International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 6 Issue: 5 220 – 225

Research Paper on Software Cost Estimation Using Fuzzy Logic

Nishi
M. Tech Scholar
B.P.S.M.V University, Sonepat
nishisinghal54@gmail.com

Mr. Vikas Malik
Assistant Professor
B.P.S.M.V University, Sonepat
vikassmalik@gmail.com

Abstract: Software cost estimation is one of the biggest challenges in these days due to tremendous completion. You have to bid so close so that you can get the consignment if your cost estimation is too low are too high in that cases organization has to suffer that why it becomes very crucial to get consignment. 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 our research was to examine the application of applying fuzzy logic in software cost estimation that can perform more accurate result. In fuzzy logic there are various membership function for example Gaussian, triangular, trapezoidal and many more. Out of these by hit and trial method we find triangular membership function (MF) yields least MRE and MMRE and this MRE must be less than 25%. In our research this value came around 15% which is very fair enough to estimate. Cost can be found out using the equation if payment is known Cost = Effort * (Payment Month). Therefore the effort needed for a particular software project using fuzzy logic is estimated. In our research NASA (93) data set used to calculate fuzzy logic COCOMO II. From this table size of code and actual effort has been taken. In end after comparing the result we found that our proposed technique is far superior to base work.

Keywords- *COCOMO, Fuzzy, Effort Multiplier, Software, WBS, Accurate, Risk, MMRE, MRE*

*****

I. INTRODUCTION

Software project failures have been an important subject in the last decade. Software Cost estimation is a prediction of the cost of the resources that will be required to complete all of the work of the software project. Uncertainties are referred as a risk .Risk is a measure of future uncertainties in achieving program performance goals and objectives within defined cost, schedule and performance constraints [1-2].

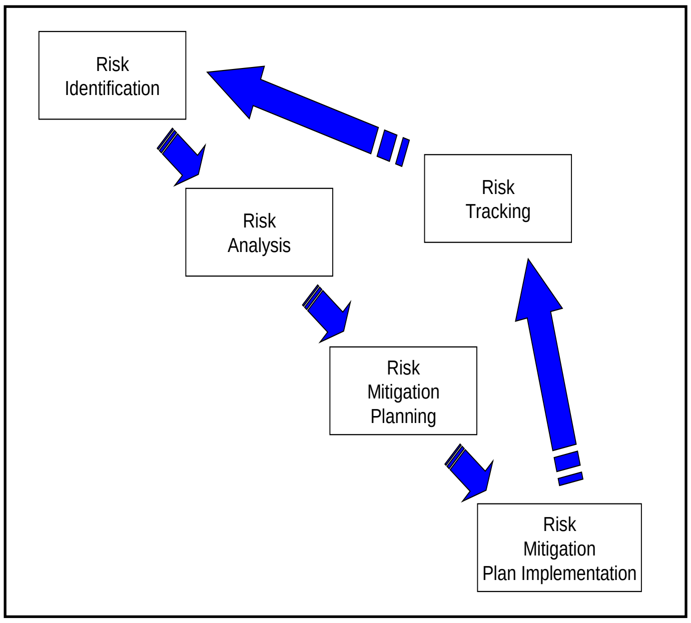


Fig. 1 Risk Management Process

Risk can be associated with all aspects of a program (e.g., threat, technology maturity, supplier capability, design maturation, performance against plan,) as these aspects relate across the Work Breakdown Structure (WBS) and Integrated Master Schedule (IMS) [3-5]. Risk addresses the potential variation in the planned approach and its expected outcome. While such variation could include positive as well as negative effects, this guide will only address negative future effects since programs have typically experienced difficulty in this area during the acquisition process

II. BACKGROUND HISTORY

The Constructive Cost Model (COCOMO) was developed by Barry Boehm of TRW and published in 1981 . Based on his analysis of 63 software-development projects, Boehm developed an easy-to-understand model that predicts the effort and duration of a project, based on inputs relating to the size of the resulting systems and a number of “cost drivers” that Boehm believes affect productivity [6]. A simplified version of the essential COCOMO effort equation for the Basic Model (the Intermediate and Detailed Models are discussed later) is of the form

$$MM = C (KDSI)^{k},$$

Where;
M = number of man-months1 (= 152 working hours),
C = a constant,

KDSI = thousands of “delivered source instructions” (DSI), and
k = a constant.

Boehm defines DSI as program instructions created by project personnel that are delivered as part of the final product [7]. They exclude comments and unmodified utility software, and include job control language, format statements, and data declarations. In Boehm’s development of COCOMO, he found that the Basic Model predicted effort within a factor of 1.3 only 29 percent of the time and within a factor of 2 only 60 percent of the time for his 63-project database. In an effort to improve the model’s accuracy, he refined the equation to include the effects of 15 “cost drivers,” which are attributes of the end product, the computer used, the personnel staffing, and the project environment. He believes that these 15 factors affect the project’s productivity and calls this version the Intermediate Model. The COCOMO Detailed Model is very similar to the Intermediate Model except that the project is divided into four phases: Product design, detailed design, Coding/Unit Test, and Integration/Test. The 15 cost drivers are estimated and applied to each phase separately, rather than to the project as a whole [8].

(A): Bailey-Basil Model

This model developed by Bailey-Basil between delivered lines of source code and formulates a relation:

$$\text{EFFORT} = 5.5\ (\text{KLOC})^{1.16}$$

(B): Halstead Model: This model developed by Halstead between delivered lines of source code and formulates a relation:

$$\text{EFFORT}=0.7(\text{KLOC})^{1.50}$$

(C) Putnam Model: The Putnam model is an empirical software effort estimation model. Putnam used his observations about productivity levels to derive the software equation:

Technical constant C= size * B1/3 * T4/3

Total Person Months B=1/T4 *(size/C) 3

T= Required Development Time in years

Size is estimated in LOC

Where: C is a parameter dependent on the development environment and is determined on the basis of historical data of the past projects.

Rating: C=2,000 (poor), C=8000 (good) C=12,000 (excellent).

The Putnam model is very sensitive to the development time: decreasing the development time can greatly increase the person-months needed for development. One significant problem with the Putnam model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) of the software to be developed. There is often great uncertainty in the software size. It may result in the inaccuracy of cost estimation**.**

In year 2011, researcher performed a work “Analysis of Software Cost Estimation using COCOMO II”. This paper shows how to make cost estimates using COCOMO II for a sample project, and outlines basic steps, terms, and tools used. Obviously, ad hoc estimates are prone to error. COCOMO II make it easy for you to clarify not only an expected project cost and duration, but also prompt you to verify all basic sides of a software project by providing clear, compact, and concise terms, methodology, which are tested on a wide range of real life projects and thus reduce essentially project risks and provide reasonable grounds for communication with a project stockholder. Paper presents difference in between estimation by COCOMO II and actual time taken by the project.

III. METHODOLOGY

The term COCOMO stands for “Constructive Cost Model”. It helps to calculate effort, cost and schedule for software project. This model was proposed by Barry W. Boehm. COCOMO first version named as “COCOMO81” and it became one of the popular cost estimation models of 1980’s.COCOMO Model may apply to three different modes of software project [12]. COCOMO provides an estimate of effort for a software project using a single property. The property used for the calculation of the corresponding estimate is the size of the software. This property is expressed in terms of KLOC (Kilo Lines of Code) and is calculated using the following formula:

$$\text{Effort}=X\times (\text{Size})^{y}$$

X, Y: Constants (depend upon the software project mode).

Most software systems, especially the large ones, consist of different components or subsystems. The complexity for each of these components or subsystems is most likely different from others. Taking this into account entity, the complete COCOMO considers the system as heterogeneous in nature. According to the complete COCOMO, a software system is composed of various components or subsystems where each of these components or subsystems has different attributes from each other. COCOMO-II was published in 1995 having three sub models; an application-composition model, an early design model and a post-architecture model. COCOMO-II has, a set of seventeen Effort Multipliers (EM) or cost drivers as an input, which are used to adjust the nominal effort (PM) to reflect the software product being developed.

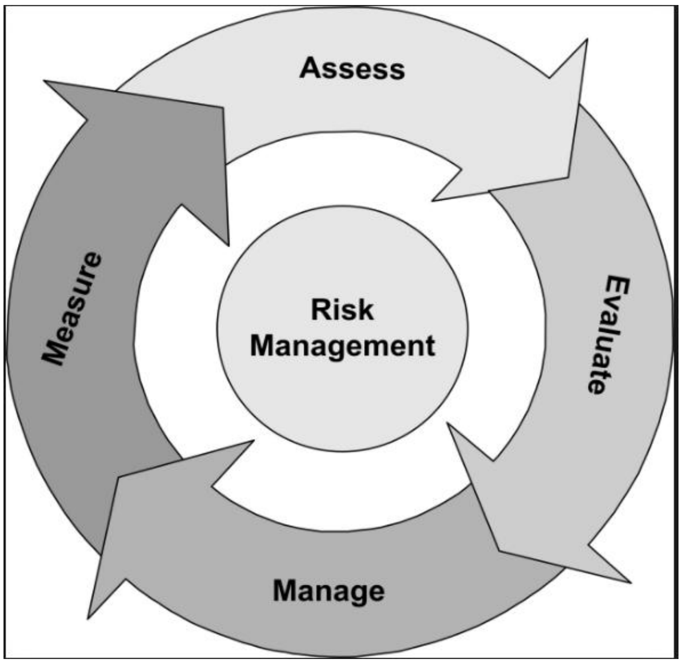


Fig.2 Risk Management Cycle

The concept of fuzzy logic (FL) is not a bearing methodology, however could be a means of process knowledge by permitting partial set membership rather than crisp set membership or non-membership. It supports fuzzy set theory. Fuzzy systems are knowledge based or rule based system. The heart of fuzzy systems is a knowledge base consisting of the so called Fuzzy "If Then rules" in which some words are characterized by continuous member functions. The popular fuzzy logic systems can be classified into three types: pure fuzzy logic system, 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 fuzzy logic system with fuzzifier and defuzzifier [10-11]. It was first proposed by Mamdani. It has been successfully applied to a variety of industrial processes and consumer products as show in Fig3 below:

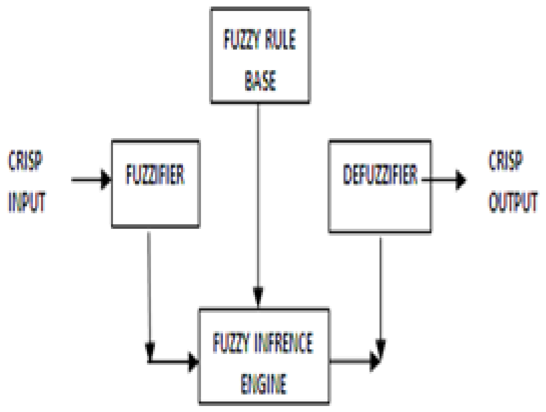


Fig.3 Fuzzy Logic System

FUZZIFIER - It converts the crisp input into a fuzzy set, and to describe situation graphically, membership functions are used.

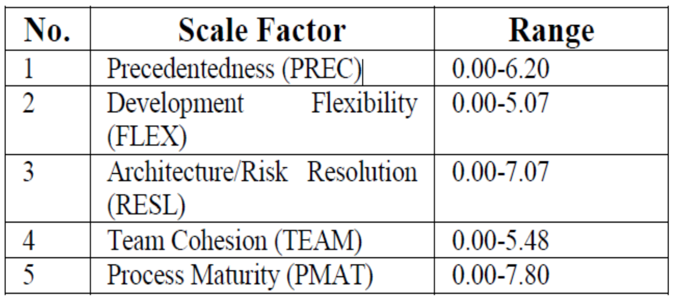
FUZZY Rule Base- It uses "if-then rules" formulae.

FUZZY Inference Engine- A collection of if -then rules stored in fuzzy rule base is known as inference engine. It performs two functions i.e., aggregation and composition.

Defuzzification- It converts fuzzy output into crisp output.

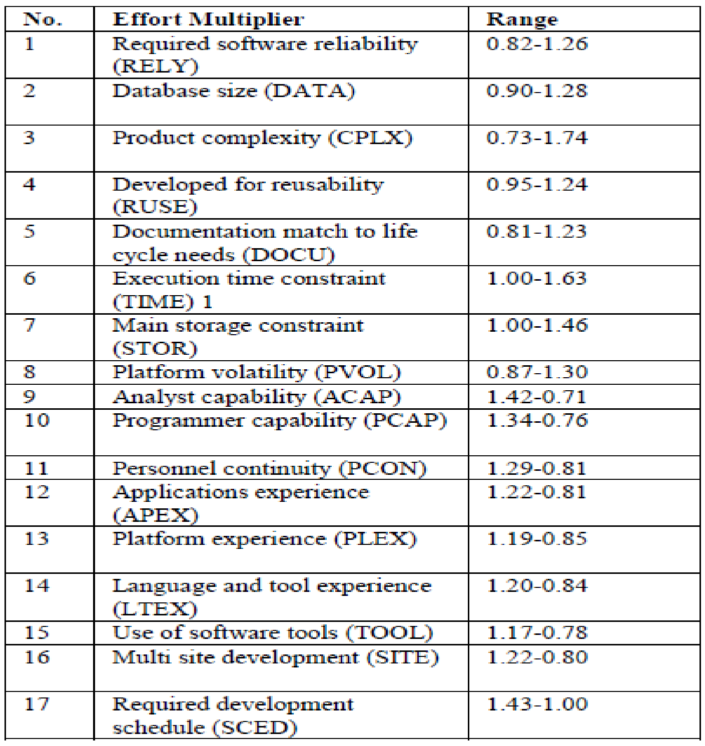
IV. EXPERIMENTAL RESULT

SOFTWARE: MATLAB R2015A: It is powerful software that provides an environment for numerical computation as well as graphical display of outputs. In Matlab the data input is in the ASCII format as well as binary format. It is high-performance language for technical computing integrates computation, visualization, and programming in a simple way where problems and solutions are expressed in familiar mathematical notation.



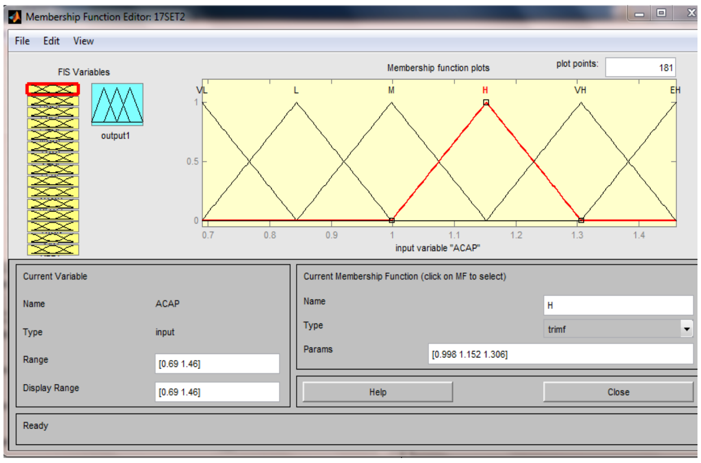
No.	Scale Factor	Range
1	Precedentedness (PREC)	0.00-6.20
2	Development Flexibility (FLEX)	0.00-5.07
3	Architecture/Risk Resolution (RESL)	0.00-7.07
4	Team Cohesion (TEAM)	0.00-5.48
5	Process Maturity (PMAT)	0.00-7.80

Fig.4 Scale Factor



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

Fig.5 Efforts Multiplier

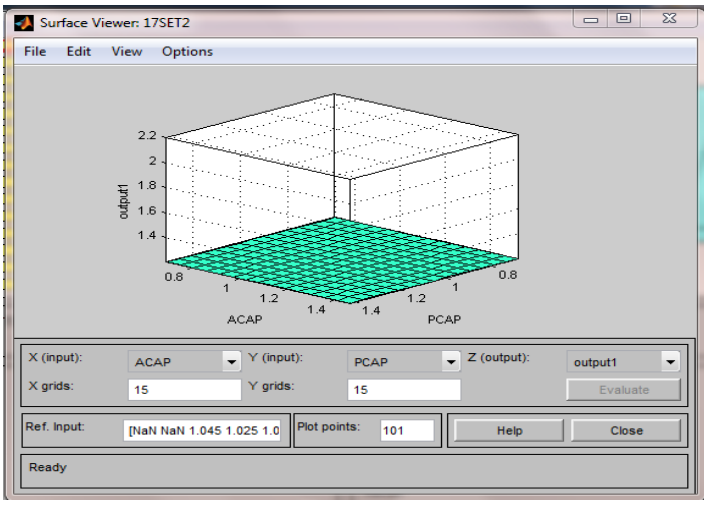


Current Variable
Name ACAP
Type input
Range [0.69 1.46]
Display Range [0.69 1.46]

Current Membership Function (click on MF to select)
Name H
Type trimf
Params [0.998 1.152 1.306]

Ready

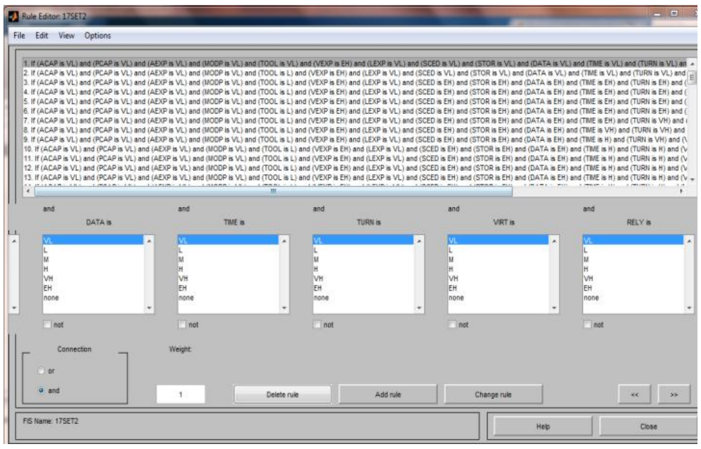
Fig.6 Effort Multiplier triangular function



X (input): ACAP
Y (input): PCAP
Z (output): output1
X grids: 15
Y grids: 15
Ref. Input: [NaN NaN 1.045 1.025 1.0
Plot points: 101

Ready

Fig.9 Effort Multiplier 3 D view for surface

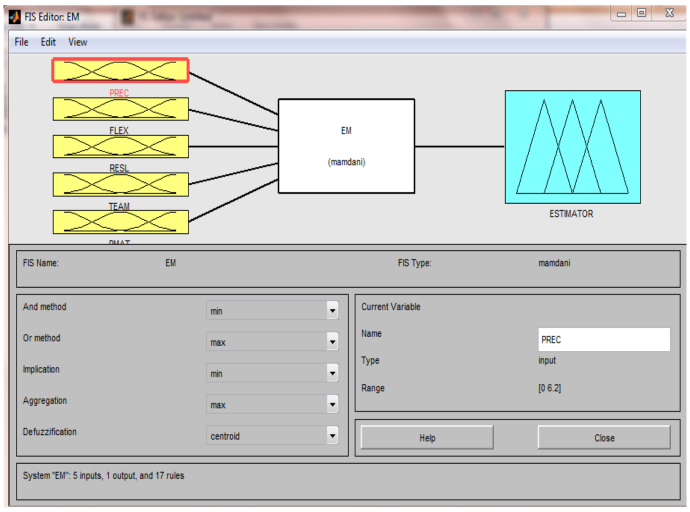


1. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is VL) and (VEXP is EH) and (LEXP is VL) and (SCED is VL) and (STOR is VL) and (DATA is VL) and (TIME is VL) and (TURN is VL) an
2. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is VL) and (STOR is VL) and (DATA is VL) and (TIME is VL) and (TURN is VL) and
3. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is EH) and (TURN is EH) and (
4. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is EH) and (TURN is EH) and (
5. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is EH) and (TURN is EH) and (
6. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is EH) and (TURN is EH) and (
7. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is EH) and (TURN is VH) and (
8. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is VH) and (TURN is VH) and
9. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is H) and (TURN is VH) and (V
10. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is H) and (TURN is H) and (V
11. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is H) and (TURN is H) and (V
12. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is H) and (TURN is H) and (V
13. If (ACAP is VL) and (PCAP is VL) and (AEXP is VL) and (MODP is VL) and (TOOL is L) and (VEXP is EH) and (LEXP is VL) and (SCED is EH) and (STOR is EH) and (DATA is EH) and (TIME is H) and (TURN is H) and (V

Weight: 1

FIS Name: 17SET2

Fig.7 Effort Multiplier rule

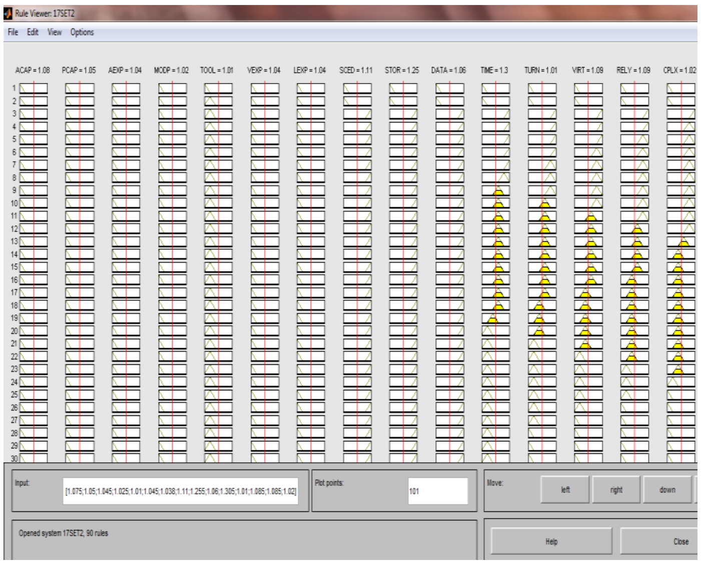


FIS Name: EM
FIS Type: mamdani
And method min
Or method max
Implication min
Aggregation max
Defuzzification centroid

Current Variable
Name PREC
Type input
Range [0 6.2]

System "EM": 5 inputs, 1 output, and 17 rules

Fig.10 Fuzzy Logic for effort Multiplier

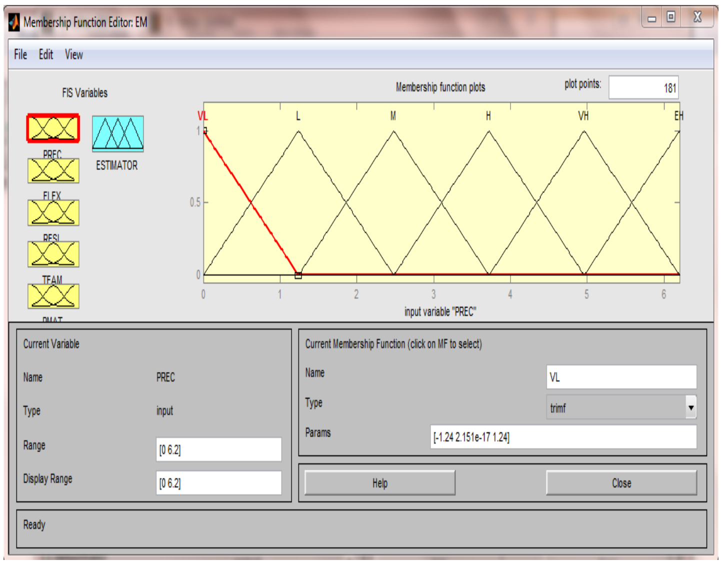


ACAP = 1.08 PCAP = 1.05 AEXP = 1.04 MODP = 1.02 TOOL = 1.01 VEXP = 1.04 LEXP = 1.04 SCED = 1.11 STOR = 1.25 DATA = 1.06 TIME = 1.3 TURN = 1.01 VIRT = 1.09 RELY = 1.09 CPLX = 1.02

Input: [1.075;1.05;1.045;1.025;1.01;1.045;1.038;1.11;1.255;1.06;1.305;1.01;1.085;1.085;1.02]
Plot points: 101

Opened system 17SET2, 90 rules

Fig.8 Effort Multiplier output

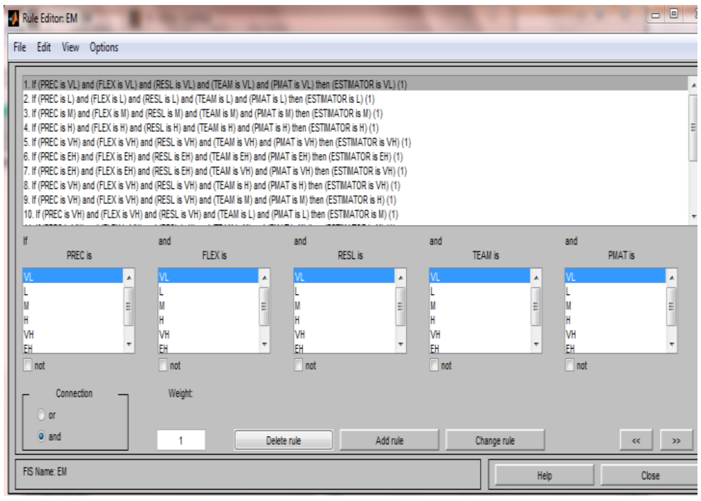


Current Variable
Name PREC
Type input
Range [0 6.2]
Display Range [0 6.2]

Current Membership Function (click on MF to select)
Name VL
Type trimf
Params [-1.24 2.151e-17 1.24]

Ready

Fig.11 Scale factor triangular membership function

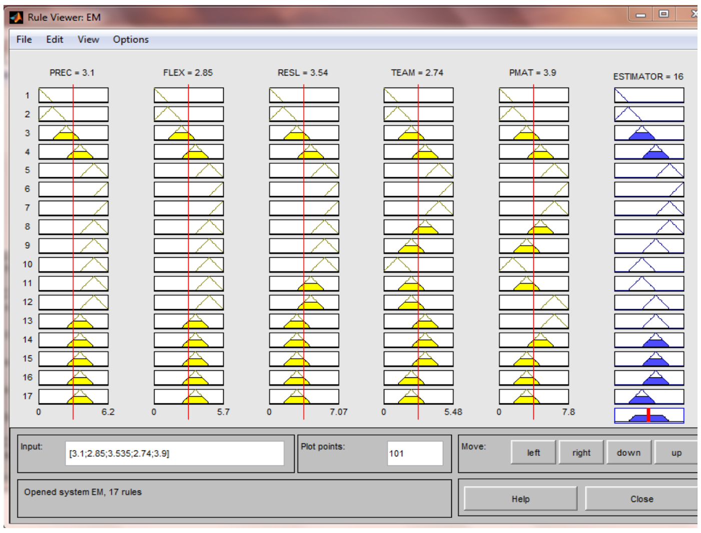


1. If (PREC is VL) and (FLEX is VL) and (RESL is VL) and (TEAM is VL) and (PMAT is VL) then (ESTIMATOR is VL) (1)
2. If (PREC is L) and (FLEX is L) and (RESL is L) and (TEAM is L) and (PMAT is L) then (ESTIMATOR is L) (1)
3. If (PREC is M) and (FLEX is M) and (RESL is M) and (TEAM is M) and (PMAT is M) then (ESTIMATOR is M) (1)
4. If (PREC is H) and (FLEX is H) and (RESL is H) and (TEAM is H) and (PMAT is H) then (ESTIMATOR is H) (1)
5. If (PREC is VH) and (FLEX is VH) and (RESL is VH) and (TEAM is VH) and (PMAT is VH) then (ESTIMATOR is VH) (1)
6. If (PREC is EH) and (FLEX is EH) and (RESL is EH) and (TEAM is EH) and (PMAT is EH) then (ESTIMATOR is EH) (1)
7. If (PREC is EH) and (FLEX is EH) and (RESL is EH) and (TEAM is VH) and (PMAT is VH) then (ESTIMATOR is VH) (1)
8. If (PREC is VH) and (FLEX is VH) and (RESL is VH) and (TEAM is H) and (PMAT is H) then (ESTIMATOR is VH) (1)
9. If (PREC is VH) and (FLEX is VH) and (RESL is VH) and (TEAM is M) and (PMAT is M) then (ESTIMATOR is H) (1)
10. If (PREC is VH) and (FLEX is VH) and (RESL is VH) and (TEAM is L) and (PMAT is L) then (ESTIMATOR is M) (1)

Weight: 1

FIS Name: EM

Fig.12 Scale factor rule

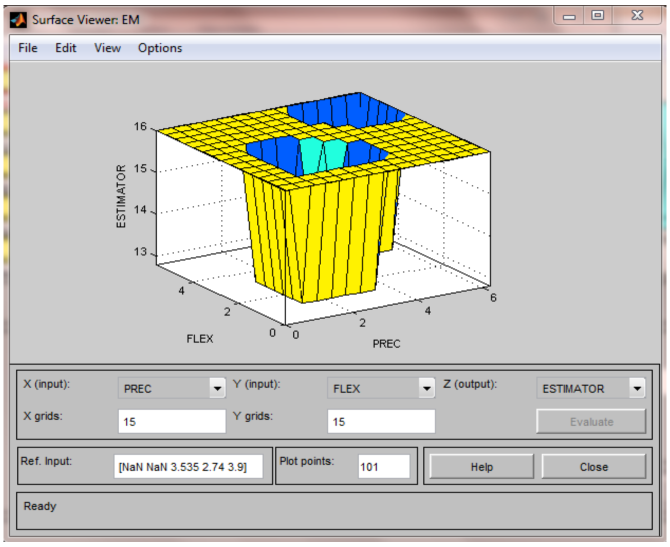


Input: [3.1;2.85;3.535;2.74;3.9]

Plot points: 101

Opened system EM, 17 rules

Fig.13 Scale factor output



X (input): PREC

Y (input): FLEX

Z (output): ESTIMATOR

X grids: 15

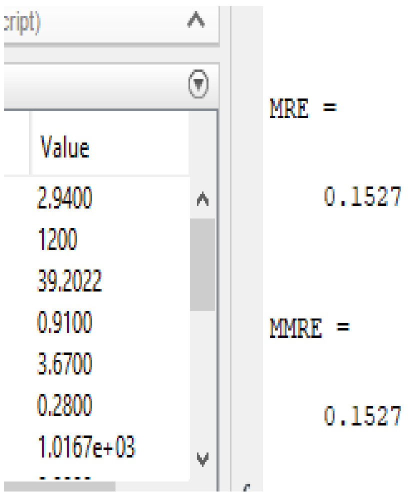
Y grids: 15

Ref. Input: [NaN NaN 3.535 2.74 3.9]

Plot points: 101

Ready

Fig.14 Scale factor 3 D view for surface



```
Value
2.9400
1200
39.2022
0.9100
3.6700
0.2800
1.0167e+03

MRE =

     0.1527

MMRE =

     0.1527
```

Fig.15 MMRE and MRE value

V. CONCLUSION

With pace of time day by day advanced technology floored in market. Today our main focus is to compete with advanced technology that's why we try to develop such an advance research which will be easy to use, more accurate, cost effective and reliable. Some time back in process of software development one issue is very crucial is accurate and reliable estimation of cost of software, manpower and time. Now a days in research area we use fuzzy logic tool box which is fourth generation technology and important thing regarding this tool box is that if you know basic very well then it will be very easy to develop new rule. Fuzzy logic can overcome the uncertainty and vagueness of software attributes. The objective of this paper was to examine the application of applying fuzzy logic in software cost estimation that can perform more accurate result. We have to calculate mean magnitude relative error (MMRE) and magnitude relative error (MRE) using triangular membership function for various scale factor and different efforts multiplier. 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.

REFERENCES

[1] Agarwal.R, Kumar.M, Malick.S, Bharadwaj.R.M, and Anantwar.D, "Estimating Software projects", ACM SIGSOFT, Vol 26, 2001.

[2] Basavaraj.M.J, Dr. Shet.K.C, "Software Estimation using Function Point Analysis Difficulties and Research Challenges", T.Sobh (ed.), Innovations and Advanced Techniques in Computer and Information Sciences and Engineering, PP 111-116. © 2007 Springer

[3] Azath.H, Wahidabanu.R.S.D, "Function Point: A Quality Loom for the Effort Assessment of Software Systems", International

Journal of Computer Science and Network Security, IJCSNS Vol.8 No12, Dec 2008

[4] 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.

[5] 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.

[6] 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).

[7] 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.

[8] Burgess C.J. and Lefley M., —Can genetic programming improve software effort estimation? A comparative evaluation‖, Information and Software Technology, 2001, Vol. 43, No. 14, pp. 863 -873.

[9] A. Idri, A. Abrian, and L. Kjiri, —COCOMO Cost Model using Fuzzy Logic‖, International Conference on Fuzzy Theory and Technology Atlantic, New Jersey, 2000.

[10] B.Boehm, C. Abts, S.Chulani,‖Software Development Cost Estimation Approaches: A Survey,‖ University of Southern California Centre for Software Engineering, Technical Report, USC-CSE-2000-505, 2000.

[11] B. Boehm, B. Clark, E. Horwitz, R. Madachy, C. Abts, S.Chulani, A.W.Brown and B. Steece, —COCOMO II model definition manual‖, Universityof South California Center for Software Engineering, 2000.

[12] M. Jorgenson and D.I.K. Sjoberg, —The impact of customer expectation on software development effort estimates‖, International Journal of Project Management,2004, Vol. 22, No. 4, pp. 317-325