Software Effort Estimation using Fuzzy Logic Membership Functions

Archana A. Savalgi, Swati Jain, Andanappa S.S, Swathi J. Narayan

Department of Computer Science & Engineering,
VIT Vellore, India.

I. INTRODUCTION

Software engineering practitioners have become more and more concerned about accurately predicting the cost and effort required for developing the software product. Software effort estimation is the process of predicting the realistic amount of effort/cost required for the completion of software projects. Software effort estimation holds good at early stages of software development to meet competitive demands of today’s industry. It refers to the predictions of the likely amount of effort, time, and staffing level required to build a software system. Accuracy in effort estimation plays a vital role in the success of software project management. Software effort estimation techniques are broadly classified into two categories: Algorithmic models and Non-algorithmic models.

Algorithm Models: The algorithmic techniques are based on some mathematical models and formulas which are used to calculate effort estimate. Algorithmic methods are using historical data and inputs like LOC (Lines of Code), number of functions to be performed(FP), various cost drivers(COCOMO) on which estimation have done. The best known algorithmic model for calculating effort is COCOMO (Constructive Cost Model), published by Barry Boehm in 1981. Boehm projected three levels of the model called Basic COCOMO, Intermediate COCOMO and Detailed COCOMO or COCOMO II.

Non-algorithmic models: In 1990’s the non-algorithmic models was discovered and have been used in effort and cost estimation of projects. The methodologies to achieve this are Expert Judgment, Delphi and soft computing. Soft computing is performed by using Fuzzy logic and Neural Networks.

Fuzzy systems are knowledge based or rule based systems. A fuzzy IF-THEN rule is an IF-THEN statement in which some words are characterized by continuous membership functions. Thus fuzzy logic can be used to handle the imprecision and uncertainty present in the early stages of the project to predict the effort more accurately by incorporating total transparency in the prediction system.

In the present paper we mainly focus on the Post Architecture Model of COCOMO II and fuzzy logic to calculate effort of the software developed. Fuzzy logic comes under soft computing, which is based on non-algorithmic technique. This COCOMO II uses a set of 15 effort multipliers reflecting personnel capability, product and project characteristics, 5 scale factors and size of the project. Fuzzy logic introduces the concept of fuzzy set theory. Fuzzy set is characterized by membership function. A membership function (MF) is a curve that defines how each point in the input space is mapped to a membership value (or degree of membership) between 0 and 1. For our problem, we have used 8 types of membership functions as triangular, trapezoidal, sigmoidal, psigm, gbell, 2-sided Gaussian, Gaussian, dsigm. It is implemented using MATLAB. There are two types of fuzzy inference system one is Sugeno method and other is Mamdani system. In our paper we considered Mamdani System because it is intuitive, has widespread acceptance and well suited for human input.

Section 2 provides the literature survey done for this paper. Section 3 gives information about COCOMO II, Fuzzy Logic and Membership Functions. Section 4 introduces the methodology used. Section 5 gives details about the experimental results.

II. LITERATURE SURVEY

In this paper, we have made a review on our topic improving the accuracy of software effort estimation by reading different kinds of papers and analyzing different techniques which are being used in these papers published by authors which are discussed as follows:

M.Kazemifard et al. [1] proposed a novel emotional Constructive Cost Model II for software cost estimation where the characteristics of team members such as communication skills, personality, mood and capabilities of team members are considered. Prasad Reddy P.V.G.D et al.
[4] compared software development effort using Fuzzy Triangular Membership Function and GBell Membership Function with COCOMO. The results were analyzed using different criterions like VAF, MARE, VARE, MMRE, Prediction and Mean BRE. Ashita Malik et al[5].,explored a soft computing technique to overcome the uncertainty and imprecision in software cost estimation. They used fuzzy logic in improving the effort estimation accuracy using COCOMO II by characterizing inputs parameters using Gaussian, trapezoidal and triangular membership functions and comparing their results. Ravishankar. S et al., utilize an adaptive fuzzy logic model to improve the accuracy of software effort estimation and results are compared with COCOMO II and Alaa Sheta Model.

III. BACKGROUND

A. The COCOMO II

For the COCOMO II models, three different sizing options are available: object points, function points, and lines of source code.

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:-

1. Plan and requirement.
2. System design.
3. Detailed design.
5. Integration and test.

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

$$\text{EFFORT}_{PM} = A \times \text{size}^{B+\sum_{i=1}^{17} EM_i}$$

where $A=2.94$ and $B=0.91$ , $EM=Effort$ Multipliers and $SF=Scale$ Factors.

The inputs are the Size of software development, a constant($A$), an exponent $E$ and a number of values called effort multipliers ($EM$). The COCOMO II includes several software attributes such as: 15 Effort Multipliers ($EM$), 5 Scale Factors ($SF$), Software Size ($SS$), and Effort estimation that are used in the Post Architecture Model of the COCOMO II. 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. 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 cost drivers 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 table 1 shows the range of Scale Factors ($SF$) used. We have referred these values from promise data repository[2].

Table 1: The range of COCOMO II SFs

<table>
<thead>
<tr>
<th>No.</th>
<th>Scale Factor</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Precededness (PADC)</td>
<td>0.00-2.00</td>
</tr>
<tr>
<td>2</td>
<td>Development Flexibility (PLEX)</td>
<td>0.00-5.07</td>
</tr>
<tr>
<td>3</td>
<td>Architecture Risk Resolution (ARS)</td>
<td>0.00-7.07</td>
</tr>
<tr>
<td>4</td>
<td>Team Cohesion (TEAM)</td>
<td>0.00-5.48</td>
</tr>
<tr>
<td>5</td>
<td>Process Maturity (PMAT)</td>
<td>0.00-7.80</td>
</tr>
</tbody>
</table>

The table 2 shows the range of Effort Multipliers ($EM$) used. These values are taken from dataset[2].

Table 2: The range of COCOMO II EMs

<table>
<thead>
<tr>
<th>No.</th>
<th>Effort Multiplier</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Analyze Capability (CE)</td>
<td>0.71-1.46</td>
</tr>
<tr>
<td>2</td>
<td>Programmers Capability (PC)</td>
<td>0.70-1.42</td>
</tr>
<tr>
<td>3</td>
<td>Application Experience (AE)</td>
<td>0.82-1.29</td>
</tr>
<tr>
<td>4</td>
<td>Network programming Practices (NP)</td>
<td>0.82-1.24</td>
</tr>
<tr>
<td>5</td>
<td>Use of Software Tools (SOT)</td>
<td>0.83-1.24</td>
</tr>
<tr>
<td>6</td>
<td>Virtual machine Experience (VME)</td>
<td>0.50-1.21</td>
</tr>
<tr>
<td>7</td>
<td>Language Experience (LE)</td>
<td>0.85-1.14</td>
</tr>
<tr>
<td>8</td>
<td>Schedule Constraint (SC)</td>
<td>1.00-1.25</td>
</tr>
<tr>
<td>9</td>
<td>Main Memory Constraint (MMC)</td>
<td>1.00-1.56</td>
</tr>
<tr>
<td>10</td>
<td>Database Size (DBS)</td>
<td>0.54-1.16</td>
</tr>
<tr>
<td>11</td>
<td>Time Constraint (TC)</td>
<td>1.00-1.86</td>
</tr>
<tr>
<td>12</td>
<td>Turnaround Time (TAT)</td>
<td>0.87-1.15</td>
</tr>
<tr>
<td>13</td>
<td>Machine Reliability (MR)</td>
<td>0.87-1.10</td>
</tr>
<tr>
<td>14</td>
<td>Process Complexity (PC)</td>
<td>0.75-1.40</td>
</tr>
<tr>
<td>15</td>
<td>Required software reliability (RSR)</td>
<td>0.70-1.65</td>
</tr>
<tr>
<td>16</td>
<td>Reliability (R)</td>
<td>0.75-1.25</td>
</tr>
<tr>
<td>17</td>
<td>Platform Volatility (PV)</td>
<td>0.80-1.25</td>
</tr>
</tbody>
</table>

B. Fuzzy Logic

In 1965, Lofti Zadeh formally developed multi-value set theory, and introduced the term fuzzy logic. Fuzzy set theory provides a natural method for dealing with linguistic terms (i.e. worst, good and best) of the linguistic variables (i.e. food). A general fuzzy system includes the following elements:

1. Fuzzification process, in which the membership functions (MF) are applied to the numerical value of input variables, to determine how much the input variables fit the linguistic terms.
2. Knowledge base, which is a set of expert control rules (knowledge) needed to achieve a goal. The knowledge base is usually expressed as a number of ‘IF–THEN’ rules based on the domain expert’s knowledge.

3. A fuzzy inference mechanism, which performs various fuzzy logic operations by using knowledge base to convert fuzzy inputs to fuzzy outputs.

4. Defuzzification process, in which if the conclusions of the fuzzy rule set are fuzzy subsets themselves, then it is necessary to translate these subsets into a crisp number before the results can be used in practice.

In this paper, we have used eight kinds of MF: triangular, trapezoidal, sigmoidal, psigm, gbell, 2-sided Gaussian, Gaussian, dsigm. The details about cocomo ii and fuzzy logic is taken from the paper[4].

IV. METHODOLOGY

Our model is established based on the COCOMO II and Fuzzy Logic. The COCOMO II includes a set of input software attributes: 17 Effort Multipliers (EMs), 5 Scale Factors (SFs), one Size in SLOC (SZ) and one output, Effort. The figure 4 shows the fuzzy logic model for software effort estimation.

A. Inputs

There are three set of inputs

- Size in SLOC
- 15 Effort Multipliers
- 5 Scale Factors

All these inputs are provided as crisp data. These inputs are directed to fuzzification module.

B. Fuzzification

This module is sub divided into three sub modules. SZ-Fuzzification module converts the Size input to the fuzzy variable, SF-Fuzzification module converts Scale Factors input to fuzzy variables whereas EM- Fuzzification module converts the Effort Multiplier input to Fuzzy variable. The Terms Very Low, Low, Nominal, High, Very High and extra High have been defined for each variable.
C. Inference Engine

Inference engine operates on a set of fuzzy rules. The Fuzzy Model rules contain the linguistic variables related to the project. We have implemented this model using FIS tool in MATLAB. A sample of the rules is presented below:

- if (PREC is VL) then (EFFORT is XH)
- if (PREC is LO) then (EFFORT is VH)
- if (FLEX is VL) then (EFFORT is XH)

D. Defuzzification

This module calculates and converts the fuzzy output into crisp data form.

This process is referred from the base paper [3].

V. EXPERIMENTAL STUDY

This paper is implemented using MATLAB which contains Mamdani Fuzzy Inference System. FIS contains FIS editor where we specify our input variables, output variables and the type of fuzzy system we use. The figure 6 shows the sample of FIS editor which contains our 15 EM’s, 5 SFs and size for triangular function.

The figure 7 shows the fuzzification process applied for Resolution (RESL) Scale Factor, triangular membership function is used.

Now the rules are to be defined based on EM’s and SF’s and size. The figure 8 shows the sample fuzzy rules we have used. We have considered 252 rules for calculating effort of the software.

After writing the rules, there is need of Defuzzification process. For every project there will be certain EM’s, SF’s, and size of the project which are predefined. By considering these factors we will calculate the effort. We have used PROMISE Software Engineering Repository Data set for the inputs. The figure 9 shows the output after Defuzzification process which contains the calculated effort.

We compare the effort calculated using fuzzy and the one calculated by COCOMO II. The evaluation methods used are Magnitude Relative Error (MRE) and Mean Magnitude of Relative Error (MMRE). The Magnitude of Relative Error (MRE) is defined as:
where E=Actual effort and E’=Predicted effort.

The MRE value is calculated for each observation 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:

$$MMRE = \frac{\sum_{i=1}^{n} MRE_i}{n}$$

where n is the total number of observations and MRE is magnitude relative error.

The table 2 shows the comparison results of various membership function used in our paper. The table 3 shows the values Magnitude of Relative Error(MRE) and Mean Magnitude Relative Error(MMRE) for various membership functions. The result shows that psig membership function calculates effort approximately equal to actual effort.

Table 3. Comparison of effort using different membership functions

Table 4. MRE and MMRE of various membership functions

VI. CONCLUSION

One of the most important factor in software project management is calculating effort required for developing the project. Fuzzy Logic can overcome vagueness and uncertainty of software attributes. Referring to table 4 effort estimation using psigmf yields better results as compared to other membership functions. It is not possible to achieve 100% results, by suitably adjusting the scale factors and effort multipliers effort estimation can be reduced.

ACKNOWLEDGMENT

We sincerely thank our guide Assistant professor Swathi J. N, for the extensive guidance throughout the implementation and for helping in proposing effective methodology.

REFERENCES


