Software Effort Estimation by Tuning COOCMO Model Parameters Using Differential Evolution

Sultan Aljahdali, Alaa F. Sheta

Abstract—**Accurate estimation of software projects costs represents a challenge for many government organizations such as the Department of Defenses (DOD) and NASA. Statistical models considerably used to assist in such a computation. There is still an urgent need on finding a mathematical model which can provide an accurate relationship between the software project effort/cost and the cost drivers. A powerful algorithm which can optimize such a relationship via tuning mathematical model parameters is urgently needed. In [1] two new model structures to estimate the effort required for software projects using Genetic Algorithms (GAs) were proposed as a modification to the famous COnstructive COst MOdel (COCOMO). In this paper, we follow up on our previous work and present Differential Evolution (DE) as an alternative technique to estimate the COCOMO model parameters. The performance of the developed models were tested on NASA software project dataset provided in [2]. The developed COCOMO-DE model was able to provide good estimation capabilities.**

I. Introduction

The dimension and complication of computer based-systems grown noticeably during the past few decades [3]–[6] and the tendency will certainly continue in the future. The cost of developing software is growing and the software is becoming the major cost of the system. Some NASA and Air Force projects have estimated that the cost of software development could be up to 50% of their development cost. The reason is many NASA software projects are considered highly complex system with both hardware/software. For example, the NASA Space Shuttle flies with approximately 500 thousand lines of software code on board and approximately 3.5 million lines of code in ground control station. According to Dr. Patricia Sanders, the Director of Test Systems Engineering and Evaluation at OUSD, in her 1998 Software Technology Conference keynote address,"40% of the DoD's software development costs are spent on reworking the software, which on the year 2000 equal to an annual loss of $18 billion". Furthermore, Sanders stated that only 16% of software development would finish on time and within budget.

Although many research papers appears since 1960 providing numerous models to help in computing the effort/cost for software projects, being able to provide accurate effort/cost estimation is still a challenge for many reasons. They include: 1) the uncertainty in collected measurement, 2) the estimation methods used which might have many drawbacks and 3) the cost drivers to be considered along with the development environment which might not be clearly specified.

In this paper, we provide a detailed study on the use of Differential Evolution as an optimization algorithm which can be used to tune the COCOMO model parameters such that a better effort estimate can be provided. The performance of the developed model was tested on NASA software project dataset provided in [2] and compared to the models presented in [7]. The developed models were able to provide good estimation capabilities compared to other models provided in the literature [1], [7].

II. Related Work

In the past, Soft Computing techniques were explored to build efficient effort estimation models structures [8], [9]. In [10], author explored the use of Neural Networks (NNs), Genetic Algorithms (GAs) and Genetic Programming (GP) to provide a methodology for software cost estimation. Later authors in [11], provided a detailed study on using Genetic Programming (GP), Neural Network (NN) and Linear Regression (LR) in solving the software project estimation. Many data sets provided in [12], [13] were explored with promising results. In [14], authors provided a survey on the cost estimation models using artificial neural networks. Fuzzy logic and neural networks were used for software engineering project management in [15]. A fuzzy COCOMO model was developed in [8].

Recently, many questions were introduced about the applicability of using Soft Computing and Machine Learning Techniques to solve the effort and cost estimation problem for software systems. In [1], author provided an innovative set of models modified from the famous COCOMO model with interesting results. Later on, many authors explored the same idea with some modification [16]–[19] and provided a comparison to the work presented in [1]. Exploration of the advantages of Fuzzy Logic using the Takagi-Sugeno (TS) technique on building a set of linear models over the domain of possible software Kilo Line Of Code (KLOC) were investigated in [20]. Authors in [7] presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. On doing this, Particle Swarm Optimization (PSO) was used to tune the parameters of the COCOMO model. The performance of the developed model was evaluated using NASA software projects data set [2]. A comparison between COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided with excellent modeling results.

S. H. Aljahdali is the Dean of the College of Computers and Information Systems, Taif University, Taif, Saudi Arabia (aljahdali@tu.edu.sa)

A. F. Sheta is a Professor of Computer Science and Engineering, College of Computers and Information Systems, Taif University, Taif, Saudi Arabia (asheta@tu.edu.sa)

III. Software Cost Estimation Models

A project manager needs to clearly identify the cost estimate of software development so that he/she can evaluate the project progress against expected budget, expected schedule and potentially improve resource utilization in [3]. It was found that the main cost driver for software development is the effort, where effort is translated into cost. The primary element which affects the effort estimation is the developed kilo line of code (KLOC). The KLOC include all program instructions and formal statements [21].

A. The COnstructive COst MOdel (COCOMO)

Many software cost estimation models where proposed to help in providing a high quality estimate to assist project manager in making accurate decision about their projects [22], [23]. A well known mathematical model for software cost estimation is the COCOMO model. COCOMO model was first provided by Boehm [22], [23]. This model was built based on 63 software projects. The model helps is defining mathematical equations that identify the developed time, the effort and the maintenance effort. COCOMO model is used to make estimates based upon three different software project estimates.

The three ways of estimating software project effort/cost with increasing levels of accuracy are simple, intermediate and complex models. These three models are defined using increasingly detailed mathematical relationship between the developed time, the effort and the maintenance effort [3]. The estimation accuracy is significantly improved when adopting models such as the Intermediate and Complex COCOMO models [23]. The COCOMO model has the form given in Equation 1.

$$E = a(KLOC)^b \tag{1}$$

E presents the software effort computed in man-months. The values of the parameters a and b depend mainly on the class of software project. Software projects were classified based on the complexity of the project into three categories. They are:

- Organic
- Semidetached
- Embedded.

These models exhibit some nonlinearity characteristics. Extensions of COCOMO, such as COMCOMO II, can be found [24], however, for the purpose of research reported, in this paper, the basic COMCOMO model is used. The three models are given in Table I. These models are expected to give different results according to the type of software projects [25] (i.e. Organic, semi-detached and embedded) [22], [23].

B. Other Software Effort Estimation Models

Typical models for software effort estimation are given in Table II. These models have been derived by studying large number of completed software projects from various organizations and applications to explore how project sizes mapped into project effort.

TABLE I
Basic COCOMO Models

Model name	Effort (E)	Time (D)
Organic Model	$E = 2.4(KLOC)^{1.05}$	$D = 2.5(E)^{0.38}$
Semi-Detached Model	$E = 3.0(KLOC)^{1.12}$	$D = 2.5(E)^{0.35}$
Embedded Model	$E = 3.6(KLOC)^{1.20}$	$D = 2.5(E)^{0.32}$

TABLE II
Known Effort Estimation Models

Model name	Model equation
Halstead	$E = 5.2(KLOC)^{1.50}$
Walston-Felix	$E = 0.7(KLOC)^{0.91}$
Bailey-Basili	$E = 5.5 + 0.73(KLOC)^{1.16}$
Doty (for KLOC > 9)	$E = 5.288(KLOC)^{1.047}$

IV. Differential Evolution

Differential Evolution (DE) was first presented in [26], [27] and has proven to be a promising evolutionary process for nonlinear function optimization. DE is very simple technique, fast to converge and easy to implement due to the limited number of control parameters. DE algorithm is similar to Genetic Algorithms (GAs) since they are a population based approach; it also uses similar operators such as crossover, mutation and selection [27].

The main difference between DE and GAs is that genetic algorithms consider crossover as the main operator and mutation as the background operation; while DE relies on mutation operation. The mutation operator in DE is generated using randomly sampled vectors of solutions in the population and combined them to generate the mutant vector using a selected scheme. The evolutionary process is directed by using a selection mechanism.

In [28], author presented a comprehensive study of the state of the art on DE, directions for future research and a MATLAB toolbox for Differential Evolution. The latest DE-based optimization software is avaliable in several programming languages (C, C++, MATLAB, Mathematica, Java, Fortran90, Scilab, Labview) and can be found in [27].

A. Population

The initial population for the DE algorithm is more likely to be generated randomly such as in the case of GAs, unless there is *a priori* knowledge about the problem under study. Assuming the variables to be optimized is defines as $X = [x_1, x_2, \ldots, x_n]$ where $x_i^{low} \leq x_i \leq x_i^{high}$. Thus, the initial population α_0 can be defined for $i = 1, 2, \ldots, n$ as given in Equation 2.

$$\alpha_0 = rand[0,1](x_i^{high} - x_i^{low}) + x_i^{low} \tag{2}$$

In [29], authors investigated the effect of population size on the quality of solutions and the computational effort required by the DE algorithm. They claim that there is a significant influence of the population size on the performance of DE as well as interactions between mutation strategies, population size and dimensionality of the problems. This is an issue we plan to investigate in this paper.

B. Mutation

DE utilize a parameter vectors known as $x_{i,G}$, where $i = 1, 2, \ldots, NP$, while G stands for the generation number. A mutant vector can be generated according to Equation 3.

$$v_{i,G+1} := x_{r1,G} + F \cdot (x_{r2,G} - x_{r3,G}) \quad (3)$$

$x_{r1,G}, x_{r2,G}$ and $x_{r3,G}$ are three distinct vectors selected randomly from population G. Note that the selected vectors should be different from each other and also different from the current index point $x_{i,G}$. Thus, the number of parameter vector in a population must be at least four.

F is a scaling factor, known as the step size. It is selected in the domain $(0, 2]$. F controls the amplification of the difference vector $(x_{r2,G} - x_{r3,G})$. Thus, if the population gets close to the optimal F should be decreased. The step-sizes need to be self-adapt over time according to the location of the population of individuals in the search space. This helps producing efficient search to find the optimal solutions [30].

C. Crossover

Crossover helps increasing the diversity in the population. This is similar to the process of crossover in GAs. CR is the crossover rate which is defined in the domain of $[0; 1)$. For each target vector $x_{i,G}$, a crossover vector $u_{i,G+1} = \{u_{1i,G+1}, u_{2i,G+1}, \ldots, u_{Di,G+1}\}$ is produced as given in Equation 4.

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (r_j \leq CR) \text{ or } j = m, \\ x_{ji;G} & \text{if } (r_j > CR) \text{ and } j \neq m \end{cases} \quad (4)$$

where $j = 1, 2, ..., D$; and $r_j \in [0, 1]$ is a random number uniformly distributed; and $m \in (1, 2, ..., D)$ is the randomly chosen index which ensures that $u_{i,G+1}$ gets at least one element from $v_{i,G+1}$ [31].

D. Selection

New individuals are selected for the next population if and only if they achieve a better value for the desired fitness value. The selection mechanism of DE can be presented as in Equation 5, assuming we are minimizing a function f.

$$x_{i,G+1} = \begin{cases} u_{i,G+1} & \text{if } (f(u_{i,G+1} \leq f(x_{i,G}) \\ x_{i;G} & \text{Otherwise} \end{cases} \quad (5)$$

E. Strategies

Different strategies can be adopted in DE algorithm depending upon the type of problem for which DE is applied. The strategies can vary based on the vector to be perturbed, number of difference vectors considered for perturbation, and finally the type of crossover used [26].

The generally used notation for DE is DE/x/y/z. DE stands for Differential Evolution, x represents a string denoting the vector to be perturbed, y is the number of difference vectors considered for perturbation of x, and z denotes the crossover scheme.

The strategy to be adopted for each problem is to be determined separately by trial and error. A strategy that works out to be the best for a given problem may not work well when applied for a different problem [32].

V. Evaluation Criteria

In order to check the performance of the developed model, two evaluation criteria will be adopted.

- we will compute the Variance-Accounted-For (VAF) performance criterion to measure how close the measured values to the values developed using the fuzzy models. Given that $E, \hat{E}$ are the actual effort and the estimated effort, respectively. The VAF is computed as follows:

$$VAF = \left[1 - \frac{var(E^{actual} - E^{estimated})}{var(E^{actual})}\right] \times 100\% \quad (6)$$

- In [33], authors provided an empirical study for data modeling in software engineering application and used Radial Basis Function (RBF) to develop effort estimation model. They considered the Mean Magnitude of Relative Error (MMRE) as the main performance measure. MMRE is defined as:

$$MMRE = \frac{1}{N} \sum_{i=1}^{N} \frac{|E^{actual} - E^{estimated}|}{E^{actual}} \quad (7)$$

We will also adopt these two criteria's for evaluating the cost estimation models investigated here.

VI. Experimental Results

Experiments have been conducted on a data set presented by Bailey and Basili [2] to explore strengthen of the developed DE based model. The dataset consist of the following variables:

- Kilo Line of Code ($KLOC$),
- Methodology (ME)
- Effort (E).

KLOC is described in Kilo Line of code (KLOC) and the E is in man-months. The dataset is presented in Table III. The data was split to two sets training (i.e. 13 projects) and testing/validation (i.e. 5 projects). We used the MATLAB Differential Evolution Toolbox [28] to produce our results.

VII. COCOMO-DE Effort Model based KLOC

In this paper, we plan to use DE to estimate COCOMO model parameters. This will allow us to compute the effort developed for the NASA software projects. The estimated parameters will significantly generalize the computation of the developed effort for all projects.

We run DE to estimate the parameters of the COCOMO model presented in Equation 1. We used the set of parameters given in Table IV to manage the evolutionary process of the DE. In [29], authors pointed that the selection of appropriate population size has a significant effect on the evolutionary process based DE. We computed the values of the COCOMO model parameters a and b using DE with various population sizes and presented the results in Table V. The model with optimal set of parameters is presented in Equation 8.

$$Effort = 2.4649(KLOC)^{0.8781} \quad (8)$$

TABLE III

NASA SOFTWARE PROJECT DATA

Project No.	$KLOC$	ME	Measured Effort E
1	90.2000	30.0000	115.8000
2	46.2000	20.0000	96.0000
3	46.5000	19.0000	79.0000
4	54.5000	20.0000	90.8000
5	31.1000	35.0000	39.6000
6	67.5000	29.0000	98.4000
7	12.8000	26.0000	18.9000
8	10.5000	34.0000	10.3000
9	21.5000	31.0000	28.5000
10	3.1000	26.0000	7.0000
11	4.2000	19.0000	9.0000
12	7.8000	31.0000	7.3000
13	2.1000	28.0000	5.0000
14	5.0000	29.0000	8.4000
15	78.6000	35.0000	98.7000
16	9.7000	27.0000	15.6000
17	12.5000	27.0000	23.9000
18	100.8000	34.0000	138.3000

TABLE V

COCOMO-DE: THE COMPUTED MODEL PARAMETERS FOR THE NASA DATA SET

Population size	a	b	VAF	MMRE
20	2.4649	0.8781	96.0189	0.0074
30	2.4649	0.8781	96.0189	0.0074
40	2.4649	0.8781	96.0189	0.0074
50	2.4649	0.8781	96.0189	0.0074
100	2.4649	0.8781	96.0189	0.0074
200	2.4649	0.8781	96.0189	0.0074

Figures 1 and 2 show the convergence process for DE (i.e. the best so far curves of the $\sqrt{\frac{1}{N}\sum(E^{actual} - E^{Estimate})}$, N is the whole set of measurements, and the convergence of the DE model parameters after each generation.

In Table VI, we show the actual measured and estimated effort over the given 18 projects using COCOMO-DE technique. The estimated effort are close in values to the original computed effort for the real NASA projects.

VIII. COMPARISON WITH OTHER SOFT COMPUTING TECHNIQUES

In [7], authors presented an extended work on the use of Soft Computing Techniques to build a suitable model structure to utilize improved estimations of software effort for NASA software projects. On doing this, Particle Swarm Optimization (PSO) was used to tune the parameters of the COCOMO model. A comparison between COCOMO-PSO, Fuzzy Logic (FL), Halstead, Walston-Felix, Bailey-Basili and Doty models were provided.

In Table VII, we show the MMRE criteria computed over all data set. It can be seen that the COCOMO-DE model and the COCOMO-PSO models have the same performance. They also found the same optimal set of parameters as described in Equation 8. They outperform the Halstead, Walston-Felix, Bailey-Basili and Doty models.

It is also shown that the FL model is the model which provided the minimum MMRE since there are three models

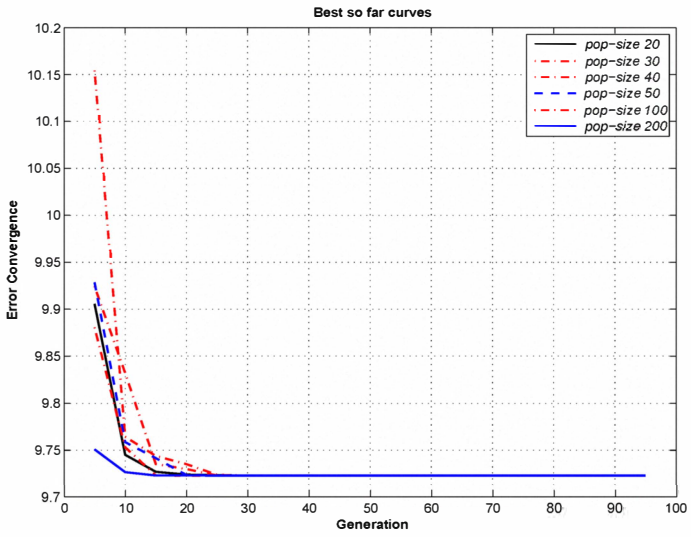

Fig. 1. Best so far curves with various population sizes

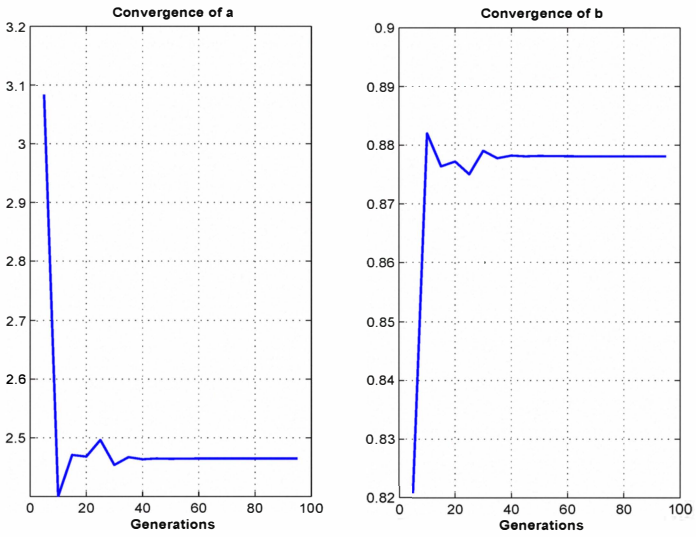

Fig. 2. Convergence of the parameters a and b with population size 20

with various membership functions. This gives an advantage of the FL model over other effort estimation models.

IX. CONCLUSIONS AND FUTURE WORK

In this paper we proposed a new model structure to estimate the software effort for projects sponsored by NASA using differential evolution. The performance of the developed model were tested on NASA software projects data presented in [2]. The developed COCOMO-DE model was capable of providing good effort estimation of compared to other known model in the literature such as Halstead, Walston-Felix, Bailey-Basili and Doty models. We suggest the use of Genetic Programming (GP) technique to build suitable model structure for the software effort estimation. GP can find a more advanced mathematical function for both the DLOC and ME such that the computed effort is more accurate.

TABLE IV
TUNING PARAMETERS FOR THE DE

D	2	Number of parameters of the objective function
Domain for a	1:10	Vector of lower higher bounds for a
Domain for b	0:1	Vector of lower higher bounds for b
NP	20, 30, 40, 50, 100, 200	NP number of population members
itermax	100	Itermax maximum number of iterations (generations)
F	0.3	DE-stepsize F ex [0, 2]
CR	0.8	CR: crossover prob. constant ex [0, 1]
Strategy	7	Strategy no. 7

TABLE VII
THE COMPUTED MMRE CRITERION FOR ALL MODELS

Model	COCOMO based DE	COCOMO based PSO	Fuzzy Model	Halstead Model	Walston-Felix Model	Bailey-Basili Model	Doty Model
MMRE	0.0074	0.0074	0.0046	0.1479	0.0822	0.0095	0.1848

TABLE VI
COCOMO-DE: MEASURED AND ESTIMATED EFFORT IN BOTH TRAINING/TESTING CASES BASED THE KLOC

Project No.	Measured Effort	DE Estimated Effort
1	115.8000	118.3116
2	96.0000	67.6772
3	79.0000	68.0439
4	90.8000	77.6870
5	39.6000	48.6343
6	98.4000	92.8783
7	18.9000	23.1770
8	10.3000	19.6445
9	28.5000	35.7351
10	7.0000	7.0942
11	9.0000	9.1414
12	7.3000	15.3272
13	5.0000	5.1249
14	8.4000	10.5737
15	98.7000	105.4666
16	15.6000	18.3868
17	23.9000	22.7226
18	138.3000	129.8115

REFERENCES

[1] 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.

[2] J. W. Bailey and V. R. Basili, "A meta model for software development resource expenditure," in *Proceedings of the International Conference on Software Engineering*, pp. 107–115, 1981.

[3] C. F. Kemere, "An empirical validation of software cost estimation models," *Communication ACM*, vol. 30, pp. 416–429, 1987.

[4] J. W. Park R, W. Goethert, "Software cost and schedule estimating: A process improvement initiative," tech. rep., 1994.

[5] M. Boraso, C. Montangero, and H. Sedehi, "Software cost estimation: An experimental study of model performances," tech. rep., 1996.

[6] K. Pillai and S. Nair, "A model for software development effort and cost estimation," *IEEE Trans. on Software Engineering*, vol. 23, pp. 485–497, 1997.

[7] A. Sheta, D. Rine, and A. Ayesh, "Development of software effort and schedule estimation models using soft computing techniques," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE CEC 2008) within the 2008 IEEE World Congress on Computational Intelligence (WCCI 2008), Hong Kong, 1-6 June*, pp. 1283–1289, 2008.

[8] J. Ryder, *Fuzzy COCOMO: Software Cost Estimation.* PhD thesis, Binghamton University, 1995.

[9] A. C. Hodgkinson and P. W. Garratt, "A neuro-fuzzy cost estimator," in *Proceedings of the Third Conference on Software Engineering and Applications*, pp. 401–406, 1999.

[10] M. A. Kelly, "A methodlolgy for software cost estimation using machine learning techniques," Master's thesis, Naval Postgratuate School, Monterey, California, 1993.

[11] J. J. Dolado and L. F. andez, "Genetic programming, neural network and linear regression in software project estimation," in *Proceedings of the INSPIRE III, Process Improvement through training and education*, pp. 157–171, British Company Society, 1998.

[12] A. J. Albrecht and J. R. Gaffney, "Software function, source lines of code, and development effort prediction: A software science validation," *IEEE Trans. Software Engineering*, vol. 9, no. 6, pp. 630–648, 1983.

[13] J. E. Matson, B. E. Barret, and J. M. Mellinchamp, "Software develop emnet cost estimation using function points," *IEEE Trans. Software Engineering*, vol. 20, no. 4, pp. 275–287, 1994.

[14] M. Shepper and C. Schofield, "Estimating software project effort using analogies," *IEEE Tran. Software Engineering*, vol. 23, pp. 736–743, 1997.

[15] S. Kumar, B. A. Krishna, and P. Satsangi, "Fuzzy systems and neural networks in software engineering project management," *Journal of Applied Intelligence*, vol. 4, pp. 31–52, 1994.

[16] H. Mittal and P. Bhatia, "A comparative study of conventional effort estimation and fuzzy effort estimation based on triangular fuzzy numbers," *International Journal of Computer Science and Security*, vol. 1, no. 4, pp. 36–47, 2007.

[17] H. Mittal and P. Bhatia, "Optimization criteria for effort estimation using fuzzy technique," *CLEI ELECTRONIC JOURNAL*, vol. 10, no. 1, pp. 1–11, 2007.

[18] M. Uysal, "Estimation of the effort component of the software projects using simulated annealing algorithm," in *World Academy of Science, Engineering and Technology*, vol. 41, pp. 258–261, 2008.

[19] P. S. Sandhu, M. Prashar, P. Bassi, and A. Bisht, "A model for estimation of efforts in development of software systems," in *World Academy of Science, Engineering and Technology*, vol. 56, pp. 148–152, 2009.

[20] A. Sheta, "Software effort estimation and stock market prediction using takagi-sugeno fuzzy models," in *Proceedings of the 2006 IEEE Fuzzy Logic Conference, Sheraton, Vancouver Wall Centre, Vancouver, BC, Canada, July 16-21*, pp. 579–586, 2006.

[21] T. Menzies, D. Port, Z. Chen, J. Hihn, and S. Stukes, "Validation methods for calibrating software effort models," in *Proceedings of*

the 27th international conference on Software Engineering (ICSE'05), (New York, NY, USA), pp. 587–595, ACM Press, 2005.
[22] B. Boehm, *Software Engineering Economics.* Englewood Cliffs, NJ, Prentice-Hall, 1981.
[23] B. Boehm, *Cost Models for Future Software Life Cycle Process: COCOMO2.* Annals of Software Engineering, 1995.
[24] B. Boehm and et all, *Software Cost Estimation with COCOMO II.* Prentice Hall PTR, 2000.
[25] O. Benediktsson, D. Dalcher, K. Reed, and M. Woodman, "COCOMO based effort estimation for iterative and incremental software development," *Software Quality Journal*, vol. 11, pp. 265–281, 2003.
[26] R. Storn, "On the usage of differential evolution for function optimization," in *Proceedings of the 1996 Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS 1996)*, pp. 519–523, 1996.
[27] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: a Practical Approach to Global Optimization.* Berlin: Springer, 2004.
[28] U. Chakraborty, *Advances in Differential Evolution (Studies in Computational Intelligence).* Berlin: Springer, 2008.
[29] R. Mallipeddi and P. N. Suganthan, "Empirical study on the effect of population size on differential evolution algorithm," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC 2008), World Congress on Computational Intelligence, June 1-6, Hong Kong*, pp. 3664–3671, 2008.
[30] A. W. Iorio and X. Li, "Incorporating directional information within a differential evolution algorithm for multi-objective optimization," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (GECCO'06)*, (New York, NY, USA), pp. 691–698, ACM, 2006.
[31] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," in *WSEAS Int. Conf. on Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, 2002.
[32] R. Storn, "Differential evolution design of an IIR filter," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 268–273, IEEE Press, 1996.
[33] M. Shin and A. L. Goel, "Emprirical data modeling in software engineering using radial basis functions," *IEEE Trans. on Software Engineering*, pp. 567–576, 2000.