American International Journal of Research in Science, Technology, Engineering & Mathematics

Available online at http://www.iasir.net

ISSN (Print): 2328-3491, ISSN (Online): 2328-3580, ISSN (CD-ROM): 2328-3629

AIJRSTEM is a refereed, indexed, peer-reviewed, multidisciplinary and open access journal published by International Association of Scientific Innovation and Research (IASIR), USA

(An Association Unifying the Sciences, Engineering, and Applied Research)

Optimization of COCOMO II Effort Estimation using Genetic Algorithm

Astha Dhiman[1], Chander Diwaker[2]

Research Scholar[1], Assistant Professor[2], Department of Computer Engineering

University Institute of Engineering & Technology

Kurukshetra University, Kurukshetra -136 119, Haryana.

INDIA

Abstract: *Software effort estimation is one of the essential steps to be carried out in the project planning. The effective and efficient development of the software requires accurate estimates. Software researchers are providing many cost estimation methods for several decades. Among those methods, COCOMO II is the most commonly used model because of its simplicity for estimating the effort in person-month for a project at the different stages. Today's effort estimation models are based on soft computing techniques as neural network, genetic algorithm, the fuzzy logic modeling etc. for finding the accurate predictive software development effort and time estimation. As there is no clear guideline for designing neural networks approach and also fuzzy approach is hard to use. Genetic Algorithm can offer some significant improvements in accuracy and has the potential to be a valid additional tool for software effort estimation. This work aims to propose a genetic algorithm for optimizing current coefficients of COCOMO II model to achieve more accuracy in estimation of software development effort.*

Keywords: *COCOMO II; Genetic Algorithm; Effort Multipliers; Scale Factors; Size*

I. Introduction

Software cost estimation is process of predicting the effort required to develop a software engineering project. While the software cost estimation may be simple in concept, it is difficult and complex in reality [1]. The major part of cost of software development is due to human-effort and most cost estimation methods focus on this aspect and give estimates in terms of person-month It determines the amount of effort necessary to complete a software project in terms of its scheduling, acquiring of resources, and meeting of budget requirements. The effective and efficient development of the software requires accurate estimates. Software researchers are providing many cost estimation techniques for several decades but the main problem persist in software engineering field. Early software estimation models are based on the regression analysis or mathematical derivations. Among those methods, COCOMO II is the most commonly used model. Today's models are based on simulation, neural network, genetic algorithm, soft computing, the fuzzy logic modeling etc [3]. In this paper, COCOMO II model used the most frequently and widely used genetic algorithm approach for optimizing the current coefficients that estimate the optimized predictive effort required for the development of software project. Genetic algorithms are optimization algorithms in the evolutionary computing techniques and proposed in 1975 by a scientist Holland. It is a natural heuristic algorithm that is used to find exact and approximate solutions. Algorithm is based on iterative improvement of current solution, but a solution set is used instead of one solution [2].

II. COCOMO II Model

COCOMO model is a regression based software cost estimation model. It was first introduced by Dr. Barry Boehm in 1981. This is one of the best-known and best-documented software effort and time schedule estimation methods. COCOMO I also called COCOMO'81 was a stable model on that time. The underlying software lifecycle is the waterfall lifecycle [4]. It has been experiencing difficulties in estimating the cost of software developed to new life cycle processes and capabilities including rapid-development process model, reuse-driven approaches and object-oriented approaches. For these reasons the newest version COCOMO II was developed [7]. The capabilities of COCOMO II are size measurement in KLOC, Function Points, or Object Points. COCOMO II adjusts for software reuse and reengineering. This new model served as a framework for an extensive current data collection and analysis effort to further refine and calibrate the model's estimation capabilities. This model has three sub-models defined below [7]:

(i) *Application Composition Model:* involves prototyping efforts to resolve potential high-risk issues like as user interfaces, software/system interaction, performance, or technology maturity. It uses object points for sizing.

(ii) *Early Design Model:* involves exploration of alternative software/system architectures. It involves use of function points for sizing and a small number of additional cost drivers.

(iii) *Post-Architecture (PA) Model:* involves actual development and maintenance of a software product. It uses source instructions and/or function points for sizing, with modifiers for reuse and software breakage [7]. Unfortunately not all of the extensions of COCOMO II are already calibrated and therefore still they are experimental. Only Post Architecture model is implemented in a calibrated software tool.

COCOMO II describes 17 Effort Multipliers (EMs) that are used in the Post-Architecture model. Table 1 summarizes COCOMO II effort multipliers or cost drivers by the categories of Product, Computer, Personnel and Project factors and also with their rating values [6].

Cost Drivers	Description	Type	Rating					
			Very low	Low	Nominal	High	Very high	Extra high
RELY	Required software reliability	Product	0.82	0.92	1.00	1.10	1.26	-
DATA	Data base Size	Product	-	0.90	1.00	1.14	1.28	-
RUSE	Developed for Reusability	Product	-	0.95	1.00	1.07	1.15	1.24
DOCU	Documentation needs	Product	0.81	0.91	1.00	1.11	1.23	-
CPLX	Product complexity	Product	0.73	0.87	1.00	1.17	1.34	1.74
TIME	Execution Time Constraints	Computer	-	-	1.00	1.11	1.29	1.63
STOR	Main Storage Constraints	Computer	-	-	1.00	1.05	1.17	1.46
PVOL	Platform Volatility	Computer	-	0.87	1.00	1.15	1.30	-
ACAP	Analyst Capability	Personnel	1.42	1.19	1.00	0.85	0.71	-
PCAP	Programmer Capability	Personnel	1.34	1.15	1.00	0.88	0.76	-
APEX	Application Experience	Personnel	1.22	1.10	1.00	0.88	0.81	-
PLEX	Platform Experience	Personnel	1.19	1.09	1.00	0.91	0.85	-
LTEX	Language ad tool experience	Personnel	1.20	1.09	1.00	0.91	0.84	-
PCON	Personnel Continuity	Project	1.29	1.12	1.00	0.90	0.81	-
TOOL	Use of Software Tools	Project	1.17	1.09	1.00	0.90	0.78	-
SITE	Multi site Development	Project	1.22	1.09	1.00	0.93	0.86	0.80
SCED	Required development schedule	Project	1.43	1.14	1.00	1.00	1.00	-

Table 1 Cost drivers for COCOMO-II PA model [6] [8]

According to paper [9], COCOMO II post architecture method calculates the software development effort (in person months) by using the following equation:

$$\text{Effort} = A \times (\text{SIZE})^{E} \times \Pi_i \ EM_i \quad [9] \quad \ldots\ldots (1)$$

Where, A- multiplicative constant with value 2.94 that scales the effort according to specific project conditions.
Size - Estimated size of a project in Kilo Source Lines of Code or Unadjusted Function Points.
E - An exponential factor that accounts for the relative economies or diseconomies of scale encountered as a software project increases its size.
EM_i - Effort Multipliers.
The coefficient E is determined by weighing the predefined scale factors (SFi) and summing them using following equation:

$$E = 0.91 + 0.01 \sum_i SF_i \quad [9] \quad \ldots\ldots (2)$$

The development time (TDEV) is derived from the effort according to the following equation:

$$\text{TDEV} = C \times (\text{Effort})^{F} \quad [9] \quad \ldots\ldots (3)$$

Latest calibration of the method shows that the multiplier C is equal to 3.67 and the coefficient F is determined is a similar way as the scale exponent by using following equation:

$$F = 0.28 + 0.002 \sum_i SF_i \quad [9] \quad \ldots\ldots (4)$$

According to paper [9], when all the factors and multipliers are taken with their nominal values, then the equations for effort and schedule are given as follows:

$$\text{Effort} = 2.94 \times (\text{Size})^{1.1} \quad [9] \quad \ldots\ldots (5)$$

Duration: TDEV = 3.67 × (Effort) $^{3.18}$ [9] ...… (6)

COCOMO II is clear and effective calibration process by combining Delphi technique with algorithmic cost estimation techniques. It is tool supportive and objective. This model is repeatable, versatile [6]. But its limitation is that most of extensions are still experimental and not fully calibrated till now [9].

III. Genetic Algorithm Based Approach for optimize Estimated Effort

Today's Software development effort estimation models are based on soft computing techniques as neural network, genetic algorithm, the fuzzy logic modeling etc. for finding the accurate predictive software development effort and time estimation. As there is no clear guideline for designing neural networks approach and also fuzzy approach is hard to use. Genetic Algorithm can offer some significant improvements in accuracy and has the potential to be a valid additional tool for software effort estimation. It is a non-parametric method since it does not make any assumption about the distribution of the data and derives equations according only to the fitted values . Genetic Algorithm is one of the evolutionary methods for the effort estimation. The solution is achieved by means of a cycle of generations of candidate solutions that are pruned by criteria 'survival of the fittest' [10]. Genetic Programming (GP) is a global search technique which makes it less likely to get stuck in the local optimum. This is different from other techniques such as neural networks and gradient descent which is prone to the local optimal values. It is particularly well suited for hard problems where little is known about the underlying search space and the concept is easy to understand [11].

IV. Related Work

A. Objective - As today's project evaluation based on old coefficients of COCOMO II Post Architecture(PA) model may not match the required accuracy [13] [14]; therefore by calibration, the accuracy of results in this method will be increased and the aim of this research is to use the concept of genetic algorithm to optimize the COCOMO II PA model coefficients to achieve accurate software effort estimation and reduce the uncertainty of COCOMO II post architecture model coefficients i.e. a, b, c and d using genetic algorithm.

B. Dataset Description - Experiments have been conducted on Turkish and Industry data set presented by Ekananta Manalif [8] to optimize effort. The dataset consists of three variables. They are the Size in Kilo Line of code (KLOC), Actual effort and the predicted effort using COCOMO II PA model. The dataset is given in Table 2. Effort multipliers and scale factors rating from Very Low to Extra High related to fifteen projects are taken from Appendix B of [8].

Pr. No.	Size(KLOC)	Actual Effort (PM)	COCOMO II PA Model Predicted Effort (PM)
1	002.00	002.00	002.90
2	114.28	018.00	294.00
3	064.10	332.00	256.70
4	023.11	004.00	063.20
5	001.37	001.00	000.90
6	001.61	002.10	002.00
7	031.85	005.00	147.10
8	131.00	619.90	745.20
9	010.00	003.00	036.20
10	015.00	004.00	063.20
11	004.25	004.50	009.30
12	004.05	002.00	002.30
13	019.90	074.60	092.70
14	003.00	001.20	003.60
15	040.53	022.00	028.60

Table 2 Data sets with their size and effort values [8]

C. Proposed Algorithm - To optimize the COCOMO II Post Architecture model coefficients, the genetic algorithm is proposed. The main steps of algorithm are defined following [2]:

Step 1 - Generate Individuals randomly, for e.g. x x x x | x x x x | x x x x | x x x x

a b c d

Each of above four coefficients is expressed by 4 genes, containing integer numbers from 0 to 9. First gene is integral part of a coefficient and remaining 3 are fractions part.

Step 2 - Calculate predicted development Effort for each Project j in the dataset using the coefficients from an Individual i.

Step 3 - Calculate Individual i fitness for the Project j, the equation used is:

$$Fitness_{ij} = | Actual^{j} - Estimated^{ij} | / Actual^{j} \quad \ldots\ldots\ldots [2][8] \quad \ldots.. (7)$$

i - the individual number,

j – the project number,

$Actual^{j}$ - is actual software project effort

Estimated^{ij} _ is the estimated or predicted software project effort, using individual i.
A positive fitness value means the actual project effort is higher than the estimated effort, and negative fitness value means the actual project effort is smaller than estimated project effort [8].
Step 4 - Individual fitness is calculated as the average value of all Project specific fitness values of an Individual obtained during steps 2 and 3. The fitness value depends on the difference between real and predicted development effort. So, the fitness function value should be minimized.

$$Fitness_i = 1/n * \sum_{j=1...n} Fitness_{ij} \quad \text{..... [2]} \quad \text{…….. (8)}$$

i - the individual number, j - the project number, n – the number of projects.
Step 5 - The stopping condition defines when the algorithm must be terminated. As an alternative condition the best (minimal) and average fitness of population can be used.
Step 6 - Selection - The tournament selection method is used.
Step 7 - Using Individuals, as selected in a previous step, new Individuals are generated. The uniform crossover strategy is used.
Step 8 - None or a small number of Individuals is randomly selected in mutation. The probability to be selected the individual should be low, typically about 10%. For each Individual, random scheme of mutating genes is defined.
Step 9 - In this step, the new population is formed and best Individuals among parents and children are chosen using same selection method applied to step 6. As an option the elitism can be added to the strategy insuring that best solution will not be lost [2].

V. Result Analysis

All the working through the implemented tool has been performed. The tuning parameters for the GA evolutionary process, to optimize the COCOMO II PA model coefficients, which include the uniform crossover strategy, tournament selection mechanism and mutation rate having value 0.15. The best result is achieved using many iterations and a solution set is received from which the best individual is chosen i.e. a solution with the best fitness function value. The final fittest individual value obtained is:

2 8 1 4 | 0 8 0 4 | 3 5 1 8 | 0 3 4 3

According to this individual, the resulting optimized COCOMO II PA model coefficients are the following: a=2.814; b=0.804; c=3.518; d=0.343. Current COCOMO II PA model coefficients are the following: a=2.94; b=0.91; c=3.67; d=0.28. Using the optimized coefficients of COCOMO II, the mean relative estimation value or best fitness value for the dataset records is 3.96. It is 1.51 times less than the mean relative estimation value obtained with the current COCOMO II Post Architecture models coefficients, which is equal to 5.99. The following Table 3 shows the comparison among the Actual Effort values and Predicted Effort (person month) values for the last five project dataset using the genetic algorithm optimized and current COCOMO II PA model coefficients with their estimated project size. At the same time, in the results obtained using the coefficients optimized by genetic algorithm, the error is much lower, but it still persists.

Pr. No.	Project Size (KSLOC)	Actual Effort (PM)	Calculated Effort (PM) using coefficients optimized by GA	Calculated Effort (PM) using COCOMO II PA model current coefficients
11	4.25	4.5	7.63	9.30
12	4.05	2.0	1.89	2.30
13	19.90	74.6	71.10	92.70
14	3.00	1.20	2.97	3.60
15	40.53	22.0	17.92	28.60

Table 3 Predicted development Effort values

The graphical comparison among three effort values described in Table 3 is shown in following Figure 1. It shows clearly that optimized coefficients produces more accurate results than the old coefficients. So, Genetic Algorithm can offer some significant improvements in accuracy and has the potential to be a valid additional tool for the software effort estimation.

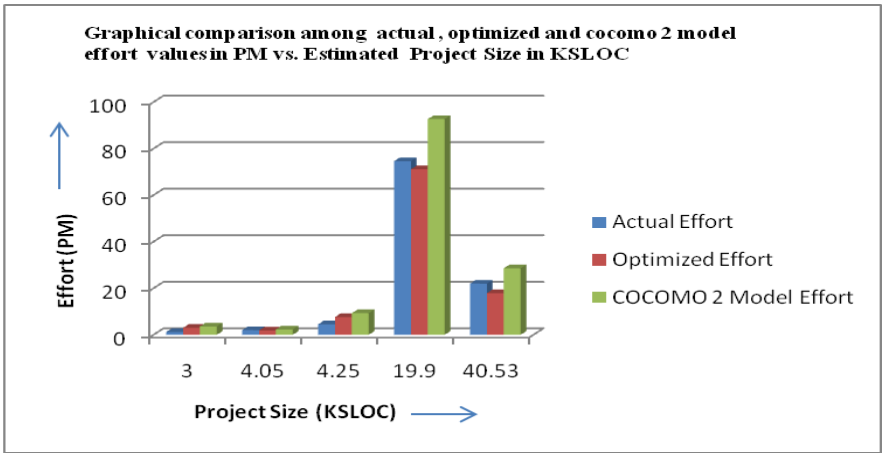

Figure 1 Graph showing comparison among Effort values vs. Size

VI. Conclusion

In this paper, the proposed algorithm is tested on TURKISH and INDUSTRY dataset and the obtained results are compared with the ones obtained using the current COCOMO II PA model coefficients. The proposed model is able to provide good estimation capabilities. It is concluded that

- By comparing the results, it can be stated that having the appropriate statistical data describing the software development projects, GA based coefficients can be used to produces better results in comparison with the results obtained using the current COCOMO II PA model coefficients.
- The results show that in most cases the results obtained using the coefficients optimized with the propose algorithm are close to the ones obtained using the current coefficients.
- The results also shows that in most cases the results obtained using the coefficients optimized with the propose algorithm are less than the real effort values.

This research indicates directions for further research. The proposed framework can be analyzed in terms of feasibility and acceptance in the industry. Trying to improve the performance of existing methods and introducing the new methods for estimation based on today's software project requirements can be future works in this area. So the research is on the way to combine different techniques for calculating the best estimate.

VII. References

[1] Maged A. Yahya, Rodina Ahmad, Sai Peck Lee, "Effects of Software Process Maturity on COCOMOII's Effort Estimation from CMMI Perspective" In 2008 IEEE, Department of Software Engineering, University of Malaya, pp. 255-256.

[2] Anna Galinina, Olga Burceva, Sergei Parshutin, "The Optimization of COCOMO Model Coefficients Using Genetic Algorithms" Riga Technical University, 2012/15, pp. 45-50.

[3] Kavita Choudhary, "GA Based Optimization of Software Development Effort Estimation" International Journal of Computer Science and Technology, Vol. 1, No. 1, Sep. 2010, ISSN: 0976-8491, pp. 38-40.

[4] Nancy Merlo–Schett, "Seminar on Software Cost Estimation" Requirements Engineering Research Group, Department of Computer Science, WS 2002/2003, pp. 3-19.

[5] Liming Wu, "The Comparison of the Software Cost Estimating Methods" University of Calgary, wul@cpsc.ucalgary.ca, pp. 2 - 12.

[6] Chander Diwaker, Astha Dhiman, Sanjeev Dhawan, "Size and Effort Estimation Techniques for Software Development" International Association of Scientific Innovation and Research, Vol. 1 & 2, No. 4, ISSN: 2279-0071, pp. 35-37.

[7] Barry Boehm, Chris Abts, Sunita Chulani, "Software development cost estimation approaches –A survey" Annals of Software Engineering 10, 2000, pp. 182-190.

[8] Ekananta Manalif, "*Fuzzy Expert-COCOMO Risk Assessment and Effort Contingency Model in Software project Management*" For the degree of Master of Engineering Science, The School of Graduate and Postdoctoral Studies, The University of Western Ontario London, Ontario, Canada.

[9] Bogdan Stępień, "Software Development Cost Estimation Methods and Research Trends" Computer Science, Vol. 5, 2003, pp. 68-82.

[10] Sumeet Kaur Sehra, Yadwinder Singh Brar, Navdeep Kaur, "Soft Computing Techniques for Software project Effort Estimation" International Journal of Advanced Computer and Mathematical Sciences, Vol. 2, No. 3, 2011, ISSN 2230-9624, pp 162-163.

[11] Jin-Cherng Lin, Chu-Ting Chang, Sheng-Yu Huang, "Research on Software Effort Estimation Combined with Genetic Algorithm and Support Vector Regression" International Symposium on Computer Science and Society, In 2011 IEEE, pp. 349-352.