IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 9, November 2014.
www.ijiset.com
ISSN 2348 – 7968

Software Project Estimation Techniques - Effort and Cost

Garima, Kailash Bahl

Department of Computer Science & Engineering
Patiala Institute of Engineering & Technnology[1], Patiala[1]

Abstract-The main objective of software project estimation covering effort and cost is to have an idea about the workload and financial implications of implementing a project. Software cost estimation is a complex activity as there are many parameters that affect the outcome of project. The essential ingredient needed for estimation is data from similar projects implemented in past and the people involved in implementing those past projects. In this paper many techniques have been discussed that have been implemented and an attempt has been made to suggest new ideas to arrive at precise estimation of such parameters as cost and effort.

Keywords – LOC, Function Points, COCOMO, Use case point, Person- month etc.

I. INTRODUCTION

Software has played an increasingly important role in systems acquisition, engineering and development, particularly for large and complex systems. accurate es- timates of the software costs are a critical part of effective program management. The bulk of the cost of software devel- opment is due to the human effort and most cost estimation methods focus on this aspect and give estimates in terms of person-months. Accurate cost estimates are critical to both developers and customers. They can be used for request for proposal, contract negotiations, scheduling, monitoring and control. Underestimating the costs may result in management approving proposed systems that then exceed their budgets, with underdeveloped functions and poor quality, and failure to complete on time. Overestimating may result in too many resources committed to the projects, or, during contract bidding, result is not winning the contract, which can lead to loss of jobs. Software project failures have been an important subject in the last decade. Software projects usually don't fail during the implementation and most project fails are related to the planning and estimation steps. Hence accurate cost estimation is become a challenge for IT industries. 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 modeling methods and the analogy-based methods. In software project estimation, it is important to balance the relationships between effort, schedule and quality. It is widely accepted that simply estimating one of these aspects without considering the others will result in unrealistic estimations.

The size of the project determining the scope is modeled as a main input of such models as COCOMO [1]. 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. In recent years, a flexible and competitive method, Bayesian Belief Network (BBN), has been proposed for software project estimation [8].

SOFTWARE METRICS

This section provides some background information on software conventional metrics used in software cost and effort estimation [4], [20].

A. Size Oriented Metrics:

i) Source Lines of Code (SLOC): is software metric used to measure the size of software program by counting the number of lines in the text of the program's source code. This metric does not count blank lines, comment lines, and library. SLOC measures are programming language dependent. They cannot easily accommodate nonprocedural languages. SLOC also can be used to measure others, such as errors/KLOC, defects/KLOC, pages of documentation/KLOC, cost/KLOC, ... etc.

ii) Delivered Source Instruction (DSI): DSI attempt to measure the distinction between a "if-then-else" statement, it would be counted as one SLOC but might be counted as several DSI [5].

B. Function Oriented Metrics:

Function Point (FP): FP defined by Allan Albrecht at IBM in 1979, is a unit of measurement to express the amount software functionality [5]. Function point analysis (FPA) is the method of measuring the size of software. The advantage is that it can avoid source code error when selecting different programming languages. FP is programming language independent, making ideal for applications using

ISSN 2348 – 7968

conventional and nonprocedural languages. It is base on data that are more likely to be known early in the evolution of project. Function types are as:

- External Inputs (EI): it originates from user or transmit ted from another application.
- External Outputs (EO) : it is derived data within application that provides information to the user.
- External Enquiries (EQ) : it is online i/p that results in the generation of some immediate s/w response in the form of an online output.
- Internal Logical Files (ILF) : is logical grouping of data that resides within the applications boundary and maintained via EI.
- External Interface Files (EIF) : is logical grouping of data that resides external to application but provides information that may be of use to the application.

1.1.2 Test Point (TP)

Test point used for test point analysis (TPA), to estimate test effort for system and acceptance tests [15]. However, it is important to note that TPA itself only covers functional testing. Hence, it is always used with FPA, that does not cover system and acceptance tests. Consequently, FPA and TPA merged together provide means for estimating both, white and black box testing efforts. There are a lot of dependent and independent factors that need to be taken into account

1.1.2.1 Test effort estimation using UCP [15], is based upon use cases (UC). UC is a systems behaviour under various conditions, based on requests from a stakeholder. UC capture contractual agreements between these stakeholders about the systems behaviour. Thus, the primary task of UCP is to map use cases (UC) to test cases (TC). Hereby, each scenario together with the corresponding exception flow for each UC serves as input for a specific TC. Basically, the amount of test cases identified through this mapping results in the corresponding test effort estimation

COCOMO (Constructive Cost Models): This family of models proposed by Barry Boehm , is the most popular method which is categorized in algorithmic methods. This method uses some equations and parameters, which have been derived from previous experiences about software projects for estimation. The models have been widely accepted in practice. In the COCOMOs, the code-size S is given in thousand LOC (KLOC) and Effort is in person-month. Three models of COCOMO given by Barry Boehm

Simple COCOMO: It was the first model suggested by Barry Boehm, which follows following formula:

$$\text{Effort} = a*(\text{KLOC})^{b}$$

where S is the code-size, and a, b are are complexity factors. This model uses three sets of a, b depending on the complexity of the software only as given in table IX. The basic COCOMO model is simple and easy to use. As many cost factors are not considered, it can only be used as a rough estimate.

Intermediate COCOMO: In the intermediate COCOMO, a nominal effort estimation is obtained using the power function with three sets of a, b, with coefficient a being slightly different from that of the basic COCOMO as shown in table 1.

Model	a	b
Organic (Simple in terms of size and complexity)	3.2	1.05
Semi ditched (Average in terms of size and complexity)	3.0	1.15
Embedded (Complex)	2.8	1.20

Table 1: Depicting model and values for a and b

Expertise Based Estimation is the most frequently applied estimation strategy for software projects. There is no substantial evidence for use of estimation based models however there are situations where one can expect expert based estimation to be more precise than formal methods. This method is usually used when there is limitation in finding data and gathering requirements. Consultation is the basic issue in this method. The following expert estimation best practice guidelines are considered aiming at reducing the size of situational and human biases in expert estimation [21].

- Evaluate estimation accuracy, but avoid high evaluation pressure.
- Avoid conflicting estimation goals.
- Ask the estimators to justify and criticize their estimates.
- Avoid irrelevant and unreliable estimation information.
- Use documented data from previous development tasks.
- Find estimation experts with relevant domain background and good estimation records.
- Estimate top-down and bottom-up, independently

ISSN 2348 – 7968

of each other.
- Use estimation checklists.
- Combine estimates from different experts and estimation strategies.
- Assess the uncertainty of the estimate.

1.1.3 Delphi Method: The aim of Delphi method is to combine expert opinion and prevent bias due to positions, status or dominant personalities. Delphi arranges an especial meeting among the project experts and tries to achieve the true information. Delphi includes some steps as,

a) The coordinator gives an estimation form to each expert.
b) Each expert presents his own estimation (without discussing with others).
c) The coordinator gathers all Forms and sum them up up and start another iteration.
d) steps (b-c) are repeated until an approval is gained.

2) Rule Based Systems: Uses human expert knowledge to solve real-world problems that normally would re- quire human intelligence. Expert knowledge is often represented in the form of rules or as data within the Personnel Capability = Low THEN Risk Level = High

1.1.4 Bayesian Belief Network

Software cost and effort estimation is the process of fore- casting the software effort to estimate software costs of both development and maintenance. In the past decades, various kinds of software cost and effort estimation methods have been proposed. However, there is no optimal approach to accurately predict the effort needed for developing a software system. Because, the information gathered at the early stages of software system development is insufficient for providing a precise effort prediction. It is a complex activity that requires knowledge of a number of key attributes. Bundle of data is needed, which is often impossible to get in needed quantities. Hence, Bayesian Belief Networks are effective for cost and effort estimation [9], [14]. BBNs are especially useful when the information about the past and/or the current situation is vague, incomplete, conflicting, and uncertain. They are a very effective method of modeling uncertain situations that depend on cause and effect. They are compact networks of probabilities that capture the probabilistic relationship between variables, as well as historical information about their relation- ships. BBNs are very effective for modeling situations where some information is already known and incoming data is uncertain or partially unavailable. An important fact to realize about Bayesian Belief Networks is that they are not dependent on knowing exact historical information or current evidence.

CONCLUSION

This paper focused on the existing software estimation methods. Also, presented background information on software project models and software metrics to be used for effort and cost estimation. No model can estimate the cost of software with high degree of accuracy. Estimation is a complex activity that requires knowledge of a number of key attributes. At the initial stage of a project, there is high uncertainty about these project attributes. As we learn that BBNs are especially useful when the information about the past and/or the current situation is vague, incomplete, conflicting, and uncertain. Conventional estimation techniques focus only on the actual development effort furthermore, this paper also described test effort estimation. In fact, testing activities make up 40% total software development effort. Hence, test effort estimation is crucial part of estimation pro- cess.

REFRENCES

[1] Jin Yongqin, Li Jun, Lin Jianming, Chen Qingzhang, ”Software Project Cost Estimation Based On Groupware”, World Congress on Software Engineering, IEEE, 2009.

[2] Chen Qingzhang, Fang Shuojin, Wang Wenfu, ”Development of the Decision Support System for Software Project Cost Estimation”, World Congress on Software Engineering,IEEE, 2009.

[3] Y. F. Li, M. Xie, T. N. Goh, ”A Study of Genetic Algorithm for Project Selection for Analogy Based Software Cost Estimation, IEEE, 2007.

[4] 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.

[5] Pichai Jodpimai, Peraphon Sophatsathit, and Chidchanok Lursin- sap, ”Analysis of Effort Estimation based on Software Project Models”, IEEE, 2009.

[6] Eduardo Aranha, Paulo Borba, ”An Estimation Model for Test Execution Effort”, First International Symposium on Empirical Software Engineering and Measurement, IEEE, 2007.

[7] Xiaochun Zhu, Bo Zhou, Li Hou, Junbo Chen, Lu Chen, ”An Experience-Based Approach for Test Execution Effort Estimation”, 9th International Conference for Young Computer Scientists, IEEE, 2008.

[8] Hao Wang, Fei Peng, Chao Zhang, Andrej Pietschker, ”Software Project Level

ISSN 2348 – 7968

Estimation Model Framework based on Bayesian Belief Networks", Sixth International Conference on Quality Software (QSIC'06), IEEE, 2006.

[9] angyang Yu, Charlottesville, "A BBN Approach to Certifying the Reliability of COTS Software Systems", annual reliability and maintainability symposium, IEEE, 2003.

[10] 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.

[11] Khaled Hamdan, Hazem El Khatib, Khaled Shuaib," Practical Software Project Total Cost Estimation Methods", MCIT 10, IEEE, 2010.

[12] Jairus Hihn, Hamid Habib-agahi, "Cost Estimation of Software Intensive Projects:A Survey of Current Practices", IEEE, 2011.