J. Basic. Appl. Sci. Res., 3(2)159-166, 2013

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

An Optimization-Based Method to Increase the Accuracy of Software Development Effort Estimation

Vahid Khatibi Bardsiri, Amid Khatibi and Elham Khatibi

Department of Computer Engineering
Bardsir Branch, Islamic Azad University, Kerman, Iran

ABSTRACT

Software development effort estimation has become a challenging issue for developers, managers and customers during the last years. Some of the reasons behind this challenge are inconsistency of software projects, complexity of production process, intensive role of humans, unclear requirements and so on. In order to remedy the challenge, quite many estimation methods have been proposed in the last decades; and the attempts to increase the accuracy of estimates have not been stopped yet. Among all the existing estimation methods, analogy based estimation (ABE) is the most popular one that has been extensively used in field of software development effort estimation. This is because ABE is applicable to be used at early stages of software projects based on a smooth and clear estimation process. Despite advantages, ABE is confronted by high number of outliers and irrelevant projects frequently appeared in software project datasets. This paper proposes a hybrid model in which the genetic algorithm and ABE are combined to make a high performance estimation model. Indeed, the process of attribute weighting is adjusted so that the performance of ABE is improved. A real dataset is utilized to evaluate the accuracy of the proposed hybrid model. The comparison between the proposed model and different types of ABE certified that the performance metrics have been improved by the proposed model.
KEYWORDS: Effort Estimation; Analogy Method; Attribute Weighting; Genetic Algorithm.

INTRODUCTION

Development effort is a critical metric in the software projects. The developers and project managers are confronted by many difficulties to estimate this metric, accurately. High level of no-normality and vague specifications of software projects are the main obstacles against accurate effort estimation. The first idea for software effort estimation returns to 1950 by presenting the manual rule of thumb [1]. By increasing the number of software projects and need of user society to earn high quality software, some models based on the linear equations were presented as the software effort techniques in 1965 [2]. As the pioneers of software estimation methods we can consider the name of Larry Putnam , Barry Bohem and Joe Aron [1]. Afterward in 1973, the IBM researchers presented the first automated tool ,Interactive productivity and Quality (IPQ) [1]. Barry Boehm proposed a new method based on computing some of software project factors by means of several mathematical equations called COCOMO [3] . In addition, Boehm explained several algorithms in his book “Software Engineering Economics” [3] that still are used by researchers. Other models such as Putnam Lifecycle Management (SLIM) [4] and Software Evaluation and Estimation of Resources – Software Estimating Model (SEER-SEM) continued the principals of COCOMO [2, 5]. Introducing the Function Point (FP) as a metric for software size estimation by Albrecht [6] was the other important event in that decade. Analogy based method was proposed in 1997[7] and its usage was increased significantly because it follows the human manners to solve the problems. Although this method usually present acceptable results but there are some constraints. For example, the software project indicators may demonstrate unnecessary or unreal specifications of the projects and particularly in some projects the amount of effort is surprising. In addition, achieving to detailed information in many projects is impossible and we have to predict the amount of effort with limited features. Several studies have tried to improve the performance of analogy methods and overcome the mentioned limitations by using mathematical and statistical methods [8-12]. Since software projects are usually complicated and relations between features are hard to understand, soft computing techniques are widely used to improve the performance of analogy methods [13-18]. In current study we are going to use genetic algorithm and hybrid optimization functions to improve the performance of analogy method. This paper is organized in six sections. ABE method is described in Section1. Section 2 explains the genetic algorithm.

***Corresponding Author:** Vahid Khatibi Bardsiri, Department of Computer Engineering, Bardsir Branch, Islamic Azad University, Kerman, Iran.

The performance metrics are explained in Section 3. Section 4 comprises of the proposed model while the numerical results are explained in section 5. Finally, section 6 includes the conclusion and future works.

1. ANALOGY BASED ESTIMATION (ABE)

ABE method was introduced by Shepperd in 1997 [7]. This method relies on comparison of projects in order to estimate the effort. The process of estimation in ABE is completely simple and includes four main components as follows:

i. historical dataset
ii. similarity function
iii. solution function
iv. the associated retrieval rules

Each component can be described as follows:

i. Gathering the previous projects data and creating the historical dataset
ii. Choosing new proper features of the project such as (FP) and Line of Code (LOC)
iii. Retrieving the previous projects and calculating the similarities between the target project and the previous projects. Usually the weighted Euclidean distance and the weighted Manhattan distance are used at this stage.
iv. Estimating the effort of the target project

1.1 Similarity Function

ABE uses a similarity function which compares the attributes of two projects. There are two popular similarity functions, Euclidean Similarity (ES) and Manhattan Similarity (MS) [7]. The equation 1 shows the Euclidean Similarity function.

$$Sim(p,p') = \frac{1}{\left[\sqrt{\sum_{i=1}^{n} w_i \, Dis(f_i, f_i')} + \delta\right]} \qquad \delta = 0.0001 \qquad (1)$$

$$Dis(f_i, f_i') = \begin{cases} (f_i - f_i')^2 & \text{if } f_i \text{ and } f_i' \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases}$$

Where, p and p' are the projects, w_i is the weight assigned to each feature and varies between 0 and 1. Fi and f_i' display the ith feature of each project and n demonstrates the number of features. δ is used for obtaining the none zero results. The MS formula is very similar to the ES but it computes the absolute difference between the features. The equation 2 shows the Manhattan similarity function.

$$Sim(p,p') = \frac{1}{\left[\sum_{i=1}^{n} w_i \, Dis(f_i, f_i') + \delta\right]} \qquad \delta = 0.0001 \qquad (2)$$

$$Dis(f_i, f_i') = \begin{cases} |f_i - f_i'| & \text{if } f_i \text{ and } f_i' \text{ are numerical or ordinal} \\ 0 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i = f_i' \\ 1 & \text{if } f_i \text{ and } f_i' \text{ are nominal and } f_i \neq f_i' \end{cases}$$

1.2 Solution Functions

After choosing the K most similar projects, it is possible to estimate the effort and cost of the new project according to the selected features. The common solution functions are: the closest analogy as most similar project [19], the average of most similar projects [7], the median of most similar projects [20] and the inverse distance weighted mean [21].
The mean describes the average of the effort of K most similar projects, where K > 1. The median describes the median of the effort of K most similar projects, where K > 2.The inverse distance weighted mean adjusts the portion of each project in estimation by using equation 3.

$$C_p = \sum_{k=1}^{K} \frac{Sim(p, p_k)}{\sum_{i=1}^{K} Sim(p, p_i)} C_{p_k} \quad (3)$$

Where p shows the new project, p_k illustrates the kth most similar project, C_{pk} is the effort value of the kth most similar project p_k, $Sim(p,p_k)$ is the similarity between projects p_k and p , and K is the total number of most similar projects.

2. GENETIC ALGORITHM

Genetic algorithm is a search based algorithm which follows the concept of natural evolution. Optimization problems are the main domain of genetic algorithm usage. An initial solution is determined as a genome or a chromosome. A population including several solutions (chromosomes) is constructed and it is treated as the first generation. Each solution (chromosome) is given a fitness value based on its merit and next generation is produced by using some operators called Selection, Mutation and crossover. Some irrelevant and unsuitable solutions are omitted during the generation producing. The main operators of genetic algorithm are described as below:

2.1 Selection Operator

This operator selects the best solutions to go to the next generation. The amount of fitness value is the most important factor considered by selection operator.

2.2 Crossover Operator

Crossover makes the genetic algorithm, different as compared to the other optimization methods. The idea behind this operator is that by combining two parents (chromosomes) we can obtain two new children which are better than their parents. Some random interchanges are performed on two parents and two new children are produced. Several types of crossover are used in genetic algorithm based on user definitions.

2.3 Mutation Operator

This operator is used to hold the diversity of the population and is comparable with biological mutation. In addition, by using mutation, the problem of local minimum will be solved because chromosomes will be sufficiently different in each population. Mutation operator changes some bits in a solution and produces new solution which may be better than first one.
The overall genetic algorithm based on previous concepts can be simply described as following:

i. randomly generate a population
ii. compute the fitness of each individual in population
iii. Repeat
1. Select parents from population
2. Performing the Crossover on parents to generate next population
3. Performing the Mutation on parents to generate next population
4. Compute the fitness of each individual in new population
iv. Until the best individuals are collected

3. PERFORMANCE METRICS

Performance of estimation methods is evaluated by several metrics including Relative Error (RE), Magnitude of Relative Error (MRE) and Mean Magnitude of Relative Error (MMRE), which are computed as the following equations [7].

$$RE = \frac{(Estimate - Actual)}{Actual} \quad (4)$$

$$MRE = \frac{|Estimated - Actual|}{Actual} \quad (5)$$

$$MMRE = \frac{\sum_{i=1}^{N} MRE}{N} \quad (6)$$

The other parameter used for evaluating the performance is Percentage of the Prediction (PRED) determined as:

$$PRED(X) = \frac{A}{N} \quad (7)$$

Where, A is the number of projects with MRE less than or equal to X while N is the number of considered projects. Usually, the acceptable level of X in software cost estimation methods is 0.25 and the various methods are compared based on this level. Decrease of MMRE and increase of PRED are the main aim of all estimation techniques used in field of software development effort.

4. PROPOSED METHOD

The method proposed in this paper is a combination of genetic algorithm and analogy based effort estimation. As seen in Equation 1, ABE allocates a weight to each project attribute prior to comparison of projects. This is because the importance level of attributes may be different that leads to different weights. Efficient attribute weighting can ensure the accurate comparisons performed by similarity function. The hybrid method proposed in this paper is depicted in Figure 1.

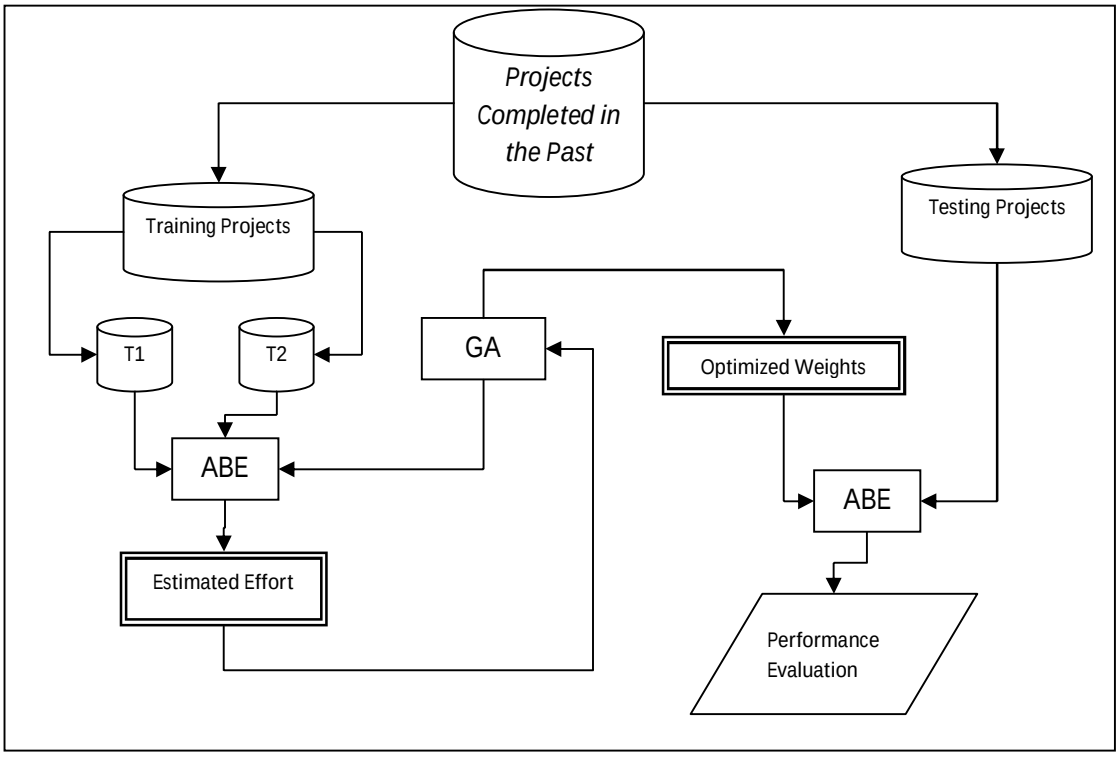

Figure 1: Structure of the proposed model

The first step is dividing all historical projects into two sets called training and testing sets. Two third of projects are located in training set and the remaining ones in the testing set. Training projects are again divided into two sets of T1 and T2 having the same size (or almost the same). Distribution of projects in sets of T1 and T2 is carried out randomly. T1 is treated as the historical data while T2 as testing data. One of the projects that exist in T2 is selected to be estimated by ABE. Genetic algorithm suggests initial weights to be employed by ABE. The development effort related to the selected project is estimated through the weights and historical data. The estimated effort and the actual effort are compared to compute the performance metric of MRE. Afterward, the next project is selected from the set of T2 to be estimated. While there is an unvisited project in T2, the mentioned steps are iterated. Once all the projects in T2 are estimated the performance metric of MMRE is computed and sent to genetic algorithm.

Based on the value of MMRE, the genetic algorithm adjusts the weights so that MMRE is minimized. The process of estimation is commenced once again using new weights. The process of attribute weighting is continued as long as the stop criteria of genetic algorithm are satisfied. Finally, the optimized weights are determined by genetic algorithm. At this moment, in order to evaluate the performance of attribute weighting, the development effort of testing projects must be estimated by means of the optimized weights. At the last step, MMRE and PRED(0.25) are computed for all the testing projects.

4.1 Dataset

Evaluation of an estimation model must be carried out using a real dataset to ensure the reliability of data and to validate the conclusions. Dasharnais [22] is one of the most popular datasets in field widely used in order to estimate the software development effort. However Desharnais is relatively old, it has been employed by many recent research works [14, 23-24]. It includes 81 software projects performed in a Canadian company, which four out of them are missing; therefore 77 projects are selected to evaluate the proposed method. Nine attributes are employed to describe a project in this dataset, which one them is categorical and the remaining ones are numerical. Moreover, the first eight attributes are treated as independent attributes and the final attribute (effort) as dependent variable.

Table 1: Desharnais dataset specifications

Feature	Description	Mean	Std Dev	Min	Max
TeamExp	Team experience	2.30	1.33	0	4.00
ManagerExp	Manager's Experience	2.65	1.52	0	7.00
Length	Length of project	11.30	6.79	1.00	36.00
Transactions	Number of transactions	177.47	146.08	9.00	886.00
Entities	Number of entities	120.55	86.11	7.00	387.00
AdjustFctor	Sum of complexity factors	27.45	10.53	5.00	52.00
PointsAdjust	N Number of adjusted function points	298.01	182.26	73.00	1,127.00
Language	Programming language	1.56	0.72	1.00	3.00
Effort	Development effort	4.83	4.189	0.55	23.94

5. NUMERICAL RESULTS

It is very optimistic to use the same data for training and testing of the proposed model. Therefore, three-fold cross validation technique is utilized for evaluating purpose. In this technique, all the data are randomly classified into three same-size sets. In each fold, two sets are treated as training data and the last set as testing data so that at the end of three folds, each set has been considered as testing set for one time. The performance metrics are computed at the end of each fold, and finally after completion of all fold the average of performance metrics is considered as the final result. Table 2 shows the results of three-fold cross validation for the proposed model when Desharnais dataset is utilized.

Table 2: Three-fold cross validation results

Folds/Metrics	MMRE	PRED(0.25)
Fold 1	0.41	0.51
Fold 2	0.52	0.46
Fold 3	0.44	0.47
Average	0.46	0.48

Since the initial goal of this paper is to improve the performance of analogy based effort estimation, different types of ABE are involved in the comparison process to show the improvement achieved by the proposed model. Four different types of ABE are considered in this section as follows: the first one is ABE with the structure explained in this paper, the second one is Regression adjusting the ABE (RABE) [25] while the third method is Linear Adjusting the ABE (LABE) [19] and the last method is Non linear adjusting the ABE (NABE) [14]. The comparison between the proposed method and different types of ABE is helpful to certify whether the combination of ABE and genetic algorithm can increase the accuracy of estimates achieved by ABE or not.

Figure 2 displays the results of MMRE obtained from three-fold cross validation in four mentioned methods as well as the propose method. It is observed that the value of MMRE for the proposed method is less than other ones, which implies increase of accuracy in estimates.

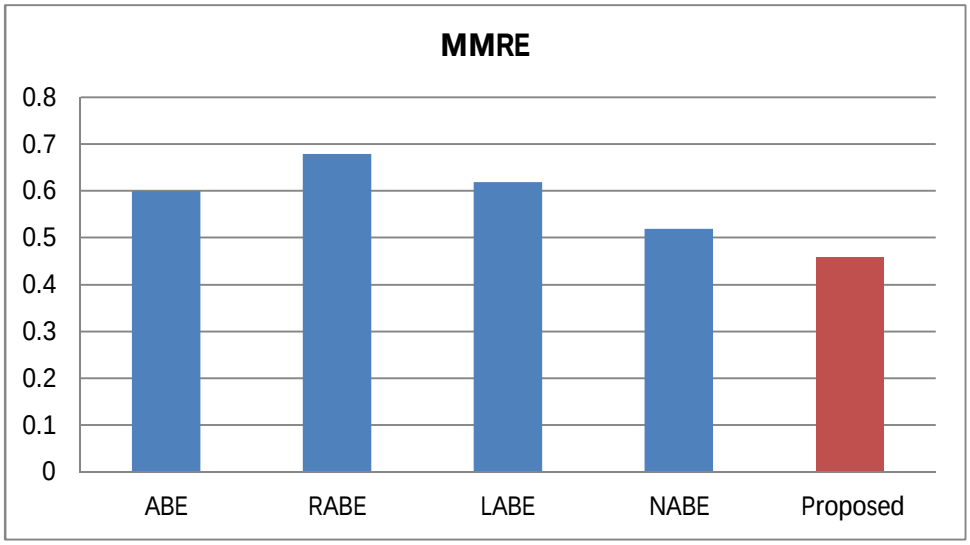

Figure 2: Proposed method versus other methods based on MMRE

PRED(0.25) is another metric that must be compared among estimation methods to show the percentage of estimates having an accuracy below 0.25. Figure 3 depicts the results of PRED(0.25) for the different types of ABE and the proposed method, as well. As seen in the figure, the proposed method achieves the highest value of PRED(0.25) among the other methods, which certifies that the most number of accurate estimates are related to the proposed method.

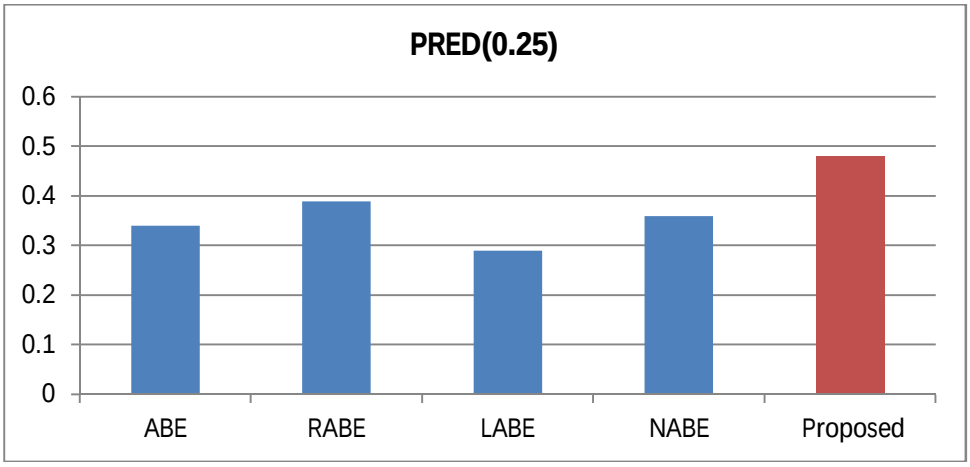

Figure 3: Proposed method versus other methods based on PRED(0.25)

In order to be ensure of the supremacy of the proposed model, both of MMRE and PRED(0.25) must be considered in the evaluation process. If MMRE shows a very low value while PRED(0.25) is also very low, this means that only a few accurate estimates have changed the value of MMRE. In other words, there are several estimates that have biased the average of error. This is why both of performance parameters have been considered in current paper.

6. CONCLUSION

Software projects as compared to the other types of project have particular specifications such as intangible product, unclear requirements and flexible production process. These differences make the process of effort estimation to be completely difficult and complicated. Since project management strongly depends on having accurate and reliable estimates, the researchers must take the problems of software development effort estimation into account in the future works. The current paper devoted the special effort to improve the performance of ABE method, as the most popular estimation method in field of software development effort estimation. Since ABE is constructed based on the comparison of projects, the main idea behind this paper is customization of comparison process so that the accuracy of estimates is increased. A model was designed to adjust the attribute weights employed by ABE. The genetic algorithm was utilized to find the most appropriate weights through a trial and error process conducted by the proposed model. A real software project dataset was employed to evaluate the performance of proposed model. The performance metrics achieved by the proposed model were compared with the different types of ABE resulted in supremacy of the proposed model. Therefore, it can be concluded that the combination of ABE and genetic algorithm can be improve the quality of comparisons and subsequently the accuracy of estimates. Applying the other optimization techniques to the proposed model can be the future work.

REFERENCES

1. Jones, C., Estimating software costs: Bringing realism to estimating. 2nd ed. 2007, New York: NY: McGraw-Hill.
2. Boehm, B.W. and R. Valerdi, Achievements and Challenges in Cocomo-Based Software Resource Estimation. IEEE Softw., 2008. **25**(5): p. 74-83.
3. Boehm, B.W., Software engineering economics. 1981, Englewood Cliffs: NJ: Prentice Hall.
4. Putnam, L.H., A general empirical solution to the macrosoftware sizing and estimating problem. IEEE Transactions on Software Engineering, 1987. **4**(4): p. 345-361.
5. Khatibi.B, V. and D.N.A. Jawawi, Software Cost Estimation Methods: A Review. Journal of Emerging Trends in Computing and Information Sciences, 2011. **2**(1): p. 21-29.
6. Albrecht, A.J. and J.A. Gaffney, Software function, source lines of codes, and development effort prediction: a software science validation. IEEE Trans Software Eng. SE, 1983. **9**(6): p. 639-648.
7. Shepperd , M. and C. Schofield, Estimating Software Project Effort Using Analogies. IEEE Transaction on software engineering,, 1997. **23**(11): p. 736-743.
8. Jianfeng, W., L. Shixian, and T. Linyan. Improve Analogy-Based Software Effort Estimation Using Principal Components Analysis and Correlation Weighting. in International conference on Software Engineering . 2009.
9. Keung, J.W. Theoretical Maximum Prediction Accuracy for Analogy-Based Software Cost Estimation. in Software Engineering Conference, 2008. APSEC '08. 15th Asia-Pacific. 2008.
10. Jingzhou, L. and R. Guenther, Analysis of attribute weighting heuristics for analogy-based software effort estimation method AQUA+. Empirical Softw. Eng., 2008. **13**(1): p. 63-96.

11. Tosun, A., B. Turhan, and A.B. Bener, Feature weighting heuristics for analogy-based effort estimation models. Expert Systems with Applications, 2009. **36**(7): p. 10325-10333.

12. Keung, J.W. and B. Kitchenham. Optimising Project Feature Weights for Analogy-Based Software Cost Estimation using the Mantel Correlation. in 14th Asia-Pacific Conference on Software Engineering. 2007. Aichi.

13. Pahariya, J.S., V. Ravi, and M. Carr. Software cost estimation using computational intelligence techniques. in Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on. 2009.

14. Li, Y.F., M. Xie, and T.N. Goh, A study of the non-linear adjustment for analogy based software cost estimation. Empir Software Eng, 2009. **14**: p. 603-643.

15. Li, Y.F., M. Xie, and T.N. Goh, A study of project selection and feature weighting for analogy based software cost estimation. Journal of Systems and Software, 2009. **82**(2): p. 241-252.

16. N. H. Chiu, S.J.H., The Adjusted Analogy-Based Software Effort Estimation Based on Similarity Distances. Journal of Systems and Software, 2007. **80**: p. 628-640.

17. Oliveira, A.L.I., et al., GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation. Information and Software Technology, 2010. **52**(11): p. 1155-1166.

18. Khatibi Bardsiri, V., et al., A PSO-based model to increase the accuracy of software development effort estimation. Software Quality Journal: p. 1-26.

19. Walkerden, F. and R. Jeffery, An Empirical Study of Analogy-based Software Effort Estimation. Empirical Softw. Eng., 1999. **4**(2): p. 135-158.

20. Angelis, L. and I. Stamelos, A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering, 2000. **5**(1): p. 35-68.

21. Kadoda , G., et al. Experiences Using Case-Based Reasoning to Predict Software Project Effort in International Conference on Empirical Assessment and Evaluation in Software Engineering. 2000. KEELE University.

22. Desharnais, J., Analyse statistique de la productivitie des projets informatique a partie de la technique des point des foncti on. 1989, University of Montreal.

23. Auer, M., et al., Optimal project feature weights in analogy-based cost estimation: improvement and limitations. IEEE Transactions on Software Engineering, , 2006. **32**(2): p. 83-92.

24. Jodpimai, P., P. Sophatsathit, and C. Lursinsap. Estimating Software Effort with Minimum Features Using Neural Functional Approximation. in International Conference on Computational Science and Its Applications (ICCSA). 2010. Brazil.

25. Jorgensen, M., U. Indahl, and D. Sjoberg, Software effort estimation by analogy and regression toward the mean. J Syst Softw, 2003. **68**: p. 253-262.