Hindawi Publishing Corporation
Advances in Software Engineering
Volume 2013, Article ID 351913, 10 pages
http://dx.doi.org/10.1155/2013/351913

Research Article

Tuning of Cost Drivers by Significance Occurrences and Their Calibration with Novel Software Effort Estimation Method

Brajesh Kumar Singh, Shailesh Tiwari, K. K. Mishra, and A. K. Misra

CSED, MNNIT, Allahabad 211004, India

Correspondence should be addressed to Shailesh Tiwari; shail.tiwari@yahoo.com

Received 5 June 2013; Revised 27 August 2013; Accepted 8 November 2013

Academic Editor: Henry Muccini

Estimation is an important part of software engineering projects, and the ability to produce accurate effort estimates has an impact on key economic processes, including budgeting and bid proposals and deciding the execution boundaries of the project. Work in this paper explores the interrelationship among different dimensions of software projects, namely, project size, effort, and effort influencing factors. The study aims at providing better effort estimate on the parameters of modified COCOMO along with the detailed use of binary genetic algorithm as a novel optimization algorithm. Significance of 15 cost drivers can be shown by their impact on MMRE of efforts on original 63 NASA datasets. Proposed method is producing tuned values of the cost drivers, which are effective enough to improve the productivity of the projects. Prediction at different levels of MRE for each project reflects the percentage of projects with desired accuracy. Furthermore, this model is validated on two different datasets which represents better estimation accuracy as compared to the COCOMO 81 based NASA 63 and NASA 93 datasets.

1. Introduction

Estimation is an important part of software engineering projects, and the ability to produce accurate effort estimates has an impact on key economic processes, including budgeting and bid proposals and deciding the execution boundaries of the project [1]. Effort estimation is a critical activity for planning and monitoring software project development and for delivering the product on time and within budget. Also, feasibility of project in terms of cost and ability to meet customer's requirements is considered in the process of estimation [2]. The prediction of the effort to be consumed in a software project is, probably, the most sought after variable in the process of project management. The determination of the value of this variable in the early stages of a software project drives the planning of remaining activities. The estimation activity is plagued with uncertainties and obstacles, and the measurement of past projects is a necessary step for solving the question. The problem of accurate effort estimation is still open and the project manager is confronted at the beginning of the project with the same quagmires as a few years ago [3]. The software industry's inability to provide accurate estimates of development cost, effort, and/or time is well known [4].

Over the past few years, software development effort is found to be one of the worst estimated attributes. Significant over- or underestimates can be very expensive for company and the competitiveness of a software company heavily depends on the ability of its project managers to accurately predict in advance the effort required to develop the software systems [5]. It is also found that efforts need to be estimated reliably in order to complete the projects on time and within budget as less than one-quarter of the projects is estimated accurately.

Many model structures evolved in the literature and these structures consider modeling relationship between software effort, developed line of code (DLOC), and influencing factors:

$$\text{Effort} = f\left(\text{DLOC}, \text{influencing factors}\right). \quad (1)$$

Building such a relationship as a function helps project managers to accurately allocate the available resources for the project [6]. Among the others, Constructive Cost Model (COCOMO) is a widely known effort estimation model where developed lines of code (DLOC) are the primary element which affects the effort estimation. The DLOC

include all program instructions and formal statements [6, 7]. The aim here is to provide a basis for the software effort estimation through a systematic review of previous research papers [8]. Some research studies have demonstrated that the level of accuracy in software effort estimates is strongly influenced by selection of the input values of the parameters of these methods. Combination of input features selection and parameters optimization of machine learning methods improves the accuracy of software development effort [9].

Recently, the uses of search based methods have been suggested to address the software development effort estimation problem [10, 11]. Such a problem can be formulated as an optimization problem where we have to identify the estimation model which provides the best prediction [5]. This study aims at providing the better effort estimate on the parameters of modified COCOMO along with the detailed use of binary genetic algorithm as a novel optimization algorithm. The performance in terms of estimation accuracy of the developed model was tested on 93 and 63 NASA dataset projects and compared to the preexisting COCOMO. The developed model is able to provide better estimation capabilities.

The whole paper is organized in 7 sections. Section 2 illustrates the problem and the techniques as a part of problem, Section 3 depicts the solution approach, and Section 4 describes the proposed algorithm for solving the problem. Four submodels are introduced in this section. Evaluation criteria and data analysis are discussed in Section 5. Result analysis is made with help of proposed method in Section 6. Finally the paper has been concluded in Section 7.

2. Problem Illustration

2.1. Problem Statement. Software development effort estimates are likely to be highly inaccurate and systematically overoptimistic due to the valence effect of prediction, anchoring, and planning fallacy and cognitive effects. Empirical evidence suggests that the causes of the problem, to some extent, were due to the influence of irrelevant and misleading information, for example, information regarding the client's budget, present in the estimation material [12]. Previous researches have shown that the average effort overrun in software development projects is about 30%–40% [1, 4]. Estimating techniques have emerged continually, and attempts have been made to compare these techniques and derive the best practices [1, 4, 13].

Empirical software estimation models are mainly based on cost drivers and scale factors. These models show the problem of instability due to values of the cost drivers and scale factors, thus affecting the sensitivity in terms of accurate effort estimation. Also, most of the models depend on the size of the project and a small change in the size leads to the proportionate change in the effort. Miscalculations of the cost drivers have even more noisy data as a result too. For example, a misjudgment in personnel capability cost driver in COCOMO between "very high to very low" will result in 300% increase in effort. Similarly in SEER-SEM, changing security requirements values from "low" to "high" will result in 400% increase in effort. In PRICE-S, 20% change in effort will occur due to small change in the value of the productivity factor [14]. Above statements reveal that, all models have one or more inputs for which small changes will result in large changes in effort. The input data problem is further compounded in that some inputs are difficult to obtain, especially early stages in a program development. The size must be estimated early in a project using one or more sizing models. Some sensitive inputs, such as analyst and programmer capability in cost drivers, are based on individual and are often difficult to determine. Many studies like the one performed by [15] show that personnel parameter data are difficult to collect.

2.2. Algorithmic Models. Many software estimation models have been proposed by various researchers and can be categorized according to their basic formulation schemes: analogy based estimation schemes [16–19], expert-judgment estimation [20], and algorithmic models including empirical methods [21], rule induction methods [22], Bayesian network approaches [23], decision tree based methods [24], artificial neural network based approaches [25, 26], and fuzzy logic based estimation schemes [27].

Some of the famous algorithmic models among these diversified models, COCOMO, SLIM, SEER-SEM, and FP analysis methods, are very much popular in practice in the empirical category [28], while COCOMO and Function Points allow us to guess the size (in KLOC) of the software ourselves. Albrecht observed in his research that Function Points were highly correlated to lines of code, so in effect they are complementary [29]. Function Points calculate the logical source lines of codes and COCOMO is based on physical source lines of codes. These empirical models work with certain inputs, accurate estimate of specific attributes, such as source lines of code (SLOC), multiplicative factors, interfaces and complexity, and number of user screens, which are not always easy to acquire during the early stage of software project development. Models based on historical data have limitations. Understanding and calculation of these models are difficult because inherent complex relationships between the related attributes to predict software development effort could change over time and/or differ for software development environments [26]. They are unable to handle categorical data as well as lack reasoning capabilities [30].

The limitations in algorithmic models have led to the exploration of nonalgorithmic models which are soft computing based [30].

2.3. Constructive Cost Model. The original Constructive Cost Model abbreviated as COCOMO was first published by Dr. Barry Boehm in 1981. The word "constructive" prevails that the complexity of the model can easily be understood due to the openness of the model, which exhibits exactly why the model gives the estimates. Since the inception of the software development techniques, many efforts were done in the improvement of estimation; COCOMO is the best documented, most transparent and reflects the software development practices of these days. The main focus in COCOMO is upon the estimation of the influence of 15 cost drivers on the development effort cost. The model does not support project management in estimating the size of

the software. COCOMO has been derived from a database of 63 projects, executed between 1964 and 1979 by the American Company TRW Systems Inc. The projects considered during this time era were differing strongly in type of their application, size, and programming language [31].

Boehm introduced three levels of the estimation model: basic, intermediate, and detailed.

(i) The basic COCOMO 81 is a single-valued, static model which provides an approximate estimation of software development effort and cost as a function of program size expressed in thousand delivered source of instructions (KDSI).

(ii) The intermediate COCOMO 81 describes software development effort as a function of program size in LOC and a set of fifteen "effort multipliers known as cost drivers." These cost drivers incorporate subjective assessments of product, project, personnel, and hardware attributes.

(iii) The advanced or detailed COCOMO 81 reduces the margin of error in the final estimate by incorporating all characteristics of the intermediate version with the determination of the cost driver's impact on each step, that is, analysis and design of the software engineering process.

COCOMO assumes that the effort grows more than linearly with software size. The value of few multipliers is required to be increased to decrease the effort. For few other multipliers, the values are required to be decreased to decrease the effort; that is, person-months $= a * (\text{KSLOC})^b * c$. Here, a and b are domain-specific parameters, KSLOC denotes kilo source lines of code which is estimated directly or computed from a function point analysis, and c is the product of fifteen effort multipliers (EM_i); here $i = 1$ to 15.

So the following equation can be represented as:

$$\text{Person-months} = a * (\text{KSLOC})^b * (\text{EM1} * \text{EM2} * \cdots * \text{EM15}) \quad \text{see [7, 25]}. \tag{2}$$

3. Solution to the Problem

3.1. Nonalgorithmic Models. Contrary to the algorithmic models, since inception in 1990s the proposed nonalgorithmic models are based on computational intelligence, analytical comparisons, and inferences to project cost estimation. They have the capability to model the complex set of relationship between the dependent variables (cost, effort) and the independent variables (cost drivers) collected earlier in the project lifecycle and to learn from historical projects data. For using the nonalgorithmic models, information about those previous projects datasets is required which are similar to the projects under estimate. Usually, in these methods estimation process is done according to the analysis of the historical datasets. Many software researchers have shown their interest in the research to new approaches of nonalgorithmic models that are based on soft computing, that is, artificial neural networks, fuzzy logic and evolutionary algorithms. These methods are being used for the assessing because of their popularity and a large number of papers about their usage have been published in the recent past years [26, 32–34]. Decision of choosing a suitable technique is a difficult one and requires the support of a well-defined evaluation scheme to rank each evolutionary computation technique as and when it is applied to any optimization problem. In this present research study an effective model based on evolutionary computation has been proposed to overcome the problem of uncertainty and to acquire better results.

3.2. Genetic Algorithms. Evolutionary computational methods are generally used in software engineering methodologies such as test case generation [35, 36], effort estimation, cost estimation, and many more. Genetic algorithms are a simple and almost generic evolutionary computational method inspired by Darwin's theory of natural evolution to solve the complex optimization problems. Genetic algorithm requires a careful and suitable selection of parent's selection methods, mutation methods, population size, and so forth, to find good solutions. If improper parameters and methods are chosen, there may have longer program runs or even bad optimization results [37]. In nature, competition among individuals for scanty resources results in the fittest individuals dominating over the weakest ones [24, 38].

3.2.1. Working Principle

(i) Genetic algorithm starts with randomly generated initial population as a set of solutions, which are represented by chromosomes.

(ii) The algorithm then generates a sequence of individuals as new population. At each iteration, the algorithm uses the individuals of current generation to create the next generation of population. To create the new population, the algorithm works with the following steps.

  (a) Score each individual member of the current population by computing its fitness value.

  (b) Scale the raw fitness scores to convert them into a more desired range of values.

  (c) Select the good individuals, called parents, based on the value of fitness function.

  (d) Few of the individuals in the current population that are having lower fitness are selected as elite. These individuals are directly sent to the next generation of population for elitism.

  (e) Produce offsprings from the parents. Offsprings are produced either by making mutation of a single parent by combining the chromosome of a pair of parents with the help of crossover operator.

  (f) Update the current population with the offsprings to form the new generation.

(iii) The algorithm terminates only on the condition that any one of the stopping criteria is reached, that is, number of generations or desired fitness value.

Sub-Model 1:
Step 1. Generate the MMRE (M) for Available N Projects using actual and COCOMO estimated efforts.
(i) [BEGIN]
(ii) Input the 15 cost drivers, KLOC, Actual Effort for NASA projects.
(iii) [LOOP]
for j = 1 to no. of projects (say n)
EAF[j] = D1 ∗ D2 ∗ ⋯ ∗ D15
Estimated Effort[j] = a[j] ∗ (kloc[j]∧b[j]) ∗ EAF[j]
MRE[j] = |Actual Effort[j] − estimated Effort[j]|/Actual Effort[j]
MMRE (original) += MRE[j]
MMRE (original) /= n [The original MMRE is obtained and noted down]
(iv) [END OF LOOP]
(v) [END]
Sub-Model 2:
Step 2. for I = 1 to 15
temp = Emi
Set Emi = 1
Calculate Influenced MMRE(MN)
List[I, 1] = I;
List[I, 2] = MN~M;
Emi = temp;
end for
Sub-Model 3:
Step 3. Sort the list according to the second parameter in descending order
For I = 1 to 14
For j = 2 to 15
If (list[I, 2] < list[j, 2])
then
swap (list[I], list[j])
end if
end for
end for
Step 4. Sig = list[I, 1] represent the order of Significance occurrences.
Sub-Model 4:
Step 5. for I = 1 to 15
for j = very low to Extra high (Six rating of cost driver)
Select Projects (P) as an input for calculating the fitness value using fitness function F1 = MMRE(P).
Set the range R as {Rmax, Rmin}
Generate initial population for the cost driver with Range R.
performs The Genetic operations for K generations.
(1) Tournament Selection
(2) Crossover with Pc = 0.8
(3) Mutation with Pm = 0.3
Select the individual (CDNEW) with the best MMRE
Step 6. Calculate the MMRE(Mmod) by replacing CDNEW with CDij
if (Mmod < M)
then update the value of CDij and M.
else
discard the value
end if
end for
end for

Algorithm 1

4. Proposed Algorithm for Solving the Problem

Algorithm Description (see Algorithm 1*).* Proposed algorithm is divided into 4 submodels. In submodel 1, we calculate the mean magnitude of relative error (MMRE) of all projects according to the results obtained by COCOMO. Here, we first calculate the estimated efforts by considering 15 COCOMO cost drivers, project modes, and kilo lines of codes (KLOC). Estimated efforts along with actual efforts produced in

Table 1: COCOMO cost drivers.

Cost drivers	Very low	Low	Nominal	High	Very high	Extra high
acap	1.46	1.19	1	0.86	0.71	
pcap	1.42	1.17	1	0.86	0.7	
aexp	1.29	1.13	1	0.91	0.82	
modp	1.24	1.1	1	0.91	0.82	
tool	1.24	1.1	1	0.91	0.83	
vexp	1.21	1.1	1	0.9		
lexp	1.14	1.07	1	0.95		
sced	1.23	1.08	1	1.04	1.1	
stor			1	1.06	1.21	1.56
data		0.94	1	1.08	1.16	
time			1	1.11	1.3	1.66
turn		0.87	1	1.07	1.15	
virt		0.87	1	1.15	1.3	
cplx	0.7	0.85	1	1.15	1.3	1.65
rely	0.75	0.88	1	1.15	1.4	

various projects are used as the input parameters to calculate the mean of relative error (MRE) of each project. MMRE for COCOMO results is recorded as the original MMRE.

In submodel 2, influenced MMRE is calculated on the basis of occurrences of 15 cost drivers. This influenced MMRE shows the effectiveness of each cost driver in the sequence of development of efforts in terms of person-months. In this process, we take sample data having 18 input parameters, that is, 15 cost drivers, modes, source lines, and actual effort. The estimated efforts are calculated for the sample data by nullifying the effect of cost driver one by one. These efforts are used to calculate the influenced MMRE for each cost driver corresponding to the actual effort provided in the sample data. The difference between influenced MMRE and original MMRE is recorded in the list along with driver.

In submodel 3, the list with the difference of influenced MMRE and original MMRE is sorted in the descending order of the difference to provide the significant occurrence of the driver. This order has been named as Sig.

In submodel 4 we will try to minimize the MMRE by updating the value of cost driver with the help of genetic algorithm in the order of their significance. This is done by selecting the projects falling in the category of particular cost driver and then using the genetic algorithm operator. The results obtained are evaluated using the fitness function as MMRE. If the MMRE is reduced, the cost driver value for particular rating is updated, otherwise discarded. The reduced MMRE will be recorded as Mmod which will be used as the MMRE for the remaining cost drivers.

5. Evaluation Method

5.1. Conceptual View. Software cost estimation models need to be quantitatively evaluated in terms of estimation accuracy to improve the modeling process. Some rules or the measurements must be provided for model assessment purpose. This measurement of accuracy defines how close the estimated result is with its actual value. Software cost estimates play significant role in delivering software projects. As a result, researchers have proposed the most widely used evaluation criterion to assess the performance of software prediction models, that is, the mean magnitude of relative error (MMRE), to evaluate the opulence of prediction systems. MMRE is usually computed by following standard evaluation processes such as cross-validation [39]. It is independent of size scale and effort units. Comparisons can be made across data sets and prediction model types [40].

COCOMO computes effort on the basis of source lines of codes. In intermediate COCOMO, Boehm used 15 more predictor variables called cost drivers, which are required to calibrate the nominal effort of a project to the actual project environment. The values are set to each cost driver according to the properties of the specific software project. These numerical values of 15 cost drivers are multiplied to get the effort adjustment factor, that is, EAF.

5.2. Data Analysis. Performance of estimation methods is usually evaluated by several ratio measurements of accuracy metrics including RE (relative error), MRE (magnitude of relative error), and MMRE (mean magnitude of relative error) which are computed as follows:

$$\begin{aligned} \text{RE} &= \frac{\text{Estimated efforts} - \text{Actual efforts}}{\text{Actual Efforts}}, \\ \text{MRE} &= \frac{\text{Estimated Efforts} \sim \text{Actual Efforts}}{\text{Actual Efforts}}, \\ \text{MMRE} &= \sum \frac{\text{MRE}}{N}. \end{aligned} \tag{3}$$

Another parameter used in evaluation of performance of estimation method is PRED (percentage of prediction) which is determined as

$$\text{PRED}\,(X) = \frac{A}{N}, \tag{4}$$

Table 2: Significant occurrences of cost drivers.

1	acap
2	pcap
3	aexp
4	rely
5	Virt
6	vexp
7	time
8	modp
9	cplx
10	data
11	tool
12	sced
13	lexp
14	turn
15	stor

where A is the number of projects with MRE less than or equal to level X and N is the number of considered projects. Usually, the acceptable level of X is 0.25 and the various methods are compared based on this level. Decreasing of MMRE and increasing of PRED are the main aim of all estimation techniques.

6. Results and Discussion

6.1. Dataset Description. Experiments were done by taking 63 COCOMO 81 based dataset used by NASA and various other calculations performed on it. 93 NASA projects from different centers for projects from the years of 1971 to 1987 were collected by Jairus Hihn, JPL, NASA, Manager of SQIP Measurement and Benchmarking Element. The proposed model is validated by these datasets. These are one of the most analyzed data sets. The independent variable used is "adjusted delivered source instructions," which takes into account the variation of effort when adapting software. COCOMO is built upon these data points, by introducing many factors in the form of multipliers.

These datasets include 156 historical projects with 17 effort drivers and one dependent variable of the software development effort.

6.2. Result Analysis. Cost drivers play a vital role in estimation of the efforts and cost to be incurred. They show characteristics of software development that influence effort in carrying out a certain project. Cost drivers are selected based on the arguments that they have a linear effect on effort. COCOMO cost drivers are the basis for the analysis of proposed algorithm. Table 1 depicts the COCOMO effort multipliers.

Significance of 15 cost drivers can be shown by their impact on MMRE of efforts on original 63 NASA datasets. The significance occurrences of 15 cost drivers are calculated by applying step 1 to step 4 which are shown in Table 2.

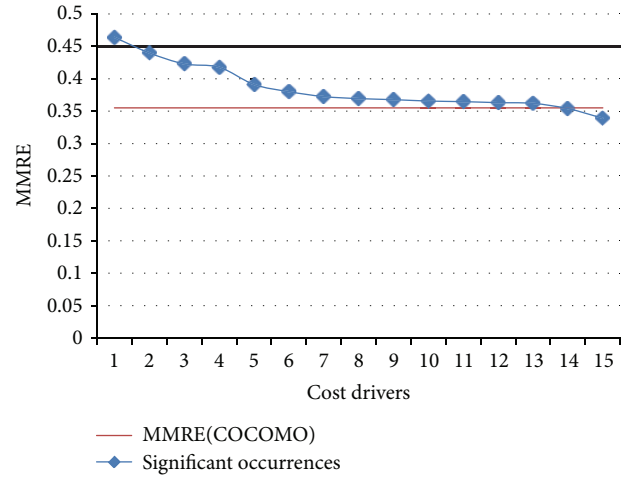

Figure 1: Relationship between MMRE and cost drivers.

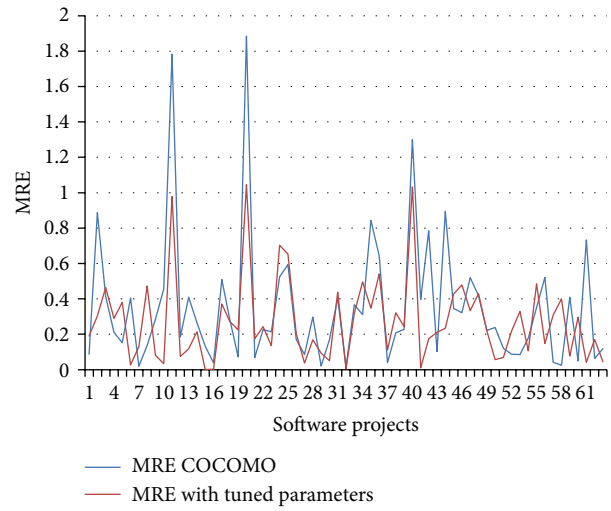

Figure 2: Comparison of MRE for NASA 63 projects.

The occurrence of each cost driver is having linearity with the MMRE calculated between actual efforts produced and estimated effort with COCOMO. In Figure 1, each cost driver moves against MMRE that is constant for all cases. The effect of each cost driver on the MMRE is the significant aspect of deriving the occurrence of cost drivers. The proportionate relationship can be seen from Figure 1, where the higher influenced MMRE with each independent cost driver against constant value of MMRE is the most significant and those with lower values are less significant.

Once significant occurrences of the cost drivers are found, the sequence of cost drivers is used to produce tuned values for different ratings of various cost drivers. Step 5 and step 6 are used to generate the new values of available cost drivers. Table 3 reveals the tuned values in preexisting cost drivers.

The proposed algorithm is validated with two different datasets of NASA projects. According to the evaluation criteria, the proposed method has marginal difference in efforts with actual project efforts in comparison to COCOMO

Table 3: Proposed algorithm based cost drivers.

Cost drivers	Very low	Low	Nominal	High	Very high	Extra high
acap	1.46	1.19	0.9	0.86	0.71	
pcap	1.42	0.9	1	0.86	0.7	
aexp	1.29	1.4	1	0.91	0.82	
modp	1.38	0.92	1	0.91	0.82	
tool	1.24	1.1	0.99	0.93	0.83	
vexp	1.38	1.03	1	0.9		
lexp	1.14	1.08	0.9	0.95		
sced	1.23	1.08	0.99	1.04	1.1	
stor			1	1.06	1.19	1.38
data		1.03	0.9	1.06	1.38	
time			0.9	1.11	1.3	1.66
turn		0.97	0.92	1.03	0.9	
virt		0.87	1	1.15	1.3	
cplx	0.7	0.85	1.11	1.15	1.16	1.65
rely	0.75	0.88	1	1.25	1.4	

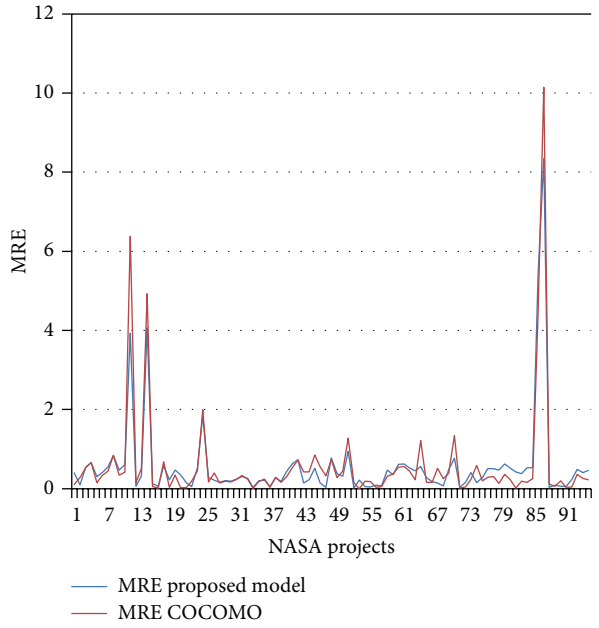

Figure 3: Comparison of MRE for NASA 93 projects.

generated efforts, shown in Figure 2. Most of the results are kept near to the mean of MRE for 63 data values. Other 93 datasets were also used to evaluate the projects with the proposed method (Figure 3 and Figure 4).

A comparison is made between proposed method and other estimation methods by MMRE in Table 4. Proposed method is having average error 0.27 with actual efforts and COCOMO produces a bit higher percentage of error with actual efforts. Proposed model is working efficiently for other 93 datasets as well.

Essentially, we want to measure useful functionality produced per time unit. Productivity is another measurement of effectiveness of the model. It is a measure of the rate or ratio at which individual software developers involved in software development produce software and associated documentation.

Higher productivity reflects the better quality achievement for the project development. Proposed method is having productivity 0.29 which is closer to the actual efforts 0.27 as productivity. Seven percent of proposed method productivity is increased and 9 percent of COCOMO productivity is decreased in comparison with actual productivity (Table 5). So the percentage of difference between proposed method and COCOMO results is approximately 17.95.

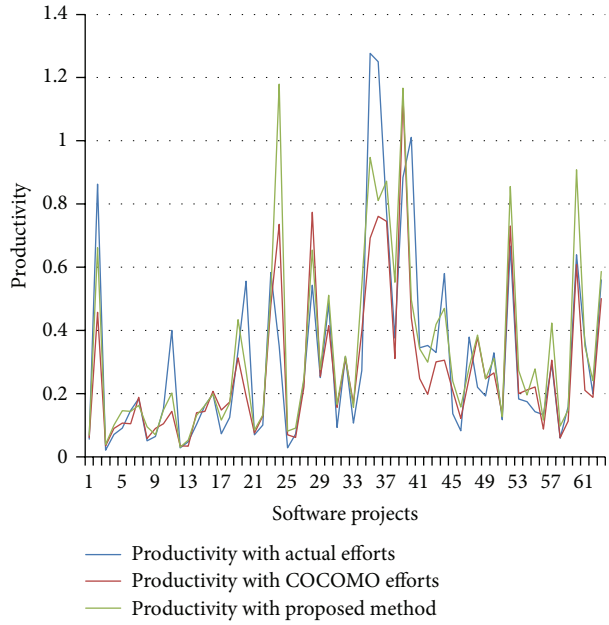

Figure 4: Comparison of productivity for NASA 63 projects.

Table 4: The MMRE for two different methods.

MMRE (for 63 datasets)		MMRE (for 93 datasets)	
COCOMO versus actual	Proposed method versus actual	COCOMO versus actual	Proposed method versus actual
0.36	0.27	0.59	0.56

Table 5: The productivity of various approaches.

	Productivity (COCOMO)	Productivity (proposed method)	Productivity (actual)
	0.25	0.30	0.28
Difference from actual	0.03	0.02	

Table 6: MMRE of NASA 63 projects for various project modes.

Project mode	No. of projects (63)	MMRE for proposed method	MMRE for COCOMO
Embedded	27	0.29	0.39
Organic	25	0.28	0.37
Semidetached	11	0.22	0.23

Table 7: MMRE of NASA 93 projects for various project modes.

Project mode	No. of projects (93)	MMRE for proposed method	MMRE for COCOMO
Embedded	21	0.72	0.82
Organic	3	0.8	0.88
Semidetached	69	0.51	0.51

Tables 6 and 7 depict the presence of error in all three categories of project modes for two different types of datasets. The comparison was made between proposed model generated results versus COCOMO results. We also evaluate the different type of project application categorically; 80% of total datasets are producing the results which are better than the COCOMO based results (Table 8).

PRED was calculated with the two separate approaches and Table 9 depicts that, for 3 different PRED assumptions, proposed method is producing approximately 6.665%, 8.01%, and 8.34% increase in PRED, respectively.

7. Conclusion

Work carried out in the paper explores the inter-relationship among different dimensions of data driven software projects, namely, project size and effort. The above-mentioned results demonstrate that applying proposed method to the software effort estimation is by far the most feasible approach for addressing the problem of apprehension and ambiguity existing in software effort drivers. Order of occurrence of various cost drivers has a significant impact on overall efforts in project estimation. Small adjustments to the COCOMO cost drivers bring significant improvements to the quality criteria applied to the proposed approach. Proposed method is producing tuned values of the cost drivers, which are effective enough to improve the productivity of the projects. Prediction at different levels of MRE for each project reflects

Table 8: Description of projects on application basis.

Type of application	No. of projects	MMRE COCOMO	MMRE proposed method
Application_ground	2	0.28	0.25
Avionics	11	0.95	0.80
Avionics monitoring	30	0.66	0.55
Batch data processing	2	0.08	0.12
Communications	1	0.18	0.05
Data capture	3	0.09	0.07
Launch processing	1	0.32	0.46
Mission planning	20	0.38	0.34
Monitor_control	8	0.20	0.50
Operating system	4	3.82	3.63
Real data processing	3	0.12	0.06
Science	2	0.18	0.41
Simulation	4	0.17	0.29
Utility	2	0.12	0.31

Table 9: Pred calculation at different values for both the models.

	PRED					
	COCOMO			Proposed method		
	10	20	30	10	20	30
Percentage of 63 NASA datasets	23.81	39.68	57.14	25.4	42.86	61.91

the percentage of projects with desired accuracy. Furthermore, this model is validated on two different datasets which represents better estimation accuracy as compared to the COCOMO 81 based NASA 63 and NASA 93 datasets. The utilization of proposed algorithm for other applications in the software engineering field can also be explored in the future.

Conflict of Interests

The authors certify that there is no actual or potential conflict of interests in relation to this paper. The American Company TRW Systems Inc. has been referred to as the company where Barry W. Boehm, the developer of COCOMO, worked.

References

[1] K. M. Furulund and K. Moløkken-Østvold, "Increasing software effort estimation accuracy—using experience data, estimation models and checklists," in *Proceedings of the 7th International Conference on Quality Software (QSIC '07)*, pp. 342–347, Portland, OR, USA, October 2007.

[2] Q. Alam, P. Bhatia, and S. Sarwar, *Systematic Review of Effort Estimation and Cost Estimation*, Institute of Management Studies, Roorkee, India, 2012.

[3] J. J. Dolado, *On the Problem of the Software Cost Function*, Facultad de Informatica, Universidad del Pais Vasco-Euskal Herriko Unibertsitatea, Gipuzkoa, Spain, 2000.

[4] K. Molokken and M. Jorgensen, "A review of software surveys on software effort estimation," in *Proceedings of the International Symposium on Empirical Software Engineering (ISESE '03)*, pp. 220–230, 2003.

[5] F. Ferrucci, C. Gravino, R. Oliveto, and F. Sarro, "Genetic programming for effort estimation: an analysis of the impact of different fitness functions," in *Proceedings of the 2nd International Symposium on Search Based Software Engineering (SSBSE '10)*, pp. 89–98, IEEE Computer Society, DMI, University of Salerno, Benevento, Italy, October 2010.

[6] A. F. Sheta, "Estimation of the COCOMO model parameters using genetic algorithms for NASA software projects," *Journal of Computer Science*, vol. 2, no. 2, pp. 118–123, 2006.

[7] B. W. Boehm, *Software Engineering Economics*, Prentice Hall, IEEE, 1984.

[8] J. Magne and M. Shepperd, "A Systematic Review Of Software Development Cost Estimation Studies," *IEEE Transactions on Software Engineering*, vol. 33, no. 1, pp. 33–53, 2007.

[9] P. L. Braga, A. L. I. Oliveira, and S. R. L. Meira, "A GA-based feature selection and parameters optimization for support vector regression applied to software effort estimation," in *Proceedings of the 23rd Annual ACM Symposium on Applied Computing (SAC '08)*, pp. 1788–1792, Ceará, Brazil, March 2008.

[10] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Software Technology*, vol. 43, no. 14, pp. 833–839, 2001.

[11] J. Clarke, J. J. Dolado, M. Harman et al., "Reformulating software engineering as a search problem," *IEE Proceedings: Software*, vol. 150, no. 3, pp. 161–175, 2003.

[12] M. Jørgensen and S. Grimstad, "Avoiding irrelevant and misleading information when estimating development effort," *IEEE Software*, vol. 25, no. 3, pp. 78–83, 2008.

[13] A. L. Lederer and J. Prasad, "A causal model for software cost estimating error," *IEEE Transactions on Software Engineering*, vol. 24, no. 2, pp. 137–148, 1998.

[14] S. Basha and P. Dhavachelvan, "Analysis of empirical software effort estimation models," *International Journal of Computer Science and Information Security*, vol. 7, no. 3, pp. 68–77, 2010.

[15] B. L. Barber, *Investigative search of quality historical software support cost data and software support cost-related data [M.S. thesis]*, 1991.

[16] N. H. Chiu and S. J. Huang, "The adjusted analogy-based software effort estimation based on similarity distances," *Journal of Systems and Software*, vol. 80, no. 4, pp. 628–640, 2007.

[17] G. Kadoda and M. Shepperd, "Using simulation to evaluate prediction techniques," in *Proceedings of the 7th International Software Metrics Symposium (METRICS '01)*, pp. 349–359, IEEE Press, London, UK, 2001.

[18] M. J. Shepperd and G. F. Kadoda, "Comparing software prediction techniques using simulation," *IEEE Transactions on Software Engineering*, vol. 27, no. 11, pp. 1014–1022, 2001.

[19] M. J. Shepperd and C. Schofield, "Estimating software project effort using analogies," *IEEE Transactions on Software Engineering*, vol. 23, no. 11, pp. 736–743, 1997.

[20] M. Joørgensen and D. I. K. Sjøberg, "The impact of customer expectation on software development effort estimates," *International Journal of Project Management*, vol. 22, no. 4, pp. 317–325, 2004.

[21] J. Kaczmarek and M. Kucharski, "Size and effort estimation for applications written in Java," *Information and Software Technology*, vol. 46, no. 9, pp. 589–601, 2004.

[22] R. Jeffery, M. Ruhe, and I. Wieczorek, "Using public domain metrics to estimate software development effort," in *Proceedings of the 7th International Software Metrics Symposium (METRICS '01)*, pp. 16–27, IEEE Computer Society, Washington, DC, USA, April 2001.

[23] G. H. Subramanian, P. C. Pendharkar, and M. Wallace, "An empirical study of the effect of complexity, platform, and program type on software development effort of business applications," *Empirical Software Engineering*, vol. 11, no. 4, pp. 541–553, 2006.

[24] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, chapter 1–8, Addison-Wesley, New York, NY, USA, 1989.

[25] A. Heiat, "Comparison of artificial neural network and regression models for estimating software development effort," *Information and Software Technology*, vol. 44, no. 15, pp. 911–922, 2002.

[26] K. Srinivasan and D. Fisher, "Machine learning approaches to estimating software development effort," *IEEE Transactions on Software Engineering*, vol. 21, no. 2, pp. 126–137, 1995.

[27] S. J. Huang, C. Y. Lin, and N. H. Chiu, "Fuzzy decision tree approach for embedding risk assessment information into software cost estimation model," *Journal of Information Science and Engineering*, vol. 22, no. 2, pp. 297–313, 2006.

[28] M. van Genuchten and H. Koolen, "On the use of software cost models," *Information and Management*, vol. 21, no. 1, pp. 37–44, 1991.

[29] 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, pp. 639–648, 1983.

[30] I. Attarzadeh and S. H. Ow, "A novel algorithmic cost estimation model based on soft computing technique," *Journal of Computer Science*, vol. 6, no. 2, pp. 117–125, 2010.

[31] F. J. Heemstra, *Software Cost Estimation Models*, University of Technology Department of Industrial Engineering, IEEE, 1990.

[32] M. Jørgensen, B. Boehm, and S. Rifkin, "Software development effort estimation: formal models or expert judgment?" *IEEE Software*, vol. 26, no. 2, pp. 14–19, 2009.

[33] Y. F. Li, M. Xie, and T. N. Goh, "A study of genetic algorithm for project selection for analogy based software cost estimation," in *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management (IEEM '07)*, pp. 1256–1260, Singapore, December 2007.

[34] H. Liu and L. Yu, "Toward integrating feature selection algorithms for classification and clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 491–502, 2005.

[35] A. Kumar, S. Tiwari, K. K. Mishra, and A. K. Misra, "Generation of efficient test data using path selection strategy with elitist GA in regression testing," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 9, pp. 389–393, Chengdu, China, July 2010.

[36] K. K. Mishra, S. Tiwari, A. Kumar, and A. K. Misra, "An approach for mutation testing using elitist genetic algorithm," in *Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT '10)*, vol. 5, pp. 426–429, Chengdu, China, July 2010.

[37] S. Sarmady, *An Investigation on Genetic Algorithm Parameters*, P-COM0005/07(R), P-COM0088/07, School of Computer Sciences, Universiti Sains Malaysia, Penang, Malaysia, 2007.

[38] K. F. Man, K. S. Tang, and S. Kwong, *Genetic Algorithms: Concepts and Designs*, Chapter 1–10, Springer, New York, NY, USA, 2001.

[39] L. C. Briand, K. El-Emam, and I. Wieczorek, "Explaining the cost of European space and military projects," in *Proceedings of the International Conference on Software Engineering (ICSE '99)*, pp. 303–312, ACM Press, May 1999.

[40] L. C. Briand, T. Langley, and I. Wieczorek, "Replicated assessment and comparison of common software cost modeling techniques," in *Proceedings of the International Conference on Software Engineering (ICSE '22)*, pp. 377–386, ACM Press, June 2000.