Volume 4, Issue 5, May 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Comparison of Cost Estimation Technique

Jyoti Rani
Pursuing M.Tech
#MANAV RACHNA INTERNATIONAL UNIVERSITY, FARIDABAD

Rachna Behel
Assistant Professor
#MANAV RACHNA INTERNATIONAL UNIVERSITY, FARIDABAD

ABSTRACT-:- In the software engineering researchers have proposed many cost estimation techniques. This research paper shows review of mainly three cost estimation techniques (function point, use case point and line of code). These technique shows that how we can calculate the cost of software according to the requirements. To Predict cost of software is very complex task for software developers. more accurate cost estimation helps in completion of software in limited time period and budget. For this task developers must have knowledge about all modes and skill which are used in cost estimation

.

Keywords—line of code(LOC),Function point ,Use case point

I. INTRODUCTION:

Software cost estimation strongly belongs to how long and how many people are required to complete a software project. The estimation process includes overall cost of the project. Software development has become an essential question because many projects are still not completed on schedule, with under or over estimation of efforts leading to their own particular problems. Therefore, in order to manage budget and schedule of software projects , various software cost estimation models have been developed. Accurate software cost estimates are critical to both developers and customers.They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control.

- It can help to classify with respect to an overall business plan.
- It can be used to determine what resources to commit to the project and how well these resources will be used.
- It can be used to assess the impact of changes and support preplanning.
- Projects can be easier to manage and control when resources are better matched to real needs.
- Customers expect actual development costs to be in line with estimated costs.

In the cost estimation of software we have to face lot of problems given below:

- A big problem in any estimate is to understand and define the system to be estimated.
- A software cost estimate done early in the project life cycle is generally based on less precise inputs and less detailed design specifications.
- Software development include many interrelated factors, which affect development effort and productivity, and whose relationships are not well understood.
- Lack of a historical database of cost measurement it means historical data is sometimes incomplete, inconsistent, or inaccurate.
- Lack of trained estimators and estimators with the necessary expertise. Software is intangible, invisible, and intractable so it is more difficult to understand and estimate a product or process that cannot be seen and touched.
- While too low effort estimates may lead to project management problems, delayed deliveries, budget overruns and low software quality, too high effort estimates may lead to lost business opportunities and inefficient use of resources.

II. Cost Estimation Methods

We use generally two methods: algorithmic and non-algorithmic. Algorithmic methods use a formula to calculate the software cost estimate. Non-algorithmic methods do not use a formula to calculate the software cost estimate. The main aim of this paper is to provide a review of these existing models and methods.

A. Non Algorithmic Based Estimation Methods:

1. Expert Judgment Method: These techniques involve consulting with software cost estimation experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. To provide a satisfactorily broad communication bandwidth for the experts to exchange the volume of information necessary to calibrate their estimates .

The estimating steps:

a. Coordinator presents each expert with a specification and an estimation form.
b. Experts fill out forms anonymously
c. Coordinator calls a group meeting in which the experts discuss estimation issues with the coordinator and each other.
d. Coordinator prepares and distributes a summary of the estimation on an iteration form.
e. 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. It has been highly successful in combining the free discuss advantages of the group meeting technique.

2. Estimating by Analogy: It 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.

The estimating steps :

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 : 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 . The leading method using this approach is Putnam model. This method is more applicable to early cost estimation when only global properties are known.

4. Bottom-up Estimating Method: By this 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 [22].

5. Parkinson's Law : The cost is determined by the available resources rather than based on an objective assessment. If the software has to be delivered in 12 months and 5 people are available, she effort is estimated to be 60 person-month . Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates.

6. Price-to-win: The estimation is based on the customer's budget instead of the software functionality. For example, if a reasonable estimation for a project costs 100 person-months but the customer can only afford 60 person-months, it is common that the estimator is asked to modify the estimation to fit 60 person -months effort in order to win the project. This is again not a good practice since it is very likely to cause a bad delay of delivery or force the development team to work overtime.

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.

1. COCOMO Model (Constructive Cost Model) is widely used algorithmic software cost model. It was proposed by Boehm [2] [4]. It has following hierarchy-

Model 1 (Basic COCOMO Model):- It computes software development effort and cost as a function of program size expressed in estimated lines of code (LOC) .

The basic steps in this Model are:-

a. Obtain an initial estimate of the development effort from the estimate of thousands of delivered lines of source code.
b. Determine a set of 15 multiple factors from different attributes of the project.
c. Adjust the effort estimate by multiplying the initial estimate with all the multiplying factors .

EFFORT = a* (KLOC)b The value of constants a and b depend on the project type.

Model 2 (Intermediate COCOMO Model):- Intermediate COCOMO Model computes software development effort as a function of program size and set of ―cost drivers‖ that include subjective assessment of the products, hardware, personnel and project attributes.

The basic model is extended to consider a set of ―cost driver attributes‖ that can be grouped into four major categories:

1. Product attributes

a. Required software reliability
b. Size of application data base
c. Complexity of the product

2. Hardware attributes

a. Run-time performance constraints
b. Memory constraints
c. Volatility of the virtual machine environment
d. Required turnaround time

3. Personnel attributes

a. Analyst capability
b. Software engineer capability
c. Applications experience
d. Virtual machine experience
e. Programming language experience

4. Project attributes

a. Use of software tools

b. Application of software engineering methods

c. Required development schedule each of the 15 attributes is rated on a 6 point scale that ranges from ―very low to ―extra high (in importance or value). Based on the rating, an effort multiplier is determined from tables published by Boehm, and the product of all effort multipliers results is an *effort adjustment factor* (EAF).

Model 3 (Detailed COCOMO Model)

The detailed COCOMO Model incorporates all characteristics of the intermediate version with an assessment of the cost driver‘s impact on each step (analysis, design, etc) of the software engineering process.

2. COCOMO II model

It is a collection of three variants, Application composition model, Early design model, and Post architecture model. This is an extension of intermediate COCOMO model [14] and defined as:-

EFFORT = 2.9 (KLOC) 1.10 .

3. SEL – Model

The Software Engineering Laboratory (SEL) of the University of Maryland has established a model i.e. SEL Model for estimation [18].

Estimation of effort according to SEL model is defined as follows:- EFFORT = 1.4 * (Size) 0.93, Duration D = 4.6 (KLOC) 0.26 (4) Effort (Person-Months) and lines of code (size in thousands of lines of code i.e. KLOC) are used as predictors.

4. Walston-Felix Model

Walston and Felix (1977) developed their effort model from a various aspects of the software development environment such as user database of sixty projects collected in IBM's Federal Systems division. This model constitutes participation, customer-oriented changes, memory constraints etc. According to Walston and Felix model, effort is computed by [26] [14]:

EFFORT = 5.2 (KLOC) 0.91, Duration D = 4.1 (KLOC) 0.36

5. Bailey-Basil Model

This model developed by Bailey-Basil between delivered lines of source code and formulates a relation . EFFORT = 5.5 (KLOC) 1.16

6. Halstead Model

This model developed by Halstead between delivered lines of source code and formulates a relation .

EFFORT = 0.7 (KLOC) 1.50

7. Doty (for KLOC > 9) This model developed by Doty between delivered lines of source code and formulates a relation [18] EFFORT = 5.288 (KLOC) 1.047

8. Putnam Model

Putnam model 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).

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

9. 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.

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%.

The collection of function point data has two primary motivations. One is the desire by managers to monitor level of productivity. Another use of it is in the estimation of software development cost.

DISADVANTAGES OF FUNCTION POINT ANALYSIS

(A) Function points require subjective measurement with lot of judgement involved.

(B)Lots of trial and cost skills are based on line of code, so that function point required to be converted.

(C)Lack of searched data is exist on function points as compared with line of code.

(D)This is the task that can perform after creation of design specifications.

. **Use Case Point (UCP)**

Test effort estimation using UCP 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 behavior. 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. UCP is comprised of six basic steps that determine a projects required test effort:

i) Calculate Unadjusted Actor Weights (UAW)

It is the sum of all actors multiplied by corresponding actor weights, based on the actor type, By following formula

UAW = Actor Weight * Actortype.

- Simple actors(a weight factor of 1) are system actors that communicate through an API.
- Average actors(a factor of 2) are system actors that communicate through a protocol or data store.
- Complex actor(a factor of 3) are human actors that intract normally through a GUI or other human interface.

ii) Determine Unadjusted UC Weights (UUCW). This is the weighted sum of all the use cases.

For calculate the use case points we use the formula:

UCPs= (UUCW+UAW)*TCFs*ECFs

For example, for a project with a UUCW of 50, UAW of 10,1.02 TCFs, and 1.04 ECFs,

UCPs= (50+10)*1.02*1.04=63.648

So, applying a PF of 20 would yield an estimate of 1,272.96 person-hours for the project.

DISADVANTAGES OF USE CASE POINT

(A)Not usable during initial stage.

(B)No standard use case document.

(c)Maintenance estimation problems.

(D)Actor identification need technical details.

LINES OF CODE

This general approach is subdivided into two different areas: SLOC(Source Lines of Code, and SDI(Source Delivered Instructions).The difference between these two is that the first, SLOC takes into account all the housekeeping which must be done by the developer, such as headers and embedded comments. The second, SDI, Only takes into account the number of executable lines of code.

The best known technique using LOC (Line of code) is the COCOMO(constructive cost model),developed by Boehm. This model , along with other SLOC/SDI based model, uses not only the LOC, but also other factor such as product attributes, hardware limitations, personnel, and development environment. These different factors lead to one or more "adjustment" factors which adjust the direct evaluation of the effort needed.In COCOMO's case, there are fourteen such .

(A)Lack of accountability.

(B)Lack of cohesion with functionality.

(C)Adverse impact on estimation.

(D)Developer's experience.

(E)Difference in languages.

(D)Advent of GUI tool.

(E)Problem with multiple languages.

(F) Lack of counting standards.

(G)psychology.

III. Conclusion

Function point , loc and use case point analysis are three cost comparison techniques. In these case use case technique are good as compare to function point and loc. The loc is not very widely used. And use case point and function point is very widely used.

REFERENCES

[1] G.N. Parkinson, ―Parkinson's Law and Other Studies in Administration‖, Houghton-Miffin, Boston, 1957.

[2] 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.

[3] L.H. Putnam, ―A general empirical solution to the macro software sizing and estimation problem,‖ IEEE Transactions on Software Engineering, pp. 345–361, , July 1978.

[4] B.W. Boehm, ―Software Engineering Economics,‖ Prentice- Hall, Englewood Cliffs, NJ, USA, 1981.

[5] 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.

[6] Chris F.K, ―An Empirical Validation of Software Cost Estimation Models‖, Management Of Computing-Communications of ACM, Vol: 30, No. 5, pp.416-429, , May 1987.

[7] Kemerer, C. ―An empirical validation of software cost estimation models‖, Communications of the ACM, 30(5), 416-429. doi: 10.1145/22899. 22906, 1987.

[8] Liming Wu ―The Comparison of the Software Cost Estimating Methods‖ University of Calgary.

[9] H. Agahi, S. Malhotra and J. Quirk, ―Estimating Software Productivity and Cost for NASA Projects‖, Journal of Parametrics, pp. 59-71, 1998.

[10] S. Chulani, B. Boehm and B. Steece, ―From Multiple Regression to Bayesian Analysis for Calibratuing COCOMO‖, Journal of Parametrics, vol. 15(2), pp. 175-188, 1999.

[11] ―COCOMO II Model definition manual‖, version 1.4, University of Southern California.

[12] S. Devnani-Chulani, "Bayesian Analysis of Software Cost and Quality Models‖. Faculty of the Graduate School, University Of Southern California May 1999.

[13] Leungh, Zhangf, ―_Software cost estimation ‖ in _Handbook of software engineering and knowledge engineering‘ (World Scientific Pub. Co, River Edge, NJ, 2001)

[14] Y. Singh, K.K. Aggarwal, Software Engineering Third edition, New Age International Publisher Limited New Delhi.

15] M. Lefley and M. J. Shepperd, ―Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets‖, LNCS, Genetic and Evolutionary Computation — ISBN: 978-3-540-40603-7, page-208, GECCO 2003.

[16] Murali Chemuturi, **"**Analogy based Software Estimation,**"** Chemuturi Consultants.

[17] Moløkken, K., & Jørgensen, M.‖ A review of surveys on software effort estimation. International Symposium on Empirical Software Engineering‖, 223-231. Retrieved from ACM Digital Library database, 2003.

[18] O. Benediktsson and D. Dalcher, ―Effort Estimation in incremental Software Development,‖ *IEEE Proc. Software*, Vol. 150, no. 6, pp. 351-357, December 2003.

[19] K. Moløkken-Østvold et al., *Project Estimation in the Norwegian Software Industry - A Summary*. 2004, Simula Research Laboratory.

[20] Andriano L.I. Oliveira, ―Estimation of Software Project Effort with Support Vector Regression‖, www.journals.elsevier.com/neurocomputing, Vol.- 69, Issues 13–15, pp. 1749–1753, August 2006.

[21] Galorath, D. D., & Evans, M. W. ―Software sizing, estimation, and risk management: When performance is measured performance improves‖. Boca Raton, FL: Auerbach, 2006.

[22] Caper Jones, ―Estimating software cost‖ tata Mc- Graw -Hill Edition 2007.

[23] Magne J and Martin S, ― A Systematic Review of Software Development Cost Estimation Studies ‖, IEEE Transactions On Software Engineering, Vol. 33, No. 1, pp. 33-53, January 2007.

[24] A. S. Andreou, E. Papatheocharous, ― Software Cost Estimation using Fuzzy Decision Trees‖, 23rd IEEE/ACM International Conference on Automated Software Engineering, pp. 371 - 374, 2008.

[25] K. Vinaykumar, V. Ravi, M. Carr and N. Rajkiran, ―Software cost estimation using wavelet neural networks,‖ Journal of Systems and Software, pp. 1853-1867, , 2008.

[26] Pressman. Software Engineering - a Practitioner‘s Approach. 6th Eddition Mc Graw Hill international Edition, Pearson education, ISBN 007 - 124083 – 7.

[27] Marcel Korte, Dan Port, ―Confidence in software cost estimation results based on MMRE and PRED,‖ *PROMISE'08, Leipzig, Germany*, May 12-13, 2008.

[28] 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

[29] C. S. Reddy, K. Raju, ― An Improved Fuzzy Approach for COCOMO‘s Effort Estimation using Gaussian Membership Function‖, Journal of Software, VOL. 4, NO. 5, pp. 452-459, 2009.

[30] M.V. Deshpande and S.G. Bhirud, ―Analysis of Combining Software Estimation Techniques,‖ *International Journal of Computer Applications* (0975 – 8887) Volume 5 – No.3, 2010

[31] J. S. Pahariya, V. Ravi, M. Carr, M. Vasu, ―Computational Intelligence Hybrids Applied to Software Cost Estimation‖, International Journal of Computer Information Systems and Industrial Management Applications (IJCISIM),Vol. 2, pp. 104-112, 2010.

[32] Prasad Reddy P.V.G.D, Hari CH.V.M.K and Srinivasa Rao, ―Multi Objective Particle Swarm Optimization for Software Cost Estimation,‖ International Journal of Computer Applications, Vol.-32, 2011.

[33] Sweta Kumari and Shashank Pushkar, ―Comparison and Analysis of Different Software Cost Estimation Methods‖, International Journal of Advanced Computer Science and Applications, Vol. 4, No.1, 2013.

