A Study On Software Cost Estimation

Josephin.M.Baby Anusha M.E , Rajeswari Mukesh M.E, Ph.d  
Hindustan Institute of Technology and Science,  
Josephin.anusha@gmail.com, rajeswarim@hindustanuniv.ac.in

Abstract: Software Cost Estimation is one of the most important part in software development. It involves in estimating the effort and cost in terms of money to complete the software development. Software Cost Estimation is very important when lines of code for the particular project exceeds certain limit, also when the software deployed with too many bugs and uncovered requirements the project will go incomplete. Software cost estimation of a project plays a vital role in acceptance or rejection of its development. There have been lot of Software cost estimation models in use. Appropriate cost estimation guarantees software to finish on budget and time. The conclusion is that there is no technique available that is best for all projects in every kind of scenarios. Hence a thorough understand and a careful analysis is required for the comparison of all methods in order to come up with best possible method which may lead to the most accurate and realistic estimates. This paper deals with the progress of finding out best Software Cost Estimation method to give better accuracy on Software Cost Estimation.

Keywords: project estimation, effort estimation, cost estimation, COCOMO

1. INTRODUCTION

In the Competitive environment of Software Industry, the victorious organization will be the one, which the capability to develop and deliver the software product to the customers or end-users within the promised period of time while staying in financial budgetary boundaries. Hence proper estimates are the drivers which may steer to achieve the milestones. In other words it may be said it is quiet necessary to understand and control the cost by proper estimation for the proper management, enhanced quality and better understanding of the software project. The overall process of developing a cost estimate for software is not different from the process for estimating any other element of cost. There are, however, aspects of the process that are peculiar to software estimating. Some of the unique aspects of software estimating are driven by the nature of software as a product. Other problems are created by the nature of the estimating methodologies. Software cost estimation is a continuing activity which starts at the proposal stage and continues through the lift time of a project. Continual cost estimation is to ensure that the spending is in line with the budget.

In the last three decades, many quantitative software cost Estimation models have been developed. They range model uses data from previous projects to evaluate the current project and derives the basic formulae from analysis of the particular database available. An analytical model, on the other hand, uses formulae based on global assumptions, such as the rate at which developer solves problems and the number of problems available. Most cost models are based on the size measure, such as Lines of Code and Function Points, obtained from size estimation. The accuracy of size estimation directly impacts the accuracy of cost estimation. But none of the above leads to an accurate estimate.

2. RELATED WORK

In current situation of software industries, successful project completion within time is most important task for any industry. The management point of view effort prediction is complicated task. Report says that around 65-80 per cent of the project faces overrun of delivery date. Effort overrun directly proportional to cost overrun, so accurate effort prediction is also important. There are a lot many models available for prediction of software development effort and cost. COCOMO (COnstructive COst MOdel)[27] is most commonly used model. But machine learning methods for software prediction are more appropriate, because they are more adaptable. When we are talking about specifically software development effort prediction problem, the output (effort value) of the system is very complexly dependent on input parameters, such as size of the problem, experience and many other. Now this complex relationship cannot be described or expressed using simple mathematical equations. In such situations neural network is more suitable to use, the reason can easily understand if we see at architecture of neural networks.

Many researchers have applied the neural networks approach to estimate software development effort. Many different models of neural networks have been proposed. They may be grouped in two major categories. First one is feed forward networks where no loops in the network path occur. Another one is feedback networks that have recursive loops. The feed forward multilayer perceptron with back propagation learning algorithm are the most commonly used in the cost estimation field. Another study by Samson et al, uses an albus multilayer perceptron in order to predict software effort. They use Boehm’s COCOMO dataset. Srinivasan and Fisher report the use of a neural network with a back propagation learning algorithm. They found that the neural network outperformed other techniques. Some primary work in the use of neural network in estimating software cost by Karunanithi et al. produced very accurate results, but the major setback in their work was due to the fact the accuracy of the result relied heavily on the size of the training set. COCOMO is arguably the most popular and widely used software
estimation model, which integrates valuable expert knowledge. Understanding the adversity in applying neural networks, Nasser Tadayon has proposed a dynamic neural network that will initially use COCOMO II Model. COCOMO, however, has some limitations. It cannot effectively deal with imprecise and uncertain information, and calibration of COCOMO is one of the most important tasks that need to be done in order to get accurate estimations. So, there is always scope for developing effort estimation models with better predictive accuracy.

The rest of this paper is organized as follows: Section 3 describes the Software Cost Estimation Techniques, Section 4 introduces the comparison of the Estimation methods, Section 5 Future Work and finally section 6 includes conclusion.

3.1 ESTIMATION TECHNIQUES

Generally, there are many methods for software cost estimation, which are divided into two groups: Algorithmic and Non-algorithmic. Using of the both groups is required for performing the accurate estimation. If the requirements are known better, their performance will be better. In this section, some popular estimation methods are discussed.

a. Non Algorithmic Methods (NA)

Contrary to the Algorithmic methods, methods of this group are based on analytical comparisons and inferences. For using the Non Algorithmic methods some information about the previous projects which are similar the under estimate project is required and usually estimation process in these methods is done according to the analysis of the previous datasets. Here, three methods have been selected for the assessing because these methods are more popular than the other None Algorithmic methods and many papers about their usage have been published in the recent years.

1. Estimation By Analogy

In this method, several similar completed software projects are noticed and estimation of effort and cost are done according to their actual cost and effort. Estimation based on analogy is accomplished at the total system levels and subsystem levels. By assessing the results of previous actual projects, we can estimate the cost and effort of a similar project. The steps of this method are considered as:

i. Choosing of analogy
ii. Investigating similarities and differences
iii. Examining of analogy quality
iv. Providing the estimation

2) Expert judgment

Estimation based on Expert judgment is done by getting advice from experts who have extensive experiences in similar projects. This method is usually used when there is limitation in finding data and gathering requirements. Consultation is the basic issue in this method. One of the most common methods which works according to this technique, is Delphi. Delphi arranges an especial meeting among the project experts and tries to achieve the true information about the project from their debates. Delphi includes some steps:

i. The coordinator gives an estimation form to each expert.
ii. Each expert presents his own estimation (without discussing with others)
iii. The coordinator gathers all forms and sums up them (including mean or median) on a form and asks experts to start iteration.
iv. Steps (ii-iii) are repeated until an approval is gained.

3) Machine learning Models

Most techniques about software cost estimation use statistical methods, which are not able to present reason and strong results. Machine learning approaches could be appropriate at this filed because they can increase the accuracy of estimation by training rules of estimation and repeating the run cycles. Machine learning methods could be categorized into two main methods, which are explained in the next subsections.

a) Neural networks

Neural networks include several layers which each layer is composed of several elements called neuron. Neurons, by investigating the weights defined for inputs, produce the outputs. Outputs will be the actual effort, which is the main goal of estimation.

Back propagation neural network is the best selection for software estimation problem because it adjusts the weights by comparing the network outputs and actual results. In addition, training is done effectively. Majority of researches on using the neural networks for software cost estimation.

b) Fuzzy Method

All systems, which work based on the fuzzy logic try to simulate human behavior and reasoning. In many problems, which decision making is very difficult and conditions are vague, fuzzy systems are an efficient tool in such situations. This technique always supports the facts that may be ignored. There are four stages in the fuzzy approach:

Stage 1: Fuzzification: to produce trapezoidal numbers for the linguistic terms.
Stage 2: to develop the complexity matrix by producing a new linguistic term.
Stage 3: to determine the productivity rate and the attempt for the new linguistic terms.
Stage 4: Defuzzification: to determine the effort required to complete a task and to compare the existing method.

b. Algorithmic Models (A)

To date most work carried out in the software cost estimation field has focused on algorithmic cost modeling. In this process costs, are analyzed using mathematical formulae linking costs or inputs with metrics to produce an estimated output. The formulae used in a formal model arise from the analysis of historical data. The accuracy of the model can be improved by calibrating the model to your specific development environment, which basically involves adjusting the weightings of the metrics. There are a variety of different models available, the best known are Boehm's COCOMO [BOEHM-81], Putman's SLIM, and Albrecht's' function points [ALBR-83]. In terms of the estimation process, nearly all algorithmic models deviate from the classical view of the cost estimation process.
These models work based on the especial algorithm. They usually need data at first and make results by using the mathematical relations. Nowadays, many software estimation methods use these models. Algorithmic Models are classified into some different models. Each algorithmic model uses an equation to do the estimation:

The Differences among the existing algorithmic methods are related to choose the cost factors and function. All cost factors using in these models are:

- **Product factors**: required reliability; product complexity; database size used; required reusability; documentation match to life-cycle needs;
- **Computer factors**: execution time constraint; main storage constraint; computer turnaround constraints; platform volatility;
- **Personnel factors**: analyst capability; application experience; programming capability; platform experience; language and tool experience; personnel continuity;
- **Project factors**: multisite development; use of software tool; required development schedule.

1) **Linear Models**

Commonly linear models have the simple structure and trace a clear equation as shown in equation 1 below:

$$ Effort = a_0 + \sum_{i=1}^{n} a_i x_i $$  \hspace{2cm} (1)

Where, $a_1, a_2, \ldots, a_n$ are selected according to the project information.

2) **Multiplicative models**

The form of the multiplicative model can be represented by using the below equation 2:

$$ Effort = a_0 \prod_{i=1}^{n} a_i^{x_i} $$  \hspace{2cm} (2)

where, $a_1, \ldots, a_n$ are selected according to the project information. In this formula, only allowed values for $x_i$ are -1,0 and +1.

3) **Source Line of Code**

SLOC is an estimation parameter that illustrates the number of all commands and data definition but it does not include instructions such as comments, blanks, and continuation lines. This parameter is usually used as an analogy based on an approach for the estimation. After computing the SLOC for software, its amount is compared with other projects which their SLOC has been computed before, and the size of project is estimated. SLOC measures the size of project easily. After completing the project, all estimations are compared with the actual ones.

Thousand Lines of Code (KSLOC) are used for estimation in large scale. Using this metric is common in many estimation methods. SLOC Measuring seems very difficult at the early stages of the project because of the lack of information about requirements. Since SLOC is computed based on language instructions, comparing the size of software which uses different languages is too hard. Anyway, SLOC is the base of the estimation models in many complicated software estimation methods.

4) **Function Point Size Estimates**

At first, Albrecht (1983) presented Function Point metric to measure the functionality of project. In this method, estimation is done by determination of the following indicators, which are referred as User Inputs, User Outputs, Logi Files, Inquiries and Interfaces.

According to the previous experiences, function point could be useful for software estimations because it could be computed based on requirement specification in the early stages of project.

5) **Seer-Sem**

SEER-SEM model has been proposed in 1980 by Galorath Inc (Galorath, 2006). Most parameters in this method are commercial and, business projects usually use SEER-SEM as their main estimation method. Size of the software is the most important feature in this method and a parameter namely $S_e$ is defined as effective size. $S_e$ is computed by determining the five indicators: newsize, existingsize, reimp and retest. The effective size ($S_e$) is calculated using the equation 3 as shown below:

$$ S_e = \text{Newsize} + \text{ExistingSize} (0.4\text{Redesign} + 0.25\text{reimp} + 0.35\text{Retest}) $$  \hspace{2cm} (3)

After computing the $S_e$ the estimated effort is calculated as shown in equation 4:

$$ Effort = t_d = D^{-0.2} X \left( \frac{S_e}{C_{te}} \right)^{0.4} $$  \hspace{2cm} (4)

Where $D$ is relevant to the staffing aspects; it is determined based on the complexity degree in staffs structure. $C_{te}$ is computed according to productivity and efficiency of the project method is used widely in commercial projects.

6) **COCOMO**

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, so all of the details are published.

COCOMO-II is the latest version of COCOMO that predicts the amount of effort based on Person-Month (PM) in the software projects. It uses function point or line of code as the size metrics, Effort Multipliers and scale factors. Some rating levels are defined for scale factors including very low, low, nominal, high, very high and extra high. A quantitative value is assigned to each rating level as its weight.
COCOMO II has some special features, which distinguish it from other ones. The Usage of this method is very wide and its results usually are accurate.

Equations 5 and 6 are used to estimate the effort and schedule as below:

\[ PM_{NS} = A^* Size^E * \prod_{j=1}^{17} EM_i \] (5)

Where E = B + 0.01 \sum_{j=1}^{5} SF_j, A = 2.94, B = 0.91

\[ TDEV = C^* (PM)^F \] (6)

Where F = D + 0.2*0.01* \sum_{j=1}^{5} SF_j = D + 0.2*(E-B)

C = 3.67, D = 0.28

7) Putman’s model

This model has been proposed by Putman according to manpower distribution and the examination of many software projects (Kemerer, 2008). The main equation for Putnam’s model is equation 7 as shown below:

\[ S = EX (Effort)^{1/3} T_d^{4/3} \] (7)

where, E is the environment indicator and demonstrates the environment ability. T_d is the time of delivery. Effort and S are expressed by person-year and line of code respectively. Putnam presented another formula for Effort as shown in equation 8:

\[ Effort = D_0 X T_d^3 \] (8)

Where, D_0 is the manpower build-up factor.

SLIM (Software Life Cycle Management) is a tool that acts according to the Putnam’s model.

IV. COMPARISON OF THE COST ESTIMATION METHODS

At this section according to the previous presented subjects, it is possible to compare mentioned estimation methods based on advantages and disadvantages of them. This comparison could be useful for choosing an appropriate method in a particular project. On the other hand, selecting the estimation technique is done based on capabilities of methods and state of the project. Below table shows a comparison of mentioned methods for estimation. For doing comparison, the popular existing estimation methods have been selected.

<table>
<thead>
<tr>
<th>Method</th>
<th>Type</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>COCOMO</td>
<td>A</td>
<td>Clear results, very common</td>
<td>Much data is required, It’s not suitable for any project,</td>
</tr>
<tr>
<td>Expert</td>
<td>NA</td>
<td>Fast prediction, Its success depend</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 1: Comparison of the existing methods

Function Point  
A  | Language free, Its results are better than SLOC  | Mechanization is hard to do , quality of output are not considered

Analogy  
NA  | Works based on actual experiences, having especial expert is not important  | A lots of information about past projects is required, In some situations there are no similar project

Neural Network  
NA  | Consistent with unlike databases, Power of reasoning  | There is no guideline for designing, The performance depends on large training data

Fuzzy  
NA  | Training is not required, Flexibility  | Hard to use, Maintaining the degree of meaningfulness is difficult

V. FUTURE WORK

We have tried to cover most of the models in our survey and for sure there are many remaining but what actually is demonstrated through our research paper is that cost estimation provide basis for better result in the field if software development. Estimating the project using good technique and according to the nature of the software, can help in achieving better results in term of right allocation of resources, time and budget. Cost estimation is more reliable and sophisticated and it can’t be neglected. For proper management of any project, proper measurement must be adopted. So a deep study and analysis of Software Cost Estimation methods and techniques is required by estimators (Project Managers) in order to ensure the project to be completed on time and on budget. Our research is on progress for developing a Software Cost Estimation method with better accuracy compared to the above discussed already existing SCE methods.

VI. CONCLUSION

Finding the most important reason for the software project failures has been the subject of many researches in last decade. According to the results of several researches, the root cause for software project failures is inaccurate estimation in early stages of the project. So introducing and focusing on the estimation methods seems necessary for achieving to the accurate and reliable estimations. In current
study most of the present estimation techniques have been illustrated systematically. Since software project managers are used to select the best estimation method based on the conditions and status of the project, describing and comprising of estimation techniques can be useful for decreasing of the project failures. There is no estimation method which can be present the best estimates in all various situations and each technique can be suitable in the special project. It is necessary understanding the principals of each estimation method to choose the best. Because performance of each estimation method depends on several parameters such as complexity of the project, duration of the project, expertise of the staff, development method and so on. Some evaluation metrics and an actual estimation example have been presented in this paper just for describing the performance of an estimation method. Trying to improve the performance of the existing methods and introducing the new methods for estimation based on today’s software project requirements is the future work of our research.

References


AUTHOR

Josephin Anusha received BE. degree in Computer Science and Engineering from St. Xaviers College of Engineering and ME degree in Computer Science and Engineering from Noorul Islam College of Engineering in 2003 and 2005, respectively. During 2005-2007, she worked as a Lecturer in Vemana Institute of Technology, Bangalore. During 2007 – Till Date she is working in Software Industry.