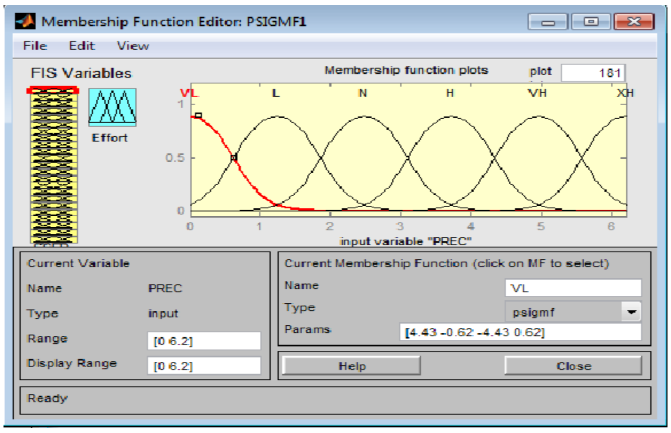
And method: min
Or method: max
Implication: prod
Aggregation: sum
Defuzzification: mom

Current Variable
Name: PREC
Type: input
Range: [0 6.2]

System "PSIGMF1": 23 inputs, 1 output, and 230 rules

Fig3: FIS tool in MATLAB software.

The figure4 shows the fuzzification process of the PREC scale factor using FIS tool available in MATLAB.



Current Variable
Name: PREC
Type: input
Range: [0 6.2]
Display Range: [0 6.2]

Current Membership Function (click on MF to select)
Name: VL
Type: psigmf
Params: [4.43 -0.62 -4.43 0.62]

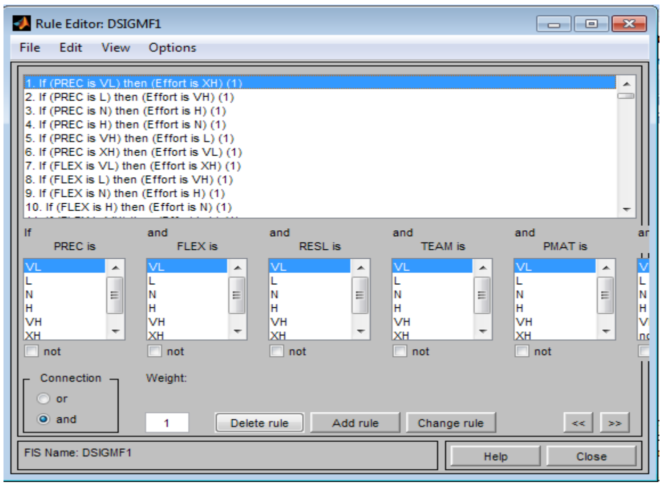
Ready

Fig4: The fuzzification of PREC Scale factor using FIS tool in the MATLAB software.

The fuzzy rules for the FL-COCOMO II were defined through the linguistic variables in the fuzzification process. The fuzzy rules were also defined based on the connective AND between input variables. Some of the examples of rules framed were shown below:

If (PREC is VL) then (Effort is XH)
If (PMAT is VH) then (Effort is L)
If (CPLX is H) then (Effort is H)
If (PREC is L) and (KLOC is VH) then (Effort is VH)

The figure 5 shows the rule editor in the fuzzy logic tool box, which was used for framing rules.



```
1. If (PREC is VL) then (Effort is XH) (1)
2. If (PREC is L) then (Effort is VH) (1)
3. If (PREC is N) then (Effort is H) (1)
4. If (PREC is H) then (Effort is N) (1)
5. If (PREC is VH) then (Effort is L) (1)
6. If (PREC is XH) then (Effort is VL) (1)
7. If (FLEX is VL) then (Effort is XH) (1)
8. If (FLEX is L) then (Effort is VH) (1)
9. If (FLEX is N) then (Effort is H) (1)
10. If (FLEX is H) then (Effort is N) (1)
```

FIS Name: DSIGMF1

Fig5: The fuzzy rule generation using FIS tool in the MATLAB software

The defuzzification of the output "Effort" is performed using the Mean of Maximum (MOM) technique.

3. EVALUATION METHODS

The evaluation methods used are Mean Magnitude of Relative Error (MMRE) and PRED(25%). The Magnitude of Relative Error (MRE) is defined as :

$\text{MRE}_i = (\text{Actual Effort}_i - \text{Predicted Effort}_i) / \text{Actual Effort}_i$

The MRE value is 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) as follows:

$$\text{MMRE} = (1/N) \sum_{i=1}^{N} \text{MRE}_i$$

where N is the total number of observations.

PRED (25%) [1] is defined as the number of observations which have got MRE less than 0.25.

Table 3: MMRE and PRED (25%) for various membership functions

Sl.No.	Membership Function	MMRE	PRED(25%)
1	dsigmf	0.37127	33.333
2	gauss2mf	0.37625	34.92063
3	gaussmf	0.359206	36.50794
4	gbellmf	0.374428	36.50794
5	pimf	0.35373	36.50794
6	psigmf	0.346349	36.50794
7	trapezoidalmf	0.358095	36.50794
8	triangularmf	0.675961	33.333

Software cost is estimated using the equation which is given below:

$\text{Cost} = \text{Effort} * (\text{Payment}_{\text{Month}})$

(Assuming $\text{Payment}_{\text{Month}}$ = Rs 20000)

4. RESULTS

The figure 6 shows the comparison of MMRE for various membership functions

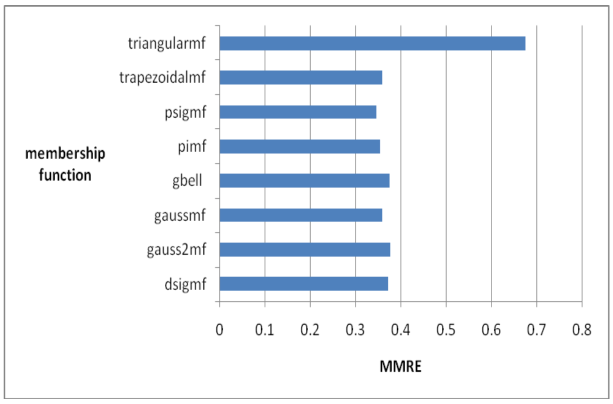


Fig6: Comparison of MMRE for various membership functions.

The figure 7 shows the comparison of PRED (25%) for various membership functions.

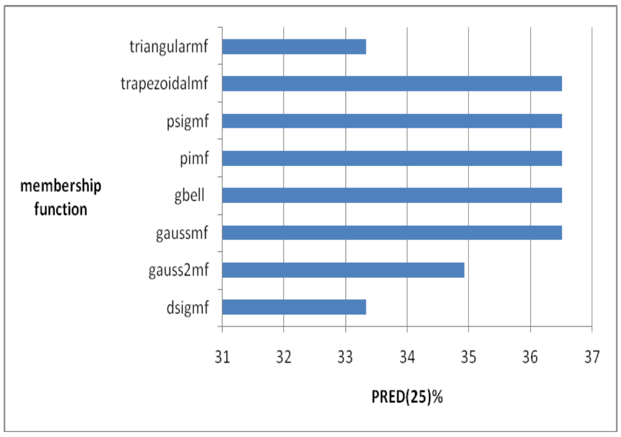


Fig 7: Comparison of PRED(25%) for various membership functions.

The table 3 shows the MMRE and PRED (25%) values estimated for the proposed FL_COCOMOII model when different membership functions available in fuzzy logic were used.

The figure 8 shows the estimated effort values for the NASA's 63 project data set using FL_COCOMOII method when the Product Sigmoid (psigmf) Membership Function is used.

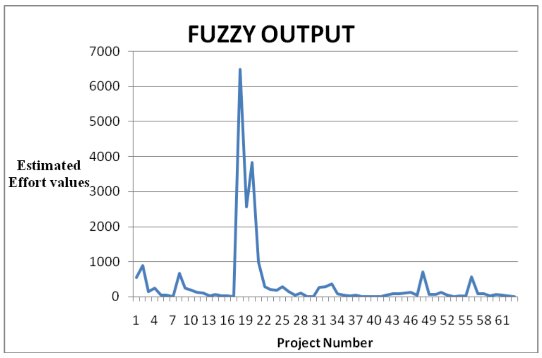


Fig 8: Estimated effort values for the NASA's 63 project dataset using psigmoid Membership Function

5. 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 paper was to examine the application of applying fuzzy logic in software cost estimation that can perform more accurate result.

Hence from the table 3, the Product Sigmoid membership function (psigmf) yields least MMRE and best PRED (25%).

Cost can be found out using the equation if payment is known

$$\text{Cost} = \text{Effort} * (\text{Payment}_{\text{Month}})$$

Therefore the effort needed for a particular software project using fuzzy logic is estimated. Also the effort is calculated using various membership functions and compared the result based on the MMRE and PRED (25%) obtained for each of the membership functions.

6. REFERENCES

[1] Iman Attarzadeh and Siew Hock Ow," Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model" IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, June 27-30, 2011.

[2] Mohd. Sadiq, Farhana Mariyam, Aleem Ali, Shadab Khan, Pradeep Tripath, "Prediction of Software Project Effort Using Fuzzy Logic" IEEE International Conference on Fuzzy Systems, March 2011.

[3] Mohd. Sadiq, Abdul Rahman, Shabbir Ahmad, Mohammad Asim, Javed Ahmad", esrcTool: A Tool to Estimate the Software Risk and Cost", IEEE Second International Conference on Computer Research and Development, pp. 886-890, July 2010.

[4] Prasad Reddy P.V.G.D, Sudha K.R , Rama Sree P & Ramesh S.N.S.V.S.C,"Fuzzy Based Approach for Predicting Software Development", International Journal of Software Engineering (IJSE), Volume (1): Issue (1).

[5] 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. problem", IEEE Transactions on Software Engineering,4(4),pp.345-361,1978.