An Analytical Study on Software Effort Estimation Techniques
Pawandeep Kaur1, Rupinder Singh2, Jaspreet Kaur3
1Student of CSE Department, 2Asst. Professor of CSE Department, 3Student of CSE Department
Chandigarh University,
Gharuan, Punjab, INDIA

Abstract: Software effort and cost estimation has turned out to be the major challenge in IT industries. In this paper, different software effort and cost estimation techniques like algorithmic methods, expert judgment, analogy based estimation and soft computing methods and their various aspects are discussed. All these techniques are discussed in detailed manner in this paper. Software effort estimation is followed by cost assessment that is useful for both customers and developers. Effort estimation is further converted into cost estimation by converting effort/person per month into dollars. In this paper we made a comparison of various software effort estimation techniques on the basis of their input, output, biasness, pros, cons type of method. No single model is good or best for effort estimation we can choose a method based on the requirements and conditions of our system to get good results. We can use the combination of two or more different techniques for better estimation results.

Keywords: Software Effort Estimation Techniques, Fuzzy logic, Neural Networks, Accurate estimation

I. Introduction

The Software effort estimation is the process of predicting the realistic amount of effort/cost required for the completion of software projects. Effective software project estimation is one of the most demanding and essential activities in software development. Software effort estimation holds good at early stages of software development to meet competitive demands of today’s industry. Now the days, people are looking for new and good quality software but at low cost and in lesser time [1]. So the cost-benefit analysis is to be done by client or developer [2]. Analyzed estimation is translated into dollars to get cost estimation. As the demand of software effort estimation has increased in industry it is becoming a most important task that has to be done at early stages of development so that feasibility can be evaluated. Accuracy in effort estimation plays a vital role in the success of software project management [3]. Resource allocation is the biggest issue in estimation as if excess amount of resources are allocated then it is called overestimation and if lesser then it is called underestimation. Accurate effort prediction is useful in reducing the impact of risks. Good software estimation is difficult task at early stages as having a lot of uncertainties in inputs like change in requirement, platform change, size of project, budget constraints, complexity etc.

A. Importance of accurate estimation

- To determine what amount of resources and manpower required to complete the project and how well they will be used.
- To find effort in terms of time, cost and technical resources.
- To assess the impact of changes in project and support re-planning.

B. Ingredients to accurate estimation

- **Working Environment**: The environment in which project is to developed or executed is very important.
- **Tools**: Tools and techniques that are used to execute or evaluate the project play an important role in software development.
- **Consistency**: The historical data is useful to estimate similar projects which were developed earlier. On occasion new observations, obstacles, issues will occur so it will make the get experience to tackle the situation.

II. Work at Background

A Magne Jorgensen et al. [4, 5] focused on the reason of occurrence of error in software effort estimation by interviewing the employees in different labs in software organizations. Then they focused on the expert judgment process with some useful, easily implemented practices. They found that by mixing planning, estimation and bidding leads to less accurate results. They also define that by combining the estimation methods having different weakness and strengths increase the estimation accuracy because estimation increases with more number of
sources. They define the guidelines for the selection of experts and estimation methods, and to provide learning opportunities.

Iman Attarzadeh et al. [6] described the use of fuzzy logic modeling for effort estimation. They proposed a new fuzzy logic model and provide training for better estimation. After training they compare it with COCOMO to get estimation results.

K.K.Aggarwal et al. [7], explored the use of neural network for estimating the lines of code (LOC) of the software projects. They are using the International Software Benchmarking Standards Group (ISBSG) Repository Data (release 9) for the software projects having record of LOC. They perform experiments on various algorithms and find out the best one, having minimum Mean Absolute Percentage Error (MAPE).

Magne Jorgensen et al. [8], discussed a lesions learning technique for effort estimation and uncertainty assessment. They explained that this method is based on learning process; they create two groups-control group and learning group. They found that uncertainty assessment results better in case of control group. So lesions learning technique will be better if designed carefully and having minimum bias.

Ning Nan et al. [9] described the affect of pressure due to budget and schedule compression on cycle time and cost of software development either positive or negative. They proposed a nonlinear relationship between budget pressure and schedule pressure on software development. They proposed four hypotheses and collect the cross sectional data from international technology firm to test those hypotheses. They assess the possibility of achieving effort and improved cycle time, minimum estimation bias with proper management of pressure in software development. Analysis suggests that if software clients cooperate with development teams in dealing with schedule compression then organization has beneficial effects.

Ali Idri et al. [10] described the architecture of neural network. They explained the usage of neural networks in software effort estimation and the easiness with which neural networks were used in estimation purposes.

E.Praynlin et al. [11] proposed ANFIS for software effort estimation and using Gaussian Membership Function, Trapezoidal Function and Triangular Function and analyzed the results in the view of MRE.

Lotfi Zadeh [12] gave the idea about fuzzy values and the usage of fuzzy systems in software effort estimation. Ashish Sharma et al. [13], proposed a systematic and accurate method for software development effort estimation on the basis of requirement based complexity of the new developing software i.e. requirement based software development effort estimation (RBDEE) from IRBC(Improved Requirement Based Complexity) using requirements as input from SRS. Firstly they develop RBDEE method and then compare it with other existing estimation methods to check its robustness. As this technique is used at early stage of software development, so it will give accurate results than the techniques using function point as input, based on expert judgment (may be biased).

Elham Khatibi et al. [14], proposed a systematic framework for the effort estimation in the organization. Basically indictors are divided into four groups. The proposed method has indicators which are helpful to discover the strength and weakness of the organization in respect to effort estimation.

Chander Diwaker et al. [15] explained the various cost estimation techniques/methods developed like algorithmic methods, analogy based estimation, expert judgment method, price to win method, top-down method, and bottom-up method. They also discussed the metrics and techniques for estimation. The Strengths and weakness of different approaches were explained. They discussed that for known projects, we should use the expert judgment method or analogy based method and for large and lesser known projects, Jyoti G. Borade et al. [16] explained the various software metrics- size oriented metrics (SLOC, DSI), function oriented metrics (FP, UAFP), object points, test points, use case points. After that they discussed several estimation techniques and they also explained recent trends in software estimation like Bayesian Belief Network and Test execution effort. They enlighten the usefulness of BBN when required information is uncertain, vague, incomplete and conflicting.

III. Techniques for Software Effort Estimation

Software estimation techniques are broadly classified into two categories: Algorithmic models and Non-algorithmic models.

Algorithmic models are used to calculate the effort from the historical data objects and are based on formulas. These models calibrate the prescribed data and formulas for estimating effort from historical data. Input to these models are SLOC, reliability, programming language etc. Non-Algorithmic models are used to capture the experience and knowledge of estimator who provides the estimates based on the past similar projects. It basically depends upon the experience and knowledge of estimator. If estimator is experienced with similar projects as ours then there are chances of good estimation results. Soft Computing methods are basically the machine learning techniques like fuzzy logic, neural networks, genetic algorithms etc. are widely used these days to get good results in case of uncertainty and vagueness.
A. **Algorithmic techniques**
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, various cost drivers on which estimation have done. Different algorithmic techniques are:

- **LOC Based:**
  LOC based estimation is the simplest among all. This method measures the size of project simply by counting number of source instructions (excluding comment lines and header lines). but it is not helpful at the early stages of software development.

- **FP Based:**
  Function Point metric is defined by Albrecht (1983) to measure the functionality of project. It overcomes the limitations of LOC based. Estimation is done by calculating following factors like Number of User Inputs, Number of User Outputs, Number of Logic Files, Number of Interfaces, Number of Inquiries. By calculating complexity degree indicator we can evaluated UFP (Unadjusted Function Point) and Technical Complexity Factor (TCF) we can calculate FP

\[
FP = UFP \times TCF
\]

Function Points are free of languages, tools, construction methods. But FP is complex method.

- **COCOMO (Cost Constructive Model):**
  The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm in 1981 (also called COCOMO’81). This model is based on a regression formula with parameters using historical data and current characteristics of projects. This is the first single-value, static model for effort estimation. This model is based on a nonlinear equation, estimating effort in PM (person months) and schedule in months.

\[
Effort = a \times (KLOC)^b
\]

Here KLOC is kilo lines of code and a, b are complexity factors. COCOMO is set of three models: Basic COCOMO, Intermediate COCOMO, and Detached COCOMO having three different modes each. As this model have been experiencing difficulties in estimating new life cycle processes, reused projects, etc. So a new technique has been developed i.e. COCOMO II. COCOMO II is the successor of COCOMO’81 and developed in 1997 by Barry W. Boehm and his colleagues. It has 17 cost drivers, 5 scale drivers and one Line of Code (LOC). Now COCOMO model is useful for projects having code-reusability, off the shelf components etc. This model consists of application composition model, early design model, post architecture model having equations for effort estimation. The cost drivers of COCOMO II are rated on scale from very low to extra high as in the COCOMO 81[9]. The post architecture model is given as:

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

Where B=1.01+0.01\times j, A is Multiplicative Constant, Size is line of code(KLOC).

- **Putnam’s model:**
  In 1978 Lawrence Putnam created a model for software estimation. Putnam’s model is an empirical model and describes the time and effort required to finish the software project. Software equation for Putnam model:

\[
S = E \times \text{Effort}^{1/3} \times t_d^{1/3}
\]
where \( t_d \) is time required for development, \( E \) is the environmental factor which effect the development, \( S \) is the size in KLOC and Effort is in Person months. SLIM is the name give to the tools of Putnam’s model for cost estimation and manpower scheduling.

B. Non-Algorithmic models

In 1990’s non-algorithmic models was discovered and have been used in effort and cost estimation of projects.

- **Expert judgment:**
  Expert Judgment the most frequently applied technique for effort estimation of software projects. This model is used when there is limitation in finding and collecting accurate information. As it is based on experts, they must be experienced and there is probability of biasness in results. Process will be done by taking suggestions from the experts who had experience in similar projects.

- **Delphi:**
  Delphi method basically combined the expert’s opinion and reduces the biasness. In this technique a meeting is held with the experts and one coordinator. They will give their suggestions without discussing it with others. Coordinator will gather all the forms from experts and sums up them, process will repeat until result is approved by everyone.

- **Soft Computing Methods:**
  In 1990’s, soft computing techniques came up and are widely used in IT industry. Soft computing techniques are used in researches to handle uncertainty, imprecision due to its inherent nature. Generally used soft computing techniques are Fuzzy logics, Neural Networks for software effort prediction.

  - **Fuzzy Logic:**
    Now the days, fuzzy logic has becoming the most active technology to deal with uncertainty and vagueness in software project estimation as it bear a resemblance to human’s decision making ability. Fuzzy logics provide the linguistic representation of input and output data and also provide the knowledge based approach. In 1965, Loﬁt Zadeh developed a multi-valued set theory called fuzzy set theory [12].

  - **Neural Networks:**
    In 1943 Warren McCulloch and Walter Pitts created a computational model for neural networks based on mathematics and algorithms [6]. Neural networks are group of interconnected nodes (neurons) which are capable of computation and machine learning. Neural networks are used in software engineering due to its learning ability from previous data. The Neural Network is initialized with random weights and gradually learns the relationships implicit in a training data set by adjusting its weights when presented with these data. An Artificial Neural network can be single layer or multilayer perceptron [10].

IV. Comparison of software effort estimation techniques

<table>
<thead>
<tr>
<th>Techniques</th>
<th>Type- Algo/Non- Algo</th>
<th>Input</th>
<th>Output</th>
<th>Biasness</th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>Algorithmic</td>
<td>LOC</td>
<td>LOC</td>
<td>---</td>
<td>Simple and easy</td>
<td>Not useful in early stages</td>
</tr>
<tr>
<td>FP</td>
<td>Algorithmic</td>
<td>User inputs</td>
<td>FP</td>
<td>---</td>
<td>Language free, results are better than LOC</td>
<td>Mechanism is hard to do, quality of output is not so good</td>
</tr>
<tr>
<td>COCOMO</td>
<td>Algorithmic</td>
<td>KLOC</td>
<td>Effort(PM)</td>
<td>---</td>
<td>Clear results, common method</td>
<td>Much data is required</td>
</tr>
<tr>
<td>Putnam</td>
<td>Algorithmic</td>
<td>SLOC</td>
<td>Effort, Time</td>
<td>---</td>
<td>Results based on curve</td>
<td>Users input may affect results</td>
</tr>
<tr>
<td>Expert Judgment</td>
<td>Non-Algorithmic</td>
<td>Past project data</td>
<td>Suggestions from experts</td>
<td>Yes</td>
<td>Fast prediction, Adapt to special projects</td>
<td>Success depends upon experts</td>
</tr>
<tr>
<td>Delphi</td>
<td>Non-Algorithmic</td>
<td>Previously used data</td>
<td>Refined ideas</td>
<td>Yes</td>
<td>Less biasness than expert judgment</td>
<td>Experts may be biased, usually is done incomplete</td>
</tr>
<tr>
<td>Fuzzy</td>
<td>Non-Algorithmic</td>
<td>Linguistic values</td>
<td>Fuzzified output</td>
<td>No</td>
<td>Flexibility, reduces uncertainty</td>
<td>Maintaining the meaningfulness is difficult</td>
</tr>
<tr>
<td>Neural</td>
<td>Non-Algorithmic</td>
<td>Neurons</td>
<td>Trained output</td>
<td>No</td>
<td>Consistent with unlike databases, provide training</td>
<td>No guideline, performance depends upon large dataset</td>
</tr>
</tbody>
</table>
V. Conclusion and future scope

This paper presented an overview on various software effort estimation techniques presently available in IT industry. Software effort prediction is very important task in software development because the future of the project depends upon estimation task. The techniques discussed are LOC-Based estimation, FP-Based, COCOMO, Expert judgment, Delphi, Analogy-Based, Soft Computing techniques. A comparison is made between various estimation techniques on different factors like input, output, type of technique, biasness, pros, cons is helpful to know which technique should be used in particular situation. In earlier days algorithmic and non-algorithmic techniques were used only but now days soft computing and machine learning techniques are used due to their learning, feature extraction, dealing with complex data structure capabilities. In this paper we were concluded that no single estimation technique is best for all the situations we can choose the technique according to our requirements to get good estimation results.

References