International Journal of Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-3, Issue-4)

Research Article | **April 2014**

Software Cost Estimation Techniques

Kusuma Kumari B.M *
Department of Computer Science,
University College of Science, Tumkur University

Abstract—

Project planning is one of the most important activities in software projects. Poor planning often leads to project faults and dramatic outcomes for the project team. If cost and effort are determined pessimistic in software projects, suitable occasions can be missed; whereas optimistic predictions can be caused to some resource losing. Nowadays software project managers should be aware of the increasing of project failures. The main reason for this problem is imprecision of the estimation. Software cost estimation is a complex activity that requires knowledge of a number of key attributes that affect the outcomes of software projects, both individually and in concert. Accurate cost estimation helps to complete project within time and budget. Many estimation models have been proposed over the last 30 years. This paper provides a detail overview of existing software cost estimation models and techniques.

Keywords—Cost Estimation, COCOMO, Estimation Techniques, Non Algorithmic method, Algorithmic method, function point

I. INTRODUCTION

It has been surveyed that nearly one-third projects overrun their budget and late delivered and two-thirds of all major projects substantially overrun their original estimates. The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for project managers as well as system analysts and developers. Without reasonably accurate cost estimation capability, project managers cannot determine how much time and manpower cost the project should take and that means the software portion of the project is out of control from its beginning; system analysts cannot make realistic hardware-software tradeoffs analyses during the system design phase; software project personnel cannot tell managers and customers that their proposed budget and schedule are unrealistic. This may lead to optimistic over promising on software development and the inevitable overruns and performance compromises as a consequence. But, actually huge overruns resulting from inaccurate estimates are believed to occur frequently.

There are a number of competing software cost estimation methods available for software developers to predict effort and test effort required for software development, from the intuitive expert opinion methods to the more complex algorithmic modelling methods and the analogy-based methods [1], [2], [3], [4]. In software project estimation, it is important to balance the relationships between effort, schedule and quality, which form the three essential aspects of the famous ”magic” triangle. It is widely accepted that simply estimating one of these aspects without considering the others will result in unrealistic estimations. Classical estimation models are established based on linear or non-linear regression analysis, which incorporate fixed input factors and fixed outputs. The size of the project determining the scope is modelled as a main input of such models [5], [6], [7], [8]. One representative of such models is COCOMO [9], [10], [11]. The most critical problem in such an approach is the wealth of data that is needed to get regression parameters, which is often impossible to get in needed quantities. It really limits their usage for project estimation. In recent years, a flexible and competitive method, Bayesian Belief Network (BBN), has been proposed for software project estimation [12], [13], [14], [15].

It is very difficult to estimate the cost of software development. Many of the problems that plague the development effort itself are responsible for the difficulty encountered in estimating that effort. One of the first steps in any estimate is to understand and define the system to be estimated. Software, however, is intangible, invisible, and intractable. It is inherently more difficult to understand and estimate a product or process that cannot be seen and touched. Software grows and changes as it is written. When hardware design has been inadequate, or when hardware fails to perform as expected, the "solution" is often attempted through changes to the software. This change may occur late in the development process, and sometimes results in unanticipated software growth.

II. ESTIMATION TECHNIQUES

Generally, there are many methods for software cost estimation, which are divided into two groups: Algorithmic and Non-algorithmic. Using of the both groups is required for performing the accurate estimation. If the requirements are known better, their performance will be better. In this section, some popular estimation methods are discussed.

A. Non Algorithmic Based Estimation Methods

1) *Expert Judgment Method:* Expert judgment techniques involve consulting with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost.

Generally speaking, a group consensus technique, Delphi technique, is the best way to be used. The strengths and weaknesses are complementary to the strengths and weaknesses of algorithmic method.

To provide a sufficiently broad communication bandwidth for the experts to exchange the volume of information necessary to calibrate their estimates with those of the other experts, a wideband Delphi technique is introduced over standard Delphi technique [16],[17],[18].

The estimating steps using this method:

1. Coordinator presents each expert with a specification and an estimation form.
2. Coordinator calls a group meeting in which the experts discuss estimation issues with the coordinator and each other.
3. Experts fill out forms anonymously
4. Coordinator prepares and distributes a summary of the estimation on an iteration form.
5. Coordinator calls a group meeting, specially focusing on having the experts discuss points where their estimates varied widely.
6. Experts fill out forms, again anonymously, and steps 4 and 6 are iterated for as many rounds as appropriate.

The wideband Delphi Technique has subsequently been used in a number of studies and cost estimation activities [18]. It has been highly successful in combining the free discuss advantages of the group meeting technique and advantage of anonymous estimation of the standard Delphi Technique.

2) *Estimating by Analogy:* Estimating by analogy means comparing the proposed project to previously completed similar project where the project development information id known. Actual data from the completed projects are extrapolated to estimate the proposed project. This method can be used either at system-level or at the component-level [19]. The estimating steps using this method are as follows:

a. Find out the characteristics of the proposed project.
b. Select the most similar completed projects whose characteristics have been stored in the historical data base.
c. Find out the estimate for the proposed project from the most similar completed project by analogy.

3) *Top-Down Estimating Method:* Top-down estimating method is also called Macro Model. Using top-down estimating method, an 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 leading method using this approach is Putnam model. This method is more applicable to early cost estimation when only global properties are known. In the early phase of the software development, it is very useful because there is no detailed information available [17] [18].

4) *Bottom-up Estimating Method:* Using bottom-up estimating method, the cost of each software components is estimated and then combines the results to arrive at an estimated cost of overall project. It aims at constructing the estimate of a system from the knowledge accumulated about the small software components and their interactions. The leading method using this approach is COCOMO's detailed model [18].

B. Algorithmic Method

The algorithmic method is designed to provide some mathematical equations to perform software estimation. These mathematical equations are based on research and historical data and use inputs such as Source Lines of Code (SLOC), number of functions to perform, and other cost drivers such as language, design methodology, skill-levels, risk assessments, etc. The algorithmic methods have been largely studied and many models have been developed, such as COCOMO models, Putnam model, and function points based models [17] [20].

1) COCOMO Model

One very widely used algorithmic software cost model is the Constructive Cost Model (COCOMO) It was proposed by Boehm [21],[22].. The basic COCOMO model has a very simple form:

MAN-MONTHS = K1* (Thousands of Delivered Source Instructions) ^{K2}

Where K1 and K2 are two parameters 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. Some of these factors are:

Complexity of the software

1. Required reliability
2. Size of data base
3. Required efficiency (memory and execution time)
4. Analyst and programmer capability
5. Experience of team in the application area
6. Experience of team with the programming language and computer
7. Use of tools and software engineering practices

Many of these factors affect the person months required by an order of magnitude or more. COCOMO assumes that the system and software requirements have already been defined, and that these requirements are stable. This is often not the case.

COCOMO model is a regression model. It is based on the analysis of 63 selected projects. The primary input is KDSI. The problems are:

1. In early phase of system life-cycle, the size is estimated with great uncertainty value. So, the accurate cost estimate cannot be arrived at.
2. The cost estimation equation is derived from the analysis of 63 selected projects. It usually have some problems outside of its particular environment. For this reason, the recalibration is necessary.

According to Kemmerer's research, the average error for all versions of the model is 601%. The detailed model and Intermediate model seem not much better than basic model.

The first version of COCOMO model was originally developed in 1981. Now, it has been experiencing increasing difficulties in estimating the cost of software developed to new life cycle processes and capabilities including rapid-development process model, reuse-driven approaches, object-oriented approaches and software process maturity initiative.

For these reasons, The newest version, COCOMO 2.0, was developed. The major new modelling capabilities of COCOMO 2.0 are a tailorable family of software size models, involving object points, function points and source lines of code; nonlinear models for software reuse and reengineering; an exponent-driver approach for modeling relative software diseconomies of scale; and several additions, deletions, and updates to previous COCOMO effort-multiplier cost drivers. This new model is also serving as a framework for an extensive current data collection and analysis effort to further refine and calibrate the model's estimation capabilities.

2) Putnam Model: The Putnam model is an empirical software effort estimation model. Putnam used his observations about productivity levels to derive the software equation:

Technical constant C= size * B1/3 * T4/3

Total Person Months B=1/T4 *(size/C)3

T= Required Development Time in years Size is estimated in LOC

Where: C is a parameter dependent on the development environment and is determined on the basis of historical data of the past projects.

Rating: C=2,000 (poor), C=8000 (good) C=12,000 (excellent).

The Putnam model is very sensitive to the development time: decreasing the development time can greatly increase the person-months needed for development [16] [18].

One significant problem with the Putnam model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) of the software to be developed. There is often great uncertainty in the software size. It may result in the inaccuracy of cost estimation.

3) Function Point Analysis: The Function Point Analysis is another 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 adopted a function point type of approach, such as ESTIMACS and SPQR/20. This is a measurement based on the functionality of the program and was first introduced by Albrecht [23]. The total number of function points depends on the counts of distinct (in terms of format or processing logic) types. There are two steps in counting function points:

a. Counting the user functions:- The raw function counts are arrived at by considering a linear combination of five basic software components: 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 function counts (FC).

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

III. Selection and Use of Estimation Methods

1) The selection of Estimation methods

From the above comparison, we know no one method is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. According to the experience, it is recommended that a combination of models and analogy or expert judgment estimation methods is useful to get reliable, accurate cost estimation for software development.

For known projects and projects parts, we should use expert judgment method or analogy method if the similarities of them can be got, since it is fast and under these circumstance, reliable; For large, lesser known projects, it is better to use algorithmic model. In this case, many researchers recommend the estimation models that do not required SLOC as an

input. I think COCOMO2.0 is the first candidate because COCOMO2.0 model not only can use Source lines of code (SLOC) but also can use Object points, unadjusted function points as metrics for sizing a project. If we approach cost estimation by parts, we may use expert judgment for some known parts. This way we can take advantage of both: the rigor of models and the speed of expert judgment or analogy. Because the advantages and disadvantages of each technique are complementary, a combination will reduce the negative effect of any one technique, augment their individual strengths and help to cross-check one method against another.

2) Use of Estimation Methods

It is very common that we apply some cost estimation methods to estimate the cost of software development. But what we have to note is that it is very important to continually re-estimate cost and to compare targets against actual expenditure at each major milestone. This keeps the status of the project visible and helps to identify necessary corrections to budget and schedule as soon as they occur.

At every estimation and re-estimation point, iteration is an important tool to improve estimation quality. The estimator can use several estimation techniques and check whether their estimates converge. The other advantages are as following:

- Different estimation methods may use different data. This results in better coverage of the knowledge base for the estimation process. It can help to identify cost components that cannot be dealt with or were overlooked in one of the methods
- Different viewpoints and biases can be taken into account and reconciled. A competitive contract bid, a high business priority to keep costs down, or a small market window with the resulting tight deadlines tends to have optimistic estimates. A production schedule established by the developers is usually more on the pessimistic side to avoid committing to a schedule and budget one cannot meet.

IV. CONCLUSION

The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for project managers as well as system analysts and developers. There are many software cost estimation methods available including algorithmic methods, estimating by analogy, expert judgment method, top-down method, and bottom-up method. No one method is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. To understand their strengths and weaknesses is very important when you want to estimate your projects.

For a specific project to be estimated, which estimation methods should be used depend on the environment of the project. According to the weaknesses and strengths of the methods, you can choose some methods to be used.

REFERENCES

[1] Y. F. Li, M. Xie, T. N. Goh, ”A Study of Genetic Algorithm for Project Selection for Analogy Based Software Cost Estimation, IEEE, 2007.
[2] Khaled Hamdan, Hazem El Khatib, Khaled Shuaib,” Practical Software Project Total Cost Estimation Methods”, MCIT 10, IEEE, 2010.
[3] Chetan Nagar, ”Software efforts estimation using Use Case Point approach by increasing technical complexity and experience factors”, IJCSE, ISSN:0975-3397, Vol.3 No.10 , Pg No 3337- 3345,October 2011.
[4] Chetan Nagar, Anurag Dixit, ”Software efforts and cost estima- tion with systematic approach”, IJETCIS, ISSN:2079-8407, Vol.2 No.7, July 2011.
[5] Chen Qingzhang, Fang Shuojin, Wang Wenfu, ”Development of the Decision Support System for Software Project Cost Estimation”, World Congress on Software Engineering,IEEE, 2009.
[6] Yinhuan Zheng, Yilong Zheng, Beizhan Wang, Liang Shi, ”Esti- mation of software projects effort based on function point”, 4th International Conference on Computer Science and Education, 2009.
[7] Jairus Hihn, Hamid Habib-agahi, ”Cost Estimation of Software Intensive Projects:A Survey of Current Practices”, IEEE, 2011.
[8] Yunsik Ahn, Jungseok Suh, Seungryeol Kim, Hyunsoo Kim, ”The software maintenance project effort estimation model based on function points”, Journal of software maintenance and evoloution, 2003.
[9] Jin Yongqin, Li Jun, Lin Jianming, Chen Qingzhang, ”Software Project Cost Estimation Based On Groupware”, World Congress on Software Engineering, IEEE, 2009.
[10] Pichai Jodpimai, Peraphon Sophatsathit, and Chidchanok Lursin- sap, ”Analysis of Effort Estimation based on Software Project Models”, IEEE, 2009.
[11] Nancy Merlo Schett, ”Seminar on software cost estimation”, University of Zurich,Switzerland, 2003.
[12] Hao Wang, Fei Peng, Chao Zhang, Andrej Pietschker, ”Software Project Level Estimation Model Framework based on Bayesian Belief Networks”, Sixth International Conference on Quality Software (QSIC’06), IEEE, 2006.
[13] angyang Yu, Charlottesville, ”A BBN Approach to Certifying the Reliability of COTS Software Systems”, annual reliability and maintainability symposium, IEEE, 2003.
[14] Ying Wang, Michael Smith, ”Release Date Prediction for Telecommunication Software Using Bayesian Belief Networks”, E Canadian Conference on Electrical and Computer Engineer- ing, IEEE, 2002.

[15] S. Bibi, I. Stamelos, L. Angelis, ”Bayesian Belief Networks as a Software Productivity Estimation Tool,IEEE.
[16] Liming Wu ―The Comparison of the Software Cost Estimating Methods‖ University of Calgary.
[17] ―COCOMO II Model definition manual‖, version 1.4, University of Southern California.
[18] Caper Jones, ―Estimating software cost‖ tata Mc- Graw -Hill Edition 2007.
[19] Murali Chemuturi, “Analogy based Software Estimation,” Chemuturi Consultants.
[20] 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 Madrid (UC3M),Volume33, Issue 1, pp.133-150, March, 2008
[21] R.K.D. Black, R. P. Curnow, R. Katz and M. D. Gray, BCS Software Production Data, Final Technical Report, RADC-TR-77-116, Boeing Computer Services, Inc., March 1977.
[22] B.W. Boehm, ―Software Engineering Economics,‖ Prentice- Hall, Englewood Cliffs, NJ, USA, 1981.
[23] A.J. Albrecht and J.E. Gaffney, ―Software function, source lines of code, and development effort prediction: a software science validation,‖ IEEE Transactions on Software Engineering, , pp. 639–647, 1983.