Study Of Using Evolutionary Computational Tools In The Software Effort Prediction By Analogy Method

Thamarai. I. (Author)
Research Scholar, Sathyabama University, Chennai , India.

Dr. S. Murugavalli (Co-Author)
Research Supervisor, Sathyabama University, Chennai , India.

Abstract

Software Estimation is a very important and crucial task in the software development process because of the intangible nature of software. It is difficult to predict the effort correctly, due to which many projects have failed. Many number of options are available to predict the software effort such as algorithmic models, non-algorithmic models etc. Estimation of Analogy has been proved to be most effective method. In the Analogy method, the estimation of software is based on the similar projects that have been successfully completed already. If the parameters of the project, matches well with the past projects then it is easy to calculate the effort for current project. The main problems faced are Feature Selection and Similarity Measure between the projects. The success rate of the effort prediction largely depends on finding the most similar past projects. To find the most relevant past projects, the computational intelligence tools are used. The role of evolutionary computation algorithms in this area is very significant. A study has been made to analyze the various available methods in software effort prediction and a new method is proposed in this paper

***Index Terms*— Expert Judgment, COCOMO, Genetic Algorithm, Genetic Programming, Differential Evolution.**

1. Introduction

This paper provides an insight to the methods available in the prediction of software effort. The main objective is to initiate progress in the research in this field. This paper proposes a more efficient way to predict the effort in the software development process. Software effort prediction is one of the major activities in the software development process. Estimation of software is important for project planning, budgeting, staff allocation, etc. Many projects have failed due to wrong estimation [12]. Effort prediction is important to assist in scheduling resources and evaluating risk factors. There are many methods available to estimate the software effort. In this paper some of the most popular approaches are studied as shown in the following figure:

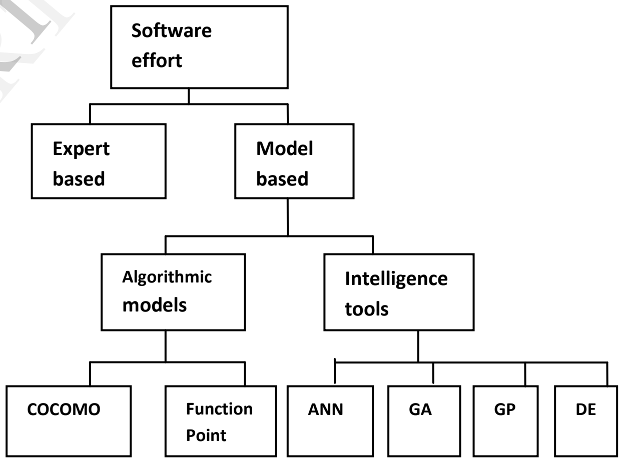

Fig 1: Estimation Models

Expert based methods are based on the judgment based quantification step where as the formal models are based on a mechanical quantification such as a formula. The evaluation of information in Expert based method are judgment based processes. In case of models, the evaluations are based on the statistical analysis. Detailed study has been made on these estimation methods and they are summarized in the following sections.

1.1. Expert Based Estimation

In this method, estimation is based on the experience of the experts in the field. The success depends on the knowledge acquired by the experts in the implementation of previous projects. In [13], M.Jorgenson provides an extensive review of studies related to expert estimation in software development effort. In this paper, he gives twelve guidelines to be followed to ensure best estimation through expert judgment. The guidelines includes avoiding conflicting estimation goals, asking the estimators to justify and criticize their estimates, avoiding irrelevant and unreliable estimation information, etc.

To minimize the errors in the Expert judgment Method, some techniques were developed in it that consists of set of steps to mitigate the potential mistakes. The most important techniques are Delhi Estimation Method and Work Breakdown Structure. In Delphi method, the members of a group are asked to make the estimation without discussing with any of the other member in the group. A variation of this technique is Wideband Delphi Technique which allows group discussions. In the Work Breakdown Structure, the software process is divided into sub tasks in hierarchy levels. The effort required for each subtask is calculated separately and summed up to find the total effort required for the complete project. Experts were used to decide the most useful component structure.

The main problem with the Expert Judgment method is that, the results are always subjective and cannot be proved scientifically. It is also very difficult to document the methods used by the experts. Also M.Jorgenson in his paper [2] says that expert judgment leads to human biases. Such disadvantages are not present in the Model based models

2. Model based Estimation

In this method, software effort estimation is based on the use of one or more formula. This is called as quantification step. Sometimes models are created as a combination of many methods and it has also been proved to be successful. Some of the popular Models are discussed in the next sections briefly.

2.1 Algorithmic Models

In this method, estimation is based on the mathematical formula that has been derived through statistical data analysis. The Algorithmic models calculate effort as a function of a number of variables. It takes the following form:

$$\text{Effort} = f(x1, x2, ..xn)$$

Where (x1, x2,…) are called cost factors

The cost factors can be Product attributes, Hardware attributes, Personnel attributes and Project attributes. Eg : Function Point (FP), COCOMO

2.1.1 Function Point [FP]

Function point metrics was proposed by Albrecht. This metric overcomes many of the disadvantages of the LOC metric. The idea of the function point metric is that the size of a software product is directly dependent on the number of functions it supports. Function point is computed in two steps. The first step computes the unadjusted function point (UFP). The second step computes Technical complexity factor [TCF]

Step1: Calculation of UFP

UFP= (Number of inputs)*4+ (Number of outputs) * 5 + (Number of inquires) * 4 + (number of files) * 10 + (number of interfaces) * 10

Step2: Calculation of TCF

This depends on 14 factors such as transaction rate, reusability, data communications, backup and recovery, performance etc. Based on the calculation of UFP and TCF, function point is calculated

$$FP = UFP * TCF$$

The main advantage of Function Point is that, it is language independent.

2.1.2 COCOMO

The COCOMO model was proposed by Barry Boehm. There are three types of COCOMO models namely Basic Model, Intermediate Model and Complete Model. Each of these types estimates the effort by dividing the project into three categories or modes based on the size, as Organic, Semi-detached and Embedded. In the Basic Model, the calculation of effort is based on the project size (L) and the equations are given below:

Organic Mode= $2.4 * L1^{.05}$

Semi detached Mode= $3.0 * L^{1.12}$

Embedded Mode= $3.6 * L^{1.20}$

In the Intermediate COCOMO, the effort estimation is computed based on the project size and a set of 15 cost drivers or attributes. Each of the 15 attributes gets a rating value. The product of these values gives Effort Adjustment Factor [EAF]. The equations are given below:

Organic Mode= $3.2 * L^{1.05} * EAF$

Semi detached Mode= $3.0 * L^{1.12} * EAF$

Embedded Mode= $2.8 * L^{1.20} * EAF$

In the Complete COCOMO model, the effort is calculated for each step of the development cycle and added to get the total effort. This approach reduces the error. In recent years, the use of COCOMO model has largely reduced as it is difficult to use this model in the multiple platforms. Tim Menzies et al proposed a new model called COSEEKMO whose operators will reduce the large deviations and also improve the mean errors for model base estimation [1]

2.2 Computational Intelligence Tools

In recent years, Machine learning methods such as Artificial Neural Network, Genetic Algorithm are used in the prediction of software effort. These techniques reflect some of the functions of the human mind to solve highly complex problems. So they are called computational Intelligence Tools. Particularly, it is observed that the computational intelligence tools can be most effectively used in the Analogy method of estimating the effort.

Estimation by analogy method requires one or more completed projects that are similar to the proposed new project. The success of this method depends on the selection of the most relevant and similar projects. Different methods have been proposed by many researchers. Tuan Khan Le et al proposed the use of Effort Inconsistency identifier [EID] for filtering inconsistent data in the historical projects [3]. Ekrem Kocaguneli et al identified the essential assumptions of Analogy based Effort Estimation [9]. It has been observed by them that whenever the assumptions listed in their research paper are violated, those situations are removed. Then, modified system is built so that there is an increase in the accuracy in the estimation of the project. The limitation is that their experiments ignored hard training cases. Jaifeng Wen et al used the concept of genetic programming using arithmetic mean, harmonic mean and geometric mean [11].

2.2.1 Artificial Neural Network [ANN]

ANN is a system that has certain performance characteristics in common with the biological neural networks. This type of networks has two layers, namely input layer and output layer that have links between them. These links carries weights. There can be hidden layers between these layers. Back propagation Algorithm is the most popular method for training Multi Layer Perceptron [MLP]. In this model, there are two passes, a forward pass and a backward pass. In the forward pass, the weights are fixed and during the backward pass, the weights are adjusted accordingly to the error correction rule. The adjustments in weights are based on the error produced between the desired and actual output. Chao-Jung Hsu et al made a study to improve the software effort estimation using ANN and many other methods. The uses of linear weights were suggested for the combination of estimation methods [10].

2.2.2 Genetic Algorithm [GA]

Genetic Algorithm is a search based algorithm to get an optimal solution. It is an evolutionary computation method. Genetic Algorithm creates population of individuals consecutively due to which, we get optimal solution for the given problem. The search process depends on the following components:

a) A list of solutions to the problem
b) A fitness function
c) Initialization of initial population
d) Selection operator
e) Reproduction operator

The main problems that are faced in the effort prediction by analogy are feature selection, no. of analogies to use, similarity measure, scaling , budget and Schedule pressure [5][7]. The commonly used similarity function is the weighted Euclidean Distance given as below:

$$\text{Distance}(p1,p2) = \text{sqrt}\left[\sum_{i=1}^{i=l} wi\,(f1-f2)2\right]$$

Where p1 and p2 denotes any two members of the project data sets, l is the number of features of the project, f1 and f2 denotes the features and wi is the weight of each feature. The software effort estimation with minimum features can be done as classification by using a feed forward neural network [6]. The selection of projects for the Analogy based Software Cost Estimation using Genetic Algorithm has been proposed by Y.F.Li et al [8]. In that paper, Genetic Algorithm is used as the optimization technique for project selection. It is shown, that performance of Analogy Based Estimation has improved by adopting Genetic Algorithm, The feasibility of the method was validated by applying to well known Albrecht data set and Desharnais Data set. However, it is also said that simultaneous optimization of historical data sets and feature weights could lead to better optimization.

2.2.2 Genetic Programming [GP]

Genetic Programming is an extension of Genetic Algorithm. It does not have the restriction that the representation of individual has to be of fixed length binary string as in the case of Genetic Algorithm. In Genetic Programming, the chromosome is some type of program normally in the form of binary tree representation. Genetic Programming offers flexibility to perform operations in a hierarchical way. In [14], Colin J. Burgess and Martin Lefly evaluate the potential of Genetic Programming in software effort estimation in terms of accuracy and ease of use.

2.2.3 Differential Evolution [DE]

Differential Evolution is also a evolutionary computational method developed in 1995 by R.Storn and K.V.Price. It is a stochastic, population based optimization algorithm. It differs from other Evolutionary Computation tools in its way of operation. In this method, mutation is applied first to generate a trial vector which is then used with a crossover operator to produce the offspring. Further, the step sizes are influenced by the difference between the individuals of the current population and not from the prior known probability distribution function. When compared to most other EAs, DE is much more simple and straightforward to implement. The space complexity of DE is low as compared to some of the most competitive real parameter optimizers [15]. Due to this feature, DE is used for handling large scale and expensive optimization problems

3. Using differential evolution Algorithm in Estimation by Analogy

Estimation by Analogy is the more effective methodology than other methods, as it is very simple and easy to understand. It is also easy to relate the output with the input. The estimation is almost accurate, if the most similar completed projects are selected. The different steps involved in the proposed method are given below:

1. Collect all the past relevant projects

2. Analyze each project and find the necessary parameters

3. Select the most relevant projects

4. Estimate the effort of the current project by comparing with the selected few most relevant projects

The selection of the most relevant projects would simplify the process of estimation. The principle of differential evolution is proposed to be used for this selection process. Differential Evolution Algorithm is proposed so that the exploration ability is improved [4].

In the proposed algorithm, the Primary population (Pp) set consists of selected individuals. The secondary population (Ps) serves as an archive of those offspring rejected by the selection operator. The steps for the algorithm are given below:

```
1.  Set the counter for generation t=1
2.  Initialize the control parameters
3.  Create and initialize the Primary Population
    Pp (1) of n individuals
4.  While terminating condition not true
    For each individual xi(t) in Pp (t) do
     Evaluate the fitness f(xi(t))
     Create a sample vector vi(t) by applying the
    mutation operator
    Create an offspring xi'(t) by applying cross
    over operator
      If f(xi'(t)) is better than f(xi(t) then
            add xi'(t) to Pp (t+1)
            xr(t) = xi(t)
      else
            add xi(t) to Pp (t+1)
            xr(t) = xi'(t)
      end
    // Grouping rejected offspring in the
    Secondary Population (Ps)
     if (t==1)
            include xr(t) in the Secondary
            Population(Ps)
     else
             if f((xr(t)) is better than f(xia(t) then
             replace  xia(t) with xr(t)
      end
     end
    end
    End
```

3.1 Comparing the use of Differential Algorithm with other Evolutionary Algorithms in the estimation of software effort

Differential Algorithm has got many similarities with other evolutionary algorithms like Genetic Algorithm, Artificial Neural Networks, Genetic Programming and others. But it differs in the fact that the information about the distance and direction between the individuals in the current population is used to guide the search process. These are the good indication of the diversity in the population. If the distance is more, the individual should take large step sizes and if the distance is less, the step sizes should be small to exploit local areas. This feature can be used in the selection of relevant projects in the estimation of analogy method.

4. Conclusion

A new method is proposed in this paper to find the most similar past projects to be used in the estimation by analogy models. The idea is derived from differential evolution. Differential evolution is stochastic, population based search strategy. This algorithm can be used to get more accurate results. The similarities between the projects such as the key attributes and features can be compared by using this algorithm. Less informative and less needed attributes can be removed.

References

[1] Tim Menzies , Zhihao Chen, Jairus Hihn and Karen Lum, Selecting Best Practices for Effort Estimation, IEEE transactions on Software Engineering ,2006

[2] M.Jorgenson , A Review of Studies in Expert estimation of software development effort , Journal of systems and software ,pp 37-60, 2004

[3] Tuan Khan H Le-DO, Kyuang-A Yoon , Yeong-Seok Seo, Doo-Hwan Bae, Filtering of inconsistent Software Project Data for Analogy- based Effort Estimation, IEEE Computer Software and Applications Conference, 2010.

[4] M.XM.Ali and A.Torn, ,Population set based global optimization Algorithms : some modifications and numerical studies, Computers and Operation Research,1703-1725,2004

[5] Juan J.Cuadrado-Gallego,Pablo Rodriguez-Sorio, Borja Martin-Herrera , Analogies and differences between Machine Learning and Expert based Software Project Estimation, ACIS International Conference on Software Engineering , Artificial Intelligence, Networking and Parallel/Distributed Computing, 269-275, 2010

[6] Jin-Cherng Lin,Chu-Ting Chang and Sheng-Yu Huang, Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression, International Symposium on Computer Science and Society, 349-352, 2011

[7] Ning Nan and Donald E. Harter Impact of Budget and Schedule Pressure on Software Development CycleTime and Effort,IEEE Transactions of Software Engineering, 624-637, 2009

[8] Y.F.Li, M.Xie , T.N.Golt, A study of Genetic Algorithm for Project Selection for Analogy Based Software Cost Estimation , IEEE Transactions of Software Engineering, 2007

[9] Ekrem Kocaguneli, Tim Menzies, Ayse Bener and Jacky W. Keung, Exploiting the essential assumptions of Analogy based Effort Estimation, IEEE Transactions of Software Engineering, 2011

[10] Chao-jung Hsu, Nancy Urbina Rodas, Chin-yu Huang, and Kuan – li Peng, A study of improving the accuracy of software effort estimation using linearly weighted combinations, 34th Annual IEEE computer software and application conference workshops – 2010

[11] Jaifeng Wen, Shixian Li, Linyan Tang, Improve analogy based software effort estimation using principal component analysis and correlation weighting, IEEE Transactions of Software Engineering, 2009

[12] L.RosenGrance, “Survey : Poor Communication causes most IT project failures”, Computer World, 2007

[13] M.Jorgensen, “A Review of Studies on Expert Estimation of Software Development Effort”, 2002

[14] Colin J.Burgess, Martin Lefly, “Can Genetic Programming inprove Software Effort Estimation? A comparative Evaluation:, Elsevier , 2001

[15] Swagatam Das, Ponnuthurai Nagaratnam Suganthan, ”Differential Evolution : A Survey of the State of Art”, IEEE transactions on Evolutionary Computation,2011