A Review on Cost Estimation Models for Effort Estimation

Tajinder Kaur, Jaspreet Singh

Abstract— Software cost estimation is an important phase in software development. It predicts the amount of effort and development time required to build a software system. It is one of the most critical tasks and an accurate estimate provides a strong base to the development procedure. There are many cost estimation models for the estimation of the software. COCOMO model is the basic model. In this paper,we have discussed about various cost estimation models and their limitations.

Index Terms— COCOMO, KLOC, Cost estimation, Function Points, Person-month,Effort Estimation.

1 INTRODUCTION

Software engineering is an engineering discipline that is concerned with all aspects of software production[1]. Useful software systems often have a very long lifetime. For example, large military or infrastructure systems, such as air traffic control systems, may have a lifetime of 30 years or more. Business systems are often more than 10 years old. Software cost a lot of money so a company has to use a software system for many years to get a return on its investment[1]. Obviously, the requirements of the installed systems change as the business and its environment change. Therefore, new releases of the systems, incorporating changes, and updates, are usually created at regular intervals.

There are many sub-disciplines of software engineering which are as follow:

a) **Software Requirement:** A requirement specification is the complete description for the behavior of the system. It may be defined according to the system specification.

b) **Software Design:** It is a problem solving process and plan for solutions. Dataflow Diagrams and Flowcharts, are comes under software design. In this SRS document are transform into design form using some tools.

c) **Software Construction:**It is a formation, functioning which is in depth manner and in the construction process we can define methods of the process and its description. It helps to improve software quality.

d) **Software Testing:** It checks whether the expected results match with the actual results. It is a process to identify correctness, completeness and effectiveness of computer software.

e) **Software Maintenance:** It is the alteration of the software products for their correction after the delivery to correct faults, error and bugs in software.

f) **Software Configuration Management** : It provides the auditing, changes and report to the changes that are made. Thus, SCM is a change management.

• *Tajinder Kaur is currently pursuing Masters Degree program in Computer Science & Engineering in Chandigarh University,Mohali,Punjab,India. E-mail: er.tajinder1991@gmail.com*
• *Jaspreet Singh is working as an Assistant Professor in Computer Science & Engineering at Chandigarh University,Mohali,Punjab,India. E-mail: jaspreet.rajal@gmail.com*

g) Software Engineering Process: It is a process which is used to increase the quality of the software in the form of various factors like flexibility, testability, portability, usability, understandability, efficiency etc [2]. There are many types of process model which are considered in these categories waterfall model, prototyping model, spiral model etc.

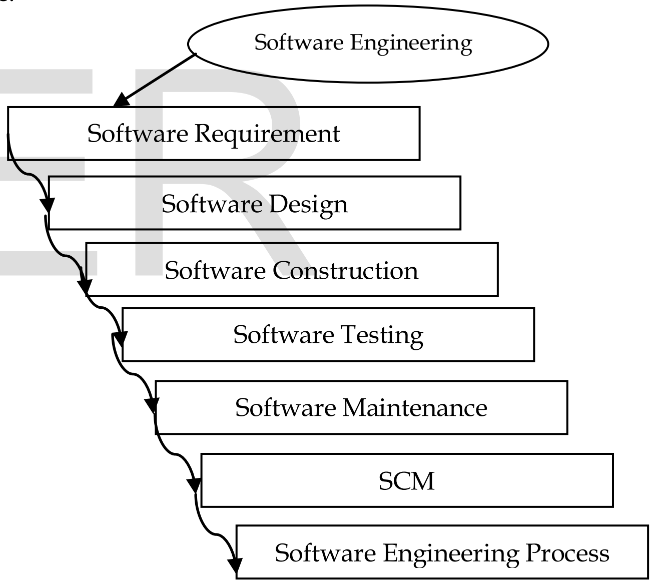

Fig1:Sub-disciplines of Software Engineering

Software Estimation: Software development effort estimation Software development effort estimation is one of the most major activities in software project management. A number of models have been proposed to construct a relationship between software size and effort; however there are many problems. This is because project data, available in the initial stages of project is often incomplete, inconsistent, uncertain and unclear. Effort estimates may be used as input to project plans, iteration plans, budgets, investment analyses, pricing processes so it becomes very important to get accurate estimates [3].

2 LITERATURE SURVEY

Chetan Nagar et al [4] “Effort Estimation by Combining the Use Case Point and COCOMO”, in this paper they combine the Use Case point and COCOMO. They predict the Line of Code with the help of Use Cases.To estimate the KLOC divide the project into module and module into the sub module until we are able to estimate the KLOC. It is concluded that Use Case used in the paper must be more specific not more generalized.

Ales Živkovič et al [5] “Automated Software Size Estimation based on Function Points using UML Models”, In this paper, they proposed the unified mapping of UML models into function points. The mapping is formally described to enable the automation of the counting procedure. Three estimation levels are defined that correspond to the different abstraction levels of the software system. The level of abstraction influences an estimate's accuracy. It is based on a small data set, proved that accuracy increases with each subsequent abstraction level. Changes to the FPA complexity tables for transactional functions will also be proposed in order to better quantify the characteristics of object-oriented software.

Bingchiang Jeng et al [6] **“**A Specific Effort Estimation Method Using Function Point”, this research suggests a different approach that simplifies and tailors a generic function point analysis model to increase ease of use. The proposed approach redefines the function type categories in the FPA model, on the basis of the target application’s characteristics and system architecture. This method makes the function types more suitable for the particular application domain. It also enables function point counting by the programmers themselves instead of by an expert. An empirical study using historical data establishes the regression model and demonstrates that its prediction accuracy is comparable to that of a FPA model.

Nancy Merlo[7]"Constructive Cost Model", introduced about COCOMO model and its sub parts and its estimation formulae. Software cost estimation is an important part of the software development process. COCOMO suite including all models offers a powerful instrument to predict software costs. Unfortunately not all of the extensions are already calibrated and therefore still experimental. Only the Post-Architecure model is implemented in a calibrated software tool. Despite this disadvantage the COCOMO II suite helps managing software projects. It supports process improvement analyses, tool purchases, architecture changes, component make/buy tradeoffs and decision making process with credible results. Many endeavors were done to measure up to the changes in software life cycles, technologies, components, tools, notations and organizational cultures since the first version of COCOMO (COCOMOI, COCOMO 81).

Zhang Dan[8] “Improving the accuracy in Software Effort Estimation”, The modified model increases the convergence speed of artificial neural network and solves the problem of artificial neural network’s learning ability that has a high dependency of the network initial weights. This model improves the learning ability of the original model and keeps the advantages of COCOMO model.

Felfernig and A. Salbrechter[9],"Applying Function Point Analysis to Effort Estimation in configurator Development" presented about the Knowledge-based configuration which is a successful function of Artificial Intelligence approaches in industry .The increasing complexity of configurable products and services necessitated improved expressiveness and maintainability of knowledge representation languages empowering the development and maintenance of large and complex configuration knowledge bases. Within the context of such configuration projects, the effectual integration of effort estimation techniques allowing for the peculiarities of configuration system development is still an open issue. They also discussed the appliance of Function Point Analysis (FPA) in the framework of knowledge-based configuration projects along with present a framework for a company-specific implementation. The application and extension of Function Point Analysis (FPA) for effort estimation in the development of knowledge-based configuration systems. A framework for a company-specific application has been presented which reduces prediction error rates compared to the application of conventional FPA approaches. Using this approach effort further work will include the analysis of domain-specific complexity.

Anupama Kaushik [10] “COCOMO Estimates Using Neural Networks”, it predicts the amount of effort and development time required to build a software system. It is one of the most critical tasks and an accurate estimate provides a strong base to the development procedure. In this paper, the most widely used software cost estimation model, the Constructive Cost Model (COCOMO) is discussed. The model is implemented with the help of artificial neural networks and trained using the perceptron learning algorithm. The COCOMO dataset is used to train and to test the network. The test results from the trained neural network are compared with that of the COCOMO model.

Charles R Symons[11] “Function Point Analysis: Difficulties and Improvements”.It described the function point analysis on the basis of Mark 11 approach.The results of some measurements of actual systems to calibrate the Mark I1 approach, and conclusions on the validity and applicability of function point analysis is discussed**.**The experience of applying Albrecht’s Function Point Method and the alternative Mark I1 approach to a variety of systems has led to three groups of conclusions:
Albrecht versus Mark I1 Function Points,Use of function points for productivity measurement,Limitations of function points.

Erik Stensrud [12] “Estimating with Enhanced Object Points vs. Function Points”, This paper describes EOP and compares the two metrics and points out some reasons why many practitioners may prefer Enhanced Object Points to Function Points.

Iman Attarzadeh et al [13]“Soft Computing Approach for Software Cost Estimation”,it concluded that greatest challenges for project managers is to predict the development effort for a software system. Most of the traditional techniques such as function points, regression models, COCOMO, etc, require a long-term estimation process. One of the new approaches that called soft computing techniques may offer an alternative for this challenge. This paper described an enhanced soft computing model for the estimation of software cost and time estimation. Result shows that the value of MMRE (Mean of Magnitude of Relative Error) applying soft computing was substantially lower than MMRE applying by algorithmic models which was previously used.

3 Models for Cost Estimation

There are different types of model for cost estimation[4]. These are as follow:

a) COCOMO
b) Use Case Point Estimation
c) Function Point Based Estimation
d) Expert Judgment
e) Estimation by Analogy

3.1 COCOMO

COCOMO is Constructive Cost Model. It is very effective and oldest model for cost estimation. It is independent model which is well documented and cannot be depended upon any software vendor .In cocomo model line of code is estimated. In this model we can understand the complexity of the system because of its openness nature. This model is constructed to evaluate the cost estimation of the software development. There are three levels in cocomo model [14]:

- Basic Cocomo
- Intermediate Cocomo
- Detailed Cocomo

3.1.1 Basic COCOMO

It computes software development cost and effort as a function of program size. It is static and single valued model. There are three modes within Basic COCOMO:

a) Organic Mode
b) Semidetached Mode
c) Embedded Mode

a) Organic Mode: In this mode development team is small and is consist of experienced persons. Here projects are not complicated. It requires less than rigid requirements.

b) Semidetached Mode: In this mode people are more experienced than organic level. This mode is more complicated than organic mode so complexity is more. It has characteristics of both modes organic and embedded. It requires rigid and less than rigid requirements.

c) Embedded Mode: In this mode software and hardware are complexly joined. It requires set of rigid requirements. It is a combination of organic and semi-detached projects.

Table 1: Classification of Basic COCOMO

Basic COCOMO	a	b	c
Organic	2.4	1.05	0.38
Semi-detached	3.0	1.12	0.35
Embedded	3.6	1.20	0.32

$$\text{Effort}=a\text{X}(\text{KLOC})^{b}\ \text{PM}$$
$$\text{Development Time},T=2.5(\text{Effort})^{c}\ \text{Months}$$

a,b,c values varires according to type or classes of software products.

3.1.2 Intermediate Cocomo

It is an addition to the basic model that computes software development effort by adding a set of cost drivers. In this 15 cost drivers are used to find out cost estimation of projects rated from very low to very high.

3.1.3 Detailed Cocomo

It is an extension of intermediate cocomo. In this cost driver is added effort multiplier at each phase to calculate the cost drivers. It uses different multiplier for each cost attribute. Cocomo 1 is also known as cocomo 81.

3.1.4 Limitations of COCOMO

There are some limitations of this model which are as follows:

1. COCOMO starting estimation from the design phase and till the end of integration phase of cost and schedule of the project. A separate estimation model should be used for remaining phase.

2.Assumptions made at the starting in this model may vary as time progresses in developing the project. It is not a realistic perfect model.

3.A new estimation may show over budget or under budget for the project when to revise the cost of the project. This may lead to a partial development of the system.

3.1.5 COCOMO2

This model simplified the cost estimation of by reducing number of parameters from 15 to 7. It suggest using functional point at initial phase and LOC is used at later phase. The parameters which used in cocomo 2 [15] are totally different from its typical value. COCOMO 2 models have two types of parameters set. First is external set and it can be matched to matrix view loosely. The vocabulary of the model can be used easily while dealing with stakeholders. The second set is internal which is used in different purpose than first set. There are many tools which are available in market which calculated the appropriate results for cost estimation. COCOMO 2 model preserve the originality of cocomo model i.e. openness of the cocomo.

There are three stages which are available in cocomo 2.

First stage is follows the prototyping model with the help of application model capability.

The next phases normally occupy investigation of architectural alternatives or incremental development strategies. When project is ready to develop then it should have a life- cycle architecture, which provide more exact information on cost driver inputs and gives more accurate cost estimates.

Table 1: Comparision of Basic COCOMO & COCOMO2

Basic COCOMO	COCOMO 2
It is Basic Model.	It is extension to basic model.
It follows waterfall model.	It follows three phase concept.
There are 15 cost drivers in basic COCOMO .	There are only 7 cost drivers in COCOMO2.
It follows reengineering concept.	It follows reusabilty concept.

3.2 USE CASE POINT ESTIMATION

A functional scope of the system is defined by the use case point. It serves as a top-down approach. It is well suited for project estimation and planning. This estimation method counts the number of transactions in each use case. A transaction is an event occurring between the system and an actor [16].The main steps of use case methods are as follow:

1.The use case actor can be simple, average and complex. Simple actor represents another system with defined API. An average actor represents another system while interacting with TCP/IP. A complex actor interacts through web pages and graphical users.

2.The use cases are also categorized as simple, average or complex categories depending on the number of transactions .A simple use case has 3 or fewer transactions ,an average use case has 4 to 7 transactions, and a complex use case has more than transactions.

3. Use case points are adjusted on the basis of value assigned to technical factor or environmental factors. Each factors value assigned from 0 to 5.

4. The environmental factors should decide the number of staff hours per use case point.

There are two methods of use case point estimation. First method of use case model is basis of counting function points to obtain an estimate of efforts based. Second method is based upon the count the line of code for estimation.

3.3 FUCTION POINT BASED ESTIMATION

FPA is an ISO organized which is used to find out the functional size of the system. The functional size indicates the amount of functionality that is important. It is not dependent upon any technology which used to implement the system. FPA express the size of the information in a number of function units[17]. So its measurement unit is function units.
The functional size may be used:
1. To enhancement costs and budget the development applications.
2. To plan the costs of the application portfolio under annual maintenance.
3. For cost estimation determine the size of the software.
4.After completion of the project to find out project pr oductivity.
A simplifies function point can be used to estimate the project size and team size.
Function Points based upon the five following parameters:
1.Input: It can be dialog-box, screens and forms which can be deleted or changed by the end user and other users.
2.Output: It include message, graphs, screens generated by an application for an end user and other users. It can process, combine and summarize the complex data and simplified it.
3.Inquiries: When we give input process apply on it and give output as a result. For the search of specific data inquiry which is used as a key to create simple out. It is used to retrieve from the database directly.
4.Logical Internal Files: The logical group or data which is controlled by the application. It is a single file or we can say flat file or table file in a relational database.
5.External Interface file: The file which is handled by another application having interaction with this application also. It includes major logical groups.
Graphical representation is necessary in it with the help of data flow diagrams.

3.4 EXPERT JUDGEMENT

Expert judgment is one of the most widely used estimation techniques. In this approach, an expert makes an educated guess of the problem size after analyzing the problem thoroughly[14]. Usually, the expert estimates the cost of the different components (i.e. modules or subsystems) of the system and then combines them to arrive at the overall estimate. However, this technique is subject to human errors and individual bias. Also, it is possible that the expert may overlook some factors inadvertently. Further, an expert making an estimate may not have experience and knowledge of all aspects of a project. For example, expert may be conversant with the database and user interface parts but may not be very knowledgeable about the computer communication part. A more refined form of expert judgment is the estimation made by group of experts. Estimation by a group of experts minimizes factors such as individual oversight, lack of familiarity with a particular aspect of a project, personal bias, and the desire to win contract through overly optimistic estimates. However, the estimate made by a group of experts may still exhibit bias on issues where the entire group of experts may be biased due to reasons such as political considerations. Also, the decision made by the group may be dominated by overly assertive members. Delphi cost estimation approach tries to overcome some of the shortcomings of the expert judgment approach. Delphi estimation is carried out by a team comprising of a group of experts and a coordinator.

3.5 ESTIMATING BY ANALOGY

Estimating by analogy means comparing the proposed project to previously completed similar project where the project development information is 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. Estimating by analogy is relatively straightforward. Actually in some respects, it is a systematic form of expert judgment since experts often search for analogous situations so as to form their opinion about similiar projects.

4 PROBLEM FORMULATION

COCOMO model is software cost estimation model of software development. The model use regression formula to estimation cost using historical data with present and future characteristics. COCOMO starts estimate from the design phase to integration phase of cost and schedule of the project. But separate estimation model should be required for remaining phase. So COCOMO model is not accurate. A simplified function point can be used to estimate the project size and team size. Function point estimation is performed after design creations. Many efforts and cost models which are based upon LOC so function points are required to be converted. It required less research data as compared to LOC. So it is more accurate than COCOMO. In the proposed work, to enhance the accuracy of the estimation model it is required to merge both COCOMO and function point estimation models i.e. “Effort Estimation Using Hybrid Technique”. As a result the model which we will get after merging will be better than the existing models and it will provide better result than other models.

5 Conclusion

Software engineering has many sub-disciplines. Software cost estimation is the major sub-discipline. There are many techniques avaliable for the cost estimation but COCOMO is the basic cost estimation model.COCOMO model has some disadvantages that it depends upon KLOC only. COCOMO starting estimation from the design phase and continues till the end of the integration phase of the project. It is not a realistic and perfect estimation model.By merging, two or more cost estimation techniques accuracy of the software cost estimation can be improved.

References

[1] Sommerville, Ian. "Software Engineering", Addison Wesley, 9th edition ,2011.

[2] Fakhar Lodhi, “Software Engineering An Introduction” 2000.

[3] Ramesh, K., and P. Karunanidhi. "Literature Survey on Algorithmic and Non-Algorithmic Models for Software Development Effort Estimation." International Journal of Engineering and Computer Science ISSN: 2319-7242.

[4] Nagar Chetan, and Anuragh Dixit. "Efforts Estimation by Combining the Use Case Point and COCOMO." Proc. of International Journal of Computer Applications (0975–8887) Volume (2012): 1-5.

[5]] Živkovič, Aleš, Ivan Rozman, and Marjan Heričko. "Automated software size estimation based on function points using UML models." Information and Software Technology 47, no. 13 (2005): 881-890.

[6] Jeng, Bingchiang, Dowming Yeh, Deron Wang, Shu-Lan Chu, and Chia-Mei Chen. "A Specific Effort Estimation Method Using Function Point." Journal of Information Science and Engineering 27, no. 4 (2011): 1363-1376.

[7] Nancy Merlo – Schett, “Constructive Cost Model”, Seminar on Software Cost Estimation,WS 2002 / 2003

[8] Zhang Dan, " Improving the accuracy in software effort estimation: Using artificial neural network model based on particle swarm optimization", International Conference on Service Operations and Logistics, and Informatics (SOLI) 2013.

[9] A. Felfernig, A. Salbrechter, “APPLYING FUNCTION POINT ANALYSIS TO EFFORT ESTIMATION IN CONFIGURATOR DEVELOPMENT” , 2003

[10] Kaushik, Anupama, Ashish Chauhan, Deepak Mittal, and Sachin Gupta. "COCOMO Estimates Using Neural Networks." International Journal of Intelligent Systems and Applications (IJISA) 4, no. 9 (2012): 22.

[11] Symons, Charles R. "Function point analysis: difficulties and improvements."Software Engineering, IEEE Transactions on 14, no. 1 (1988): 2-11.

[12] Stensrud, Erik. "Estimating with enhanced object points vs. function points." InProceedings, 13 th COCOMO/SCM Forum. 1998.

[13] Attarzadeh, Iman, and Siew Hock Ow. "Soft computing approach for software cost estimation." Int. J. of Software Engineering, IJSE 3, no. 1.

[14] Mall, Rajib, "Fundamentals of Software engineering", PHI Learning Pvt. Ltd., 2014.

[15] http://www.codeproject.com/Articles/9266/Software-Project-Cost-Estimates-Using-COCOMO-II-Mo

[16] Bente Anda ,Hege Dreiem, Dag I.K. Sjøberg and Magne Jørgensen, “Estimating Software Development Effort based on Use Cases – Experiences from Industry”, 2002

[17] http://geekswithblogs.net/Prabhats/archive/2007/03/01/107632.aspx

[18] Barry Bohem, Ray, Richard , “Cost Models for Future Software Life Cycle Processes: COCOMO 2.0”, 1998.