International Journal of Multimedia and Ubiquitous Engineering
Vol. 10, No. 2 (2015), pp. 97-110
http://dx.doi.org/10.14257/ijmue.2015.10.2.09

An Empirical Approach for Estimation of the Software Development Effort

Amit Kumar Jakhar and Kumar Rajnish

Department of Computer Science & Engineering, Birla Institute of Technology, Deemed University, Mesra, Ranchi-835 215, Jharkhand, INDIA
amitkumar.cs08@pec.edu.in, krajnish@bitmesra.ac.in

Abstract

The number of standards and methods has been developed for software estimation in the last many decades. These methods help practitioners and engineers to estimate the effort accurately, but they found their inability to estimate software precisely. In this paper, an empirical approach is developed to measure the software development effort. The well-known PROMISE data sets of 217 projects are collected and the several performance factors are used in this paper to validate the estimated results. This work also considers the most popular model for software effort estimating, i.e. COCOMO. And the result of the COCOMO model and proposed an empirical approach is compared with the actual effort, and concludes that the proposed approach estimates the software development effort better than COCOMO in several aspects.

Keywords: *COCOMO, lines of code, function points, man-months, prediction PRED (N), mean relative error*

1. Introduction

Software exists only virtually, we cannot touch it physically. Almost every type of industries is totally or partially depends on software's. Now the software engineering concerned to develop quality software within budget and within a specified development time. Therefore, they estimate the software before development start and the accuracy of estimation of effort and budget is depends on the how much the estimation of the project size is accurate. Lines of code (LOC) and the function points are mostly used to estimate the size of a project. Many estimation techniques have been developed till now to estimate the development effort of software, but accurate estimation is still a mystery. Therefore, the estimation seems difficult and if the requirement is changing frequently then it becomes a very complicated task. Inaccurate estimation of software makes a severe impact on its success. There are two categories of software effort estimation: Algorithmic and Non-Algorithmic technique. The accuracy of both models is depends on the requirements. If requirements are exact and specific better will be the estimated result. Function Points (FPs), COCOMO, SLIM, Putnam's models are fall in algorithmic category, and Analogy, Expert Judgment, machine learning model like: Neural networks, Fuzzy method are fall in non-algorithmic category.

Most of the software project failures caused by planning and estimation stages. Despite overrun with time and budget, only 30-40% of software projects are ended with successfully and rest of others become failed [6]. Many surveys find that the most to the software's are overrun in its effort, time and cost from its tolerable level. The 89% of the software overrun by their cost reported by Standish groups [8]. Several other surveys were conducted in the last

ISSN: 1975-0080 IJMUE

decades with the intension to find the cause behind the software project failures. The researchers found that the inaccurate requirement, poor planning of the software project and inaccurate estimation are the major causes for failure of software [6, 7, 10]. The researchers tried to overcome these problems, but most of the time the original data of overrun are not provided publically due to many reasons. The estimation of effort is not the problem of individuals but it is a central problem of software engineering industry, so everyone who is connected with the software industry has responsibility to contribute in accurate estimation.

The most famous and widely accepted model for software estimation is COCOMO, invented by Barry Boehm [2] in 1981. The Boehm focuses on the several parameters that can affect the software estimation. The development time and the budget of software are based on the effort, so the estimation of effort should be accurate.

Caper's John [5] published a rule of thumb method with International Software bench-marking Standards Group (ISBSG). This helps to estimate the effort. Use Case Points (UCP) [9] is helpful for object-oriented standards used for implementation. It was developed by Gustav Karner in 1993. Function Points (FP) [11] proposed by Allan Albrecht of IBM in 1979 used to measure the functionality of software projects. Function Point Analysis (FPA) is maintained by the International Function Point User Group. The five parameters: input, output, internal files, logical files and enquiries are taken into account to count function points.

The growing research trend in the field of software estimation, researchers move from formal regression model to some new approaches for better estimation. The Fuzzy logic approach [12] can also use for software estimation, by using fuzzy logic the range of values is provided rather than a specific numeric value. Other approach is the artificial neural networks [13] used to predict the effort required to develop the software. Back propagation model is mostly used for the software estimation. In the neural network, the neurons are used and arranged in different layers. The neurons of different layers are connected through connections between neurons.

In this paper, the COCOMO'81 is used to calculate the effort of 217 projects that are collected from PROMISE [3] data sets and also proposed an empirical approach for effort estimation and the outcome of both the techniques are compared with the actual effort of projects. The main objective of this paper is to improve the accuracy of the software effort estimation and reduce the mean relative error between the calculated effort and actual effort. Some other factors are also calculated to validate the proposed approach like standard deviation, correlation, total mean relative error and prediction of effort PRED (N).

The rest of the sections are organized as follows: Section 2 discusses the work related to COCOMO and its implementation. Proposed empirical approach of this paper is described in the Section 3. Section 4 describes the different performance parameters used to evaluate the performance of COCOMO model and proposed approach. Section 5 relates with the experimental results of the proposed approach and COCOMO model and their outcome. Finally, Section 6 offers the summary conclusion and future works of this research work.

2. Background of the COCOMO Model

The COnstuctive COst MOdel (COCOMO) was invented and published by Barry Boehm in 1981 [2] which is based on linear-least square regression. He used 63 software development projects to understand and predict the effort and cost of the software system, and different cost drivers are considered that can affect the productivity of developers. The Boehm makes three levels of his model: Basic, Intermediate, Detailed COCOMO model. Boehm provides the constant value of cost drivers and size of the software is calculated in

terms of Kilo Delivered Source Instruction (KDSI) also take into account during estimate the effort.

2.1. Basic COCOMO Model

Basic model uses simple parameterized equation without consideration the project characteristics attribute.

$$MM = a * (KDSI)^{b} \quad (1)$$

where MM is the man-months (152 working hours in a month) and a and b are constants given in Table 1. KDSI, is the kilo of delivered source instruction (DSI).

These instructions are developed by software developers as a final product. Generally, comments and blank lines are not included in DSI, the only executable statements are considered as a part of the source code. Parameters of Basic COCOMO are shown in Table 1, which are used in equation 1.

Table 1. Parameters of Basic COCOMO

Basic COCOMO	Organic	Semi-detached	Embedded
A	2.4	3.0	3.6
B	1.05	1.12	1.20

2.2. Intermediate COCOMO Model

In this model, the estimation of software development effort is like Basic COCOMO level model except the 15 cost drivers that are shown in Table 3. This level considered different attributes like: product, computer, personnel and project attributes, which may affect the required effort and the productivity level of programmers. Equation 2 is used to estimate the required effort with Intermediate COCOMO model.

$$MM = a * (KDSI)^{b} * EAF \quad (2)$$

Here again a, and b are constants, set out in Table 2. Effort Adjustment Factors (EAF) is calculated by multiplying the 15 cost drivers and the values of each cost drivers is presented in Table 3.

Table 2. Parameters of Intermediate COCOMO

Intermediate COCOMO	Organic	Semi-detached	Embedded
A	3.2	3.0	2.8
B	1.05	1.12	1.20

The next section relates to the proposed approach to estimate the required effort to develop software. The example is also solved in this section and the result of the proposed approach and COCOMO model is also compared with the actual effort of the project.

2.3. Advanced Detailed COCOMO Model

At this level, the effort is measured as a function of project size and weighted cost drivers according to the different stages of software development life cycle.

Only Basic and Intermediate level of COCOMO model is considered in this research works and to compare with the proposed approach.

2.4. Illustration of Basic and Intermediate COCOMO Model with an Example

This example is randomly selected from the used data set in this work. Size of the project is 13 KDSI.

The effort estimation is calculated with organic Basic COCOMO as given below:

MM (Basic COCOMO) = $2.4 * 13^{1.05}$

= 2.4 * 14.77

= 35.47 MM

Approx. 36 man-months are required to complete the above given project.

Product complexity (1.15) is only high attribute from 15 cost drivers and rest 14 cost drivers are nominal (1). The embedded Intermediate COCOMO is used to calculate the effort, which is given below:

MM (Intermediate COCOMO) = $2.8 * 13^{1.2} * 1.15$

= 2.8 * 21.71 * 1.15

=69.92 *MM*

According to Intermediate COCOMO, approximately 70 man-months are required to complete the given project. The actual effort required for this project is 60 man-months that is given in the data sets. The Basic COCOMO model estimated 35.47 man-months for this project, means the project is under-estimated and the Intermediate COCOMO model estimated 69.92 man-months required for this project, means the project is overestimated, both under-estimation and over-estimation are dangerous in software engineering aspects. The seriousness of this estimation is discussed in the Section 4.

Table 3. Cost Drivers of COCOMO Model

S. No	Cost Drivers	Attributes	Very Low	Low	Nominal	High	Very High	Extra High
1	Required Reliability	Product	0.75	0.88	1	1.15	1.4	-
2	Database Size (data)		-	0.94	1	1.08	1.16	-
3	Product Complexity		0.7	0.85	1	1.15	1.3	1.65
4	Time Constraints	Computer	-	-	1	1.11	1.3	1.66
5	Main memory Constraint		-	-	1	1.06	1.21	1.56
6	Machine Volatility (virt)		-	0.87	1	1.15	1.3	-
7	Turnaround Time (turn)		-	0.87	1	1.07	1.15	-
8	Analyst Capability		1.46	1.19	1	0.86	0.71	-

9	Application Experience (aexp)	Personnel	1.29	1.13	1	0.91	0.82	-
10	Programmers Capability (pcap)		1.42	1.17	1	0.86	0.7	-
11	Virtual m/c oxp		1.21	1.1	1	0.9	-	-
12	Language Experience		1.14	1.07	1	0.95	-	-
13	Modern Programming Practices	Project	1.24	1.1	1	0.91	0.82	-
14	Use of s/w tools (tool)		1.24	1.1	1	0.91	0.83	-
15	Schedule Constraints		1.23	1.08	1	1.04	1.1	-

3. Evaluation of Proposed Approach

The authors make a flexible way for effort estimation. The values of the different cost drivers are shown in Table 4. These values are calculated empirically and found that the proposed approach provides the more accurate estimation as compared to the COCOMO model. These values of cost drivers may be refined further for better results. The authors classify projects according to KDSI which is given in the data sets and found a constant value that is used in effort estimation. The formula which the proposed approach uses for effort estimation shown in equation 3.

$$Effort = [\{\sum_{i=1}^{15} (em_i * W) * KDSI * \{ (\prod_{j=1}^{15} em_j) - 0.07 \} * (w)\}] \quad (3)$$

Where em is the effort multipliers, which are cost drivers given in Table 3, W is the new calculated weight in this research paper by authors, which is given in Table 4, w is also a calculated weight which is based on the size of the project and KDSI is the size of projects. The result of the equation 3 is the man-months required to finish the projects and w= 0.075 is used for this project because this is constant weight which is used for all the projects whose size is 13 KDSI, if the size is smaller or larger than the value of this constant w will be changed, which is discussed later.

Now the effort is calculated with equation 3 of an example which is provided in Section 2.4.

Effort= [{((1 * 3.85) + (1 * 9.28) + (1.15 * 13.68) + (1 * 3.45) + (1 * 1.74) + (1 * 2.45) + (1 * 1.45) +
(1 * 0.84) + (1 * 2.78) + (1 * 2.54) + (1 * 1.45) + (1 * 0.30) + (1 * 0.46) + (1 * 1.94) + (1 * 7.04)) *
KDSI * (1.15-0.07)} * 0.075]
= [(55.30) * 13 * (1.15 – 0.07) * 0.075]
= 58.23 man-months.

Table 4. New Weights (W) are Assigned to Cost Drivers by Proposed Approach

S. No	Cost Drivers	Attributes	Very Low	Low	Nominal	High	Very High	Extra High
1	Required Reliability (rely)	Product	0.73	2.05	3.85	4.14	3.78	-
2	Database Size (data)			9.4	9.28	12.2	13.68	-
3	Product Complexity (cplx)		6.25	7.68	9.38	13.68	5.76	7.8
4	Time Constraints (time)	Computer	-	-	3.45	2.85	4.05	7.25
5	Main memory Constraint (stor)		-	-	1.74	2	0.81	1
6	Machine Volatility (virt)		-	0.74	2.45	0.12	1.93	-
7	Turnaround Time (turn)		-	1.35	1.45	0.11	2.18	-
8	Analyst Capability (acap)	Personnel	10.56	8.15	0.84	1.84	1.65	-
9	Application Experience (aexp)		4.15	4.58	2.78	3.84	3.79	-
10	Programmers Capability (pcap)		10.25	4.75	2.54	2.45	0.74	-
11	Virtual m/c exp.		1.06	0.45	1.45	1.69	-	-
12	Language Experience (lexp)		2.4	0.35	0.3	3.75	-	-

13	Modern Programming Practices (modp)	Project	2.89	0.85	0.46	0.82	1.55	-
14	Use of s/w tools (tool)		1.25	1.45	1.94	2.94	0.62	-
15	Schedule Constraints (sced)		3.98	3.16	7.04	1.05	1.25	-

4. Performance Methodology

This section contains the different performance parameters that are used to measure the performance of the proposed approach and COCOMO model and also used to validate the outcome of these models.

4.1. Data Collection

The data sets which are used in this research work are famous data sets. These data sets are collected from PROMISE [3]. In this work, three data sets are considered to calculate and validate the result of both the proposed approach and the COCOMO model. Total projects are 217, collected from COCOMO'81 (64 instances), cocomonasa_1 (60 instances) and cocomonasa_2 (93 instances). The entire projects in the given data sets are sorted according to size (lines of code) for evaluation of the results.

4.2. Error Analysis

The authors of this paper did hard work to match the actual effort with the estimated effort. If the calculated effort and the actual effort are the same means method is perfect and the mean relative error is 0 (zero). However, this is very rare is software estimation, the estimators always tries to reduce the mean relative error between estimated and actual effort.

A very simple way to analyze the result is the difference between the actual and calculated, but has a severe problem. For example 100 MM project has absolute error is 80 MM. This is likely to be a serious problem whereas the same problem is occurred in 1000 MM with same error is much less severe than previous estimation.

The Boehm [2] remove the above problem and recommend the following equation 4 for calculating the percentage of error.

$$Percentage \quad (\%) \_ Error \quad = \frac{MM_{cal} - MM_{act}}{MM_{act}} \tag{4}$$

The above equation solves the problem that determines the percentage of error according to its size. MM_{cal} is the effort calculated by the estimators and MM_{act} is the actual man-months are required to finish the software project. The result of the above equation is error percentage with the actual effort. This helps to concentrate on the performance of all projects. It is clear from the above equation 4. That the output generated either negative or positive or may be zero (in perfect estimation) error. So, now the question arises that what is the significance of these negative and positive generated error. In the example which is discussed

in Section 2 and Section 3, actually required effort was 60 man-months and the calculated effort by Basic COCOMO is 35.47 man-months, with Intermediate COCOMO is 69.92 man-month and by the proposed approach is 58.32 man-months. Now the % error is calculated by using equation 4. The error % of Basic COCOMO is (-40.88) %. Error % of Intermediate COCOMO is (+16.53) % and error % of the proposed approach is (-1.77) %. It is clear now the importance of negative and positive errors. The negative errors mean the project is under-estimated and positive errors mean the project is over-estimated. And both the negative and positive estimated errors have severe impact on success of the software projects. Bigger under-estimate leads the way to add more staff at the late of project when deadline approaches, known as Brooks Law [4]: "Adding man-power to a late software makes it later", because the new peoples need training about what is going on? It takes time and productive staff, by doing this the schedule slips further. Over-estimation reduces the productivity of personnel's, known as Perkinson's Law [15]: "Work expands to fill the time available for its completion". It creates another problem when the average is calculated for multiple projects errors then the negative and positive errors cancelled each other. Conte, *et al.,* [14] suggested by taking the magnitude of relative error (MRE) shown in equation 5.

$$MRE = \left| \frac{MM_{cal} - MM_{act}}{MM_{act}} \right| \tag{5}$$

Mean MRE (%) is calculated by equation 6.

$$MRE\ (\%) = \frac{1}{n} \sum_{i=1}^{n} MRE * 100 \tag{6}$$

Predictive accuracy is represented by PRED (N), in other words it discloses the results with in N% of error of actual effort, and the formula is given below:

$$PRED\ (N) = \frac{100}{n} \sum_{i=1}^{n} \begin{cases} 1, if, MRE_i \leq \frac{N}{100} \\ 0 \quad otherwise \end{cases} \tag{7}$$

Min, max, mean, standard deviation, standard error mean and the correlation are also calculated for validation of this research work, and these all above described performance factors are used to find the significance of COCOMO model and proposed approach. The consequences of all 217 projects and their performance factors are described in the next section with the help of tables and figures.

5. Experimental Results

This section contains the experimental results which have been computed with different approaches and analyze the result with the help of many performance factors which is discussed in the previous Section 4. In this research paper, the COCOMO'81 is considered to measure the effort because the scaling factors of COCOMO-II [1] are not present in the used data sets. The results and the conclusion of the proposed approach and COCOMO model are shown in separate tables. In this work, Basic COCOMO, Intermediate COCOMO and the proposed approach are used to measure the software development effort and different performance parameters are calculated to validate the results of both the approaches. The MATLAB 2010 tool is used to calculate the effort on all 217 projects and IBM SPSS tool is used to measure the correlation between the actual effort and calculated effort. Table 6 shows

the lines of code, actual effort, calculated effort by both the proposed approach and COCOMO model, and their calculated percentage mean relative error. The result is shown in Table 6 for only first 50 projects of 217 projects in ascending order. Statistics result of background data of all 217 projects are shown in Table 5. The entire calculated and actual effort of 217 projects are in man-months of 152 hours in a month.

Table 5. Calculated Results of Existing Project Data

	Min	Max	Mean	Standard Deviation	Std. Error Mean
KDSI	1.9	1150	83.47	135.74	9.21
Actual Effort MM	5.9	11400	578.9	1277.9	86.75

Table 6. Calculated and Actual Effort with Their Corresponding MRE (%)

S. No	LOC	Actual Effort	Proposed cal_effrot	MRE% Proposed	COCOMO ca1. _effort	MRE% COCOCMO
1	0.9	8.4	8.4	0.00	2.5	0.70
2	1.98	5.9	6.3	-0.06	6.5	-0.10
3	2.14	7.3	6.8	0.07	7.1	0.03
4	2.2	8.4	6.9	0.18	6.4	0.23
5	2.2	8.4	6.9	0.18	6.4	0.23
6	3	9.8	10.0	-0.02	9.9	-0.01
7	3	60.0	53.7	0.11	50.8	0.15
8	3	38.0	30.5	0.20	37.1	0.02
9	3.5	10.8	11.3	-0.04	11.1	-0.02
10	3.5	10.8	11.3	-0.04	10.5	0.03
11	3.6	23.9	25.9	-0.08	23.9	0.00
12	3.9	61	44.8	0.26	40.7	0.33
13	4	43	41.0	0.05	25.6	0.40
14	4.4	11.8	11.5	0.02	11.8	0.00
15	5.3	21.3	3.0	0.86	4.7	0.78
16	5.3	6	19.0	-2.17	21.3	-2.55
17	5.5	18	17.9	0.00	16.8	0.06
18	5.5	18	17.9	0.00	16.8	0.06
19	6	24	10.0	0.58	9.9	0.59
20	6	24	10.0	0.58	9.9	0.59
21	6.1	40	53.7	-0.34	50.7	-0.27
22	6.2	12	7.8	0.35	8.4	0.30
23	6.2	8	63.4	-6.93	77.0	-8.62
24	6.3	7.5	7.6	-0.01	7.5	0.00
25	6.5	42	42.9	-0.02	32.3	0.23
26	6.5	42	42.9	-0.02	30.2	0.28
27	6.7	57	59.6	-0.05	50.1	0.12
28	6.9	8	10.3	-0.28	9.8	-0.22
29	7.25	648	88.1	0.86	78.0	0.88
30	7.5	72	67.1	0.07	42.1	0.42
31	7.5	72	67.1	0.07	42.1	0.42
32	7.7	31.2	31.4	-0.01	24.0	0.23

33	7.7	31.2	31.4	-0.01	24.0	0.23
34	8	42	44.9	-0.07	32.7	0.22
35	8	42	44.9	-0.07	32.7	0.22
36	8.2	41	44.0	-0.07	40.1	0.02
37	8.2	36	33.5	0.07	25.6	0.29
38	8.2	36	33.5	0.07	25.6	0.29
39	9.1	37.6	39.3	-0.05	37.6	0.00
40	9.4	88	75.0	0.15	61.2	0.31
41	9.7	25.2	32.9	-0.30	30.5	-0.21
42	9.7	25.2	32.9	-0.30	33.6	-0.33
43	10	122	13.6	0.89	13.9	0.89
44	10	13.9	119.9	-7.63	82.7	-4.95
45	10	48	27.3	0.43	28.1	0.41
46	10	48	27.3	0.43	28.1	0.41
47	10.4	50	50.6	-0.01	32.9	0.34
48	10.4	50	50.6	-0.01	36.3	0.27
49	11.3	36	37.5	-0.04	28.0	0.22
50	11.3	36	37.5	-0.04	25.2	0.30

5.1. Experimental Result of Basic COCOMO and Intermediate COCOMO Model

This section contains the results of Basic COCOMO and Intermediate COCOMO model. The performance parameters are calculated according to the equations given in the previous Section 4. The result of Basic COCOMO and Intermediate COCOMO is shown in Table 7. It is clearly observed from Table 7 that the performance of both the approach is not good when we analyze all the parameters with the Table 5 of project data, but the Intermediate COCOMO model predicts the better effort of projects than Basic COCOMO model. Because the Basic COCOMO is used only constant values of a, b, and the size of project but the Intermediate COCOMO considered the 15 others cost drivers shown in Table 3 which have major effect on the productivity of the developers, due to this reason the Intermediate COCOMO provides better results than Basic COCOMO. The correlation of Intermediate COCOMO is also good than Basic COCOMO model.

5.2. Experimental Results of the Proposed Approach

This section describes the results of the proposed approach of software effort estimation which is calculated as according to equation 3. The constant weight (w) is depends on the size of projects. The authors tried to match the estimated effort with actual effort so that the mean relative error is reduced and the estimation is moved to the more accurate estimation. The result of the proposed approach is shown in Table 8 with their performance parameters. The result of Table 8 clearly indicates that the standard deviation, standard error mean are less than the COCOMO model and other performance parameters of the proposed approach are nearby the actual effort which is shown in Table 5. The correlations of all three techniques (Pearson, Kendall's, and Spearman's) are also better than the COCOMO model. Since the correlation is tending to +1 is desirable. These all results show the goodness of the proposed approach than COCOMO model.

Table 7. Result of COCOMO-Basic and COCOMO-Intermediate

	Min	Max	Mean	MRE (%)	Standard Deviation	Std. Error Mean	Correlation	
Basic COCOMO	4.9	16949	635.23	71.89	1750.2	118.57	Pearson	0.589
							Kendall's Tau_b	0.706
							Spearman's rho	0.874
Intermediate COCOMO	2.3	24727	638.1	45.4	2007	197.32	Pearson	0.695
							Kendall's Tau_b	0.824
							Spearman's rho	0.947

Table 8. Result of Proposed approach

	Min	Max	Mean	MRE (%)	Standard Deviation	Std. Error Mean	Correlation	
Proposed approach	3.0	10919	568.74	49.30	1253.52	85.09	Pearson	0.873
							Kendall's Tau_b	0.796
							Spearman's rho	0.950

5.3. Discussion

In this section, all 217 projects are analyzed that are collected from PROMISE [3]. These data sets have 15 effort multipliers, lines of code and their respective actual effort of each project. These parameters are used to measure the effort of all 217 projects. Basic COCOMO, Intermediate COCOMO model, and a developed proposed approach are used to estimate the software development effort of each project and the estimated results of all projects have been shown in the Table 6. This table also contains the MRE (%) of all projects. The overall results of COCOMO and the proposed approach are shown in Table 7-9, and Figure 1. In Figure 1, the MRE (%) of first 30 projects in ascending order is shown for both Intermediate COCOMO model and proposed approach. The Figure 1 indicates that the proposed approaches have less mean relative error than COCOMO model. In Table 9, the prediction of effort PRED (N) is shown with specified error, in other words, PRED (10) indicates that how many instances of 217 projects are estimated whose MRE(%) is less than or equal to 10% and so on. The Intermediate COCOMO model predict the effort 23.58% of all projects are within 10%, 27.52% of all projects are within 15% error, 39.06% of all projects are within 20% error, 51.61% of all projects are within 25% error, and 58.52% of all projects are within 30% error. And the proposed approach predicts 57.60%, 61.29%, 66.36%, 70.05%, 74.20%, of all projects within 10%, 15%, 20%, 25%, 30% error respectively. PRED (N) result clearly indicates that the proposed approach estimated the software required effort of 217 projects is better than the COCOMO model. Other measures like standard deviation, standard mean error, and correlation are also indicate that the proposed approach estimate better software development effort than COCOMO model. The overall result of these above performance measures validates the proposed approach which has better capability to estimate the required effort than the COCOMO model.

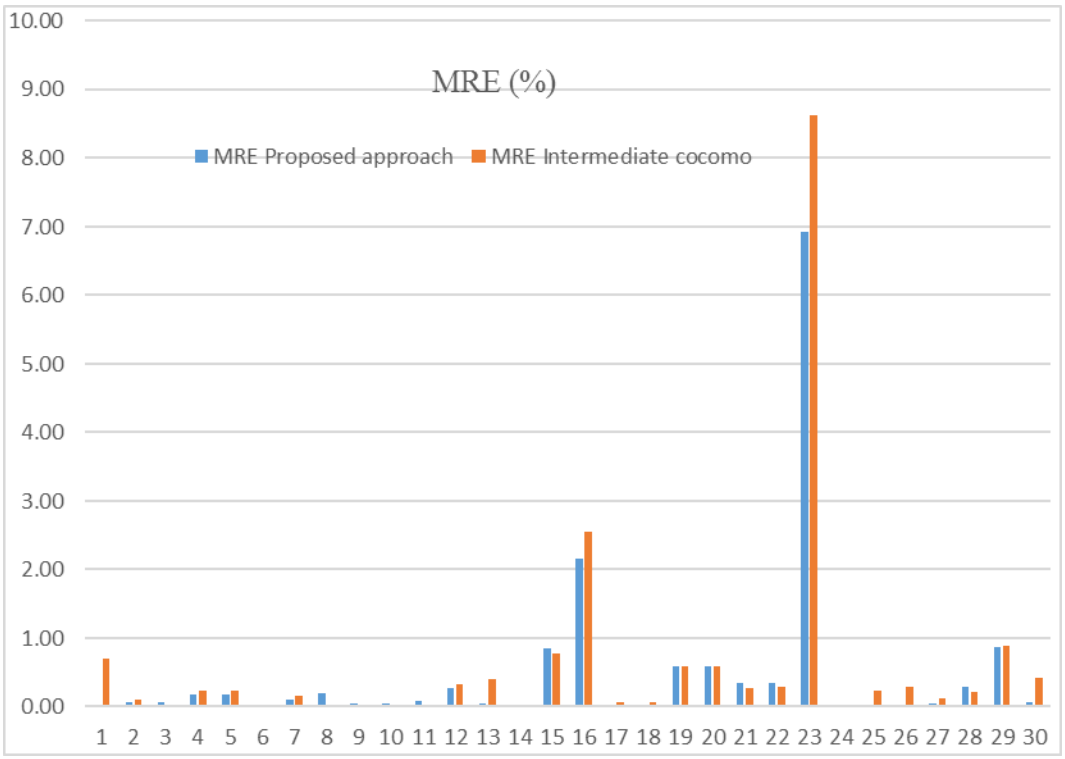

Figure 1. Plot of MRE (%) of the First 30 Projects in Ascending Order

Table 9. Prediction of Effort with Their Specified MRE (%)

	PRED (10)	PRED (15)	PRED (20)	PRED (25)	PRED (30)
COCOMO Basic	9.22%	13.36%	17.51%	23.04%	27.72%
COCOMO Intermediate	23.58%	27.52%	39.06%	51.61%	58.52%
Proposed approach	57.60%	61.29%	66.36%	70.05%	74.20%

6. Conclusion and Future Scope

The estimation of software is really a tedious job. In today's world, it seems very important when the software pays a lot in almost every type of industry. So, in this work, the authors examined 217 projects that are collected from PROMISE website. First, the Basic COCOMO, and Intermediate COCOMO model are used to calculate the effort of all 217 project of the given data sets. Secondly, the authors proposed a more flexible approach to estimate the software development effort. At last, several performance parameters are measured that helps to validate the techniques. Performance parameters are mean relative error, standard deviation, standard deviation mean, different correlations, and prediction of accuracy PRED (N) are described in Section 4 and 5 are calculated for all effort estimation approaches used in this research work. The analysis of the results concludes that the proposed effort measurement approach increases the accuracy of software effort estimation and reveals the better result than Basic and Intermediate COCOMO model.

This study is focused on the software development effort estimation. During last decades, significant improvement has been notified in software estimation. So, this work can be extended to improve the accuracy of the proposed work in several ways:

a) Weight of some parameters may be further adjusted so that the accuracy of effort estimation is improved.
b) Focus on the most significant cost drivers and passed over the others.
c) Large data sets may be utilized to further validate this work.

References

[1]. B. W. Boehm, *et al*., "Software cost estimation using COCOMO II, 1st ed. Englewood Cliffs", NJ, Prentice-Hall, **(2000)**.
[2]. B. Boehm, "Software Engineering Economics, Englewood Cliffs", N.J., Prentice Hall, **(1981)**.
[3]. http:// promise.site.uottawa.ca/SERepository.
[4]. F. P. Brooks, "The Mythical Man-Month", Addison-Wesley, Reading, Mass., (**1975)**.
[5]. C. Jones, "Software Estimating Rules-of-Thumb", http://www.compaid.com/ caiinternet/ezine/capers-rules.pdf, **(2007)** March.
[6]. K. Molkken, and M. Jrgensen, "A Review of Surveys on Software Effort Estimation", 03 Processing of the International Symposium on Empirical Software Engineering, **(2003)**, pp. 223-231.
[7]. M. Jegensen, "Practical guidelines for expert-judgment-based software effort estimation", IEEE Software, vol. 22, issue 3, **(2005),** pp. 57-63.
[8]. G. Standish, "The Chaos Report", The Standish Group, **(1994)**.
[9]. G. Karner, "Resource Estimation for Objectory Projects", Objective Systems SF AB, **(1993)**.
[10]. C. Jones, "Estimating software costs: Bringing realism to estimating" second edition, New York, NY McGraw Hill, **(2007)**.
[11]. 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, vol. 9, no. 6, **(1983),** pp. 639-648.
[12]. L. A. Zadeh, "Fuzzy Set, Information and Control", vol. 8, **(1965),** pp. 338-353.
[13]. I. Attarzadeh, S. Hock, "Proposing a New Software Cost estimation Model Based on Artificial Neural Networks", IEEE International Conference on Computer Engineering and Technology (ICCET), vol. 3, **(2010)**, pp. 487-491.
[14]. S. Conte, H. Dunsmore and V. Shen, "Software Engineering Metrics and Models", Benjamin/Cummings, Menlo Park, Calif., **(1986)**.
[15]. Parkinson, "C.N. Parkinson's Law and other studies in administration (Buccaneer Books, Cutchogue", **(1957)**.

Authors

Mr. Amit Kumar Jakhar, he is a student in the Department of Computer Science and Engineering at Birla Institute of Technology, Mesra, Ranchi, and Jharkhand, India. He received his M.E. in Computer Science & Engineering from PEC University of Technology, Chandigarh, India in the year of 2010. He received his B.E. (Honours) from MDU, Rohtak, and Haryana, India in the year of 2008. His research area is software engineering.

Dr. Kumar Rajnish he is an Assistant Professor in the Department of Computer Science and Engineering at Birla Institute of Technology, Mesra, Ranchi, Jharkahnd, India. He received his PhD in Engineering from BIT Mesra, Ranchi, and Jharkhand, India in the year of 2009. He received his MCA Degree

from MMM Engineering College, Gorakhpur, State of Uttar Pradesh, India. He received his B.Sc. Mathematics (Honours) from Ranchi College Ranchi, India in the year 1998. He has 30 International and National Research Publications. His Research area is Object-Oriented Metrics, Object-Oriented Software Engineering, Software Quality Metrics, Programming Languages, and Software Estimation.

