Abstract

Rapid growth of software industry leads to need of new technologies. Software effort estimation is one of the areas that need more concentration. Exact estimation is always a challenging task. Effort Estimation techniques are broadly classified into algorithmic and non-algorithmic techniques. In this paper we present the review of some popular estimation techniques along with their pros and cons. SLIM, COCOMO, ESTIMAC, Function point, expert estimator, fuzzy logic, genetic programming, and analogy, neural are some effort estimation techniques.

Keywords

SLIM, COCOMO, Neural Network, Fuzzy logic, ESTIMAC

I. Introduction

Software effort estimate is always a striking & mind catching field. Getting exact estimation of software development is something which is always difficult & challenging for software industry & Academia, since it was started. Software estimation always get affected by some certain factors: uncertainty, level of detail of preparing the project plan, managerial factors, lack of past data, pressure to lower estimation and estimator experience [1]. This paper represents the review of some techniques to estimate the software effort. We cannot straight that one technique is better in providing high level of accuracy than others. All techniques give different level of accuracy depending on data set taken or parameters chosen.

Some techniques which were used in the past are not in use during present time, like SLIM [28], checkpoint [2], Seer [2]. In all the way of work time, many of new advance roads have been suggested for effort estimation like Genetic programming [23], Fuzzy logic [21], Neural Network [29], data mining [18], etc.

II. Review of Literature

The era was started from the expert judgments. But it is only act as pillar when current project & Relevant Past projects are similar. Range of technique step up from COCOMO [28] to AI approaches [2].

A. SLIM

SLIM estimating method was enrooted by Larry Putnam in 1970’s. This method based on Rayleigh function. To drive software estimate linear programming is key work to SLIM [28]. Manpower Buildup Index (MBI) and Productivity Factor (PF) are used to influence the Rayleigh curve [2].

Quantitative Software Management developed three flavors of SLIM, SLIM-Estimate, SLIM-Control and SLIM-Metric. SLIM depends upon the source line of code (SLOC) [2].

B. COCOMO

In 1981 developer Barry Boehm developed COCOMO as constructive cost model [4]. An easy going & understandable model advanced by Barry Boehm could call the effort & time period of project which is bridge on input relating to the size of resulting system. COCOMO is activated for the position of 63 software projects with 15 projects attributes [28]. Simplest form:

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

where,

- \( E \) = estimate effort in man month
- \( a, b \) are the constant

Same developer who developed COCOMO later on developed COCOMO 2.0, as some problems and some misses were found in first version [14]. As software industry was growing tremendously and previous version was not up to need.

C. ESTIMAC

Howard Rubin designed this model to estimate effort at conception stage [4]. Since it is a proprietary model, equation used are not available. ESTIMAC is high level model but doesn’t provide accurate solution [3]. Rubin identified the six critical estimation dimensions: effort hour, staff size, cost, hardware resource requirement, risk, portfolio impact [2]. This model support iterative approach: -

- Input the data.
- Summarize the estimate
- Analysis of data
- Jump to step (i) depending upon result (iv)

D. Function Point

Developer Allan Albrecht developed measurement method called Function point [3] at IBM in 1979. For LOC (line of code) techniques many problems was faced as: lack of universally accepted definition for exactly what line code really is? Other side line of code is language dependence. Function point defines the complexity of software system in terms of functions that system delivers to user. It includes linear combination of five basic software components (input, output, master files, interfaces, inquiries) consider to be low, average, high [3].

Steps for Function point method include: - (i) Determine the type of Function point, (ii) Identify scope, (iii) Identify data function and transaction function, (iv) Determine the value of adjusted factor, (v) Calculate unadjusted factor [18]. In 1990, GC Low and DR. Jeffery also concluded in their paper that Function point method is more consistent then the line of code measure [12]. But there are many criticisms related to Function point method, which are then refined by using SPQR/20 Function point method [13].

E. Fuzzy Logic

Lotfi Zadeh introduced concept of fuzzy which may be a sweeping statement of concept of classical set or crisp set. Theoretically it is collections with un-sharpen boundaries and acknowledged as an addition/extension of classical set theory [21]. The body \( \mu_A(X) \) of an element \( X \) of classical set \( A \), as subset of universe \( X \) is labeled by:

\[ \mu_A(x) = 1, \text{ if } f(X) = A \text{ and } \mu_A(x) = 0, \text{ if } f(X) \neq A \]

I.e. \( X \) is member of set \( A \) if \( \mu_A(x) = 1 \)

The main reason to use Fuzzy logic for effort estimation is that software effort is often uncertain, imprecise and incomplete. Fuzzy logic works with linguistics variables (low, very low, high, and moderate) rather than exact numerical value [6]. In 2000 IDRI, A. ABRIA, and L.Kjiri applied Fuzzy logic to cost drivers of intermediate COCOMO model [22].
use of fuzzy set rather than the classical interval in COCOMO model. In 2004 Zhiwei Xua, Taghi M Khoshgoftaar presented an innovative fuzzy identification method to deal with linguistic data and highlighted that the traditional approaches COCOMO, Function point are inefficient to deal with vague or imprecise data [5]. In 2008 M. Wasif Nisar, Yong-Ji Wang, Manzoor Elahi have done the survey for software development effort estimation using fuzzy logic and conclude that combination of effort techniques provides the better result than the use of individual technique [21].

**F. Genetic programming**

Genetic programming was advanced as a back-up address to deal with general optimization complication with target search spaces [7]. The key process:-

- Achieved at hit-or-miss population of solutions that i.e. family of chromosomes.
- By putting genetic operator to fittest chromosome develop a fresher population from the last one.
- Keep going on step (ii), till you get either health of the boss solution has been assembled or to the point number of generation have been produced.

To improve the software development cost, in 2002, Y Shan, R1 McKay, CJ Lokan utilized the Grammar Guided Genetic programming {GGGP} [23]. According to work, more fitting result arrange by Genetic programming as compare to linear approaches. In 2003, M Leffley, M Shepperd concludes that the Genetic programming may not be considered as the clear winner in case of accuracy level of software estimation rather than other approaches provides more efficient results [24]. But in 2008, SJ Huang, NH Chiu, LW Chen used the integrated approach of Grey Relational Analysis with Genetic Algorithm for effort estimation. This integrated approach draw the more efficient result then the machine learning approaches [8].

**G. Case Based Reasoning**

It is a channel of analogical reasoning where the potential analogies and target is specimen, for example- software project [9]. Prototype of Case base reasoning is strategies are estimation software project effort by analogy [25].

Analogy based method is systematic form of expert judgments especially when experts search for analogies situation, so to level their opinion. The technique binds charactering the project and to do so an estimate stand is in need [25]. Freshly came into view an analogy based estimation if and guaranteed passage, with equivalent efficiency and exactness in algorithm method as in other studies and it is way better in understanding and using in action practically [9]. Certain numbers of steps are required to go through when working on estimating software project effort by analogy [25]:-

- For targeted projects, value of project metrics should be measured or estimated.
- As source analogies, one or more projects should be selected and should search a repository of already done projects for similar one to target projects.
- Use source analogies effort value as an introductory estimate for target project.
- Both target and source projects known metrics value should be weigh for match.
- With the projected answer of difference between target and source, should adjust the effort estimation.

Martin Shepperd and Chris Schofield described the use of analogies for estimated software project effort. They found estimation by analogy is able operate in circumstances, where it is not possible to generate an algorithmic model [10]. In 2011, reference M Azzeh, D Neagu, PI Cowling integrated the analogy based method with fuzzy number [11]. The authors concluded that it is more appropriate to use of integration of these approaches, because Fuzzy logic help to deal with uncertainty in early stages.

**H. Neural Network**

A Neural Network is massively distributed processor made up of simple processing elements called neuron, which model some functionality like human brain [29]. The use of Neural Network offers the some useful properties and capabilities: - Nonlinearity, Adaptivity, Evidential Response, Confidence in decision made. A primary advantage of learning systems is that they are nonparametric; predictive models can be tailored to, the data at a particular site [15].

![Fig. 1: Model of Neural Network](https://example.com/fig1)

Fig. 1, represents the model of neural network. A set of synapses or connecting links is characterized by a weight or strength of its own. A signal xj at the input of synapse j connected to neuron k is multiplied by the synaptic weight wkj. An adder for summing the input signals. An activation function for limiting the amplitude of output of a neuron.

Baskesles concluded that parametric model (COCOMO) may fail to give good estimates because it is based on restricted data [26]. In 2005, N Tadayon compares the three models COCOMO II, Neural Network and expert judgments to state the strength of different estimation techniques [27]. A precise estimation should not only consider the FPs, but should also include various elements related to development environment for its estimation, like Neural Network refines the output through training process [16]. In 2010, Jagannath Singh compares the result of different neural models and draws that cascade ANN gives better result than Feed forward, Elman and recurrent [17]. There are many training algorithm designed for Neural Network [29]. The right choice of algorithm, number of hidden layer neuron, and number of input neuron will result in high accuracy.

**I. Other Techniques**

There are many other techniques such as ordinary least square (OLS) [2], Date mining [18], Bayesian COCOMO II [2] etc. also used for the effort estimation but not discussed in this paper. Table 1 represents the summary of this paper.

**III. Conclusion**

From this we conclude that there are many techniques that used for effort estimation, each one has its own pros and cons. We cannot say, one of the approach provide best path upon other approaches but the hybrid approaches COCOMO with fuzzy [22], neuro-fuzzy [19], Neuro-Gene tic [20] can act as fly-over to achieve better
level of accuracy.

Table 1: Summary of Effort Estimation Approaches with Their Pros and Cons

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Techniques</th>
<th>Key Idea</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>B.</td>
<td>SLIM [2,3]</td>
<td>Based on the Rayleigh Function [28]</td>
<td>1. Many tools available which saves the precious time of estimator.</td>
<td>1. Depend upon the source line of code (SLOC) [4].</td>
</tr>
<tr>
<td>C.</td>
<td>COCOMO [3,4,20]</td>
<td>Predict the effort and duration of project, based on inputs relating to size of system [4].</td>
<td>1. Simple approach.  2. Good accuracy of level when source line code is known.  3. Transparent [5].</td>
<td>1. Language Dependent [3].  2. Based on SLOC  3. Historical data used may not be accurate all the time [5].</td>
</tr>
<tr>
<td>F.</td>
<td>Fuzzy logic [6,7,5,9]</td>
<td>Fuzzy logic is used for effort estimation because software effort is often uncertain, imprecise and incomplete; it deals with linguistic variables (high, very high etc.)</td>
<td>1. can used for complex problem, when mathematical approaches unable to solve problem [21].  2. Property to handle vague or imprecise data [5].</td>
<td>1. Require finer tuning and simulation while developing effort model.  2. Doesn’t always provide accurate result.</td>
</tr>
<tr>
<td>G.</td>
<td>Genetic programming [10,11]</td>
<td>Genetic programming [23] was advanced as a back-up address to deal with general optimization complication with target search spaces [7].</td>
<td>1. Doesn’t need prior knowledge [7].  2. flexible approach  3. Provide Consistent result [23].</td>
<td>1. Doesn’t provide optimum solution for Large space [10].  2. Harder to configure [7].  3. Produce complex model.</td>
</tr>
<tr>
<td>H.</td>
<td>Analogy Based (CBR approach) [14,19,10]</td>
<td>a channel of analogical reasoning where the potential analogies and target are specimen</td>
<td>1. Easy to understand and apply [9].  2. Useful where domain is difficult to model.  3. Potential to mitigate problem of outliers</td>
<td>1. Requires considerably amount of computation [25].</td>
</tr>
</tbody>
</table>

References


