International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2018 IJSRCSEIT | Volume 3 | Issue 3 | ISSN : 2456-3307

Effort Estimation Techniques - A Review

Monika*[1], Amit Khatkar[2]
*[1]Department of Computer Science and Engineering, GJ University Science & Engineering, Hisar, Haryana, India
[2]Department of Computer Science and Engineering, I.K. Gujral Punjab Technical University, Jalandhar, Panjab, India

ABSTRACT

Software Effort Estimation has always been an ongoing challenge to software engineers, as testing is one of most critical activities of SDLC. Now-a-days, accurate & reliable effort estimation is the need of software companies. There are various models and techniques used to estimate the software project effort. However, due to some factors or causes, they are unable to give accurate results. So, this paper act as torchbearer to various questions related to factors, importance of estimation, problem faced by estimation techniques, guidelines etc. of effort estimation. In this paper, we also summarized existing techniques that used for effort estimation.

Keywords: Function Point Analysis (FPA), WBS, User Case Point (UCP), Wideband Delphi Technique, Three Point Estimation, Percentage of development effort method, Percentage distribution

I. INTRODUCTION

Success of any project management largely depends upon software effort estimation. An unrealistic estimation leads to endangerment of software. If the effort estimation turns out to be underestimated then it will be very difficult to fulfill project's commitments because of shortage of time and/or funds. If overestimated, it may lead to rejection of software due to over demand of time and/or money.
Software's late delivery is the main factor in causing over cost of software development. Around 8000 projects observed from different areas. However, only 16.2 % projects delivered on under the consideration of cost as well as the user requirements. The remaining 83.8 % failed to produce the software i.e. delay approx. 200% and the cost can be increase approx. 180% on average in Standish group report [22].

The different methods based on analogy and algorithm model and others are frequently used to estimate the effort estimation in an early stage [10, 12, 15, 16].

This paper explains different techniques used to determine effort estimation to provide a better understanding to novel developers. There are different facts need to be measured during effort estimation.

These are:

1. Skill of the Team
2. Domain Knowledge
3. Application Complexity
4. Historical data
5. Cycle of the Bug for the project
6. Availability of the Resources (as if vacations, holiday, and sick days can have a great impact on your estimates).
7. Productivity variations
8. System requirements and downtime
9. Buffer time

II. TESTING ESTIMATION TECHNIQUES

Variety of methods used to measure effort estimation of software [11, 13, 20]. Based on classification introduced by Boehm; we classified the effort estimation into three categories i.e. expert judgment, algorithmic estimation, and analogy based estimation. We will discuss some popular methods.

A. Algorithmic models are a mathematical models that calculate the estimation of the cost because of major cost factors that is measured as variable in the function. It's form is given as below

$$Effort_{cost} = f(p_1, p_2, \dots, p_n)$$

Where as $p_1, p_2, \dots, p_n$ represents major cost-factors

These methods are different on the aspect of selective cost factors & the structure of the considered function, *f*. A special algorithm must follows by the considered function. As the software, projects became popular; the model for estimation also developed with the need to estimation, some of which are COCOMO (Constructive Cost Models, Intermediate COCOMO and COCOMO II Khatibi et.al (2010).

B. Expertise Based Estimation is a method that majorly based on the expertise of the consultant. This method used when there is a no clear indication of the effort estimation and constraint on the data findings as well as requirement gathering.

C. Learning Oriented Techniques: Former knowledge based model helps to create effort estimation. Some of the techniques are given below:

i. **Neural Networks**: The main principle of NN is learning from any example. NN consists of three units: neurons, algorithm used for learn and the network structure. The back propagation trained feed forward network is the most popular to develop estimation model. A series of input-output will observed to estimate the result. The data will be train and will learn to decrease the prediction error. After the training completion, the arc weights for the network will be determined. Now, the inputs are assigning with a predicted value for effort of the software.

ii. **Fuzzy Logic (FL):** As human can observed the daily life and his /her, decision varies with the surrounding environmental variable. FL helps simulate this human behavior. It helps to consider all the components that may be ignore otherwise. There are four stages in the fuzzy approaches i.e. fuzzification, complexity matrix development, productivity rate and Defuzzification

iii. **Analogy-Based Estimation:** Human tends to predict things as per his own experience. The ABE main principle is to follow this approach i.e. human problem-solving approach.

It is very easy to implement. Whenever a new project will be in need to estimate for effort or size; it tries to find similarities with historical projects. The nearly project will be use to estimate of the new project.

Table 1. Software Size Estimation Methods Comparison

Type	Strengths	Weakness
Direct-Guess	Appropriate for a typical approach	No better than participants
	May or may not be accurate	No clear justification
Simple and Structures Analogy	Experienced based	historical data dependent
	All factors of the software considered	Expertized dependent
Algorithmic Model	Objective, repeatable analysable formulae	Don't Fit for a typical projects

	May celebrate to historical data	Not futuristic approach
	Easy to use for beginners	Do not consider the components that is not involve in the definition of the model

Table 2. Estimation Models classification

Type	Examples
Productivity	Average effort Conversion factor
Parametric	COCOMO SLIM
Learning-Oriented	Decision tree learning Case-Based Reasoning Neural Networks SVM
Combined Techniques	COCOMO II MP5 algorithm
Statistical	Regression models
Probabilistic	Bayesian networks
Expert-Based	Wideband Delphi Planning Game Planning Poker WBS

Now we are discussing main techniques of software estimation briefly:

A. Best Guess Method: This method is entirely base on guesswork and experience. It is very common to use. Its main disadvantage is that the guess can be highly differing from the actually cost.

B. Ad-hoc Method: Marketing personnel/client/Managerial committee set a temporary period without ant experience. It follows until the given budget runs out. Sometimes, these estimates can also limits more than 100%.

C. Experienced Based – Analogy and Experts: this technique based on the analogy and experience. It assumes that earlier projects have similar applications. The project gathered the metrics using the test done on the projects. The subject matter expert takes inputs.

D. WBS (Work Breakdown Structure) [18,25] : This method contains seven steps to effort estimation. As suggested by the name, the project under test split into different pieces. Then the large modules are dividing into small modules i.e. sub-modules. Further, sub-modules creates fragment into functionalities, the functionality to sub-functionality. Now review has done to ensure all testing requirements are included. The total number of the task should calculate. At the end, the effort and time period needed are estimated for each task.

E. Wideband Delphi Technique: This technique divides the project into functionalities. Teams of 3-4 members will be assign a task. Then every team will estimate the period in hours [12, 23, 24]. The main advantage of the technique is that it produces good result on average. The team estimates are maintained anonymity to give confidence to everyone. However, it requires management support.

F. Three Point Estimation: It is based on statistical methods. The project is broken down into tasks and tasks are further divide into subtasks. The estimation based on the three points i.e. Optimistic estimate (O), Most likely estimate (M) and Pessimistic estimate (L)

$$\boldsymbol{Test\ Estimate} = \frac{\boldsymbol{O} + (\boldsymbol{4} \times \boldsymbol{M}) + \boldsymbol{L}}{\boldsymbol{6}}$$

$$\boldsymbol{Standard\ Deviation\ (SD)} = \frac{\boldsymbol{E} - \boldsymbol{O}}{\boldsymbol{6}}$$

G. **Function Point/ Test Point Analysis:** This technique considers the perspective of the user to estimate the effort of the projects. TMap maps the function points into test points [3, 5]. It carry out the static and dynamic test points EF, PF, initial test hour, control factor and total test hours. Some components required to calculate FPs are:

- **Unadjusted data Function Points –**
  - i. Internal files
  - ii. External interface
- **Unadjusted Transaction Function Points**
  - i. User Inputs
  - ii. User Outputs
  - iii. User Inquiries
- **Carper Jones Basic Formula –**

$\#\ test\ cases = \#\ function\ points \times 1.2$

- **Total Actual Effort (TAE) :**

$$\#\ test\ cases = \frac{Percentage\ of\ Development\ Effort}{100}$$

The main advantage is that is can be available in early stage of SDLC. it is promoted by an international and independent group. But, still it has some problem. A very detail requirement needed to implement FPA/TPA. There is no explanation about the assigned value. The major problems occur while assigning value for function points.

H. **Percentage of Development Effort Method**: Different methods such as function-point and use-case-point methods are used to estimate the project efforts. The testing efforts are directly proportional to development effort. 35% of elapsed time is used in the testing process [1, 4, 14]

I. **Percentage Distribution**: This technique distributes the total percentage into the phase of the cycle of software development in percentage. This is based on past data on similar projects. The percentage of the testing phase is also further divide. In this example :

Phase	Effort (%)	Phase	Effort (%)
Project Management	8	Requirements	8
Design	17	Coding	24
Testing (All Phase)	28	Documentation	10
Installation & Training	5		

All Testing Phase	Percen_effort
Component	15
Independent	85
Total	100

System Testing	Percen_effort
Functional	65
Non-Functional	35
Total	100

Independent Testing	Percen_effort
Integration	25
System	51
Acceptance	23
Total	100

Test Planning and Design Architecture	49 %
Review	51 %
Total	100

J. **Use Case Point Estimation:** Effort estimation based on OO methodology, Gustav Kumar introduced UCP in 1993. The first step for every type of estimation is to calculate the size of activity to perform. There are six major components for determining the size of a project Nageswaran [14].

Transactions between use cases identified and suggested to use it in UCP by [21].

- Calculate Unadjusted Actor Weights (UAW)
- Determine Unadjusted Use Case Weights (UUCW)
- Compute Unadjusted Use Case Points (UUCP)

$$\boldsymbol{UUCP = UAW + UUCW}$$

- Determine Technical Environmental Factor(TEF)

$$\boldsymbol{TCF = 0.6 + (0.01 \times TFactor)}$$

- Determine Environmental Complexity Factors(ECF)

$$\boldsymbol{ECF = 1.4 + (-0.03 \times EFactor)}$$

- Calculate Adjusted Use Case Point (AUCP)

$$\boldsymbol{AUCP = UUCP * [0.65 + (0.01 \times TEF)]}$$

- Compute final effort using a conversion factor.

$$\boldsymbol{Total\ Actual\ Effort = AUCP \times CF}$$

The main advantages of the UCP are that it estimates the project effort in a very early stage. It is easy to use and the estimation is very close to the actual effort. It also does not require any experience to implement it. In addition, the results are very convincing. Nevertheless, still there are some hurdles. The requirement must write in the form of use cases. The value of the TC and EC factors must assigned with care.

III. CONCLUSION AND FUTURE SCOPE

Table 3. Summary

	SLOC	TPA	COSMIC	FPA
Artifact	Source code	requirements	requirements	requirements
Elements	Complexity and size of the application	Complexity and size of the application	Complexity and size of the application	Volume of text
Restrictions	Language specific	Req. written as use cases	-	Req. written as use cases
Availability	After implementation	After requirement specification	After requirement specification	After requirement specification
Standardized	Work in progress	Yes	Yes	yes
Deterministic	Yes	No, result depend on expert	No, result depend on expert	No, result depend on expert
Measurement Cost	None (tool for support language)	High, lot of manual work	Very High, lot of manual work and abstract method	High, lot of manual work
Calibration Cost	N/A	Significant but not requisite	Significant but not requisite	Significant but requisite
Requirement Background	None when supported by tools	Knowledge in measurement methods and tools	Knowledge in measurement methods and tools	Knowledge in measurement methods and tools
Automation	Yes	No	No	no
Validity	Can be implemented(depend on language)	Several criteria	Result depends on who is measuring	Problems similar to FPAs

This paper collectively describes different existing techniques for effort estimations. Every technique does have its own pros and cons. A comparative study with valid and large data can give us a conclusive result.

Table 4. Comparison of the Existing Methods

Method	Type	Advantages	Disadvantages
Expert Judgment	Non-Algorithmic	Fast method Implement on special projects	Entirely dependent on expert
Analogy	Non-Algorithmic	Actual experience Not expert dependent	Similar Past project needed, not always possible
Function Point	Algorithmic	Not language dependent Much better than LOC	Manual dependent Output quality is not well-thought
COCOMO	Algorithmic	Easy To Use, Commonly Used	Cannot implement on every project Detailed data needed
Neural Networks	Non-Algorithmic	Consistent and reliable Perception authority	No design guidelines Performance depends on training
Fuzzy	Non-Algorithmic	No training, Imprecise data representation Flexible	Difficult maintenance Not easy to use

IV. REFERENCES

[1]. B. Beizer, "Software testing techniques (2nd ed.)". Van Nostrand Reinhold Co., New York, NY, 1990

[2]. Borade J. G., Khalker V. R.. Software Project Effort and Cost Estimation Techniques, International Journal of Advanced Research in Computer Science and Software Engineering, vol.3, issue 8, pp 730-739. ISSN: 2277 128X, Available online at: www.ijarcsse.com.

[3]. Fischman, L. "Evolving Function Points" Crosstalk: The Journal of Defence Software Engineering. 2001. Volume 14, issue 2. Pp24-27.

[4]. Harrold M., "Testing: a Roadmap", In Future of Software Engineering, 2nd International Conference on Software Engineering, pp.66-72. 2000.

[5]. Hemmastra F., and Kusters R., "Function point analysis : evaluation of a software cost estimation model" European Journal of Information Systems. 1991. Vol 1, No 4. Pp229-237.

[6]. Hihn J. and Habib-agahi H., "Cost estimation of software intensive projects: a survey of current practices," 1991 Proceedings] 13th International Conference on Software Engineering, Austin, TX, 1991, pp. 276-287,doi: 10.1109/ICSE.1991.130653

[7]. http://www.eurostarconferences.com/community/member/eurostar-presentations-archieve/test-effort-estimation -with-test-point-analysis.aspx

[8]. http://www.project.ed.ac.uk/methodologies/Full-software-project-template/estimation-guidelines.shtml

[9]. Karner G., , "M. E. Khatib, K. Shuaib," Practical Software Project Total Cost Estimation Methods", MCIT 10, IEEE, 2010.

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

[11]. Khatibi V., Jawawi D. N. A., Software Cost Estimation Methods: A Review, Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1 pp 21-29. 2010-11. ISSN 2079-8407.

[12]. Leung H., Fan Z., "Software Cost Estimation", Article 2001.

[13]. Li Y Y. F., Xie M. and Goh T. N., "A study of genetic algorithm for project selection for analogy based software cost estimation," 2007 IEEE International Conference on Industrial Engineering and Engineering Management, Singapore, 2007, pp. 1256-1260. doi: 10.1109/ IEEM.2007. 4419393.

[14]. Naeshwaran, S., "Test Effort estimation using use case points". Quality Week, San Fracisco, California, USA, 2001

[15]. Nagar C., "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

[16]. Nagar C., Dixit A."Software efforts and cost estimation with systematic approach", IJETCIS, ISSN:2079-8407, Vol.2 No.7, July 2011.

[17]. Pressman R. S., , "Software Engineering: A Practioner's Approach", 6th Edn., McGraw-Hill New York, USA., ISBN: 13: 9780073019338,2005.

[18]. Pritchard Cl L..,"How To Build a Work Breakdown Structure". ISBN 1-890367–12-5, 1998.

[19]. Project Management Institue, "Practice Standard for Work Breakdown Structures (Second Edition)", ISBN 1933890134, 2006.

[20]. Qingzhang C., Shuojin F., Wenfu W., "Development of the Decision Support System for Software Project Cost Estimation", World Congress on Software Engineering,IEEE, 2009.

[21]. Robiolo G., Orosco R., "Employing use cases to early estimate effort with simpler metrics", innovations in Systems and Software Engineering 4 (1) (2008) 31-43.

[22]. Standish Groups, "Chaos Report" 1995, CHAOS, Dennis,1995

[23]. Stellman, A. and Greene, J. "Applied Software Project Management". O'Reilly Media. ISBN 0-596-00948-8, Nov 2005.

[24]. Stepien B, "Software Development Cost Estimation Meth- ods and Research Trends",Computer Science, Vol.5,2003.

[25]. Swiderski., M. "PMBOK-Work Breakdown Structures", June 2013.