Volume 3, Issue 7, July 2013 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Performance Analysis of the Software Cost Estimation Methods: A Review

Sweta Kumari , Shashank Pushkar
CSE & BIT Mesra, Ranchi
India

Abstract: - Accurate cost estimation helps to complete project within time and budget. Many estimation models have been proposed over the last 30 years. This paper provides a detail overview of existing software cost estimation models and techniques. Cost estimation models are basically of two types: algorithmic and non-algorithmic. It also includes the recent developed techniques for software cost estimation field. This paper presents the strength and weakness of various software cost estimation methods. It also focuses on some of the relevant reasons that cause inaccurate estimation. In this paper a comparative analysis among existing popular models are performed and the performance is analysed and compared in terms MMRE (Mean Magnitude of Relative Error) and PRED (Prediction).

Keywords:-Software Cost Estimation Methods, KDLOC-Thousands of delivered lines of code, PM- Person Months, COCOMO- Constructive Cost Estimation, MOPSO- Multiple objective particle swarm optimization, Support vector regression

I. Introduction

The importance of software cost estimation has been increasing gradually over last three decades. Software cost estimation is related to how long and how many people are required to complete a software project. Software cost estimation starts at the proposal state and continues throughout the life time of a project. The estimation process includes size estimation, effort estimation, developing initial project schedules and finally estimating overall cost of the project. Software development has become an essential question [5] because many projects are still not completed on schedule, with under or over estimation of efforts leading to their own particular problems [4]. Therefore, in order to manage budget and schedule of software projects [3], various software cost estimation models have been developed. Accurate software cost estimates are critical to both developers and customers [22]. They can be used for generating request for proposals, contract negotiations, scheduling, monitoring and control. Accurate cost estimation is important because of the following reasons [13]:

- It can help to classify and prioritize development projects 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.

Software cost estimation historically has been a major difficulty in software development. Several reasons have been identified that affects the estimation process such as:

- It is very difficult to estimate the cost of software development. One of the first steps 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 involving many interrelated factors, which affect development effort and productivity, and whose relationships are not well understood.
- Lack of a historical database of cost measurement that means historical data is sometimes incomplete, inconsistent, or inaccurate.
- Lack of trained estimators and estimators with the necessary expertise.
- However, 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.

Other factors that affect the cost are programmer ability, experience of the developer's area, complexity of the project and reliability requirements etc. Figure 1. illustrates the inputs and outputs of the software cost estimation process. The primary cost driver is assumed to be the software requirements. It is the primary input to the estimation process. The estimator is then adjusted according to a number of cost drivers (such as experience of personnel & complexity of system) to arrive at the finale state. Financial constraints limit the amount of money that can be budgeted for the project. Calendar constraints specify a delivery date that must be met and manpower constraints limit the number of people that can be allocated to the project. Loading is the number of engineering and management personnel allocated to the project as a function of time. Effort is defined as the engineering and management effort required to complete a project. It is usually measured in person-months. Duration is the amount of time required to complete the project. The estimator can also quantify a set of cost drivers.

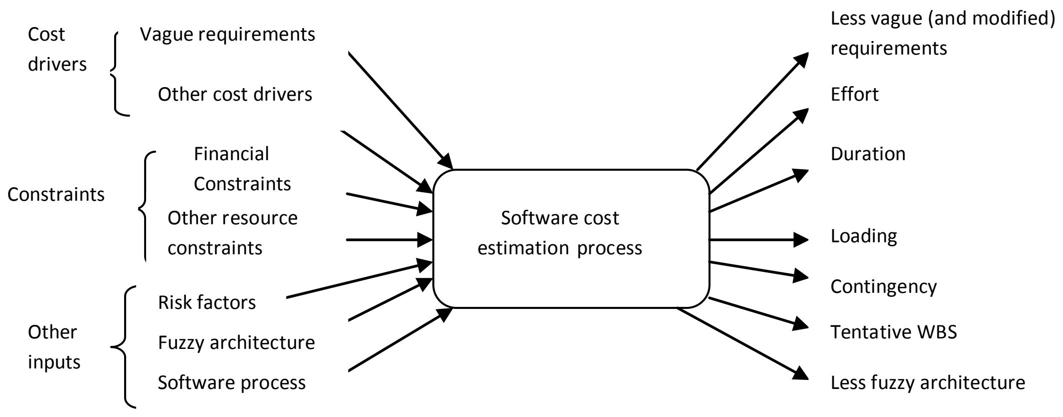

Fig.1: Inputs & Outputs to the Estimation Process

Software project failures have been frequent in the last decade. Software projects usually don't fail during the implementation and most failures are related to the planning and estimation steps. Due to time overrun and cost, approximately between 30% to 40% of the software projects are completed successfully [17]. The Standish group's CHAOS reports failure rate of 70% for the software projects. Also the cost overrun has been indicated 189% in 1994 CHAOS report [19]. It claims the reported results do not depict the real failures rate and are pessimistic. In addition, it indicate that the CHAOS report may be corrupted. Nevertheless the mentioned statistics show the deep crisis related to the future of the software projects. [19]. During the last decade several studies have been done in term of finding the reason of the software projects failure. Galorath et al. [21] performed an intensive search between 2100 internet sites and found 5000 reasons for the software project failures. Among the found reasons, insufficient requirements engineering, poor planning the project, suddenly decisions at the early stages of the project and inaccurate estimations were the most important reasons. Many researchers pointed out that the root causes for project failures is the inaccurate cost estimation [22] [7] [17]. Through the indicated statistics seem pessimistic; inaccurate estimation is a real problem in the software production's world which should be addressed. Analysing more seriously, the efficient techniques and reliable models seems necessary examined because of the dynamic nature of software development process, many methods should examine as different approaches would be suitable for different category of software projects. The rest of the paper is organized as follows: In section, II and III include detail description of some existing cost estimation methods and the comparison of existing methods. Section IV includes the recently developed techniques for software cost estimation field. Performance analysis and evaluation criteria are presented in section V and VI and finally, the concluding remarks is illustrated in section VII.

II. Existing Cost Estimation Methods

Cost estimation methods are basically of two types: algorithmic and non-algorithmic. Algorithmic methods use a formula to calculate the software cost estimate. The formula is developed from models which are created by combining related cost factors. In addition, the statistical method is used for model construction. 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

Expert judgment techniques involve consulting with software cost estimation expert or a group of the experts to use their experience and understanding of the proposed project to arrive at an estimate of its cost. It is the most usable methods for the software cost estimation. Mostly companies used this method for generating the cost of the product. Generally speaking, a group consensus technique, Delphi technique, is the best way to be used. To provide a satisfactorily broad communication bandwidth for the experts to exchange the volume of information necessary to calibrate their estimates with those of the other experts, a wideband Delphi technique is introduced over standard Delphi technique [11] [22] [8]. The estimating steps using this method are as follows:

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 [22].

2. Estimating by Analogy

Estimating by analogy 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. This method can be used either at system-level or at the component-level [16].

The estimating steps using this method are as follows:

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

Top-down estimating method is also called Macro Model. 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 or components. The leading method using this approach is Putnam model. This method is more applicable to early cost estimation when only global properties are known. In the early phase of the software development, it is very useful because there is no detailed information available [11] [22].

4. Bottom-up Estimating Method

Using bottom-up estimating 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. The leading method using this approach is COCOMO's detailed model [22].

5. Parkinson's Law

Using Parkinson's principle "work expands to fill the available volume" [1], the cost is determined (not estimated) 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, the effort is estimated to be 60 person-months. Although it sometimes gives good estimation, this method is not recommended as it may provide very unrealistic estimates. Also, this method does not promote good software engineering practice.

6. Price-to-win

The software cost is estimated to be the best price to win the project. 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, and other cost drivers such as language, design methodology, skill-levels, risk assessments, etc. The algorithmic methods have been largely studied and many models have been developed, such as COCOMO models, Putnam model, and function points based models [11] [28].

1. COCOMO Model

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

❖ ***Model 1 (Basic COCOMO Model):-*** The basic COCOMO model computes software development effort and cost as a function of program size expressed in estimated lines of code (LOC) [6] [23]. 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 (KLOC).
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.

The initial estimate (also called nominal estimate) is determined by an equation of the form used in the static single variable models, using KLOC as the measure of size. To determine the initial effort in person-months the equation used is of the type

EFFORT = a* $(\text{KLOC})^b$ (1)

The value of constants a and b depend on the project type. It has following three classes of software projects [14] [30].

Development Mode	**Basic Effort Equation**	**Time Duration (TDEV)**
Organic	Effort = 2.4 $\text{KLOC}^{1.05}$	TDEV = 2.50 * $(\text{PM})^{0.38}$
Semi Detached	Effort = 3.0 $\text{KLOC}^{1.12}$	TDEV = 2.50 * $(\text{PM})^{0.35}$
Embedded	Effort = 3.6 $\text{KLOC}^{1.20}$	TDEV = 2.50 * $(\text{PM})^{0.32}$

- ***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). Typical values for EAF range from 0.9 to 1.4 [4].

The intermediate COCOMO model takes the form:

EFFORT = a* $(\text{KLOC})^b$ * EAF (2)

Where effort in person-months and *KLOC* is the estimated number of delivered lines of code for the project.

Development Mode	**Intermediate Effort Equation**
Organic	Effort = 3.2 * $(\text{KLOC})^{1.05}$ * EAF
Semi Detached	Effort = 3.0 * $(\text{KLOC})^{1.12}$ * EAF
Embedded	Effort = 2.8 * $(\text{KLOC})^{1.20}$ * 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 $(\text{KLOC})^{1.10}$ (3)

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. It provides a relationship between delivered lines of source code. 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)

5. Bailey-Basil Model

This model developed by Bailey-Basil between delivered lines of source code and formulates a relation [12]:

EFFORT = 5.5 (KLOC) $^{1.16}$ (6)

6. Halstead Model

This model developed by Halstead between delivered lines of source code and formulates a relation [18]

EFFORT = 0.7 (KLOC) $^{1.50}$ (7)

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)

8. Putnam Model

The Putnam model is an empirical software effort estimation model. Putnam 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).

The Putnam model is very sensitive to the development time: decreasing the development time can greatly increase the person-months needed for development [22] [8].

One significant problem with the Putnam model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) 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.
This is a measurement based on the functionality of the program and was first introduced by Albrecht [5]. 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%.

III. Strength And Weakness Of The Existing Methods

Here, we describe the advantages and disadvantages of existing cost estimation methods. This description could be useful for choosing an appropriate method in a particular project. Table I. Shows a comparison of mentioned methods for estimation. For doing comparison, the popular existing estimation methods have been selected.

TABLE I. ADVANTAGES AND DISADVANTAGES OF EXISTING METHODS.

Method	**Type**	**Advantages**	**Disadvantages**
COCOMO	Algorithmic	Clear results, very common	Much data is required, It 's not suitable for any project,
Expert Judgment	Non-Algorithmic	Fast prediction, Adapt to especial projects	Its success depend on expert, Usually is done incomplete
Function Point	Algorithmic	Language free, Its results are better than SLOC	Mechanization is hard to do , quality of output are not considered
Analogy	Non-Algorithmic	Works based on actual experiences, having especial expert is not important	A lots of information about past projects is required, In some situations there are no similar project
Parkinson	Non-Algorithmic	Correlates with some experience	Reinforces poor practice
Price to win	Non-Algorithmic	Often gets the contract	Generally produces large overruns
Top-down	Non-Algorithmic	Requires minimal project detail, Usually faster and easier to implement, System level focus	Less detailed basis , Less stable
Bottom-up	Non-Algorithmic	More detailed basis, More stable , encourage individual commitment	May overlook system level costs , Requires more effort, More time consuming

According to the current comparison and based on the principals of the algorithmic and non algorithmic methods. For using the non algorithmic methods it is necessary to have the enough information about the previous projects of similar type, because these methods perform the estimation by analysis of the historical data. Also, non algorithmic methods are easy to learn because all of them follow the human behaviour. On the other hand, Algorithmic methods are based on mathematics and some experimental equations. They are usually hard to learn and they need to the much data about the current project state. But if enough data is reachable, these methods present the reliable results. In addition, algorithmic methods usually are complementary to each other, for example, COCOMO uses the SLOC and Function Point as two input metrics and generally if these two metrics are accurate, the COCOMO presents the accurate results too. Finally, for selecting the best method to estimate, looking at available information of the current project and the same previous projects data could be useful.

IV. Recently Developed Techniques For Software Cost Estimation

In recent years, researchers have attempted different approaches for software cost estimation. For example, Witting and Finnie [10] [9] describe use of back propagation learning algorithms on a multilayer perceptron in order to predict software development effort. Lefley and Shepperd [15] applied genetic programming to improve software cost estimation on public datasets with great success. Prasad Reddy et al. [32] proposed a model for software cost estimation using Multi Objective (MO) Particle Swarm Optimization. It was observed that the model gives better results when compared with the standard COCOMO model. Later, Vinaykumar et al. [25] used wavelet neural networks for the prediction of software cost estimation. Oliveira [20] provides a comparative study on support vector regression (SVR), radial basis functions neural networks (RBFNs) and linear regression for estimation of software project effort and it was observed that SVR significantly outperforms RBFNs and linear regression. Pahariya et al. [31] proposed new computational intelligence sequential hybrid architectures involving programming and Group Method of Data Handling (GMDH). This includes data mining methods such as Multi-Layer Regression (MLR), Radial Basis Function (RBF) and so on . Reddy et al [29] improved fuzzy approach for software effort of the COCOMO using Gaussian membership function which performs better than the trapezoidal function to presenting cost drivers. Andreou et al [24] used Fuzzy Decision Trees (FDTs) for predicting required effort and software size in cost estimation as if strong evidence about those fuzzy transformations of cost drivers contributed to enhancing the prediction process. Sweta and Pushkar [33] provides a relative study on support vector regression (SVR), Intermediate COCOMO and Multiple Objective Particle Swarm Optimization (MOPSO) model for estimation of software project effort and it has been observed through simulation that SVR gives better result in terms of accuracy and error rate in comparison of other estimating techniques.

V. Evaluation Criteria

According to [19] **t**here are various approaches for evaluating the estimation accuracy of software effort proposed model such as:-

MRE (Magnitude of relative error): First calculate the degree of estimating error in an individual estimate for each data point as project .It is defined as:-

$$\text{MRE} = \frac{|\textit{Predicted Value} - \textit{Actual Value}|}{\textit{Actual Value}} \qquad (9)$$

RMSE (Root Mean Square Error): It is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modelled or estimated. It is just the square root of the mean square error as shown in equation given below:-

$$\text{RMSE} = \sqrt{\frac{1}{n}}\sum_{i=1}^{n}(\textit{Actual Value} - \textit{Predicted Value})^{2} \qquad (10)$$

MMRE (Mean Magnitude of Relative Error): It is another measure and is the percentage of the absolute values of the relative errors, averaged over the N items in the "Test" set and can be written as:-

$$\text{MMRE} = \frac{1}{n}\sum_{i=1}^{n}\frac{|\textit{Predicted Value} - \textit{Actual Value}|}{\textit{Actual Value}} \qquad (11)$$

PRED (N): is the third criteria used for the comparison and this reports the average percentage of estimates that were within N% of the actual values .It is commonly used and is the percentage of predictions that fall within p % of the actual, denoted as *PRED (p)*, k is the number of projects where MRE is less than or equal to p, and n is the number of projects.

$$\text{PRED (p)} = k / n \qquad (12)$$

VI. Performance Analysis

B.W. Boehm [4] is the first researcher to look at software engineering from an economic point of view, and he came up with COCOMO dataset. The COCOMO [27] dataset includes 63 historical projects with 17 effort drivers and one dependent variable of the software development effort. The software development effort is recorded in terms of unit of person-month. The dataset is given in Table II.

TABLE II. COCOMO DATASET.

Project No.	Size	EAF	Effort
1	46	1.17	240
2	16	0.66	33
3	4	2.22	43
4	6.9	0.4	8
5	22	7.62	107
6	30	2.39	423
7	18	2.38	321
8	20	2.38	218
9	37	1.12	201
10	24	0.85	79
11	3	5.86	73
12	3.9	3.63	61
13	3.7	2.81	40
14	1.9	1.78	9
15	75	0.89	539
16	90	0.7	453
17	38	1.95	523
18	48	1.16	387
19	9.4	2.04	88
20	13	2.81	98
21	2.14	1	7.3

Table III shows the calculated efforts obtained for some of the data sets taken from COCOMO dataset. Figure 2 shows the comparative analysis of actual effort with that of the effort estimated using Intermediate COCOMO, COCOMO II, MOPSO, SVR , SEL, Walston Felix , Bailey-Basil, Halstead and Doty models.

TABLE III. ESTIMATED EFFORTS OF DIFFERENT TYPES OF MODELS

Project No.	Size	EAF	Measured Effort	Intermediate COCOMO	COCOMO II	MOPSO Effort	SVR Effort	SEL Model	Walston-Felix Model	Bailey- Basil Model	Halstead Model	Doty model (for KLOC> 9)
1	46	1.17	240	208.56	195.63	235.83	239.66	49.26	169.47	466.83	218.39	291.21
2	16	0.66	33	38.82	61.23	37.18	88.51	18.45	64.83	137.13	44.8	96.38
5	22	7.62	107	626.11	86.91	597.57	174.9	24.81	86.62	198.42	72.23	134.53
6	30	2.39	423	271.97	109.59	279.41	174.31	33.11	114.86	284.33	115.02	186.14
7	18	2.38	321	158.41	69.69	146.77	115.85	20.58	72.16	157.21	53.46	109.03
8	20	2.38	218	176.93	78.26	167.39	126.52	22.71	79.42	177.65	62.61	121.75
9	37	1.12	201	158.85	153.96	171.98	201.34	40.23	128.72	362.64	157.54	231.85
10	24	0.85	79	76.52	85.74	76.89	135.91	26.89	93.75	219.48	82.31	147.36
15	75	0.89	539	336.18	334.94	332.59	391.82	77.62	264.43	823.06	454.66	485.83
16	90	0.7	453	324.32	409.32	329.77	465.13	91.95	312.15	1016.92	597.67	588.01
17	38	1.95	523	284.42	158.55	307.51	219.21	41.24	142.43	374.04	163.97	238.41
18	48	1.16	387	216.23	205.01	246.67	263.17	51.25	176.17	490.46	232.78	304.47
20	13	2.81	98	132.89	48.73	115.24	105.03	15.21	53.66	107.78	32.81	77.55

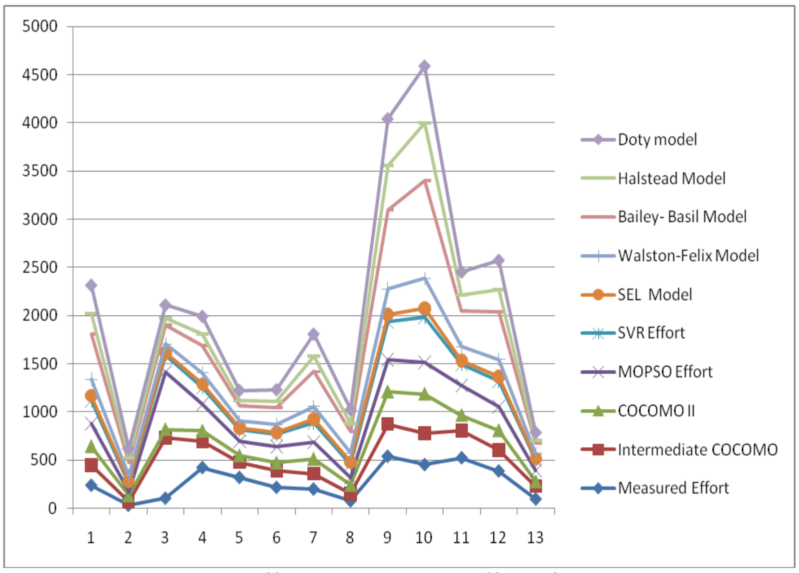

Fig.2: Measured Effort Vs Estimated Effort of various Models.

The calculated errors using different models are shown in Table IV. In this section, we are using statistical methods like MMRE and Prediction for evaluating the cost estimating models.

TABLE IV. ERRORS IN CALCULATED EFFORT USING DIFFERENT EFFORT ESTIMATION MODELS:-

Performance Criteria	Intermediate COCOMO	COCOMO II	MOPSO Effort	SVR Effort	SEL Model	Walston Felix Model	Bailey- Basil Model	Halstead Model	Doty model (for KLOC> 9)
MMRE	0.64	0.45	0.58	0.46	0.81	0.52	0.84	0.43	0.49
Prediction (10%)	0.07	0.15	0.15	0.31	0	0	0.07	0.15	0.07

The following figure 3 shows the performance measure of Intermediate COCOMO, COCOMO II, MOPSO, SVR , SEL, Walston Felix , Bailey-Basil, Halstead and Doty models.

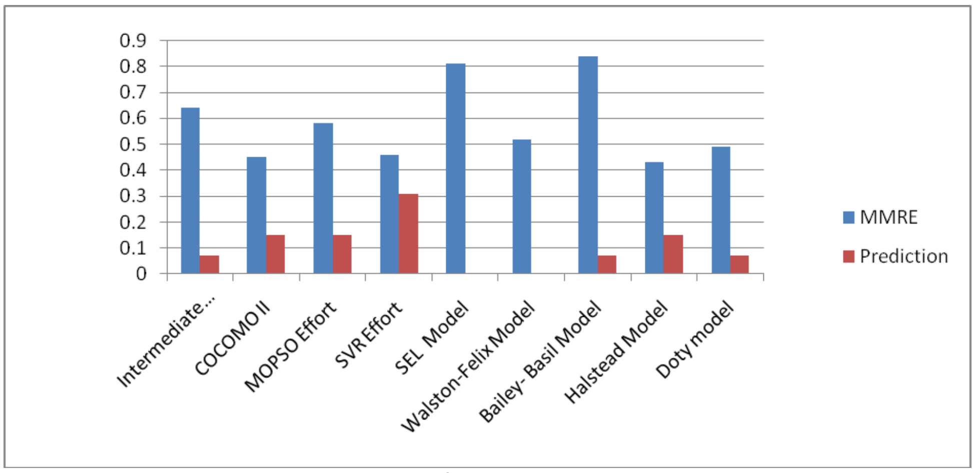

Fig.3. Performance Measure

VII. Concluding Remarks

In this paper, we provided a comprehensive overview of different types of software cost estimation methods and also describe the advantages and disadvantages of these methods. This paper also presents some of the relevant reasons that cause inaccurate estimation. To produce a meaningful and reliable estimate, we must improve our understanding of software project attributes and their causal relationships, develop effective ways of measuring software complexity and the cost estimation process needs to be thoroughly arranged and carefully followed. It has been seen that all estimation methods are specific for some specific type of projects. It is very difficult to decide which method is better than to all other methods because every method or model has an own significance or importance. To understand their advantages and disadvantages is very important when you want to estimate your projects. In recent year research, researchers worked with another field along with the software engineering like data mining and machine learning techniques for improving the accuracy of software cost estimation process. The future work is to study new software cost estimation methods and models that can be help us to easily understand the software cost estimation process.

Acknowledgment

The author would like to thanks BIT Mesra and specially Department of Computer Science and Engineering for encouraging us to do this work and also the anonymous referees for their helpful comments and suggestions.

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.