Soft Computing Approach for Software Cost Estimation

Iman Attarzadeh and Siew Hock Ow

Department of Software Engineering
Faculty of Computer Science & Information Technology
University of Malaya, 50603 Kuala Lumpur, MALAYSIA
Email: attarzadeh@perdana.um.edu.my, show@um.edu.my

ABSTRACT

Software metric and estimation is base on measuring of software attributes which are typically related to the product, the process and the resources of software development. One of the greatest challenges for software developers is predicting the development effort for a software system based on some metrics for the last decades. Project managers are required to the ability to give a good estimation on software development effort. Most of the traditional techniques such as function points, regression models, COCOMO, etc, require a long-term estimation process. One of the new approaches that called soft computing techniques may offer an alternative for this challenge. This paper described an enhanced soft computing model for the estimation of software cost and time estimation. Result shows that the value of MMRE (Mean of Magnitude of Relative Error) applying soft computing was substantially lower than MMRE applying by algorithmic models.

Keywords: Software Engineering, Algorithmic Model, Software Cost Estimation, COCOMO Model, Soft Computing Techniques

1- INTRODUCTION

1-1 Software Estimation

Software Estimation is responsive to the many problems the software industry has experienced in creating significant cost and time estimates. Software estimation is base on measuring of software attributes which are typically related to the product, the process and the resources of software development [1]. This kind of measuring can be used as parameters in project management models [2] which provide assessments to software project managers in managing software projects to avoid problems such as cost overrun and behind the schedule. One of the most widely researched areas of software measurement is software effort estimation. Software effort estimation models divided into two main categories: algorithmic and non-algorithmic. The most popular algorithmic estimation models include Boehm’s COCOMO [3], Putnam’s SLIM [4] and Albrecht’s Function Point [5].
These models require as inputs, accurate estimate of certain attributes such as line of code (LOC), complexity and so on which are difficult to obtain during the early stage of a software development project. The models also have difficulty in modeling the inherent complex relationships between the contributing factors, are unable to handle categorical data as well as lack of reasoning capabilities [6]. The limitations of algorithmic models led to the exploration of the non-algorithmic techniques which are soft computing based.

These include artificial neural network, evolutionary computation, fuzzy logic models, case-based reasoning, combinational models and so on. Artificial neural network are used in effort estimation due to its ability to learn from previous data [7] [8]. It is also able to model complex relationships between the dependent (effort) and independent variables (cost drivers) [8]. In addition, it has the ability to generalise from the training data set thus enabling it to produce acceptable result for previously unseen data. Most of the work in the application of neural network to effort estimation made use of feed-forward multi-layer Perceptron, Backpropagation algorithm and sigmoid function [9][10].

Selecting good models for software estimation is very critical for software engineering. In the recent years many software estimation models have been developed [15] [16].

Gray and MacDonell compared Function Point Analysis [2][14], Regression techniques, feed-forward neural network and fuzzy logic in software effort estimation. Their results showed that fuzzy logic model achieved good performance, being outperformed in terms of accuracy only by neural network model with considerably more input variables. Also they developed FULSOME (Fuzzy Logic for Software Metrics) which is a set of tools that helps in creating fuzzy model.

Fei and Lui [10] introduced the f-COCOMO model which applied fuzzy logic to the COCOMO model for software effort estimation. Since there was no comparison of the results between the f-COCOMO and other effort estimation models in their study, the estimation capability of the former is unknown. Roger [11] also proposed a fuzzy COCOMO model which adopted the fuzzy logic method to model the uncertainty of software effort drivers, but the effectiveness of the proposed model is not mentioned. Idri [7] [8] further defined a fuzzy set for the linguistic values of each effort driver with a trapezoid-shaped membership function for the fuzzy COCOMO model. The effort multipliers in the original COCOMO model were obtained from the fuzzy sets. This fuzzy COCOMO model was less sensitive to the software effort drivers[17] as compared to the intermediate COCOMO81. In 2004, Xue and Khoshgoftaar [13] presented a fuzzy identification effort estimation modeling technique to deal with linguistic effort drivers, and automatically generated the fuzzy membership functions and rules by using the COCOMO81 database. The
proposed fuzzy identification model provided significantly better effort estimates than the original three COCOMO models, i.e., basic, intermediate, and detailed.

1-2 The COonstrictive COst MOdel (COCOMO)

The COCOMO model is a regression based software cost estimation model. It was developed by Barry Boehm [3] in 1981 and thought to be the most cited, best known and the most plausible [18][19] of all traditional cost prediction models. COCOMO model can be used to calculate the amount of effort and the time schedule for software projects. COCOMO 81 was a stable model on that time. One of the problems with using COCOMO 81 today is that it does not match the development environment of the late 1990’s. Therefore, in 1997 COCOMO II was published and was supposed to solve most of those problems. COCOMO II has three models also, but they are different from those of COCOMO 81. They are [3]:

i. Application Composition Model – Suitable for projects built with modern GUI-builder tools. Based on new Object Points.

ii. 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.

iii. 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:

\[
\text{Effort} = A \times [\text{Size}]^B \times \prod_{i=1}^{17} \text{Effort Multiplier}_i \quad (1)
\]

where \( B = 1.01 + 0.01 \times \sum_{j=1}^{5} \text{Scale Factor}_j \)

where:

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)
1-3 Fuzzy Logic Approach

One of the famous issues in soft computing is fuzzy logic. Since fuzzy logic foundation by Lotfi Zadeh in 1965, it has been the subject of important investigations [12]. It is a mathematical tool for dealing with uncertainty and also it provides a technique to deal with imprecision and information granularity [11]. The fuzzy logic model uses the fuzzy logic concepts introduced by Lotfi Zadeh [12]. Fuzzy reasoning consists of three main components [20]: 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 function (MF). 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 membership value from all the fired rules. Defuzzification process refers to the translation of fuzzy output into objective [21][22].

2- RESEARCH METHOD

2-1 The Proposed Model

The proposed model base on COCOMO II has two input’s group from COCOMO II cost drivers and scale factors and one output, effort estimation. This model covers those three fuzzy steps, fuzzification process, inference from fuzzy rules and defuzzification process. It shows in Fig. 1 in below.

![Figure 1 The proposed model: Inputs (COCOMO II cost drivers, scale factors, Size) and Output: (effort estimation)](image-url)
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 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.

All the input variables in COCOMO II model changed to the fuzzy variables based on the fuzzification process. The terms Very Low (VL), Low (L), Nominal (N), High (H), and Very High (VH) were defined for the 22 variables, cost drivers and scale factors, in COCOMO II. For example, in the case of RELY cost driver, we define a fuzzy set for each linguistic value with a Triangular Membership Function (TRIMF), in Fig. 2. We have defined the fuzzy sets corresponding to the various associated linguistic values for each cost driver.

![Figure 2 Representation of RELY cost driver using Triangular Membership Function (Input)](image)

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

IF TOOL is Low THEN effort is Low
IF PCAP is Very_Low THEN effort is Very_High
IF RESUE is Nominal THEN effort is Nominal
IF DATA is Very_High THEN effort is Very_High

The MATLAB fuzzy inference system (FIS) was used in the fuzzy calculations, in addition to the Max-Min composition operator, the Mandani implication operator, and the Maximum operator for aggregation. The defuzzification of the output “effort” used the Mean of Maximum (MOM) technique in this work because the resulting values were more appropriate when compared to the other evaluated techniques (Center of Area (COA) and First of Maximum (FOM)).

2-2 Model Evaluation

The main parameter for the evaluation of cost estimation models is the Magnitude of Relative Error (MRE) [13] which is defined as follows:

\[ MRE_i = \frac{|Actual\ Effort_i - Predicted\ Effort_i|}{Actual\ Effort_i} \]  

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) as follows:

\[ MMRE = \frac{1}{N} \sum_{i} MRE_i \]  

A complementary criterion is the prediction at level 1, \( Pred(l) = k/N \), where \( k \) is the number of observations where MRE is less than or equal to \( l \), and \( N \) is the total number of observations. Thus, \( Pred(20) \) gives the percentage of projects which were predicted with a MRE less or equal than 0.20. In general, the accuracy of an estimation technique is proportional to the \( Pred(l) \) and inversely proportional to the MMRE [16, 18].

3- RESULTS

The proposed fuzzy model was validated by a very famous dataset, NASA’93. This dataset consists of 93 software project in NASA. This dataset is applied to the proposed fuzzy model and COCOMO II model. The comparison between the results from two models shows more accuracy in case of effort estimation by the proposed fuzzy model. The comparisons between results are shown in Tables 3 and 4.
In this research, NASA'93 dataset separately applied to the COCOMO II model and proposed fuzzy 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 93 applied projects shows the MMRE for COOCMO II model is 0.413812453 and for proposed fuzzy model the value equals to 0.3665545456. 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 46% in Pred(25%) and COCOMO II value is 39% in same Pred. As it mentioned above, Pred shows the number of projects that they have MMRE less 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.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Model</th>
<th>Evaluation</th>
<th>MMRE</th>
<th>Pred (25%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NASA'93 Dataset</td>
<td>COCOMO II</td>
<td>Proposed Model</td>
<td>0.413812453</td>
<td>0.3665545456</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>39%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>46%</td>
</tr>
</tbody>
</table>

For comparing proposed model with COCOMO model, the improvement is 11.42% based on the MMRE 0.36 and 0.41. The experimental results show that the proposed software effort estimation model shows better estimation accuracy than the COCOMO model. 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.
4- CONCLUSION

The paper suggests soft computing approach for estimating of software project development cost and time. The major difference between our work and previous works is that Triangular membership function in fuzzy technique is used for software development cost and time estimation and then it's validated with 93 software projects from NASA. Here, the advantages of fuzzy logic and good generalisation are obtained. The main benefit of this model is its good interpretability by using the fuzzy rules and another great advantage of this research is that it can put together expert knowledge (fuzzy rules) project data into one general framework that may have a wide range of applicability in software estimation.

REFERENCES


