International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 3 Issue 5, August 2019 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Review on Algorithmic and Non-Algorithmic Software Cost Estimation Techniques

Pa Pa Win, War War Myint, Hlaing Phyu Phyu Mon, Seint Wint Thu

Faculty of Information Science, University of Computer Studies, Meiktila, Myanmar

***How to cite this paper**:* Pa Pa Win | War War Myint | Hlaing Phyu Phyu Mon | Seint Wint Thu "Review on Algorithmic and Non-Algorithmic Software Cost Estimation Techniques" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-3 | Issue-5, August 2019, pp.890-895, **https://doi.org/10.31142/ijtsrd26511**

IJTSRD26511

ABSTRACT

Effective software cost estimation is the most challenging and important activities in software development. Developers want a simple and accurate method of efforts estimation. Estimation of the cost before starting of work is a prediction and prediction always not accurate. Software effort estimation is a very critical task in the software engineering and to control quality and efficiency a suitable estimation technique is crucial. This paper gives a review of various available software effort estimation methods, mainly focus on the algorithmic model and non – algorithmic model. These existing methods for software cost estimation are illustrated and their aspect will be discussed. No single technique is best for all situations, and thus a careful comparison of the results of several approaches is most likely to produce realistic estimation. This paper provides a detailed overview of existing software cost estimation models and techniques. This paper presents the strength and weakness of various cost estimation methods. This paper focuses on some of the relevant reasons that cause inaccurate estimation.

KEYWORDS: *Software cost estimation; Software Effort Estimation; algorithmic; non-algorithmic*

1. INTRODUCTION

Software cost estimation plays an important role in software engineering, often determining the success or failure of contract negotiation and project execution. The main goal of software cost and effort estimation is to scientifically estimate the required workload and its corresponding costs in the life cycle of the software system.

Accurate cost estimates activity is critical to developers and customers. Accurate cost estimation is important for the following reasons [1][2]:

- ➢ It can be used to classify and prioritize development projects with respect to the complete business plan.
- ➢ It can help to find out what resources to commit to the project and how well these resources will be used
- ➢ It can help to assess the impact of changes and supporting for preplanning. Projects can be easier to manage and control when resources are matched to real needs.
- ➢ Customers expect accurate development costs to be in line with estimated costs.

Software cost estimation activity historically has been a major difficulty in software development. Several reasons have been identified that affects the cost estimation process such as [15]

- ➢ Cost of software development estimate is difficult. The first steps in the estimate are to understand and define the system to be estimated.
- ➢ A cost estimate done early in the project life cycle is generally based on less precise inputs and less detailed design specifications.
- ➢ Software development involves many interrelated factors, which affect development effort and productivity, and whose relationships are not well understood.
- ➢ Incomplete, inaccurate or inconsistent historical database of cost measurement.
- ➢ Lack of trained estimators.
- ➢ Software is intangible, invisible, and intractable so it is more difficult to understand and estimate a product or process that cannot be seen and touched.

2. BACKGROUND

Software project failure has been an important subject in the last decade. Software projects usually don't fail during the implementation and the most project fails during the implementation and most project fails are related to the planning and estimation steps despite going to overtime and cost, approximately between 30% and 40% of the software projects are completed and the other fail(Molokken and Jorgenson,2003). During the last decade, several studies have been done in term of finding the reason for the software project failure. Galorath and Evans(2006) performed an intensive search between 2100 internet site and found 5500 reasons for software project failures. Among the found reasons, insufficient requirements engineering, poor planning the projects, suddenly decision at the early stages of the project and inaccurate estimations were the most important reasons. The other researches regarding the reason of project fail to show that inaccurate estimation is the root factor of fail in the most software project fails(Jones,2007; Jorgensen, 2003). Despite the indicated statistics may be pessimistic, inaccurate estimation is a real problem in the software product's world which should be solved. Presenting the efficient techniques and reliable models seems required regarding the mentioned problems.

The conditions of the software projects are not stable and the state is continuously changing so several methods should be presented for estimation that each method is appropriate for a special project[9].

3. ESTIMATION TECHNIQUES

Generally, there are many methods for software cost estimation, which are divided into four categories: Algorithmic, Non-Algorithmic, Parametric and Machine learning Models. All categories are required for performing an accurate estimation. If the requirements are known better, their performance will be better. In this section overview of four estimation models are discussed.

A. Algorithmic Models These models work based on the special algorithm. These model usually need data at first and make result by using the mathematical relation. Nowadays, many software estimation methods use these models. Algorithm models are classified into some different models. Each algorithmic model uses an equation to do the estimation: **Effort=f(x1,x2,……….,xn)** where (x1……xn) is the vector of the cost factor. The differences among the existing algorithmic methods are related to choosing the cost factor and function.

B. Non-Algorithmic Model 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 to the underestimate 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 accessing because these methods are more popular than the other None Algorithmic methods and many papers about their uses have been published in recent years.

C. MACHINE LEARNING METHODS Most techniques about cost estimation use statistical methods, which are not able to present reason and strong results. This approach could be appropriate because they can increase the accuracy of estimation by training rules of estimation and repeating the run cycles. It is categorized into two main methods, neural networks and fuzzy methods which are :

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. Backpropagation 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 are focused on modeling the Cocomo method, for example in [5] a neural network has been proposed for estimation of software cost according to the following figure. Figure (1) shows the layers, inputs and the transfer function of the mentioned neural network. Scale Factors(SF) and effort multipliers(EM) are an input of the neural network, pi and qj are respectively the weight of SFs and EMs.[6]

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

Stage 1**:** produce trapezoidal numbers for the linguistic terms.
Stage 2**:** develop the complexity matrix by producing a new linguistic term.
Stage 3**:** determine the productivity rate and the attempt for the new linguistic terms.
Stage 4: determine the effort required to complete a task and to compare the existing method. For example in [3] COCOMO technique has been implemented by using the fuzzy method.

Fig (2) displays all following mentioned steps.
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 [6].

4. The parametric-estimating method is a mathematical representation of cost estimating relationships that provide a logical and predictable correlation between the cost as a dependent variable and the cost estimating factors as the independent variables associated with the project being estimated (Duverlie and Clastelain, 1999; Dysert, 2003; International Society of Parametric Analysis [ISPA], 2008). Parametric models are developed by applying regression analysis to historical project data (obtained from past projects).

4. ALGORITHMIC METHODS

These methods are designed to provide some mathematical equations to perform software cost estimation. These mathematical equations are based on research and historical data and use some inputs for example Source Lines of Code, a number of functions to perform, and some cost drivers like as language, design methodology, skill-levels, risk assessments, etc. Algorithmic methods developed many models such as COCOMO models, Putnam model, and function points based models [3].

A. COCOMO Model

One very widely used algorithmic cost estimation model is the Constructive Cost Model (COCOMO) which was proposed by Boehm [4]. The basic COCOMO model has a simple form: MAN-MONTHS = K1* (KDLOC) K2 Where K1 and K2 are two parameters which are dependent on the application and development environment. Estimates from the basic COCOMO model can be made more accurate by taking into account other factors concerning the required characteristics of the software to be developed, the qualification and experience of the development team, and the software development environment. The complexity of the software has the following factor:

Reliability
- Database size
- Required efficiency for memory and execution time
- The capability of an analyst and programmer
- Team experience in the application area
- Experience of the team in the programming language and computer
- Use of software engineering and tools

Cost models generally use some cost indicator for estimation and notice to all specification artifacts and activities. COCOMO 81(constructive cost model) proposed by Barry Bohem is the most popular method which categorized in algorithmic methods. This method uses some equations and parameters, which have been derived from previous experiences about software projects for estimation. There are three forms of the constructive cost model: 1. Basic COCOMO which gives an initial rough estimate of man-months and development time. 2. Intermediate COCOMO which gives a more detailed estimate for small to medium size projects. 3. Complete COCOMO which gives a more detailed estimate for large projects.

There are three modes of development.
1. Organic mode
* Relatively small simple software projects.
* Small team with good application experience work to a set of less than rigid requirements.
Similar to previously developed projects.
* Relatively small and require little innovation.
2. Semi-Detached mode
* Intermediate(in size and complexity) software projects in which team with mixed experience level must meet a mix of rigid and less than rigid requirements.
3. An embedded mode Software project that must be developed within a set of tight hardware software and operational constraints.

BASIC COCOMO
Basic COCMO is an empirical estimation for estimating effort, cost, and schedule for software projects. It was derived from the large data set from 63 software projects ranging in size from 200 to 100000 lines of code, and programming languages ranging from assembly to PL/I. This data was analyzed to discover a set of formula that was the best fit to the observation these formula link the size of the system. In COCOMO 81 effort is be calculated as PM= a*Size^6ΠEmi i=1 to 15 Where a & b are the domain constant in the model. It contains 15 effort multipliers. This estimation scheme accounts the experience and data of the past projects which is extremely complex to understand and apply the same.

INTERMEDIATE COCMO
In 1997, an enhanced scheme for estimating the effort for software development activities, which is called as COCOMO II. COCOMO II has some special features which distinguish for another one the uses of this method are very hidden and its result usually accurate. In COCOMO II effort requirements can be calculated as PM=a* size ^B*ΠEmi i=1 to 17 Where E=B+0.01*ΣSFj j=1 to 5 COCOMO II is associated with 31 factors LOC measures as the estimation variable, 17 cost drivers, 5 scale factor, 3 adaptation percentage of modification, 3 adaptation cost drives and requirements volatility.

The Detailed COCOMO Model
The detailed COCOMO model differs from the intermediate model is only one major aspect. The detailed model uses different effort multipliers for each phase of a project. These phase-dependent effort multipliers yield better estimates than the intermediate model[11].

B. Putnam Model
The Putnam model is an empirical effort estimation model. Putnam used his productivity levels observations to derive the software equation: Technical constant C= size * B1/3 * T4/3 Total PM B=1/T4 *(size/C)3 T = Required Development Time in years Size = estimated in LOC Where: C = parameter dependent on the development environment and is determined on the basis of historical data of the past projects. Rating for C=2,000 is poor C=8000 is good C=12,000 is excellent. This model is very sensitive to the development time, decreasing the development time can greatly increase the person-months needed for development [6][12]. One significant problem with this model is that it is based on knowing, or being able to estimate accurately, the size of the software to be developed. There is often great uncertainty in the software size. It may result in the inaccuracy of estimation.

C. Function Point Analysis
The Function Point Analysis is a method of quantifying the size and complexity of a software system in terms of the functions that the systems deliver to the user. A number of proprietary models for cost estimation have adapted to this type of approach, like as ESTIMACS and SPQR/20. This is a measurement which is based on the functionality of the program. It was first introduced by Albrecht [1]. The total number of FP depends on the counts of distinct in terms of format or processing logic types. Following two steps in counting function points:

Counting to the user functions: The raw function counts are arrived at by considering a linear combination of five basic software components. These components are external inputs, external outputs, external inquiries, logic internal files, and external interfaces, each at one of three complexity levels: simple, average or complex. The sum of these numbers, weighted according to the complexity level, is the number of FC.

Adjusting for environmental processing complexity: The final function points are arrived at by multiplying function count by an adjustment factor that is determined by considering 14 aspects of processing complexity. This adjustment factor allows the function count to be modified by at most 35% or -35%.

D. Linear Models Commonly these models have the simple structure and trace a clear equation as below:
EFFORT = a0 +Σ n ai xi i=0 Where, a1, a2 ,an are selected according to the information of project, only allowed values for xi are -1, 0, +1.[16]

E. Seer-sem Models
This model has been proposed in 1980 by Galorath Inc[9]. Most parameters in seer-sem are commercial and, business projects usually use seer-sem as their main cost estimation method. Size of the software is the most important feature in seer-sem method and a parameter namely Se is defined as effective size. Se is computed by determining the five indicators: new size, existing size, ramp redesign and retest as below:
Se=new size + existing Size(0.4 redesign + 0.25 reimp + 0.35 retest) After computing the Se the estimated effort is calculated as below: **EFFORT= td = D - 0.2 × (Se / Cte)0.4** Where D = relevant to the staffing aspects It is determined based on the complexity degree in staffs structure. Cte is computed according to the productivity and efficiency of the project method. It is used widely in a commercial project. [7]

5. NON ALGORITHMIC METHODS

Non Algorithmic methods use some information about the previous projects which are similar to the underestimate project is required and usually cost estimation process in these methods is done according to the analysis of the previous datasets.

A. Expert Judgment Method Expert judgment techniques involve consulting with cost estimation expert or a group of the estimation experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. It is the most used methods for cost estimation. Most companies used expert judgment method for generating the cost of the product [4] [12]. This method using the following estimating steps: a. Project leader presents each expert with a specification and an estimation form. b. The experts fill out forms anonymously. c. Project leader calls a group meeting in which the experts discuss cost estimation issues with the project leader and each other. d. Project leader prepares and distributes a summary of the cost estimation on an iteration form. e. Again experts fill out forms, anonymously. f. Steps d and step e are iterated for as many rounds as appropriate.

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 a limitation in finding data and gathering requirements. Consultation is the basic issue in this method. One of the most common methods which work according to this technique in Delphi. Delphi arranges an especial meeting among the project experts and tries to achieve the true information about the project from there debates. Delphi includes some steps:

- ➢ The coordinator gives an estimation from each expert.
- ➢ Each expert presents his own estimation
- ➢ The coordinator gathers all forms and sums up them on a form and ask experts to start another iteration.
- ➢ Steps (ii-iii) are repeated until approval is gained.

A. Estimating by Analogy

Cost estimating by analogy means comparing the proposed project to previously completed similar project where the project development information is known. Actual data from the completed projects are extrapolated to cost estimate the proposed project. Analogy method can be used either at the system level or at the component level [12]. This method using the following estimating steps: a. Find out the necessary characteristics of the proposed project. b. Choose the most similar completed projects whose characteristics have been stored in the historical database. c. Find the estimate for the proposed project from the most similar completed project by analogy.

It means creating estimates for new projects by comparing the new projects to similar projects from the part. 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. As the Algorithmic technique have a disadvantage of the need to calibrate the model. So the alternative approach is 'Analogy by Estimation'. Estimation based on analogy is accomplished at the total system levels and subsystem levels. By accessing the result of previous actual projects. We can estimate the cost and effort of a similar project. The steps of this method are considered as 1. Choosing analogy 2. Investigating similarities and differences 3. Examining of analogy quality. 4. providing the estimation.

B. Parkinson's Law

Using Parkinson‟s Law *"*Work expands to fill the available volume"[8], the cost is determined by the available resources rather than based on an objective assessment., If the software has to be delivered in 20 months and 4 people are available, the effort is estimated to be 80 PM. Although it sometimes gives a good estimation, this method is not recommended as it may provide very unrealistic estimates. Parkinson‟s Law does not promote good software engineering practice [2]. ***E. Price-to-win*** The cost is estimated to be the best price to win the project. The cost estimation is based on the customer's budget instead of the software functionality. For example, if a reasonable estimation for a project costs 100 PM but the customer can only effort 60 PM. It is common that the estimator is asked to modify the estimation to fit 60 PM effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the estimation team to work overtime[10][2].

C. Top-Down Estimating Method

The top-down estimating method is known as Macro Model. Using this estimating method, overall cost estimation for the project is derived from the global properties of the software project, and then the project is partitioned into various low-level mechanism or components. The method of using this approach is the Putnam model. The top-down method is more applicable to early cost estimation when only global properties are known. In the early phase of the software cost estimation, top-down is very useful because there is no detailed information available [4].

D. Bottom-up Estimating Method

Using a bottom-up cost estimating method, the cost of each software component is estimated and then combines the results to arrive at an estimated cost of the overall project. The bottom-up method aims at constructing the estimate of a system from the knowledge accumulated about the small software components and their interactions. The method of using this approach is COCOMO's detailed model [4].

6. The Strength and Weakness of Algorithmic and Non-Algorithmic Software Cost Estimation Techniques

According to the study of software cost estimation techniques, it is evident that no one method is necessarily better or worse than the other, in fact, their advantage and disadvantage are often complementary to each other. The algorithmic methods are based on mathematics and some experimental equations. They are usually hard to learn and they need much data about the current project state. But if enough data is available, these methods present reliable results. On the other hand for non-algorithmic methods, it is necessary to have enough information about the similar type of previous projects, because these methods perform the cost estimation by analysis of the historical data. They are easy to learn because they follow human behavior. According to the estimation experience, it is recommended that a combination of models and expert judgment estimation methods are useful to get reliable, accurate cost estimation for software development. We should use expert judgment method or analogy method for known projects and projects parts if the similarities of them can be got since it is fast and under this circumstance, reliable. For large, lesser-known projects, it is better to use algorithmic methods. In this case, many researchers recommend the estimation models that do not require a source line of code as an input. If we approach cost estimation by parts, we may use expert judgment for some known parts. In this way, we can take advantage of both the rigor of models and the speed of expert judgment method or analogy. Because the advantages and disadvantages of each technique are complementary, a combination will reduce the weakness of anyone technique, augment their individual strengths and help to cross-check one technique against another.

Table1. The Strength of Algorithm and Non-Algorithmic Software Cost Estimation Techniques

Sr. No	Method	Type	Strength
1	COCOMO	Algorithmic	Clear results, it's very common
2	LOC	Algorithmic	Very easy in implementation to estimate the size of the software
3	Putnam model	Algorithmic	A Probabilistic model, it"s used in a very large project
4	Seer-Sem model	Algorithmic	Used in very large projects
5	Linear model	Algorithmic	It"s the best method of prediction using a linear regression technique
6	Analogy	Non-Algorithmic	Works based on actual experience and especial expert is not important
7	Expert judgment	Non-Algorithmic	Fast prediction, adapt for special projects
8	Parkinson	Non-Algorithmic	Correlates with some experience
9	Price to win	Non-Algorithmic	It"s often gets the contract
10	Top-down	Non-Algorithmic	Requires minimal project detail, usually faster and easier to implement and system-level focus
11	Bottom-down	Non-Algorithmic	More detailed basis, more stable and encourage individual commitment

Table2. The Weakness of Algorithm and Non-Algorithmic Software Cost Estimation Techniques

Sr. No	Method	Type	Weakness
1	COCOMO	Algorithmic	A lot of data is required, It is not suitable for any project
3	LOC	Algorithmic	Prediction of the line is tough in the early stages, not good for a very large project and language-dependent.
4	Putnam model	Algorithmic	For only use large projects
5	Seer-Sem model	Algorithmic	it"s required 50 input parameters which are increases the complexity and uncertainty
6	Linear model	Algorithmic	Little difference between actual and predicted results and error is also needed to calculate.
7	Analogy	Non-Algorithmic	Much information about past projects is required in some situations there are no similar project
8	Expert judgment	Non-Algorithmic	Success depends on expert, usually is done incomplete
9	Parkinson	Non-Algorithmic	Reinforces poor practice
10	Price to win	Non-Algorithmic	Generally produces large overruns
11	Top-down	Non-Algorithmic	Less detailed basis and less stable
12	Bottom -down	Non-Algorithmic	May overlook system-level costs, requires more effort, a lot of time-consuming

7. ACKNOWLEDGMENTS

I would like to express my special thanks to all my teachers who gave me their time and guidance, and all my friends who helped in the task of developing this paper. Finally, I would like especially to thank my parents for their continuous support and encouragement throughout my whole life.

8. CONCLUSION

In this paper, we have discussed a comparative study of different types of software cost estimation techniques and also described the advantages and disadvantages of these techniques. This paper presents some of the relevant reasons that cause inaccurate estimation. To produce a meaningful and reliable cost estimate, we must improve our understanding of software project attributes and their causal relationships. It has been seen that all cost estimation methods are specific for some specific type of projects. It is very difficult to decide which method is better than to all other methods because every method or model has its own significance or importance. Finding the most important reason for the software project failure has been the object of many researchers in the last decade. According to the result of this paper, the root causes for software project failure is inaccurate estimation in stages of the project. To decrease the project failures software project managers are used to select the best estimation method based on the different conditions and status of the project and also describe comparing the estimation technique. There is no estimation method which could present the best estimates in all various situation and technique can be suitable in the special project. To improve the performance of the existing method and introducing the new methods for estimation based on today's software project requirements can be the future works in this area. The future work is to study the new software cost estimation technique that can help us to easily understand the software cost estimation process.

References

[1] Leungh, Zhangf,‗"Software cost estimation" in Handbook of software engineering and knowledge engineering,, (World Scientific Pub. Co, River Edge, NJ, 2001)

[2] Sweta Kumari and Shashank Pushkar," Performance Analysis of the Software Cost Estimation Methods", International Journal of Advanced Computer Science and Applications, Vol. 3, 2013.

[3] Oscar Marbán, Antonio de Amescua, Juan J. Cuadrado, Luis García "A cost model to estimate the effort of data mining projects", Universidad Carlos III de Madri (UC3M), Volume33, Issue 1, pp.133-150, March, 2008 .

[4] B. W. Boehm", Software Engineering Economics" Prentice-Hall, Englewood Cliffs, NJ, USA, 1981.

[5] Attarzadeh, I. Siew Hock Ow, "Proposing a New Software Cost Estimation Model Based on Artificial Neural Networks", IEEE International Conference on Computer Engineering and Technology (ICCET), Volume: 3, Page(s): V3-487 - V3-491 2010.

[6] Sikka, G., A. Kaur, et al. " Estimating function points: Using machine learning and regression models". Education Technology and Computer (ICETC), 2nd International Conference on, 2010.

[7] Li, J,J. Lin, et al. " Development of the Decision Support System for Software Project cost Estimation Information Science and Engineering ". 2008. ISIS '08. International Symposium on, 2008.

[8] G.N. Parkinson" Parkinson„s Law and Other Studies in Administration" Houghton-Mifflin, Boston, 1957.

[9] Galorath, D. D., & Evans, M. W. " Software sizing, estimation, and risk management: When performance is measured performance improves". Boca Raton, FL: Auerbach,2006.

[10] Maged A. Yahya, Rodina Ahmad, Sai Peck Lee, " Effects of Software Process Maturity on COCOMO II's Effort Estimation from CMMI perspective", 978-1-4244-2379-8/08 IEEE (c),2008

[11] Leungh, Zhangf,‗" Software cost estimation" in Handbook of software engineering and knowledge engineering,, (World Scientific Pub. Co, River Edge, NJ, 2001)

[12] Yahya, M. A., R. Ahmad et al. " Effects of software process maturity on COCOMO II&#x2019;s effort estimation from CMMI perspective ". Research Innovation and vision for the future, RIVF. IEEE International Conference on, 2008.