Open access Journal International Journal of Emerging Trends in Science and Technology
IC Value: 76.89 (Index Copernicus) Impact Factor: 2.838 DOI: https://dx.doi.org/10.18535/ijetst/v3i11.08

A Comparison between Various Software Cost Estimation Models

Authors
Sanjali Gupta, Sarthak Tiwari, Himanshu Singh, Ayush Shukla, Himanshi Raghuvanshi
Scope, VIT University, Vellore (T.N.), India

ABSTRACT
Effective cost estimation is one of the most important and complex tasks in development of any software project. The stakeholders require a simple and effective model to enable them to make an efficient estimate of the software development cost. Estimation, basically, means to predict the value and predictions tend to go wrong. The whole process requires a huge amount of data, which is often challenging to collect, hence, making the whole process challenging. In this paper, we try to illustrate various existing cost estimation models proposed over the years and finally present a comparison between them. The paper also tries to pinpoint the various reasons for errors in cost estimation.
Keyword: *COCOMO, SLOC, Function Points, Fuzzy, Neural Net.*

INTRODUCTION

Cost estimation plays a crucial part in software project management. It often helps to determine the final result of the project, whether it will be successful or not. The software cost estimation process starts in the planning phase of SDLC. One can approach by defining all the primary stakeholders of the software and their goals. Then define software requirements, list out the functionalities and prepare various use cases. After which, one can estimate the effort required to implement those use cases. Experienced teams can help make good estimates of the types and quantities of various resources needed.

The main idea behind the process is to make a fair estimate of the size, effort required, time required and the corresponding cost of the project. But most of the industries are unable to make these estimates properly, thus, creating a need for efficient models which can help them estimate the overall project cost. Lack of information or trained staff can lead to inappropriate estimation of the cost. Both, overestimation and underestimation of cost can lead to failure of the project. With constantly increasing costs, it has become a major need to have an effective model which could accurately estimate the project cost.

Effective cost estimation helps in classifying and prioritizing development plans and making the management of the project easier. It helps the developers to know what resources will be required and how will they be used. Moreover, it helps in making a proper business case for the customers by fulfilling their expectation of the estimated cost being same as the actual cost.

The concept of software cost estimation began in 1960s and many cost estimation models have been proposed by various researchers since then. The categories into which these models are divided are Non-algorithmic Models (Expert judgement, Price to win, Analogy based estimation), Algorithmic Models (COCOMO, Linear Models, Functional Point Analysis)and Machine Learning techniques (Neural Networks, Fuzzy Logic). This paper summarizes all these methods and provide a comparative study based upon the effectiveness of these models.

NON-ALGORITHMIC MODELS

These models first compare the project under consideration with the previously done projects by the organization and analyses the information from the most similar projects to make the cost

estimates. Basically, these methods make use of the past experiences.

A. ANALOGY-BASED ESTIMATION

Estimation by Analogy means to compare the project at hand with previously completed projects which are similar to the proposed project and extrapolate the information from those projects to estimate its cost. It has been seen that experts often apply analogic reasoning while making cost estimates. This method is simple and flexible. Since it relies on the historical data, it can help in dealing with poorly understood areas. It can be applied at the initial phases of SDLC and can later be improved upon as more information about the proposed project becomes available.

The process for analogy based estimation goes as follows:

- Identify the requirements of the current project.
- Select the most similar projects completed previously which have their information in the database.
- Extrapolate this data to estimate the cost for the proposed project.

B. EXPERT BASED JUDGEMENT

Experts are assets of any organization as they provide valuable inputs in estimating the cost of software development. An expert can be anyone from a stakeholder to a customer. This technique is one of the most accepted ones. Experts help in providing better estimates as they have extensive experience in similar projects. This method can be used when there is limitation of information resources.[8]

Delphi is the most common methods which work according to this technique. This method relies on a panel of experts who are supposed to answer questionnaires in a number of rounds. After every round, an anonymous summary of the experts' forecasts is provided by a facilitator with the reasons for their judgments. Experts are then asked to start another iteration. This process is repeated unless a consensus is achieved.

C. PRICE TO WIN

In this technique, the cost is estimated considering the best cost to get the project and not on the functionalities or resource availability. It takes into account the customer budget rather than project requirements. Let us consider a project in which a reasonable estimation of the cost is 300 PM but the customer is ready to pay 200 PM only, then according to this technique, the cost is reduced to fit the customer budget that is, 200 PM so as to get the project. But in long run, the technique is not a good method as it usually leads to significant delays in delivery or force the development team to work extra due to improper initial estimates.

D. TOP-DOWN

The top down estimation method also known as macro model, considers effort as a function of size of the project. That is,

$EFFORT = a*SIZE^b$, [7]

where a and b are constants. At first, an overall cost is estimated, the project is then partitioned into various levels and the cost estimation of every level of the project is derived from the global properties of the software project. The overall cost estimation of the project makes it very easy to estimate costs at the start, however, one needs to revise the initial estimates as the project progresses, which leads to delays if the revisions lead to varying results from the earlier estimates. Due to the fact that very little detailed information is available at the start, this method is highly regarded in early cost estimation.

E. BOTTOM-UP

This is the exact opposite of the top down approach. In this method, we first estimate the cost for each and every small components of the project, which is then combined to the cost of overall project. It aims to consolidate the small information available and how they interact to arrive at the overall cost. COCOMO method uses this approach for cost estimation. Although a much consolidated technique, bottom up cannot be applied to projects where much detail is not known

during the start of the project. Trying to apply bottom up in these situations can lead to bad estimations.

ALGORITHMIC MODELS

In these methods, software cost estimation is provided using mathematical equations. A lot of research is done on historical data and inputs such as skill levels, risk assessments, the number of functions to perform, source lines of code etc. in order to arrive at these mathematical equations. These methods have developed a lot of commendable models such as Putnam model, COCOMO model, and function points based models.

A. Putnam Model

Being an empirical effort estimation model, it derived the following software equations based on productivity observations:

Technical constant k = $size * B^{\frac{1}{3}} * T^{\frac{4}{3}}$

Total PM B = $1/T^4 * \left(\frac{size}{k}\right)^3$

T= development time required in years

Size is estimated in LOC. [5]

Here, K is a development environment dependent parameter and is calculated using the historical data collected from projects done in the past.

For understanding K= 1000 is poor, whereas k = 14000 can be termed as excellent.

This method is inversely related to development time and the man-months needed for development can be greatly increased if the development time is decreased.

B. COCOMO Model

Constructive Cost Model or COCOMO model was proposed by Boehm and was simply represented as:

$E = l_1 * (KLOC)^{K}_{2}$

Where, l_1 and l_2 are two application and development environment dependent parameters. Also, E is the MAN- MONTHS required to do the job and KLOC is the estimated number of delivered lines of code for the project (in thousands).

If we consider the qualification and experience of the development team, the characteristics required in the desired software and the software development environment, then the estimates of the COCOMO model can be made yet more accurate.

Software complexity depends on following factors:

- Experience of the team in application development
- Memory efficiency and execution time
- Size of the database required
- Programming language and computer experience of the team
- How capable the analyst and the programmer are
- Reliability
- Software engineering and tools used.[5]

Being a regression model, it depends on the analysis of 63 selected projects and has thousands of delivered source instructions as for its primary input. The primary problems include the following:-

- It basically focuses on the analysis of 63 projects done and can do nothing in case it has some problem outside its training environment. There comes the problem of recalibration from time to time.
- It can detect the cost estimate with accuracy for a product in its early stages of SDLC since its size can't be detected accurately.

Seeing the problems faced by the initial COCOMO model in estimating the cost of software for new life cycles, COCOMO 2.0 was introduced. Its major features were involving function points, object points, source line of code, a tailor-made family of software size model and a non-linear model for software reuse and engineering.

C. Function Point Analysis

Function Point Analysis is a structured technique for problem solving. A function point is the unit to measure software just like an hour is to measure the time. It is estimate of the amount of business functionality provided to user by an information

system. The cost (in dollars or hours) of a single unit is calculated from past projects [2]. Counting the functional points should be scheduled and pre-planned.

The count of distinct format or processing logic types decides the number of function points. Counting of FP can be done by following the below steps:

A linear combination of five basic software components – external inputs and outputs, logic internal files, external inquiries and external interfaces, each being either simple, average or complex, leads to arrival of basic raw function counts.

Multiplication of an adjustment factor to the raw function count leads to the final count. 14 aspects of processing complexity are considered to arrive at this adjustment factor. This multiplication modifies the original function count by +35% or -35%.

MACHINE LEARNING METHODS

All the techniques discussed till now make use of statistical methods for cost estimation and thus, fail to provide reason and strong results. Machine learning methods on other hand, utilize training rules for estimation ad repeat the run cycles. Thus, it can be the appropriate approach as it can increase accuracy of the results.

A. NEURAL NETWORK

Neural networks have a wide application and they have a very interesting use in cost estimation. The process of functioning of neural networks can be utilized for cost estimation easily. Every neuron can represent an activity for the project, and all of the neurons give outputs based on the input they receive. The overall output gives the cost estimation of the entire project. A neural network is useful in the way that it becomes very easy for it to learn, as they modify the weights every time they are used. This leads to better and better estimation as the project tends to completion. Numerous methods exist inside neural networks and back propagation works the best in them.

B. FUZZY LOGIC

The fuzzy logic system works based upon concepts of fuzzy logic which includes fuzzification, defuzzification, fuzzy number etc. The system tries to enact the human behavior by searching for reasons. In cases where it's difficult to come up with a crisp decision, fuzzy logic can be effective in reaching the results.

This technique always tries to support the facts which might be ignored usually. There are 4 stages in this approach:

1. Produce trapezoidal numbers for the linguistic terms.
2. Develop the complexity matrix by producing a new linguistic term.
3. Determine the productivity rate and the attempt for the new linguistic terms.
4. Determine the effort required to complete a task and to compare the existing method.[3]

COCOMO can be implemented using fuzzy logic as shown below in figure 1:

Step (1) fuzzification has been done by scale factors, cost drivers and size.

Step (2) principals of COCOMO are considered.

Step (3) defuzzification is accomplished to find the effort [4].

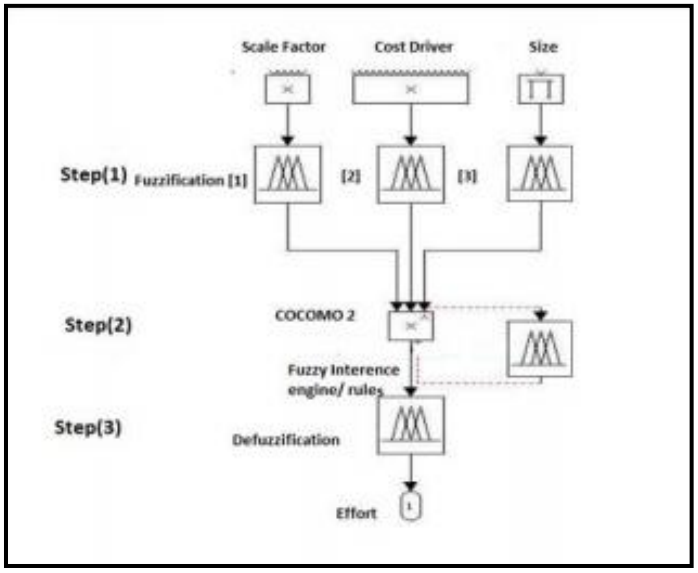

Figure 1 An example of using Fuzzy method

COMPARISON BETWEEN VARIOUS MODELS

Table I. Comparison between different Models [6]

Sr. No.	*Method*	*Type*	*Advantages*	*Disadvantages*
1.	Analogy-Based Estimation	Non-Algorithmic	Work based on the past experiences and knowledge base. No new resources or experts required.	A large amount of information about the project is required and sometimes similar project data might not be available.
2.	Expert based Judgement	Non-Algorithmic	Experts bring experience to the project and can make the process faster.	Cannot be quantified. Difficult to document the factors used by experts. Chances of biased decisions.
3.	Price to Win	Non-Algorithmic	Often wins the contract.	May cause delay of delivery or force the development team to work overtime.
4.	Bottom-Up	Non-Algorithmic	Stable as the estimation errors in the various components might balance out.	Time consuming, might be inaccurate due to shortage of information, not feasible with limited resources.
5.	Top-Down	Non-Algorithmic	Doesn't miss the cost of system level functions, faster and easier	Overlook low-level costs, no justification for decisions, less stable.
6.	Putnam Model	Algorithmic	Efficient for very large projects.	Uncertainty in the software size may result in inaccurate cost estimations.
7.	COCOMO	Algorithmic	Repeatable results can be generated, easily modifying input data, easy refining and customization of formulae, clear results.	Inaccurate cost estimate as the size is uncertain at beginning. A lot of data required, not suitable in practice.
8.	Function Point	Algorithmic	Estimation based on requirements or design specifications, tool independent	Hard to implement, not considered good enough.
9.	Neural Net	Machine Learning	Superior cost estimate, consistency.	Training data required, no standard guidelines.
10.	Fuzzy logic	Machine Learning	Flexible, no training needed.	Difficult to use and understand the concept.

Based upon the above study, we can say that selecting the best method is a difficult task as it is clear that there is no best or worst method for cost estimation rather each one of them has its advantages and disadvantages which are usually complimentary. Non-algorithmic methods take decisions based upon past data and experience whereas Algorithmic methods use mathematical equations for the prediction. At one hand algorithmic methods require some efforts to understand but with enough data provide efficient results, on the other, non-algorithmic methods are easier to implement but sometimes provide inaccurate estimates. Thus, one must use a combination of different models to have all the benefits and get the best results. Moreover, the cost estimates should be compared to actual cost and updated on regular basis throughout the project life cycle.

CONCLUSION

We saw various techniques for cost estimation which have their own advantages and disadvantages. It is very hard to point out a failsafe method, as most of the cost estimation is scenario dependent. Some method are highly accurate but cumbersome to implement, while some only work best on certain known parameters. In order to pick a method for any given project, a proper analysis of the project is much needed. A wrong estimation technique can significantly delay a project, and a right technique can make a project breeze through its deadlines. A detailed analysis of global factors must be made, or else, it becomes very difficult to give correct deadlines. Also, an emphasis on smaller details is necessary, as they can cause delays when they add up together.

ACKNOWLEDGMENT

We would like to take this opportunity to thank VIT University, Vellore for giving us this platform to showcase our technical as well as research skills through this paper. We would also like to express our heartiest gratitude to Dr Ramesh Babu for guiding us and for being a constant support throughout. We would like to thank our friends and parents for being a source of encouragement and motivation. We take this moment to thank all those who have directly or indirectly helped in making this effort successful.

REFERENCES

1. Attarzadeh, I. Siew Hock Ow,"Improving the accuracy of software cost estimation model based on a new fuzzylogic model",World Applied science journal 8(2):117-184,2010-10-2.
2. Vahid Khatibi, Dayang N. A. Jawawi " Software Cost Estimation Methods: A Review" Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1 ,2010-11
3. Narendra Sharma, Aman Bajpai, Mr. Ratnesh Litoriya, The International Journal of Computer Science & Applications (TIJCSA) ISSN – 2278-1080, Vol. 1 No.3 May 2012.
4. Roberto Meli, Luca Santillo "FUNCTION POINT ESTIMATION METHODS: A COMPARATIVE OVERVIEW" Data Processing Organization, http://web.tin.it/dpo.
5. Rekha Tripathi, Dr. P. K. Rai, "Comparative Study of Software Cost Estimation Techniques" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 6, Issue 1, January 2016 ISSN: 2277 128X.
6. Narendra Sharma, Aman Bajpai, Mr. Ratnesh Litoriya "A comparison of software cost estimation methods:A Survey " The International Journal of Computer Science & Applications (TIJCSA), Volume 1, No. 3, May 2012 ISSN – 2278-1080.
7. Syeda Binish Zahra, Mohsin Nazir "A Review of Comparison among Software Estimation Techniques", http://bujict.bimcs.edu.pk/wp-content/uploads/2013/11/1999_4974_12_6.pdf.
8. Jyoti Rani, Rachna Behel "Comparison of Cost Estimation Technique" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014 ISSN: 2277 128X.
9. Ziauddin, Shahid Kamal, Shafiullah khan, Jamal Abdul Nasir "A Fuzzy Logic Based Software Cost Estimation Model" International Journal of Software Engineering and Its Applications Vol. 7, No. 2, March, 2013.