International Journal of Research in Engineering and Science (IJRES)
ISSN (Online): 2320-9364, ISSN (Print): 2320-9356
www.ijres.org Volume 7 Issue 4 Ser. I ǁ 2019 ǁ PP. 85-92

Fluffy Based Programming Cost Assessment for Non Algorithmic Methodology

Bibhuti Bhusan Bhutia,Madan Mohan Sahu, Kedarnath Hota, Sitakanta Nayak
Department of Mechanical engineering, Gandhi Institute For Technology (GIFT), Bhubaneswar

Abstract: *Cost estimation is one of the most challenging tasks in project management. It is to accurately estimate needed resources and required schedules for software development projects. The software estimation process includes estimating the size of the software product to be produced, estimating the effort required, developing preliminary project schedules, and finally, estimating overall cost of the project. Effort is a function of size. The software industry does not estimate projects well. In this paper we have represented size in KLOC as a Fuzzy number. A new model is presented using fuzzy logic to estimate effort required in software development.*

Keywords: *estimation; budget; effort; LOC; FP.*

I. INTRODUCTION

Out of the three principal components of cost i.e., hardware costs, travel and training costs, and effort costs, the effort cost is dominant. Software cost estimation starts at the proposal state and continues throughout the life time of a project.

There are several techniques of software cost estimation:

- Algorithm Cost Model
- Expert Judgments
- Estimation by Analogy
- Top-down estimation
- Bottom-up estimation

Expert Judgment Method:

Expert judgment techniques involve consulting with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost[1].

Estimating by Analogy

Estimating by analogy means comparing the proposed project to previously completed similar project where the project development information id known. Actual data from the completed projects are extrapolated to estimate the proposed project. This method can be used either at system-level or at the component-level[1].

Top Down Estimating Method

Top-down estimating method is also called Macro Model. Using top-down estimating method, an overall cost estimation for the project is derived from the global properties of the software project, and then the project is partitioned into various low-level components[1].

Bottom up Estimating Method

Using bottom-up estimating method, the cost of each software components is estimated and then combine the results to arrive at an estimated cost of overall project. It aims at constructing the estimate of a system from the knowledge accumulated about the small software components and their interactions[1].

Algorithmic Method

The algorithmic method is designed to provide some mathematical equations to perform software estimation. These mathematical equations are based on research and historical data and use inputs such as Source Lines of Code (SLOC), number of functions to perform, and other cost drivers[1].

DIRECT APPROACH

Source lines of code (SLOC) is a software metric used to measure the size of a software program by counting the number of lines in the text of the program's source code. SLOC is typically used to predict the amount of effort that will be required to develop a program, as well as to estimate programming productivity or

maintainability once the software is produced. There are two major types of SLOC measures: physical SLOC (LOC) and logical SLOC (LLOC). Specific definitions of these two measures vary, but the most common definition of physical SLOC is a count of lines in the text of the program's source code including comment lines. Blank lines are also included unless the lines of code in a section consists of more than 25% blank lines. Logical SLOC attempts to measure the number of executable "statements", but their specific definitions are tied to specific computer languages[3].

The COCOMO cost estimation model is used by thousands of software project managers, and is based on a study of hundreds of software projects. Unlike other cost estimation models, COCOMO is an open model. COCOMO estimates are more objective and repeatable than estimates made by methods relying on proprietary models. The most fundamental calculation in the COCOMO model is the use of the Effort Equation to estimate the number of Person-Months required to develop a project. COCOMO has cost drivers that assess the project, development environment, and team to set each cost driver. The cost drivers are multiplicative factors that determine the effort required to complete your software project. number of executable "statements", but their specific definitions are tied to specific computer languages[2].

Effort is calculated by

Effort= a* $(size)^b$

Where 'a' and 'b' are empirically determined constants. Size is length of the code in KLOC.

Type of project	A	B
Organic	3.2	1.05
Semi detached	3.0	1.12
Embedded	2.8	1.20

The Effort Adjustment Factor in the effort equation is simply the product of the effort multipliers corresponding to each of the cost drivers.

For example, if your project is rated Very High for Complexity (effort multiplier of 1.34), and Low for Language & Tools Experience (effort multiplier of 1.09), and all of the other cost drivers are rated to be Nominal (effort multiplier of 1.00), the EAF is the product of 1.34 and 1.09..

The COCOMO schedule equation predicts the number of months required to complete your software project. The duration of a project is based on the effort predicted by the effort equation:

Duration=3.67*(Effort)^{SE}

Where

Effort is the effort from the COCOMO effort equation.

SE is the schedule equation exponent derived from the cost Drivers.

The Man per month is calculated by

Average staffing = (Person-Months) / (Duration)

INDIRECT APPROACH

Function Point Analysis (FPA):

It begins with the decomposition of a project or application into its data and transactional functions. The data functions represent the functionality provided to the user by attending to their internal and external requirements in relation to the data, whereas the transactional functions describe the functionality provided to the user in relation to the processing this data by the application[4].

Each function is classified according to its relative functional complexity as low, average or high. The data functions relative functional complexity is based on the number of data element types (DETs) and the number of record element types (RETs). The transactional functions are classified according to the number of file types referenced (FTRs) and the number of DETs. The number of FTRs is the sum of the number of ILFs and the number of EIFs updated or queried during an elementary process.

The data functions are:

1. Internal Logical File (ILF)
2. External Interface File (EIF)

The transactional functions are:

1. External Input (EI)
2. External Output (EO)

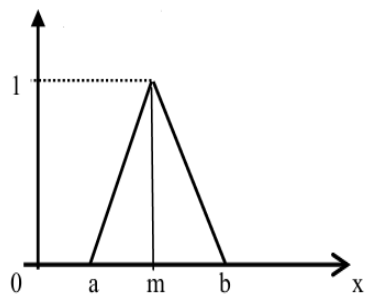
3. External Inquiry (EI)
The actual calculation process consists of three steps:
1.Determination of unadjusted function points(UFP).
2.Calculation of value of adjustment factor(VAF).
3.Calculation of final adjusted functional points.
Evaluation of Unadjusted FP:
The unadjusted Functional points are evaluated in the following manner
UFP= ∑∑Fij*Zij , for j= 1 to 3 and i = 1 to 5, where Zij denotes count for component i at level (low, average or high) j, and Fij is corresponding Function Points.
Evaluation of Value Adjusted FP:
Value Adjustment Factor (VAF) is derived from the sum of the degree of influence (DI) of the 14 general system characteristics (GSCc). General System characteristics are:
1. Data communications
2. Distributed data processing
3. Performance
4. Heavily utilized configuration
5. Transaction rate
6. On-line data entry
7. End-user efficiency
8. On-line update
9. Complex processing
10. Reusability
11. Installations ease
12. Operational ease
13. Multiple sites/organizations
14. Facilitate change

Function points can be converted to Effort in Person Hours. Numbers of studies have attempted to relate LOC and FP metrics. The average number of source code statements per function point has been derived from historical data for numerous programming languages. Languages have been classified into different levels according to the relationship between LOC and FP. Programming language levels and average numbers of source code statements per function point.

Fuzzy Logic:

Fuzzy logic is used to find fuzzy functional points and then the result is defuzzified to get the functional points and hence the size estimation in person hours. Triangular fuzzy numbers are used to represent the linguistic terms in Function Point Analysis (FPA) complexity matrixes. A fuzzy set is characterized by a membership function, which associates with each point in the fuzzy set a real number in the interval [0,1], called degree or grade of membership. The membership function may be triangular, trapezoidal, parabolic etc. Fuzzy numbers are special convex and normal fuzzy sets, usually with single modal value, representing uncertain quantitative information. A triangular fuzzy number (TFN) is described by a triplet (α ,m, β), where m is the modal value, α and β are the right and left boundary respectively.

1.**Triangular Fuzzy numbers**, TFN (α, m, β), $\alpha \leq m$, $\beta \geq m$.



The membership function (μ(x)) for which is defined as:

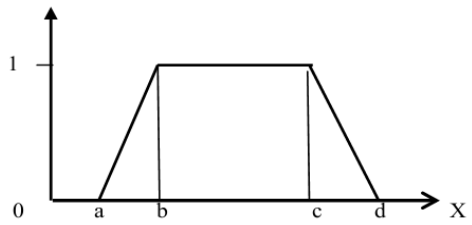
$$\mu(x)=\begin{cases} 0, & x \le \alpha \\ x-\alpha / m-\alpha, & \alpha \le x \le m \\ \beta - x / \beta - m, & m \le x \le \beta \\ 0, & x \ge \beta \end{cases}$$

II. TRAPEZOIDAL FUZZY LOGIC

Defined by its lower limit *a*, its upper limit *d*, and the lower and upper limits of its nucleus or Kernel b and c respectively:

$$T(x) = \begin{cases} 0, & (x \le a) \text{ or } (x \ge d) \\ (x-a)/(b-a), & x \in (a, b) \\ 1, & x \in (b, c) \\ (d-x)/(d-c), & x \in (c, d) \end{cases}$$

Fig.: Trapezoidal Fuzzy



The five major components mentioned above, they have to be rated as either Low, Average, or High. Ranking is commonly based on File Types Referenced, Data Element Types and Record Element Types. File Types Referenced (FTRs) represents the total number of internal logical files (ILFs) maintained, read, or referenced, and the external interface files read or referenced by the EI/EO transaction. Data Element Type (DET) can be defined as unique user recognizable non-recursive fields including foreign key attributes that are maintained on ILF/EIF. Record element type (RET) is a subgroup of data elements within an ILF/EIF. For each of the components belonging to Transactional functions, the ranking is based on the number of files updated or referenced (FTRs) and number of data element types (DETs).For the data components viz., Internal Logical Files (ILF) and External Interface Files (EIF), ranking is based on the number of Data Element Types (DETs) and number of Record Element Types (RETs).

Based on the ratings the domain character values are fuzzified using the Triangular membership function. The value thus obtained is called membership function output, whose domain is specified, usually the set of real numbers, and whose range is the span of positive numbers in the closed interval [0, 1]. Each numerical value of the domain is assigned a specific value and 0 represents the smallest possible value of the membership function, while the largest possible value is 1.

Defuzzification:

Defuzzification means the fuzzy to crisp conversions. The fuzzy results generated cannot be used as such to the hence it is necessary to convert the fuzzy quantities into crisp quantities for further processing. This can be achieved by using defuzzification process. The defuzzification has the capability to reduce a fuzzy to a crisp single-valued quantity or as a set, or converting to the form in which fuzzy quantity is present. Defuzzification can also be called as "rounding off" method. Defuzzification reduces the collection of membership function values in to a single sealer quantity. **Defuzzification** is the process of producing a quantifiable result in fuzzy logic, given fuzzy sets and corresponding membership degrees. It will have a number of rules that transform a number of variables into a fuzzy result, that is, the result is described in terms of membership in fuzzy sets. The defuzzification is applied to the value that had been obtained from the fuzzification process. The fuzzified output has to be defuzzified into the real number so that it will give the effort that has been needed for the cost estimation.

$$D(y)=\begin{cases}\mu(x)*w1 & 0<c(x)\leq 1\\ \mu(x)*w1+(1-\mu(x))*w2 & 1<c(x)\leq 2\\ \mu(x)*w2+(1-\mu(x))*w1 & 2<c(x)\leq 3.5\\ \mu(x)*w2+(1-\mu(x))*w3 & 3.5<c(x)\leq 5\\ \mu(x)*w3+(1-\mu(x))*w2 & 5<c(x)\leq 6.5\\ \mu(x)*w3+(1-\mu(x))*w5 & 6.5<c(x)\leq 8\end{cases}$$

VARIOUS CRITERIONS FOR ASSESSMENT OF SOFTWARE COST ESTIMATION MODELS
There are 4 important criterions for assessment of software cost estimation models:

1. VAF (Variance Accounted For) (%):

VAF (%) = $\left(1-\frac{var(E-\hat{E})}{var\ E}\right)*$:

2. Mean absolute Relative Error (%):

Mean absolute error (%) = $\frac{\sum f}{\sum}$ * 100

3. Variance Absolute Relative Error (%):

VAR (%) = $\frac{\sum f(R_E-mean}{\sum f}$ * 100

4. Pred (n): Prediction at level n((Pred (n)):

Var x = $\frac{\sum f(x}{\sum}$

III. EXPERIMENTAL RESULTS

Performance of the effort can be predicted based on the MARE and Prediction n method.The estimated effort of LOC is compared with the actual effort of LOC in the first graph.The estimated effort of FP is compared with the actual effort of FP in the second graph.The MARE of LOC and FP is compared in the third graph.It has been clearly identified that Function point based estimation is better than the LOC estimation.

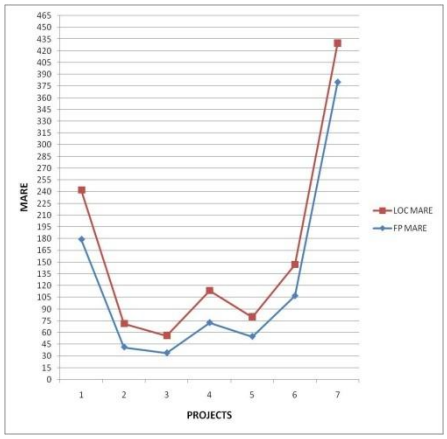
The Table 1 indicates the lines of code with the actual effort and the estimated effort using the cocomo model. Both MARE analysis and Prediction n method has been applied to the direct approach and the indirect approach.The actual effort is the original effort and the estimated effort is the one which has been done in the estimation process using the cocomo method.

.LOC	Actual effort	Estimated effort
48	1107.3	1465.83
50	84	145
39	72	112
164	246	510
200	130	625
40.5	82.5	160.7

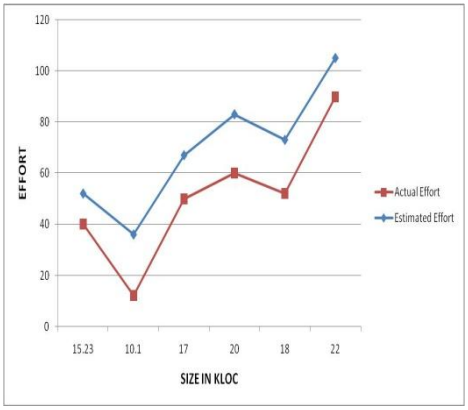
The next table shows the function point with actual effort and the estimate effort.

LOC in FP	Actual effort	Estimated effort
15.23	40	52
10.1	12	36
17	50	67
20	60	83
18	52	73
22	90	105

The graph shows the variation between the actual and estimated effort using LOC.



The following graph shows the variation between the actual and estimated effort using LOC in FP.



The MARE analysis is given as follows

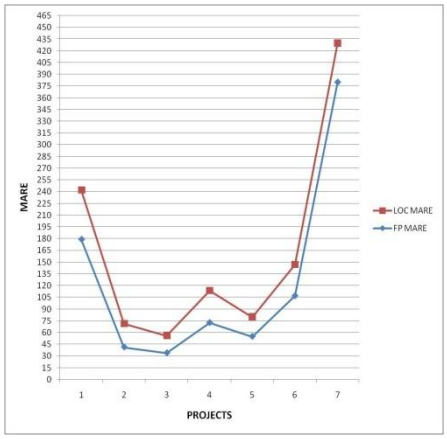


Table 2: Results and comparison of effort estimation in person months

Project ID	Actual effort	Effort in Person Months (PM)		
		COCOMO	Using triangular MF	Using trapezoidal MF
1	61	45.63	52.53	55.39
2	237	214.10	249.16	234.52
3	599	539.60	575.44	580.80
4	603	553.43	578.12	589.36
5	702	1335.10	1253.90	1146.20
6	523	278.86	314.04	365.57
7	1075	661.30	739.63	806.34
8	2455	1945.40	2016.90	2096.30
9	958	408.33	476.60	563.65
10	1063	1275.90	1209.60	1164.40

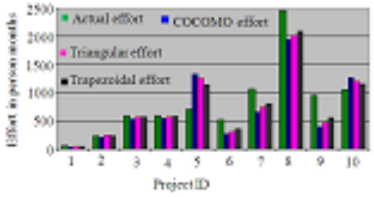


Fig. 2: Chart representing the comparisons of effort estimation

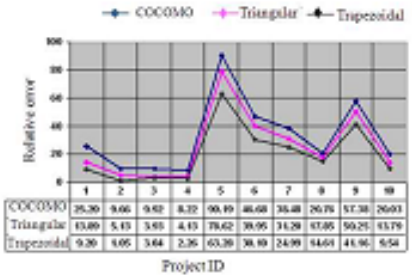


Fig. 3: Assessments of magnitude of relative errors

These graph shows the bar chart representing comparative analysis of actual effort with that of the effort estimated using COCOMO, triangular and trapezoidal membership functions with relative error. Effort in person months is scaled along with y-axis. Actual effort, COCOMO effort and effort obtained using TAMF for size and effort obtained using TPMF for size, were represented for each sample projects, which were taken along with x-axis.

IV. CONCLUSION

In this paper, we have proposed a new approach to estimate the software project cost. This approach is based on fuzzy logic.In fuzzy logic approach datas are represented by fuzzy sets.In this investigation it is projected to characterize the size of the project using Triangular Membership Function which gives superior transition from one interval to another.A new fuzzy effort estimationmodel is proposed by using trapezoidal function to dealwith the size and to generate fuzzy Membership Function and rules.After analyzing the results attained by means of applying COCOMO, triangular and trapezoidal Membership Function models, it is observed that the effort estimation of the proposed model isgiving more precise results than the other models. Theeffort estimated by means of fuzzifying size usingTrapezoidal Membership Function is yielding better estimate which is very nearer to the actual effort.

FUTURE WORK

From the experimental results, it is concluded that, by fuzzifying the size of the project using TPMF, it can be proved that the resulting estimate impacts the effort.

The effort generated using the proposed model gives better result than that of using ordinal COCOMO. This illustrates that by fuzzifying size using TPMF, the accuracy of effort estimation can be improved and the estimated effort can be very close to the actual effort.

Moreover, by capturing the uncertainty of the initial data (estimates), one can monitor the behavior (quality) of the cost estimates over the course of the software project. This facet adds up a new conceptual dimension to the models of software cost estimation by raising awareness of the decision making with regard to

the quality of the initial data needed by the model. This study can be extended by integrating with neural networks. By using this extended approach with the standard COCOMO models, we can take advantage of the features of neural network, such as learning ability and good interpretability. Therefore, a promising line of future work is to extend to the neuro-fuzzy approach.

REFERENCES

[1]. M. Boraso, C. Montangero, and H. Sedehi, "Software cost estimation: An experimental study of model performances", tech. rep., 1996.

[2]. O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, "COCOMO based effort estimation for iterative and incremental software development", Software Quality Journal, vol. 11, pp. 265-281, 2003.

[3]. T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, "Validation Meth-ods for calibrating software e_ort models", ICSE '05:Proceedings of the 27th international conference on Software engineering, (New York, NY, USA),,pp.587-595, ACM Press, 2005.

[4]. Boehm, B., Abts, C., Brown, A. W., Chulani, S., Clark, B.K., Horowitz, E., Madachy, R., Reifer, D. J., Steece,B.Software cost estimation with COCOMO II. Prentice-Hall,Upper Saddle River, NJ, February 2000.

[5]. IFPUG. Function Point Counting Practices Manual: Release 4.0. International Function Point Users Group, Princeton Junction, NJ, 1994.

[6]. Alaa f. sheta," Estimation of the COCOMO Model Parameters Using Genetic Algorithm for NASA Software Projects", Journal of Computer Science ,2(2):118-123,2006

[7]. Ali Idri, alain Abran and Laila Kijri, "COCOMO cost modeling using Fuzzy Logic", International conference on Fuzzy Theory and technology At-lantic, 7New Jersy, March 2000.

[8]. Baiely,j.w Basili,"A Meta model for Software Development Resource Expenditure", Proc. Intl. Conference Software Egg.,pp : 107-115,1981

[9]. Idri, A. and Abran, A.:"COCOMO Cost Model Using fuzzylogic".

[10]. IFPUG. *Function Point Counting Practices Manual: Release 4.0*. International Function Point Users Group, PrincetonJunction, NJ, 1994.