SOFTWARE PROJECT ESTIMATION METHODS: A REVIEW

RINA BHATTACHARYA

Abstract: Project planning is one of the most important activities in software projects. Poor planning often leads to project failure and dramatic outcomes for the project team. Whenever estimates are made, we look into the future and accept some degree of uncertainty as a matter of course. Although estimating is as much art as it science, this important activity need not to be conducted in a haphazard manner. 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 the project failure. The main reason for this problem is imprecision of the estimation. The objective of this paper is to provide a comparative study of several existing methods of software project estimation technique and their aspects.

Keywords: Cost Estimation, Effort Estimation, Project Planning, Software Project Estimation Technique.

Introduction : Software project management begins with a set of activities that are collectively called project planning. Before the project can begin, the manager and software team must estimate the work to be done, the resources that will be required, and the time that will elapse from start to finish. Whenever estimates are made we look into the future and accept some degree of uncertainty as a matter of course. Although estimating is as much art as it science, this important activity need not to be conducted in a haphazard manner. Useful techniques for time and effort estimation do exist. Process and project metrics can provide historical perspective and powerful input for the generation of quantitative estimates. Past experience can aid immeasurably as estimates are developed and reviewed. Lack of project estimation makes the project boundaries quite vague. Estimation serves as a compass, navigating the project team throughout the project life cycle. Making estimation are not difficult, but to establish accurate and realistic estimation is one of the challenging and important activity in the software development.

Project Estimation Technique

Model of various types are simply abstractions of the product or process. Effective models allow us to ignore uninteresting details and concentrate on essential aspects of the artifact described by the model. Preference should be given to the simplest models that provides adequate descriptive capability and some measures of intuitive acceptability. A good model should possess predictive capabilities rather than being merely descriptive or explanatory.

Over the last 30 years, several estimating techniques have been proposed, such as estimating by analogy, expert judgment etc. However there is no simple way to establish accurate estimates. Despite the extensive amount of experience with estimation techniques, the accuracy of the existing estimation techniques is not adequate. Initial estimates are usually based on inadequate information in a user requirements definition. The estimation techniques are divided into six different approaches: Empirical estimation approach, Heuristic approach, Statistical approach, Theory-based approach, Analytical approach and de-compositioning approach. Empirical estimation approach has five another approaches, these are Ad-hoc approaches, analogy approach, Expert-judgment approach, Pert-sizing approach and algorithmic model.

A)Empirical Estimation Approach

Empirical estimation techniques are based on making an educated guess of the project parameters. While using this techniques, prior experience with the development of similar products is helpful. Although estimation techniques are based on common sense, different activities involved in estimation have been formalized over the years.

a)ad-hoc Approach

They have little or no bias but they work, they win orders and they are useful when there is no better way. The advantages of this approach is that

1.It has quick response time.

The disadvantages are:

1.It has poor overall system performance

2.Dissatisfaction at all ends

b)Parkinsonian Approach

This techniques is based on Parkinson's law and implies that the project cost is determined, not estimated, by the available resources rather than on objective assessment. The advantages of this approach:

1.It seems to always work

The disadvantages are:

1.Work expands to fill available budget and time

2.It often leads to overestimates

c)Price-to win Approach

This approach estimates the cost with the intention

to set a good price. A good price implies a price that will win the project. Thus the estimates are based on the customer's budget rather than on their needs for certain functionality. The strength of this approach is obviously that the supplier often gets the contract. The advantages of this approach are:
1.Apparently easy and straightforward
2.Does not need expert assistance
The disadvantages are:
1.Often leads to budget overshooting
2.Schedule overruns are common
3.Bad quality and poor reputation
d)Analogy Approach
This estimation technique compares the current project with one or more similar projects that are completed. Thus the characteristics, such as effort, schedule and cost of the projects are known. The strength of this technique is that it is based actual experiences and real project data. This method deals with experienced and completed projects it is also capable of dealing with poorly understood domains. It avoids the problems associated both with knowledge elicitation and extracting and codifying the knowledge. The weakness is obviously that it is not always certain that there exists a similar project that is representative of the constraints, environment, functions etc. of the new software.
e)Expert-judgment Approach
This method is the most commonly used method. It bases the estimates on knowledge and experience of one or several experts. Each expert establishes an estimate and the estimates are compared and discussed. This approach is iterated until the experts have agreed upon an estimate. The strength of this approach is that an expert with adequate experience may provide good estimates quite fast and relatively cheap. On the other hand, if the experts does not possess sufficient experience and knowledge, the estimates will most likely suffer from inaccuracy. The disadvantages of this approach are:
1.Expert's recall capability may be limited.
2.May be wholly subjective.
3.Subject to personal bias.
f)Delphi Approach
Delphi cost estimation approach tries to overcome some of the shortcomings of expert judgment approach. It is carried out by a group of experts and a coordinator. The approach is based on the non-consultative, group consensus technique, needs access to several experts. The steps in a typical Delphi process are:
1.Coordinator explains the task to experts.
2.Specifications are supplied to each expert.
3.Each expert makes estimates anonymously.
4.Coordinator consolidates the responses and circulates the same to all experts.
5.Each experts reacts to disagreements giving reasons
6.Iterates till agreement is reached.
g)Pert-sizing Approach
Better estimates can be obtained by breaking the project down into a number of components. It undertakes estimation for individual components. It uses PERT approach for estimating the total project size. The advantage of this approach is:
1.Low-level break-up
The disadvantage is that it needs a lot of information.
h)Bottom-up
In this approach, each component of the software system is separately estimated and the results aggregated to produce an estimate for the overall system. The requirement for this approach is that an initial design must be in place that indicates how the system is decomposed into different components.
i)Top-down
This approach is the opposite of the bottom-up method. An overall cost estimate for the system is derived from global properties, using either algorithmic or non-algorithmic methods. The total cost can then be split up among the various components. This approach is more suitable for cost estimation at the early stage.
j)Algorithmic Model
Algorithmic estimation is based on the application of a cost model, i.e, mathematical formulas that have been derived through statistical analysis of completed projects. In its most general form, the software cost may be expressed as
$\text{Effort}=A \times \text{SIZE}^{B} \times M$
In which A and B represent constant factors that depends on organizational practices and the type of software that is developed. The SIZE represents the software size and may be expressed in either code size or function points. M is an effort multiplier which is determined by combining several cost factors. The different cost factors are divided into four different categories:
Product factors: reliability, complexity, size of database, reusability etc.
Computer factors: execution time constraints, storage constraints, platform volatility etc.
Personal factors: programming capability, analyst capability, application experience etc.
Project factors: use of software tool, required development schedule etc.
The advantages are:
1.Calibrated to previous projects
2.Easy to use
The disadvantages are:
1.Rely on the estimate of some parameter of finished software.
2.Controlled experiments are expensive and difficult.
3.Parameter values tend to be highly organization-

dependent.
4.Limited usefulness of past data in the light of advancing technology.
5.Dependent on the correctness of the inputs.
6.Cannot handle exceptional conditions.
7.Calibrated to past projects and not to future.
8.Impact of new technology is difficult to judge.

B) Heuristic Approach

Heuristic techniques assume that the relationships among the different project parameters can be modeled using suitable mathematical expressions. Once the basic(independent) parameters are known, the other(dependent) parameters can be easily determined by substituting the value of the basic parameters in the mathematical expression. Different heuristic estimation models can be divided into the following two classes: single variable model and multi-variable model.

Single variable estimation models provide a means to estimate the desired characteristics of a problem, using some previously estimated basic(independent)characteristic of the software product such as its size. A single variable estimation model takes the following form:

Estimated Parameter= $c_1 * e^{d1}$

In the above expression, e is a characteristic of the software which has already been estimated. Estimated parameter is the dependent parameter to be estimated. The dependent parameter to be estimated could be effort, project duration, staff size etc. c_1 and d_1 are constant. The values of the constant c_1 and d_1 are usually determined using the data collected from past project(historical data).

A multi-variable cost estimation model takes the following form:

Estimated Resource= $c_1*e_1^{d1}+c_2*e_2^{d2}+$.....

Where e_1, e_2,... are the basic (independent)characteristics of the software already estimated, and c_1, c_2,... are constants. Multivariable estimation models are expected to give more accurate estimates compared to the single variable models, since a project parameter is typically influenced by several independent parameters. The independent parameters influence the dependent parameter to different extents. The values of the constants are determined from historical data.

COCOMO

The basic COCOMO model gives an approximate estimate of the project parameters. The basic COCOMO estimation model is given by the following expressions:

Effort=a_1 X $(KLOC)^{a}2$ PM

Tdev=b_1 X $(Effort)^{b}_2$ months

Where KLOC is estimated the size of the software product expressed in Kilo Lines of Code, a1, a2, b1, b2 are constants for each category of software products, Tdev is the estimated time to develop the software, expressed in months, Effort is the total effort required to develop the software product expressed in person-months(PM).

The basic COCOMO model assumes that effort and development time are functions of the product size alone. However a host of other project parameters besides the product size affect the effort required to develop the product as well as the development time. Therefore, in order to obtain an accurate estimation of the effort and project duration, the effort of all relevant parameters must be taken into account. The intermediate COCOMO model recognizes this fact and refines the initial estimate obtained through the basic COCOMO expressions by using a set of 15 cost drivers based on various attributes of software development. A major shortcoming of both the basic and intermediate COCOMO models is that they consider a software product as a single homogenous entity.

C) Statistical Approach

C.E. Walston and C.P.Felix of IBM used data from 60 previous software projects completed by the Federal Systems Division to develop a simple model of software development effort. The metric LOC was assumed to be the principal determiner of development effort. A relationship of the form

$E=aL^b$

was assumed, where L is the number of lines of code, in thousands and E is the total effort required, in person-months. Regression analysis was used to find appropriate values of parameters a and b. the resulting equation was

$E=5.2L^{0.91}$.

Nominal programming productivity, in LOC per person-month, can then be calculated as L/E. In order to account for deviations from the derived from E, Walston and Felix also tried to develop a productivity index, I, which would increase or decrease the productivity depending upon the nature of the project.

D) Theory-Based Approach

L.H. Putnam developed a theory-based model pf the software development process based upon the assumption that the personnel utilization during program development is described by a Rayleigh curve such as the following:

$Y=Kte^{-t2/2T2}/T^2$

Where y= the number of person on the project at any time, t;

K= the area under the Rayleigh curve, equal to the total life cycle effort in person-years and

T= development time.

Putnam assumed that either overall staffing curve or the staffing curves for individual phases of the development cycle can be modeled by an equation of

this form. He then developed the following relationship between the size of the software product and the development time:

$S=CK^{1/3}T^{4/3}$,

Where S is the number of source LOC delivered;
K is the life cycle effort in person-years and
C is a state of technology constant.

E) Analytical Approach

Analytical estimation techniques derive the required results starting with certain basic assumptions required the project. Here we discuss the Halstead's software science as an example of an analytical model.

Halstaed's Software Science Model

Halstead's software science is an analytical technique to measure size, development effort, and development cost of software products. Halstead use a few primitive program parameters to develop the expressions for the overall program length, potential minimal volume, actual volume, language level, effort and development time. For a given program, let:

n_1 be the number of unique operators used in the program;

n_2 be the number of unique operands used in the program;

N_1 be the total number of operators used in the program;

N_2 be the total number of operands used in the program.

Length and Vocabulary : the length of a program as defined by Halstead, quantifies the total usage of all operators and operands in the program. Thus length $N= N_1+N_2$. Halstead's definition of the length of the program as the total number of operators roughly agrees with the intuitive notion of the program length as the total number of tokens used in the program. The program vocabulary is the number of unique operators and operands used in the program. Thus, program vocabulary n=n1+n2.

Program Volume: The length of a program depends on the choice of the operators and operands used. Thus while expressing program size, the programming language used must be taken into consideration. Let us try to understand the important idea underlying the following expression:

$V=N \log_2 n$

The program volume V is the minimum number of bits needed to encode the program. Here we need N $\log_2 n$ bits to store a program of length N. Therefore, the volume V represents the size of the program by approximately compensating for the effect of the programming language used.

Potential Minimal Volume: The potential minimal volume V* is defined as the volume of the, most succinct program in which a problem can be coded. The minimum volume is obtained when the program can be expressed using a single source code instruction. If an algorithm operates on input and output data d_1, d_2, ...,d_n, the most succinct program would be $f(d_1, d_2,.....,d_n)$for which $n_1= 2$, $n_2=n$. Therefore $V^*= (2+n_2) \log(2+n_2)$. The program level L is given by $L=V^*/V$. the concept of program level L has been introduced in an attempt to measure the level of abstraction provided by the programming language.

Effort and Time: The effort required to develop a program can be obtained by dividing the program volume by the level of programmi9ng language used to develop the code. Thus effort E=V/L, where E is the number of mental discriminations required to implement the program and also the effort required to read and understand the program. Thus the program effort E is $E= V^2/ V^*$varies as the square of the volume. The programmer's time T=E/S where S is the speed of mental discriminations. The value of S has been empirically developed from psychological reasoning.

Length Estimation: Halstead suggest a way to determine the length of a program utilizing the number of unique operators and operands used in the program. Halstead assumed that it is quite unlikely that a program has several identical parts- in formal language terminology identical substrings-of length greater than n. thus we can safely assume that any program of length N consists of N/n unique strings of length n.

F) De-compositioning Approach

The decomposition approach has two different view points: decomposition of the problem and decomposition of the process. Estimation uses one or both forms of partitioning. But before an estimate can be made, the project planner must understand the scope of the software to be built and generate an estimate of its "size".

Conclusion : Finding the most important reason for the software project failures has been the subject of many researches in the las decade. According to the results of several researches, the root cause of software project failures is inaccurate estimation in early stages of the project. So introducing and focusing on the estimation methods seems necessary for achieving to the accurate and reliable estimation. In this paper, I study most of the present estimation techniques that have been illustrated systematically. Since software project managers are used to select the best estimation method based on the condition and status of the project, there is no estimation method which can be present the best estimates in all various situations and each technique can be suitable in the special project. It is necessary for understanding the principles of each estimation method to choose the

best, because performance of each estimation method depends on several parameters such as complexity of the project, duration of the project, expertise of the staff, development method and so on. Some project estimation techniques and their merits and demerits have been presented in this paper for choosing the right estimation method to avoid the project failure.

References :

1 Balachandani, N. 2003. Software Estimating & Costing. Mona Associates.

2 Boehm, Barry W; Abts, Chris; Brow, A Winsor. 2000. Software Cost Estimation with COCOMO II. Prentice Hall

3 Coombs, Paul. 2003. IT Project Estimation – A Practical Guide to the Costing of Software. Cambridge University Press

4 Crowfoot, Norman; Hatfield, Scott; Swank, Mike. 1992. DMCM: A Knowledge-Based Cost-Estimation Tool. Xerox Corporation IAAI-92 Proceedings. 1992

5 Jones, Caper. 2007. Estimating Software Costs. Tata Mc-GrawHill Education Pvt. Ltd. 2nd Edition.

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

7 Lum, Karen; Bramble, Michaael; Hihn, jairus; Hackney, John, Khorrami, Mori; Monson, Erik. 2003. Handbook for Software Cost Estimation. JPL D-26303, Rev. 0. May 30, 2003

8 Mendes, Emilia; Warson, Ian; TRiggs, Chris; Mosley, Nile; Counsell, Steve. 2003. A Comparative Study of Cost Estimation Models for Web Hypermedia Applications. Empirical Software Engineering, 8, 163–196, 2003.

9 Milicic, Darko. 2004. Applying COCOMO II – A case study. Master Thesis No.MSE-2204-19. School of Engineering. Blekinge Institute of Technology. Sweden. August 2004.

10 Peters, Kathleen. 2000. Software Project Estimation. Methods and Tools. Global knowledge source for software development professionals ISSN 1023-4918. Summer 2000. Volume 8. Number 2.

11 Pfleeger, Shari Lawrence; Wu Felicia; Lewis, Rosalind. 2005. Software Cost Estimamtion and Sizing Methods, Issues & Guidelines. Rand Corporation.

12 Sharma, T. N.; Bhardwaj, Anil; Sharma, Anita. 2011. A Comparative study of COCOMO II and Putnam models of Software Cost Estimation. International Journal of Scientific & Engineering Research. Volume 2. Issue 11. November-2011 1 ISSN 2229-5518.

13 Tagra, Dinesh; Arora, Manisha. 2012. Cost Estimation for Commercial Software Development Organizations. Lap Lambert Academic Publishing

14 Vladan Devedzic. Software Project Management.

***

Research Scholar,
CMJ University, Shillong,
Meghalaya, India,
bh.rini@gmail.com