Software Cost Estimation using Algorithmic Model and Non-Algorithmic Model a Review

L.R. Nerkar
UG Student
BVCOE and RI,Nasik

P.M. Yawalkar
Associate Professor
MET IOE,BKC,Nasik

ABSTRACT
Software Project Estimation is the process of computing the effort & cost required to develop software. Software cost estimation is one of the most important factor in software project management. This review paper provides a general overview of software cost estimation techniques using algorithmic and non-algorithmic model. The techniques such as COCOMO model, Putnam Model, Function-Point based Model, expert judgment, estimation by analogy, Parkinson's Law and pricing to win. Each estimation technique has its own pros and cons. Also, it provides overview of Hybrid Model. The main purpose of this paper is to explain all the existing techniques of software cost estimation. Actually, it is true that, only one technique is not best for all conditions to generate the realistic estimates.

Keywords
Cost Estimation Technique, Realistic Estimates, Software Estimation, WebCost.

1. INTRODUCTION
In Software Project Management System, the most critical activity is estimating the software cost and effort, because when we estimate the software project for cost and effort, there is neither any background nor any previous experience about the project. Therefore predicting the cost as well as effort is complicated. If the requirements are clear and final, then the software estimation process is easy. But if the requirements are change continuously, then software estimation process is complicated. To predict the accurate estimation is difficult for developer as well as customer.

The main purpose of this paper is to describe the cost estimation technique including algorithmic and non-algorithmic model and also describe the Hybrid model such as WebCost model. The full paper is arranged in three sections, first gives overview of estimation techniques, second describes the techniques including algorithmic, non-algorithmic model and Hybrid Model and last is analysis of various models.

2. ESTIMATION TECHNIQUE
Today's, there are so many techniques are available for estimating the software cost. Estimation technique is divided into two models such as algorithmic and non-algorithmic model. Both model is developed for performing the accurate estimation. In this paper, some methods are described including both models.

2.1 Algorithmic Model
Cost estimation by using the algorithmic cost model is based on mathematical formulas. In this model to estimate the cost based on size of project, type of software, software team, software attributes and process being used, many estimation techniques are used. Using the algorithmic model many models have been developed, such as COCOMO Model, Putnam Model and Function points based model. Each algorithmic model uses the mathematical equation:

Effort=f(x1, x2, x3, ------, xn)

Where, x1, x2, x3, ----xn are the cost factors.

Following are some of the models are used under algorithmic model:

2.1.1 COCOMO Model
One very widely used algorithmic software cost model is the Constructive Cost Model (COCOMO). COCOMO 81 Model is proposed by Barry Boehm in 1981. Nowadays the COCOMO-II model is used in which effort estimation is based on Person-Month (PM) in software projects.

This model uses the function point or line of code as the size metrics and composes of 5 scale factor and 17 multipliers. The 5 scale factor are rated on a six-point scale from very low to extra high (5 to 0)[1]. After assigning the rating values add them, divide them by 100 and add the result to 1.01 to get the exponent that should be used [1].

Table 1. Scale Factor

Scale Factor	Explanation
Precedentedness	Reflects the previous experience of the similar project. No previous experience means very low, organization is familiar with this type of project means very high.
Development Flexibility	Reflects the degree of flexibility in the development process. Rating very low means prescribed process is used. and client sets the only general goals in case of rating extra high.
Architecture/risk resolution	Reflects the extent of risk analysis carried out. Little analysis means very low and a complete and thorough risk analysis means extra-high.
Team cohesion	Reflects how well known about team members & work together. Rating very low as very difficult interactions and rating extra-high when effective team and no any communication problems.

Suppose, any one organization is taking a project & it has low previous experience of such type of project. Also project client not defined the process being used and has not allowed time in the project schedule for significant risk analysis. A new development team must be put together to implement this system.[1] In that cases, possible values for the ratings used in exponent calculation are :

- Precedentedness : Rated low for new project for organization.(Rating : 4)
- Development Flexibility: Rated very high when no client involvement. (Rating : 1)
- Architecture/risk Resolution: Rated very low while no risk analysis carried out. (Rating : 1)
- Team Cohesion: New team so no information. Hence rated as nominal. (Rating : 3)
- Process Maturity: Rated nominal for some process control in place. (Rating : 3)

The sum of above rating value is 16. So calculate the exponent by adding 0.16 to 1.01, getting a value of 1.17.

The following Table2 shows the cost drivers that are used to adjust the initial estimates and create multiplier in the post-architecture model fall into four classes [1]:

- Product Attributes: these attributes are concerned with required features for developing software product.
- Computer Attributes: these are the restriction enforced on the software by the hardware platform.
- Personal Attributes: these are multipliers that take the experience and capabilities of the people working on the project into account.
- Project Attributes: these attributes are related to specific features of software development project.

Table 2. Project Cost Driver [1]

Attribute	Type	Description
CPLX	Product	Complexity of system modules
DOCU	Product	Required Extent of documentation
DATA	Product	Database size is used
RELY	Product	Reliability of system required
RUSE	Product	Reusable components percentage.
TIME	Computer	Execution time constraint
PEXP	Personnel	Programmer experience in project domain
PVOL	Computer	Volatility of development platform
STOR	Computer	Memory constraints
ACAP	Personnel	Capability of project analysts
SITE	Project	Extent of multisite working and quality of inter-site communications
PCAP	Personnel	Capability of Programmer
AEXP	Personnel	Analyst experience in project domain
LTEX	Personnel	Tool and Language experience
PCON	Personnel	Personnel continuity
TOOL	Project	Use of Software tools
SCED	Project	Development schedule compression

2.1.2 Putnam's Model

This model is proposed by Putnam's (Kemerer, 2008). This model was developed on the basis of manpower distribution and the research of many software projects.

The main equation for Putnam's Model is [2]:

$$S = E*(Effort)^{1/3}td^{4/3} \quad (1)$$

Where, E is the environment factor that describes the environment ability, td is the time of delivery. Effort and S are expressed by person-year and line of code respectively [2].

Putnam's Model also presented other equation for effort:

$$Effort = D_0 \times td^3 \quad (2)$$

Where, D_0 is the manpower build-up factor, varies from 8 to 27 means it varies from new software to rebuilt software. By combining equations 1 and 2, the final equation is [2]:

$$Effort = (D_0^{4/7} * E^{-9/7}) * S^{9/7} \quad (3)$$

$$t_d = (D_0^{-1/7} * E^{-3/7}) * S^{3/7} \quad (4)$$

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

2.1.3 Function-Point based Model

In 1983, the Albrecht presented a metric known as a Function Point Metric for the purpose of to measure the functionality of project. In this model, the estimation can be done using the following five factors [2]:

- User Inputs
- User Outputs
- Logic Files
- Inquiries
- Interfaces

For a simple function have less complexity, while complex function have high complexity. Complexity is measure in between 1, 2, and 3 for simple, medium and complex degree respectively.

Also for each factor the weight can be define in between 3 to 15.

2.2 Non-Algorithmic Model

In non-algorithmic model, the estimation can be done by using the previous projects previous experiences which is similar to the under estimate project. In this paper four non-algorithmic techniques are described.

2.2.1 Analogy Technique

This technique requires previous existing software project similar to proposed software project.

In this technique, the cost is estimated by comparison between the existing software project and proposed software project. The estimation by analogy can be done at either the "Component Level" or at "Subsystem Level". The Component level means, consider the all cost components of the system. While subsystem level means, providing a more detailed information about the similarities and differences between the existing project and proposed project.

The following are the steps used in analogy technique:

- Find out the characteristics of proposed software project.

- Selecting the existing projects whose characteristics is related to proposed projects.
- By using analogy technique derive the estimation of proposed project from existing similar project.

In short, the Analogy technique is a technique that compares the feature of existing project to proposed project, and then estimates the cost by using analogy.

2.2.2 Expert Judgment

In Expert Judgment technique, to estimate the cost by getting the suggestions from the experts who have experience in similar projects. This technique is very similar to the Delphi Technique.

In Delphi Technique, Some experts are presents with its coordinator. When expert designs the specification then coordinator records this specification in the form of estimates. Every expert fills the form separately without discussing any other. If in case any question arises to expert then expert can ask only coordinator. After recording all estimates, the coordinator prepares the summary report. These tasks repeat again while the specification for estimation is not finalized. Before finalizing the estimation, a meeting is arranged between the coordinators and experts regarding the estimation issue.

2.2.3 Parkinson's Law

The Parkinson's Law state that , "the Work expands to fill the available completion time. " i.e. only by using available material like hardware, software or any other resources like electricity or space etc, to compute the cost. E.g., if the customer requirement is to complete the software within 10 months, and only 4 peoples are available. In this case the cost is estimated to be 10 *4 means 40 person/months. Many times this method gives good estimation. But this method is not better for good software engineering practices. This method may provide unrealistic estimation.

2.2.4 Pricing to Win

This technique fully depends on customer budget rather than the functionality of software. In this technique, consider only the customer budget, and not how many persons are required for developing the software or any other resources for estimation. E.g., if the customer can spent 40 person/month, but actual effort is 60 person/month. Then estimator is asked to modify in estimation and fit in 40 person/months. Again this technique is not better for good software engineering practices.

2.3 Hybrid Model

Using the combination of Algorithmic and Non- Algorithmic Model, Z.B. Mansor et al developed a Hybrid Model named as WebCost Model. This model mainly focuses on the accuracy and efficiency of Cost Estimation for Web-Based application. Hence to provide more accuracy the COCOMO II model is used because more variables are considered in this model. The WebCost Model is suitable for everyone especially the Software Project Managers, Software Practitioners or Software Engineering Students.

2.3.1 WebCost Model

The WebCost model is a combination of expert judgment and COCOMO II technique to estimate the cost of web-based application. Parameters such as project type, project size, function point, cost adjustment, reuse, cost-driven and report are involved.[7]

These parameters are constructed from COCOMO II model. The experts give input according to their knowledge and experiences in the application area by linking the efforts and features of the past project. The relationship between COCOMO II and Expert Judgment is shown in Fig 1.

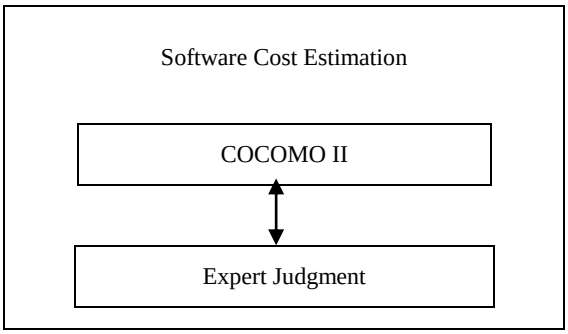

Fig 1: Relationship between COCOMO II and Expert Judgment [7]

In this Hybrid Model, both the techniques of estimation play important role as COCOMO II is used in providing a standard variables as already defined and Expert Judgment technique gives input to the variables provided in COCOMO II.

3. ANALYSIS OF ALGORITHMIC AND NON ALGORITHMIC MODEL

Type	Method	Pros	Cons
Algorithmic Model	COCOMO Model	Clear results	Much data is required,
	Putnam's Model	Fast Result.	Depends on Research
	Function-Point Based Model	Language free, Its results are better than SLOC	Mechanization is hard to do , quality of output are not considered
Non-Algorithmic Model	Analogy Technique	Estimation Result is Clear.	Required previous existing projects.
	Expert Judgment	Fast Result.	Depends on the Experts who have experience in similar projects.
	Parkinson's Law	Simple to predict	Unrealistic estimation.
	Pricing to Win	Often win the Contract.	Unrealistic estimation.

4. COMPARISION WITH EXISTING TOOL

	COCOMO II	WebCost
Estimation Technique	COCOMO II	COCOMO II & Expert Judgment
Architecture	Stand alone	Stand alone
Project Size Parameter	Source Line of Code	Function Point Based source Line of Code and judge by expert
Include Reuse Parameter	Yes	Yes

5. CONCLUSION

Researchers developed number of techniques for estimating the software cost including algorithmic or non-algorithmic model. But no one technique can estimate the software cost with high accuracy, since no estimation technique is best for all situation. So, first software project manager finds the characteristics, conditions and status of the project, and then select the estimation technique which is best for software project. Each estimation technique has its own limitations and depends on many parameters such as development process, expertise of the staff, complexity of project, duration and so on. Results show that the web cost model gives better performance as compared to the other. However combining COCOMO II with other techniques like pricing to win or Parkinson's law will certainly improve the performance of estimation.

6. ACKNOWLEDGMENTS

I specially thanks to Prof. P. M. Yawalkar for guiding me. Also I am thankful to my parents, friends and others for there help and blessings.

7. REFERENCES

[1] Software Engineering 8th Edition – Lan Sommerville.

[2] "Software Cost Estimation Methods: A Review" - Vahid Khatibi, Dayang N. A. Jawawi. Volume 2 No. 1 ISSN 2079-8407 Journal of Emerging Trends in Computing and Information Sciences ©2010-11 CIS Journal.

[3] "Software Effort Estimation Approaches - A Review"- S.K.Mohanty and A.K.Bisoi. International Journal of Internet Computing ISSN No: 2231 – 6965, VOL- 1, ISS- 3 2012.

[4] "Software Cost Estimation" - Hareton Leung, Zhang Fan

[5] "The Comparison of the Software Cost Estimating Methods" - Liming Wu, University of Calgary.

[6] "Literature Survey On Algorithmic And Non-Algorithmic Models For Software Development Effort Estimation" - K.Ramesh P.Karunanidhi, International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 2 Issue 3 March 2013 Page No. 623-632.

[7] "E-Cost Estimation Using Expert Judgment and COCOMO II" - Zulkefli Bin Mansor, Zarinah Mohd Kasirun, Noor Habibah Hj Arshad, Saadiah Yahya, *ITSIM'10,* 15-18 June 2010, Kuala Lumpur, Malaysia. Copyright 2010 IEEE 978-1-4244-6716-7/10/$26.