Accuracy of Contemporary Parametric & Non Parametric Software Estimation Models: A Comparative Analysis

Dr. Aruna Deoskar, Jyoti Mokashi, Rinku Dulloo

Principal, ATSS, India
Assistant Professor, RSCOE, India

Abstract: - In IT industry, the achievement of project depends upon the way desired product or application delivered in stipulated time, with negligible deviation on schedule & most important cost within limits. Here software project management plays a challenging and onerous role to pull off success and for which smart project planning with broad thought process is required.

This paper highlights the common size estimation metrics as a number of estimation models depend on a software size as an input. Also discuss different algorithmic & non algorithmic cost estimation models that have been anticipated and used successfully in the industry. Every cost estimation model has its own pros and cons. At the end of paper, comparative analysis of various estimation models is provided in the company of correlation of cost estimation models with project parameters.

Keywords:-Source Line of Code (SLOC), Function Point (FP), Constructive Cost Model, Software Lifecycle Management(SLIM), EAF (Effort Adjustment Factor),Cost Estimation, Effort Estimation.

I. Introduction

For software project management, cost estimation is the most demanding tasks. Software cost estimation is a composite activity that requires awareness of the number of parameters about the project for which the estimate is constructed. Software practitioner knows the significance of realistic estimation of effort to the successful organization of software projects. Pragmatic estimation at the commencement of project's life cycle permits project managers & development organizations to manage resources effectively.
Software cost estimation is usually deliberate in terms of effort. For any type of software development there are some important indicators to consider

1. Size of project
2. Effort required
3. Cost essential to develop project
4. Time/Schedule taken by the project

The full paper is organized in sections which are listed as below. Section II describes related work in estimation field, Section III describes the problem statement, Section IV discuss the literature review, Section V explains size estimation, Section VI explains various algorithmic & non algorithmic estimation techniques, Section VII describes comparative analysis of various estimation techniques, Section VIII includes proposed metric and Section IX includes the conclusion and future work.

II. Related Work

Defining the project estimation early in the development life cycle is supreme challenge. K. Ramesh et al. [4] analyze algorithmic & non-algorithmic models and provide depth review of software and project estimation techniques existing in industry. Vahid et al. [3] focused on all the existing methods for software cost estimation methods and comparing their features. It is useful for selecting the special method for each project. Lionel et al. [5] investigate data-driven approach for software cost estimation. They investigate which estimation technique produces accurate results either using typical software development cost data or organization specific data. Lalit et al. [2] represents modern idea which is based on PCA (Principal Component Analysis) with Artificial Neural Network by keeping the base of Constructive Cost Model II model. Where PCA can filters multiple input values into a few certain values. It also helps in reducing the gap between actual and estimated effort. Lionel et al.[8] replicates a comprehensive comparison of common estimation techniques within different organizational contexts. ◆

Barry Boehm et al. [6] summarizes several classes of software cost estimation models and techniques. Abedallah et al. [7] describes the issues in software cost estimation (SCE) where they mentioned that SCE is a process used in software development industry to estimate or predict the resource, efforts, cost of any development process.

III. Problem Statement

To support the cost estimation as one of the major project failure reason, it is extremely necessary to understand the correct way of such estimation(s). The basic objective of this paper is

1. To propose a consolidated document highlighting the comparative analysis of estimation techniques.
2. To propose a metric this can suggest the suitable estimation technique for different types of projects.

IV. Literature Review

Software cost estimation is totally fluctuating as it does not denote the accurate values. There are lots of reasons which affect the accurate cost estimation and the reasons are:

1. Lack of user involvement,
2. Improper use of cost estimation technique due to failure in understanding project parameters,
3. Poor Planning,
4. Requirements of projects are changing continuously,
5. New requirements are added, but the original estimation cannot be changed,
6. Lack of awareness in understanding the estimating techniques,
7. Historical data is seldom available for calibration of estimates.

V. Size Estimation

Exact estimation of development effort and cost is totally depending on accurate prediction of the software size. Two such common techniques are

1. SLOC
2. FP Size Estimation

SLOC : Source Line of Code is the oldest metric for estimating project size. SLOC is nothing but the number of lines of the delivered source code of the software; SLOC estimation of a software system can be obtained from experience, the size of previous project, the size of a competitor's project, and breaking down the system into smaller modules and estimating the SLOC of each module. SLOC is calculated by considering a as smallest, b as largest and m as most likely size (Roger S. Pressman, 2005).

Table 1: Stepwise SLOC Calculation

Steps	Formulas	Steps	Formulas

Expected SLOC for Module Ei	$E_i = \frac{a+4m+b}{6}$	Expected SLOC for entire software system E	$E = \sum_{i=1}^{n} E_i$
Standard deviation of each of the estimates Ei	$SD_i = \frac{\lvert b-a \rvert}{6}$	SD of the expected SLOC for entire software system	$SD = \sqrt{\sum_{i=1}^{n} SD_i^2}$
Note:- n is the total number of module.			

FP Size Estimation - Function point is introduced by Allan Albrecht (1983) of IBM. The FP is programming language independent. FP is based on the number of 'functions' that software has to fulfill.

Table 2: Stepwise FP Calculation

Steps	Execution
I.	Identify the function for a given indicator, Rate the function's complexity must as low, average, or high. It is necessary to define a weight for each above indicator which can be between 3 - 15.
Ii. Unadjusted function points	UFP for entire system=Sum of (Each function count X weight associated with its complexity)i.e. $UFP = \sum_{i=1}^{3}\sum_{j=1}^{5} w_{ij}x_{ij}$ Where Wij - is the weight for row i and column j Xij - is the function count in cell i,j.
III. Calculating adjusted function points	UFP do not consider environment variables for calculating effort. List of 14 general system indicators are rated from 0 to 5 with respect to their likely effect for the system being counted. $VAF = 0.65 + 0.01 \bullet \sum_{i=1}^{14} c_i$ Where Ci - Value of general system characterstic i , for 0<=Ci<=5
IV. FP	FP=UFP X VAF
V. Size in FP	Size(KLOC) = (FP X Selected Language)/1000

VI. Cost Estimation Techniques

More than 40 years, many more estimation models have been proposed. They fall in two categories

1. Algorithmic Approach
2. Non Algorithmic Approach

Algorithmic (Conventional) Software Cost Estimation

It uses parametric models which are derived from the statistical project data. Algorithmic Methods are

1.Putman Model (SLIM) 2. Seer-Sem
3.Linear Model 4. Multiplicative Model
5.Checkpoint 6. Boehm's Model (COCOMO 81 & II)

Non-Algorithmic (Non Parametric) Software Cost Estimation

It is based on soft computing technique. Soft computing consists of distinct concept & techniques which aim to overcome difficulties encountered in real world problems. Non Algorithmic methods are

1. Estimation By Analogy
2. Expert judgment
3. Machine Learning Models
   a. Neural Network
   b. Regression Model
   c. Fuzzy Logic
   d. Genetic Algorithm

1. Putman's Model

This model has been proposed by Putman according to manpower distribution and the examination of many software projects [3]. It is used for cost estimation and manpower scheduling of software. Equation is [3]

$$\text{Effort} = (D_0^{4/7} \times E^{-9/7}) \times S^{9/7}$$

Where Effort is the effort in person-year

E-Environment factor that gives development capability S-Size in LOC

D_0-Manpower build-up factor, ranges from 8(new software) to 27 (rebuilt software).

In the late 1970's, Larry Putnam developed the Software LIfecycle Model (SLIM). SLIM is based on Putnam's analysis of the life cycle in terms of a so called Rayleigh distribution of project personal level versus time [6].

2. SEER-SEM (Software Evaluation and Estimation of Resources-Software Estimating Model)

SEER-SEM model is proposed in 1980 by Galorath [3]. It covers all phases of the project life-cycle, from specification through design, development, delivery and maintenance. It grip a mixture of environmental & application configurations like client-server, standalone, distributed, graphics, etc. SEER SEM uses sizing metrics as SLOC and FP.

3. Linear models

It is used in the large, revolutionary software cost estimation study carried by System Development Corporation. Linear model consist of straightforward construction with a plain equation:

$$\text{Effort} = a_0 + \sum_{i=1}^{n} a_i x_i$$

Where x_i - Cost driver variables a_i - Set of coefficients which provide finest to a set of practical data points

4. Multiplicative Model

Multiplicative cost estimating model uses following form:

$$\text{Effort} = a_0 \prod_{i=1}^{n} a_i^{x_i}$$

Where a_0......... a_n - set of coefficients, x_i . . . . . . x_n - cost driver variables.

Here xi can obtain only 3 possible values: -1, 0, +1. This model works fine if the variables chosen are sensibly independent [3].

5. Checkpoint

Checkpoint is a commercial proprietary model developed by T. Capers Jones of Software Productivity Research, Inc and is based on actual historical information from approximately 4,700 software projects [6]. Checkpoint analyzes the project classification information like nature, scope, class and kind. An exclusive aspect of the CHECKPOINT model is based on FP. Checkpoint predicts the initial staffing, effort, schedules, and the costs of producing the project's deliverables.

6. Boehm's Model (COCOMO 81 & COCOMO II)

COCOMO model used by thousands of software project managers and it is the study of 100s software projects, this model calculate project effort and development time. It is structured into two parts

1. COCOMO I or COCOMO '81 2. COCOMO II (Advanced Model)

COCOMO I

Boehm proposed 3 levels of the model: Basic, Intermediate, Detailed COCOMO. It calculates Development Effort using:

$\text{Effort} = a * (\text{KLOC})^{b}$ ·····expressed in person months (PMs) or Man-Month (MM).

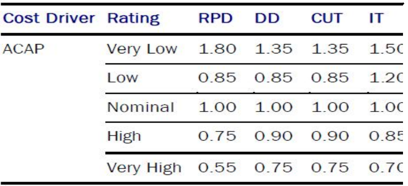
Coefficients a & b depend on mode of the development. There are 3 modes of development.

Table 3: Development modes

Development Mode	Project Characteristics			
	Size	Innovation	Constraints	Dev. Environment
Organic	Small	Little	Not Tight	Stable
Semi Detached	Medium	Medium	Medium	Medium
Embedded	Large	Greater	Tight	Complex Hardware

Table 4: Comparative Information of COCOMO I

Factors	Basic COCOMO	Intermediate COCOMO	Detailed COCOMO
Basic Info	Good for quick, early, rough estimation.	In addition, 15 cost drivers are rated to calculate effort multiplier. EAF uses 15 parameters covering Product, Personnel, Computer, and Project familiarity.	It include all uniqueness of intermediate version with an assessment of cost driver's impact on each step (analysis, design, etc.) of software engineering process [14].
Applicable	Small to medium products	Medium sized projects. Cost drivers are based on product reliability, database size, execution & storages. Team size is medium.	Large sized projects. Cost drivers are based on requirements, analysis, design, testing and maintenance. Team size is large.

<table>
<tr>
<td>Formula</td>
<td>Effort = a * (KLOC)^b</td>
<td>EAF= E1 * E2 * ... * E15
Effort = a * EAF * (KLOC)^b</td>
<td>It uses Effort Multipliers for every phase of a project. Four phases:
RPD - Requirements Planning & Product Design
DD - Detailed Design
CUT - Code & Unit Test
IT - Integrate &amp; Test</td>
</tr>
<tr>
<td>Values of a, b, c for 3 development mode</td>
<td colspan="2">
<table>
<tr><th>COCOMO Values</th><th>a (Basic)</th><th>a (Intermediate)</th><th>b</th><th>C</th></tr>
<tr><td>Organic</td><td>2.4</td><td>3.2</td><td>1.05</td><td>0.3</td></tr>
<tr><td>Semi-Detached</td><td>3.0</td><td>3.0</td><td>1.12</td><td>0.3</td></tr>
<tr><td>Embedded</td><td>3.6</td><td>2.8</td><td>1.20</td><td>0.32</td></tr>
</table>
</td>
<td>

<table>
<tr><th>Cost Driver</th><th>Rating</th><th>RPD</th><th>DD</th><th>CUT</th><th>IT</th></tr>
<tr><td>ACAP</td><td>Very Low</td><td>1.80</td><td>1.35</td><td>1.35</td><td>1.50</td></tr>
<tr><td></td><td>Low</td><td>0.85</td><td>0.85</td><td>0.85</td><td>1.20</td></tr>
<tr><td></td><td>Nominal</td><td>1.00</td><td>1.00</td><td>1.00</td><td>1.00</td></tr>
<tr><td></td><td>High</td><td>0.75</td><td>0.90</td><td>0.90</td><td>0.85</td></tr>
<tr><td></td><td>Very High</td><td>0.55</td><td>0.75</td><td>0.75</td><td>0.70</td></tr>
</table>
Analyst capability effort multiplier
</td>
</tr>
</table>

COCOMO II

COCOMO II was developed in 1995 by Barry Boehm & his team. Similar to the COCOMO I, but uses more complex formula. It reflects recent software development processes and comes in three versions:

1. Application Composition Model 2. Early Design Model 3. Post Architecture Model

Table 5: Comparative Information of COCOMO II

<table>
<tr><th>Parameters</th><th>Application composition model</th><th>Early design model</th><th>Post architecture model</th></tr>
<tr><td>Applicable for Project Like</td><td>Rapid application development or Prototype development</td><td>Useful when only requirements are available & design has not yet started.</td><td>It is used during the actual development & maintenance of software products.</td></tr>
<tr><td>Equation</td><td>Effort=NOP/PROD
where
NOP is no. of object point
PROD is the productivity rate</td><td colspan="2">$Effort = A * Size^B * \prod_{i=1}^{N} EM_i$
$b = 0.91 + 0.01 \sum_{j=1}^{5} SF_i \qquad 0.91 \le b \le 1.23$
Where Effort is in person-months & Size is in KSLOC
A- Constant derived from historical project data
B - Exponent which is replaced by 5 <b>scale factors</b>
EM$_i$- E<b>ffort multiplier</b>(7-Early design,17-Post architecture) for ith cost driver.</td></tr>
<tr><td>Sizing</td><td>Object Points are used.</td><td colspan="2">Uses FP which then converted to SLOC.</td></tr>
<tr><td>Details</td><td>Uses no of screens, reports, & 3GL components that will comprise application.
<table>
<tr><th>Object Type</th><th colspan="3">Complexity Weight</th></tr>
<tr><th></th><th>S.</th><th>M.</th><th>D.</th></tr>
<tr><td>Screen</td><td>1</td><td>2</td><td>3</td></tr>
<tr><td>Report</td><td>2</td><td>5</td><td>8</td></tr>
<tr><td>3 GL</td><td></td><td></td><td>10</td></tr>
</table>
S - Simple</td><td>7 Cost Drivers are
1. Product reliability
2. Required Reuse
3. Platform Difficulty
4. Personnel Capability
5. Personnel Experience
6. Faculties
7. Schedule</td><td>Cost Drivers are based on
1. Product,
2. Platform,
3. Personnel and
4. Project</td></tr>
</table>

	M - Medium D - Difficult	5 Scale Factors are 1. Precedent 2. Team Cohesion 3. Development/Flexibility 4. Process Maturity 5. Architecture Risk Resolution

Table 6: Differences Between COCOMO I and COCOMO II

Parameters	COCOMO I	COCOMO II
Development Life Cycle	Useful in waterfall models	Useful in non-sequential, rapid development, reengineering and reuse models of software.
Size	Delivered Source Instructions (thousands) i.e. KDSI as an input.	Object Points or FP or KSLOC.
Equation Exponent	Effort equation's exponent is determined by 3 development modes.	Effort equation's exponent is determined by 5 scale factors.
Cost Driver	15 cost drivers attributes	17cost drivers attributes
Estimation Accuracy	It provides estimates of effort and schedule.	Provides estimates that represent one standard deviation around the most likely estimate.
Data Points	63 Projects Referred	161 Projects Referred
Model Difference	Model based upon 1. Linear reuse formula 2. Assumption of reasonably stable requirements.	Other enhancements : 1. Non Linear reuse formula 2. Reuse model which looks at effort needed to understand & assimilate.

Non-Algorithmic Technique

1. Estimation by Analogy (EbA): EbA is based on finding efforts for similar projects from the project repository. EbA compare the projected project with earlier accomplished analogous project where the project development information is known. This method can be used either at the total project level or at subsystem level. [10]

Major issues are: the selection of appropriate similarity or distance functions, the selection of applicable project attributes (in our case cost-drivers), and the assessment about the number of similar projects to retrieve (analogies). EbA is comparatively straightforward. Actually in some admiration, it is a systematic form of expert decision since expert often searches for similar situations so as to inform their opinion.

2. Expert Judgment Method: Expert judgment methods rely on the use of human expertise to estimate software cost. This method takes advices from experts who have extensive experiences in similar projects. The experts provide estimates using their own methods and experience [4][14]. This method is usually used when there is limitation in finding data and gathering requirements. Consultation is the basic issue in this method [3]. Delphi provides a broad communication bandwidth for the experts to exchange the volume of information necessary to calibrate their estimates with those of the other experts [4].

3. Machine Learning Models: Machine learning explores the mechanism through which knowledge is gained based on experience. It is used to assemble a bunch of techniques which symbolize some of the facts of human mind. It covers Artificial Neural Networks (ANN), which is a simplified model of brain. ANN is a machine learning approach that models human brain & encompass number of artificial neurons. ANN is organized in 3 layers: Input Layer, Intermediate or Hidden Layer, Output Layer

ANN is used in cost estimation because of its ability to learn from earlier data. It is also able to model complex interaction between the dependent (effort) & independent variables (cost drivers).

Regression Model: Regression analysis is a statistical technique for modeling and analyzing variables. It models the relation between a set of input variables, and one or more output variables, which are considered somewhat dependent on the inputs, on the basis of a finite set of input/output observations.

Fuzzy Logic: All systems, which work based on the fuzzy logic try to replicate human behavior and reasoning. Many times, decision making is very hard and circumstances are vague, fuzzy systems are an efficient tool in such situations [3]. Fuzzy is nothing but the thing which is not accurate, understandable or distinct; blurred. Fuzzy Logic is a method to resolve troubles which are too multifaceted to be comprehend quantitatively. It is a multi-valued logic, which allows halfway values to be defined between straight evaluations like high/low, yes/no and true/false. Each problem must symbolize in terms of fuzzy set like, Fuzzy set = {Slowest, Slow, Fast, Fastest} instead of only {Slow, Fast}, Fuzzy set= {0.0-0.15, 0.15-0.30, 0.30-0.45, 0.45-0.60}

For the software cost estimation, it can be used with COCOMO. Steps involved are:

Step 1: Fuzzification has been done by scale factors, cost drivers and size .
Step 2: Principles of COCOMO are considered.
Step 3: De Fuzzification is accomplished to gain effort.

Genetic Algorithm (GA): GA is used to solve a problem for which little is known. They are very general algorithms & work well in any search space. It does not require any prior knowledge, expertise or logic related to the particular problem being solved [20]. GA generates a family of randomly generated solutions to the problem being investigated. Each of the solutions is evaluated to find out how fit it is, and a suitable value is assigned to each solution. Using GA, given a number of data values for a set of i/p parameters and one o/p parameter, construct an expression of the i/p parameters which best predicts the value of the o/p parameter for any set of values of the i/p parameters. The result obtained depends on the fitness function used.

VII. Comparative Analysis

Table 7: Comparative analysis of various estimation techniques

Model	Advantages	Disadvantages
SLIM	1. Use of SLOC. 2. Easy to modify i/p data. 3. Easy to filter & customize formulas. 4. Objectively calibrated to experience.	1. Highly dependent on the SLOC. 2. Incapable to handle exceptional conditions. 3. Some experience & factors can't be quantified. 4. Not suitable for small projects.
Seer-Sem	1. Systematize project fundamentals into WBS for convenient planning & control. 2. Estimation is based on sizable knowledge of existing projects.	1. Project exact size is key concern in this model.
COCOMO	1. Easy to adapt, use & very understandable. 2. Works on historical data & hence is more predictable & accurate. 3. Consider various factors that affect cost of project. 4. Works well on similar projects. 5. Conquer the problem of reengineering and reuse of software modules.	1. Much data is required & not suitable for all project 2. It ignores requirements and all documentation. 3. It ignores hardware issues. 4. Dependent on the totality of time spent in each phase. 5. Personnel experience may be obsolete. 6. Must know the cost drivers.

Estimating by Analogy	1. Depend on actual project data & past experience. 2. Estimators past knowledge can be utilize which is not easy to quantify.	1. Representativeness of the experience 2. Comparable projects may not exist; 3. Historical data may not be accurate.
Experts Judgment	1. Expert with significant knowledge can offer good estimation. Fast estimation. 2. Experts can factor in discrepancy between precedent project experience & necessities of the projected project.	1. Totally dependent on the 'expert' 2. This method can't be quantified. 3. Difficult to document factors used by experts. 4. Expert may be optimistic and unfair.
Neural Network	1. Highly non-linear modeling which needs less formal statistical training. 2. It can handle large amount of data sets; 3. Do not require a priori knowledge about the data. 4. Have strength & fault-tolerant capability.	1. It cannot extrapolate the results. 2. Extracting the knowledge is too difficult. 3. Immaterial variables may include further noise. 4. Input dimensionality boost computational complexity & memory requirements of model increase.
Fuzzy Logic	1. Accurate estimation & understandability. 2. It is inherently robust since it does not require precise, noise-free inputs. 3. Can control nonlinear systems 4. Training is not required.	1. Hard to use, maintaining the degree of meaningfulness is difficult. 2. Need enough expert knowledge for the formulation of the rule base, mixture of the sets and the de-Fuzzification.
Genetic Algorithm	1. Applied to optimization problem. 2. Does not rely upon specific knowledge of the problems. 3. Robust & flexible so that they applied & work well in complex systems.	1. The genetic algorithm is more complex to implement.

VIII. Researcher Proposed Model

Cost of software is heavily depending upon the software quality. Quality is a relative term and mainly relates with the customer / end user perception in terms of getting satisfaction when using that software. Quality of Software is about magnification of the extension of software desirable characteristics. Till now as per the literature survey it has been observed that costing of a project is done based on the manpower requirements and the time requirements. But project costing should consider project parameters also. Quality of software project affects project cost and software project quality depends upon software project performance. Software project performance can be measured through its functional and nonfunctional attributes. It will be good if cost estimation model can be applied after considering the software parameters and attributes depending upon software project type. Researcher would like to suggest the existing cost estimation model which can be applicable to various software projects. This is a review based analysis. Practical implications would be implemented in future for getting primary results.

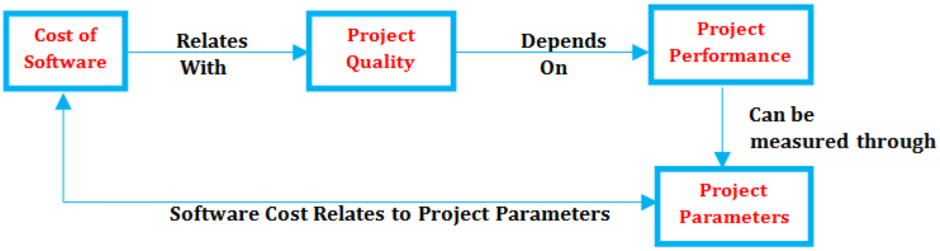


FIG. 1: Project cost performance correlation with project parameters

Table 8: Suggested cost model(S) based on project parameters

S.NO.	Project type	Project parameters	Suggested cost model(s)
1	System Software (e.g. Operating Systems, Utility Programs, Drivers)	Architecture Complexity Compatibility Memory Organization Performance Risk Management Security Development Environment, Integrity Usability	Checkpoint COCOMO Fuzzy Logic ANN Expert Judgment
2	Application Software (e.g. General Purpose, Tailor Made Software)	Configuration Maintainability Security Portability Usability Compatibility Complexity Scalability Adaptability Performance	GA Estimation By Analogy Expert Judgment
3	Research Oriented Software (e.g.Anti-Virus, Network Utilities)	Speed Performance Reliability Usability Security Maintainability Efficiency Development Mode	Checkpoint ANN COCOMO Expert Judgment

IX. Conclusion & Future Work

In this paper, Researcher(s) have compared techniques for predicting software project effort and cost. These techniques have been compared in terms of accuracy, transparency and ease of configuration. Despite finding that there are dissimilarity in forecasting precision, researchers fall out that there may be other characteristics of these technique that will have an equal, if not greater, impact upon their adoption. The results shown in all these approach demand additional investigation, particularly to explore the effect of various parameters on the models in term of improving robustness and accuracy. It also offers the potential to provide more transparent solutions but this aspect also requires further research.

References

1. Magne Jørgensen and Martin Shepperd, "A Systematic Review of Software Development Cost Estimation Studies", IEEE Transactions on Software Engineering, Vol. 33, No. 1
2. Lalit V. Patil, Rina M. Waghmode, S. D. Joshi, V. Khanna, "Generic Model Of Software Cost Estimation: A Hybrid Approach", IEEE International Advance Computing Conference (IACC)2014
3. Vahid Khatibi, Dayang N. A. Jawawi "Software Cost Estimation Methods: A Review "
4. K. Ramesh, 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
5. Lionel C. Briand, Khaled El Emam Dagmar Surmann, Isabella Wieczorek, Katrina D. Maxwell, "An Assessment and Comparison of Common Software Cost Estimation Modeling Techniques", ICSE '99 Los Angeles CA ACM 1999 I-581 13-074-0/99/05.
6. Barry Boehm, Chris Abts and Sunita Chulani, "Software development cost estimation approaches–A survey", Annals of Software Engineering 10 (2000) 177–205.
7. Abedallah Zaid, Mohd Hasan Selamat, Abdual Azim Abd Ghani, Rodziah Atan and Tieng Wei Koh," Issues in Software Cost Estimation", IJCSNS International Journal of Computer Science and Network Security, 350 VOL.8
8. Lionel C. Briand, Tristen Langley. Isabella Wieczorek, "A replicated Assessment and Comparison of Common Software Cost Modeling Techniques", ICSE 2000,Limerick Ireland
9. Brad Touesnard, "Software Cost Estimation: SLOC-based Models and the Function Points Model"

10. Hareton Leung,Zhang Fan, “Software Cost Estimation”.
11. Z. Zia, A. Rashid and K. uz Zaman, “Software cost estimation for component-based fourth-generation-language software applications”, Institution of Engineering and Technology (IET) Software, Vol. 5
12. Marcio Rodrigo Braz, Silvia Regina Vergilio, “Software Effort Estimation Based on Use Cases”, Computer Software and Applications Conference, Vol. 1, pp. 221-228, Sept. 2006.
13. Geetika Batra, Kuntal Barua, M.Tech Scholar, "A Review on Cost and Effort Estimation Approach for Software Development", International Journal of Engineering and Innovative Technology (IJEIT) Volume 3
14. Sweta Kumari , Shashank Pushkar, " Performance Analysis of the Software Cost Estimation Methods: A Review ", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 7, July 2013 ISSN: 2277 128X
15. Geeta Nagpal, Moin Uddin and Arvinder Kaur, "A Comparative Study of Estimation by Analogy using Data Mining Techniques", J Inf Process Syst, Vol.8, No.4, December 2012, pISSN 1976-913X eISSN 2092-805X
16. Ömer Faruk SARAÇ, Nevcihan DURU, "A Novel Method For Software Effort Estimation: Estimating With Boundaries",978-1-4799-0661-1/13/$31.00 ©2013 IEEE
17. Poonam Pandey," Analysis Of the Techniques for Software Cost Estimation ", 2012 Third International Conference on Advanced Computing & Communication Technologies, 978-0-7695-4941-5/12 $26.00 © 2012 IEEE DOI 10.1109/ACCT.2013.13
18. Ashita Malik, Varun Pandey, Anupama Kaushik, "An Analysis of Fuzzy Approaches for COCOMO II", I.J. Intelligent Systems and Applications, 2013, 05, 68-75 Published Online April 2013 in MECS DOI: 10.5815/ijisa.
19. Iman Attarzadeh, Siew Hock Ow, "Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model",2011 IEEE International Conference on Fuzzy Systems June 27-30, 2011, Taipei, Taiwan
20. Colin J. Burgess, Martin Lefley, "Can genetic programming improve software effort estimation? A comparative evaluation", Information and Software Technology 43(2001) 863-873
21. Capers Jones, "Social and Technical Reasons for Software Project Failures" from Software Productivity Research, LLC.
22. Chris F. Kemerer, “Reliability of Function Points Measurement. A Field Experiment,” Communications of the ACM, Vol.36, No.2, pp. 85-97.