Analysis of Software Estimation Techniques

Sahithi Akiri
Dept. of Information Technology
NITK, Surathkal
Karnataka, India
sahithi.akiri@gmail.com

***Abstract*—As the scale of software projects developed increases, there is a need to accurately identify the estimation factor effort. There are many approaches proposed for the software estimation, and the developers has to choose a right method among them. This paper aims to present a comparative analysis of three techniques namely Putnam model, COCOMO technique and ANN-COCOMO model, then identify the best technique among them. Metrics used in the comparison are MRE and MMRE.**

***Index Terms*—Estimation, Putnum, COCOMO, MRE, MMRE**

I. Introduction

Software estimation is one of the most important activities in software project management. Accurate cost estimation is crucial because it can help to classify and prioritize development projects to determine what resources to commit to the project and how well these resources will be used. In order to complete the project properly and deliver it, to the customer as scheduled, the project manager must estimate the resources, effort and time needed as well as the cost of the software product. These estimates are calculated in the early development phases of the project. So, we need a good model to calculate these parameters. An early and accurate estimation model reduces the possibilities of conflicts between members in the later stages of project development. Often, project managers resort to estimating schedules skipping to estimate size. This may be because of the timelines set by the top management or the marketing team. However, whatever the reason, if this is done, then at a later stage it would be difficult to estimate the schedules to accommodate the scope changes. While estimating, certain assumptions may be made. It is important to note all these assumptions in the estimation sheet, as some still do not document assumptions in estimation sheets. Even good estimates have inherent assumptions, risks, and uncertainty, and yet they are often treated as though they are accurate.

II. effort estimation methods

Estimation is the process of finding an estimate, or approximation, which is a value that can be used for some purpose even if input data may be incomplete, uncertain, or unstable. Estimation determines how much money, effort, resources, and time it will take to build a specific system or product. In software development, effort estimation is the process of predicting the most realistic amount of effort (expressed in terms of person-hours or money) required to develop or maintain software based on incomplete, uncertain and noisy input. Various techniques existing for the estimation are categorized into Empirical and Heuristic approaches. Empirical techniques are based on systematic guesses by experts. In heuristic approaches, the characteristics to be estimated are expressed in mathematical equations.

The survey done includes the three models among which Putnam model is Empirical and COCOMO and ANN-COCOMO are Heuristic approaches.

A. Putnam SLIM effort estimation model

Putnam model, also known as SLIM (Software Lifecycle Management) model, is an automated empirical software estimation model. The model created by Lawrence Putnam, deals with the effort and time that could be taken to complete a software project of given size.

This model follows the Rayleigh-distribution curve. As this is an empirical model which depends on observation and experience, a software equation was derived.

The two parameters of the Rayleigh curve are Time and Effort.

$$L = C_k \times K^{\frac{1}{3}} \times t_d^{\frac{4}{3}} \tag{1}$$

where

- L is size of the project in KLOC,
- t_d is the time for the development the software in years
- C_k is a constant that reflects the productivity. Generally C_k is set to 2 for poor, 8 for good and 12 for excellent software development environments.

The main advantage of the model is the simplicity to calculate.

According to this model, total effort reduces as the time for the project increases. So this is sensitive to the time of development.

B. COCOMO estimation model

The Constructive Cost Model (COCOMO) is a procedural software cost estimation model developed by Barry W Boehm. It is a regression model based on LOC, i.e number of Lines of Code. It is used to estimate cost and effort in software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry

Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

- Effort: Amount of labor that will be required to complete a task. It is measured in person-months units.
- Schedule: Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put. It is measured in the units of time such as weeks, months.

Different models of COCOMO have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below.

$$Effort = a \times [KLOC]^b \tag{2}$$

where a and b depend on different categories of system

- Organic: a = 2.4, b = 1.05
- Semi detached: a = 3.0, b = 1.12
- Embedded: a = 3.6, b = 1.20

C. Artifical Neural-Network COCOMO model

Artificial Neural-Network Constructive Cost Model(ANN-COCOMO) is developed by applying neural network concepts on the existing and highly used COCOMO model. In order to model complex non-linear problems, neural networks are used in most of the domains. ANN-COCOMO model is designed based on COCOMO 2 model. It uses a multi-layer perceptron(MLP) architecture in order to predict the effort estimation. The efficiency of the model depends on the architectural design which involves number of hidden layers, nodes in each of hidden layer. MLP has an input layer, a number of hidden layers and an output layer. There are 17 nodes in the input layer. Out of them, 15 inputs are Effort Multipliers(EM), one bias and the actual effort of project. These 17 inputs together form an input layer. To get the predicted effort, COCOMO 2 model prediction equation is applied here. The equation is as below:

$$log(Effort) = log(a \times [KLOC]^b \times_{i=1} \Pi^{15} EM_i) \tag{3}$$

where

- EMi(Effort Multiplier) represents a cost driver
- KLOC indicates kilo lines of code
- Organic: a = 2.4, b = 1.05
- Semi detached: a = 3.0, b = 1.12
- Embedded: a = 3.6, b = 1.20

The output of Eq. (3) is fed into the activation function of the model. Where it is compared with a threshold value and decides whether to activate the input or not. If it is activated, the difference between the actual effort and predicted effort is calculated and accordingly the weights and biases are updated. The inputs are fed into the model till we reach no error for all training data inputs.

III. Evaluation metrics and results

Analysis of the models are done on the datasets containing NASA projects where each of the dataset contains entries corresponding to all the metrics required in estimation calculation like Effort Multipliers, KLOC, Actual effort and development time. For evaluating the different software effort estimation models, the most widely accepted evaluation criteria are the mean magnitude of relative error (MMRE) and probability of a project having a relative error of less than or equal to 0.25 (Pred(l)). The Magnitude of Relative Error (MRE) is defined as follows

$$MRE = \frac{actualeffort - calculatedeffort}{actualeffort} \tag{4}$$

The MRE value is calculated for each observation whose effort is predicted. The average of MRE over multiple observations (N) can be achieved through the Mean

$$MMRE = \frac{{}_i\Sigma MRE}{n} \tag{5}$$

Therefore low MMRE for a model implies that the model is more accurate.

TABLE I
EFFORT CALCULATED

Project no.	**Features**	
	LOC	***Actual Effort***
1.	29.5	120
2.	9.7	25.5
3.	5.5	18
4.	16.3	82
5.	32.6	170

TABLE II
MRE

Project No.	**MRE values**		
	Putnam	***COCOMO***	***ANN-COCOMO***
1.	0.4909	0.4231	0.1342
2.	0.7650	0.5314	0.1789
3.	0.4523	0.2011	0.0363
4.	0.5673	0.3410	0.1376
5.	0.3785	0.2098	0.0940

TABLE III
MMRE

Technique	***MMRE value***
Putnam	0.5308
COCOMO	0.34128
ANN-COCOMO	0.1162

IV. Conclusion

The calculation and comparision of MRE and MMRE is done on three different models, COCOMO, Putnam and ANN COCOMO, using different datasets. It is observed that MMRE score for ANN COCOMO model is lesser than other two, which says that ANN COCOMO model is more accurate for software engineering effort estimation. It is also observed that Putnam model is sensitive to time and is less accurate than rest of two techniques. [6].

REFERENCES

[1] Promise Software Engineering Repository ”http://promise.site.uottawa.ca/SERepository/datasets-page.html”.

[2] COCOMO Nasa dataset ”http://promise.site.uottawa.ca/SERepository/datasets-page.html”.

[3] Muhammad M. Albakri1 M. Rizwan Jameel Qureshi2 ”Empirical Estimation of COCOMO I and COCOMO II Using a Case Study”.

[4] Ning Jingfeng, Jiang Yan2 and Yu Honglei ”Research and application of estimation method for software cost estimation based on Putnam model”.

[5] Anupama Kaushik,Ashish Chauhan, Deepak Mittal, Sachin Gupta , ”COCOMO Estimates Using Neural Networks”.

[6] Dharmesh Santani , Mahesh Bundele, Poonam Rijwani . ”Artificial Neural Networks for Software Effort Estimation”.