International Journal of Applied Engineering Research ISSN 0973-4562 Volume 18, Number 1 (2023) pp. 1-6

Competitive Advantages of Differential Evolution Algorithm in Software Effort Estimation

I.Thamarai*

[*]Associate Professor, Panimalar Engineering College Chennai City Campus, Chennai 600 029, Tamil Nadu, India

Abstract

Software effort estimation is the process of calculating the effort required to develop a software product based on the input parameters that are usually partial in nature. It is an important task but most difficult and complicated step in the software product development. Estimation requires detailed information about project scope, process requirements and resources available. Inaccurate estimation leads to financial loss and delay in the projects. Due to the intangible nature of software, most of the software estimation process is unreliable. But there is a strong relationship between effort estimation and project management activities. Various methodologies have been employed to improve the procedure of software estimation. This paper reviews journal articles on software development to get the direction in the future estimation research. Several methods for software effort estimation are discussed in this paper including the data sets widely used and metrics used for evaluation. The use of evolutionary computational tools in the estimation is dealt in detail. A new model for estimation using Differential Evolution Algorithm called DEAPS is proposed and its advantages discussed**.**

Keywords: Software Effort Estimation Methods, Algorithmic and Non-Algorithmic Models, Evolutionary Computation, Differential Evolution.

1.0 Introduction

Planning a software project is one of the most important activities in any software development process. Many factors are to be considered to estimate the software cost and effort. The most important factors are size of the project, number of persons involved and schedule. Prediction of software effort is a difficult and complicated task. Software is intangible in nature. So the measurement of progress is the software process is very difficult to access. Also the requirements of software project change continually which causes the change in estimation. Inaccurate estimation of effort is the usual cause of software project failures. In [1], Magne Jorgensen and Tanja M. Grushke suggested that the type of individual lesson learned processes may have effect on the accuracy of estimates. Budget and schedule pressure also plays an important role in effort calculation according to Ning Nan and Donald E.Harter [2]. In [3], Linda M.Laird lists out the reason for the inaccurate software project estimation. They are:

a) Lack of education training
b) Confusion in the schedule
c) Inability of team members
d) Incomplete and changing requirement
e) Hope based planning

In [4], Magne Jorgenson and Martin Sheppard present a systematic review of various journals. According to them, the properties of dataset impact the result when evaluating the estimation. In [5], Karel Dejaegar et al present an overview of the literature related to software effort estimation. The parameters used were project size development and environmental related attributes. The paper gives detailed study with different processing steps and addresses many issues like data quality, missing values etc. The authors confirm that the COCOMO model performed equally well as the non-linear techniques. In [6], Tim Menzies, Andrew Butcher et al evaluated the lessons that are global to multiple projects and local to particular projects in software effort estimation. They used Promise repository and concluded that software effort estimation team should build clusters from the available data and the best cluster is the one that is near the source data that is not from the same source as the test data.

In [7], Ekrem Kocaguneli et al explored the general principles of effort estimation that can guide the design of effort estimation. The author concludes that, the effort estimation shall improve significantly, if the situations when the assumptions are violated are identified and removed. Also it is said in the paper that estimation can be improved by dynamic selection of nearest neighbor with small variance. Nicolaos Mittas and Lefteris Angelis proposed a statistical framework based on multiple comparison algorithms to rank several cost estimation models [8]. In [9], Mark Harman and Afshin Mansouri propose the application of search based optimization for the software effort estimation. They list out the advantages of using search based optimization as robustness, scalability and powerful. In [10], Ekrem kocaguneli et al propose a tool called QUICK TOOL to reduce the number of features and instances required to capture the information for software effort estimation. This reduces the complexity in data interpretation. The distance between features is taken for analysis. The tool is suitable for small data sets. In [11], Ray Ashman suggested a simple Use Case based model. In this, the relationship between estimated and actual data is used to improve the future estimates. This model works best in an interactive development process. The aim of the Use Case based model is to capture the experience of specialists and enable a consistency on the timescale.

As today's software products are largely component based, the prediction is also done on the basis of individual components of the software. The performance of a system can be predicted

by creating models to calculate the performance of every single component that comprises the full system.To improve the uncertainty in cost assessments, Magne Jorgensen provide evidence based guidelines in [12]. The methods for assessing software development cost uncertainty are based on results from empirical studies. Some of the important guidelines provided are, not to rely solely on unaided, intuition based uncertainty assessment process and to apply structured and explicit judgment based process. His advice is to combine uncertainty assessments from different sources through group work and not through mechanical combination. Thus it can be seen that various aspects has to be considered for Software Effort Estimation Models. The main aim of this paper is to give a detailed description about various software effort estimation methods. The paper is organized as follows: Section 2 consists of discussion on various traditional software estimation methods and Algorithmic Models. In Section 3, we discuss about the recent trend of using evolutionary computation methods in software effort estimation. In Section 4, we propose a new model called DEAPS for the selection of most relevant project in Analogy for Effort Estimation and its experimental results. Section 5 is the conclusion and recommendation for future work.

2.0 Software Estimation Methods

There are many traditional methods such as Expert Judgment method, Function Point method, COCOMO, SLIM Model, Case Based Reasoning Model to the recent methods that uses Neural Networks, Fuzzy Logic, Genetic Algorithm, Genetic Programming, Particle Swarm Optimization etc. The Software Effort Estimation models are primarily divided into four main categories. The first and foremost is the Expert Judgment Method, where the effort is estimated by experts in the field. Algorithmic models are based on the mathematical formulas. Some of the Algorithmic Models are FP (Function Point), COCOMO (Constructive Cost Model) and SLIM (Software Life cycle Management model). These models depends on various parameters like LOC (Lines of code), Complexity, Number of Interfaces etc. The limitation of Algorithmic models such as inflexibility led to the Non-algorithmic models. Case Based Reasoning (CBR) is a popular Non-algorithmic method. Analogy is a CBR methodology. Also with the advent of soft computing techniques, new methodology of evolutionary computation came into existence. In this, many Machine Learning methods are used including Neural Networks, GA, GP and DE. These methods are discussed in the following sections.

All the approaches have their own advantages and disadvantages. It should be noted that there is not a single method which can be said to be best for all situations. Some of the research favors the combination of methods that has been proved successful. In [13], Chao-Jung Hsu et al, integrated several software estimation methods and assigned linear weights for combinations. They proved that their model is very useful in improving estimation accuracy. In [14], Magne Jorgensen and Stein Grimstad had made a detailed study on how misleading and irrelevant information can affect software estimation. They have presented the research questions and hypothesis. Based on the answers, they concluded that the field settings that led to irrelevant information have very small impact on the effort estimation than the artificial experimental settings. They also summarize that the researchers should be more aware of the different role of lab and field experiments.

3.0. Evolutionary Computation Models

The use of Evolutionary Computation Model is suggested recently to estimate the software projects. They have the advantage of handling large search spaces. The basic idea is the Darwin's theory of evolution according to which the genetic operations between chromosomes lead to the survival of the fittest individuals. These methods are the extension of machine learning algorithm such as ANN.

a) ANN (Artificial Neural Network)

ANN Models are inspired by human neural system to solve problems. An ANN is an information processing system that has certain performance characteristics in common with the biological neural network. This type of network has two layers, namely input layer and output layer. There are links between the layers. Each link carries weights. There can be hidden layers in between these layers. Back propagation Algorithm is the most popular method for training. In this, there are two passes, a forward pass and a backward pass. In the forward pass, the weights are all fixed. During the backward pass, the weights are adjusted according to the error correction rules. The weight adjustment is based on the error produced between the desired output and actual output.

Mair et al in [15] conclude that Artificial Neural Network offer accurate effort prediction but their configuration and interpretation is difficult to achieve. In [16], Gavin R.Finnie and Gerhard E.Wittig examined the performance of ANN and CBR tools for software effort estimation. The concept of back propagation neural network on Desharnais dataset and ASMA (Australian Software Metric Association) dataset were explored by them. They proved that ANN results in high level of accuracy. They concluded that ANN models are capable of providing adequate estimation models. They also said that the performance depends on the training data. In [17], Ruchika Malhotra and Ankita Jain evaluated the use of machine learning methods such as ANN, Decision tree, Support Vector Machine in effort prediction. According to their research, it is proved that decision tree method is the best method among the three. The advantage of Artificial Neural Network is that they can handle heterogeneous database but there are no guidelines for design. Accuracy largely depends on training data set.

b) GA (Genetic Algorithm)

GA is a search based algorithm to get an optimal solution. It is a evolutionary computation method. Genetic Algorithm creates consecutive population of individuals due to which, we get optimal solution for the given problem. The search process is influenced by the following components:

a) An encoding of solutions to the problem known as chromosome
b) A function to evaluate the fitness
c) Initialization of initial population

d) Selection operator
e) Reproduction operator

The important issues related with GA are the representation of solution, selection of genetic operators and choosing the best fitness solution. GA can efficiently search through the solution space of complex problem. In [18], GA is used for project selection. The steps involved are encoding, population generation, fitness function evaluation, cross over, mutation, elitism and stopping criteria.
The two real world data sets Desharnais data set and Albrecht data set are used for the experiments. The authors applied two ABE based models on these data sets. The first model uses GA to select appropriate projects subsets. This is named as PSABE (Project Selection in Analogy Based Estimation). The second model FWABE (Feature Weighting ABE) assign relevant feature weights by GA. The results are better than the other software estimation methods. In [19], Klaus Krogmaun et al presented a reverse engineering approach that combine genetic search, static and dynamic analysis to predict the performance of software application. The performance of file sharing approach is predicted using runtime byte code count. The average accuracy is proved to be better than other methods. The main disadvantage is that this approach supports only component for which java byte code is available. This method is useful for component based applications.

c) GP (Genetic Programming)

GP is a field of Evolutionary Computation that works on tree data structure. The Non continuous functions are very common in software engineering applications due to the use of branching statements. Genetic Programming can be effectively used in such situations. Using a tree based representation in Genetic Programming requires adaptive individuals and domain specific grammar GP begin with a population of randomly created programs. The programs consist of functions corresponding to the problem domain. Each program is evaluated based on fitness function. Unlike GA, mutation operation is usually not needed in GP because the crossover operation can provide for point mutation at nodes. The process of selection and crossover of individual continues till the termination criteria are satisfied. Colin J.Burgess and Martin Lefley analyzed the potential of G.P in Software Effort Estimation in terms of accuracy and ease of use [20]. The research was based on Desharnais data set of 81 software projects. The authors prove that the use of GP offer improvement in accuracy but this improvement depends on the measure and interpretation of data used.

d) DE (Differential Evolution)

Differential Evolution is an important evolutionary computation method in recent days that can be used to improve the exploration ability. Differential Evolution is similar to Genetic Algorithm, but it differs in the sense that distance and direction information from the current population is used to guide the search process. Differential evolution (DE) is a method that optimizes a problem, iteratively to improve a solution. There are many types of DE such as Simple DE, Population based DE, Compact DE, etc. DE performs well than any other contemporary algorithm and it is proved that it offers good optimization due to higher number of local optima and higher dimensionality. In a simple DE algorithm an initial population is created by random set of individuals. For each generation, three individuals say x_1, x_2 and x_3 are selected. An off spring x'_{off} is generated by mutation as

$$x'_{off} = x_1 + F(x_2 - x_3)$$

Here, F is a scale factor. Then crossover is done based on some condition. In [21], we have proposed a new method of using Differential Evolution Algorithm for the selection of similar projects in analogy method. In the proposed algorithm which is based on population based DE, the Primary population (Pp) set consists of selected individuals. The secondary population (Ps) serves as an archive of those offspring rejected by the selection operator.

4.0 Selection of most relevant project using Differential Evolution Algorithm

The main advantages of DE are its ability to provide multiple solutions. It can be easily applied to real problems despite noisy and multidimensional space. It is simple but has effective mutation process that ensures search diversity. Here, we propose a new model called DEAPS (Differential Evolution in Analogy for Project Selection). The following Figure gives the framework for using Differential Evolution Algorithm to select the relevant project from the available set of historical projects

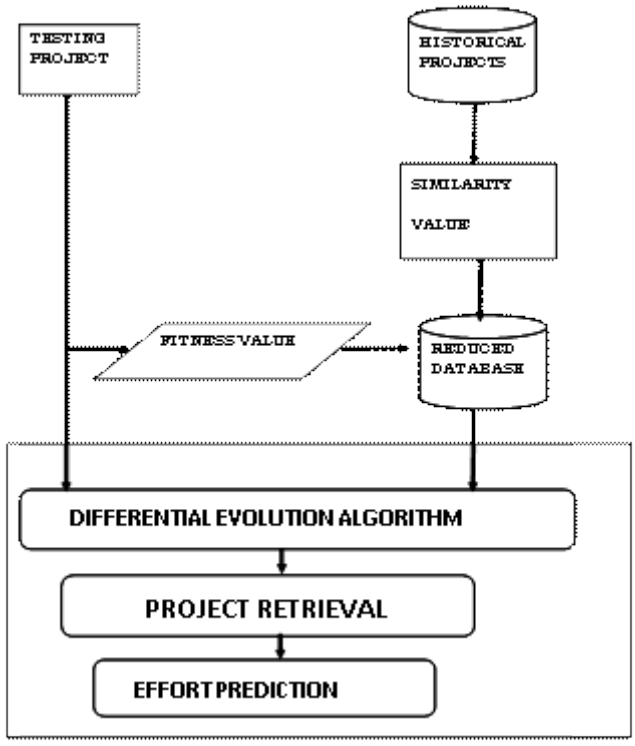

Figure 1: Differential Evolution in the selection of relevant project.

The proposed method combines Analogy concept with Differential Evolution Algorithm, The retrieval of most similar project is done in two stages. In the 1st stage, there is a reduction of historical database to a set of most similar projects using Similarity Measure. In the 2nd stage, DE is applied to retrieve the most relevant project.

4.1 Performance Evaluation Metrics

Evaluation criteria are essential for the validation of Effort Estimation Models. Metrics are used for this purpose. The most commonly used metrics are given below:

a) MRE (Magnitude of Relative Error): Relative Error is the difference between the actual value and estimated value. MRE is the absolute value of the relative error.

$$\text{MRE} = \frac{|A - E|}{|A|}$$

Where A is the Actual Effort Value and E is the Estimated Effort Value.

b) MMRE (Mean Magnitude of Relative Error): MMRE is the average percentage of the MRE over an entire dataset

$$\text{MMRE} = \sum_{i=1}^{i=n} \left| \frac{A_i - E_i}{A_i} \right| * \frac{100}{n}$$

Where A_i is the Actual Effort and Ei is the Estimated Effort of the ith project, n is the number of projects.

c) Pred(q): The prediction level pred(q), is the average percentage of prediction that falls within a specified percentage (q%) of the actual value. If the value of pred(q) is high, then the estimation is good

$$\text{pred(q)} = \frac{p}{n}$$

Where, p is the number of projects whose MRE is less than or equal to q. The commonly used metric is pred(.25) which is the percentage of predictions that is less than 25% of the actual value

$$\text{Pred (.25)} = \frac{1}{n} \sum_{i=1}^{n} \left(\frac{|A_i - E_i|}{|A_i|} \right) <= 0.25$$

4.2 Experimental Results

The input is the project parameters from the Albrechdt dataset whose values are slightly changed. It is found that, by using the DE Algorithm, the most relevant project is retrieved. The results are given in the following table:

Table 1: Results of DEAPS Model on Albrechdt Dataset

Project Id	Estimated Effort	Actual Effort	MRE	Pred (0.25)
1	100	102.4	0.0234	1
2	94	105.2	0.1065	1
3	25	11.1	1.2523	0
4	15	21.1	0.2891	0
5	35	28.8	0.2153	1
6	6	10	0.4000	0
7	10	8	0.2500	1
8	1	4.9	0.7959	0
9	20	12.9	0.5504	0
10	20	19	0.0526	1
11	10	10.8	0.0741	1
12	10	8	0.2500	1
13	9	7.5	0.2000	1
14	10	12	0.1667	1
15	1	0.5	1.0000	0
16	12	15.8	0.2405	1
17	15	18.3	0.1803	1
18	6	8.9	0.3258	0
19	30	38.1	0.2126	1
20	51	38.1	0.3386	0
21	5	3.6	0.3889	0
22	11	11.8	0.0678	1
23	1	0.5	1.0000	0
24	2	6.1	0.6721	0

Table 2 summarizes the results of various effort estimation methods on Albrechdt dataset. The corresponding test results are compared with the previous research results [22] and the comparison is also shown in diagrammatically in Figure 2, 3 and 4.

Table 2: Comparison of result with previous Models

S.No	Methods	MMRE	PRED(0.25)	MdMRE
1	ABE	0.49	0.13	0.49
2	FWABE	0.42	0.25	0.46
3	PSABE	0.39	0.38	0.45
4	ANN	0.49	0.25	0.51
5	DEAPS	0.38	0.54	0.36

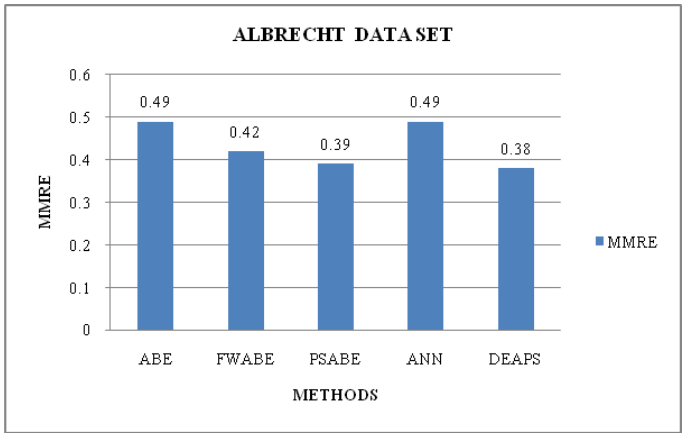

Figure 2: Comparison of MMRE values

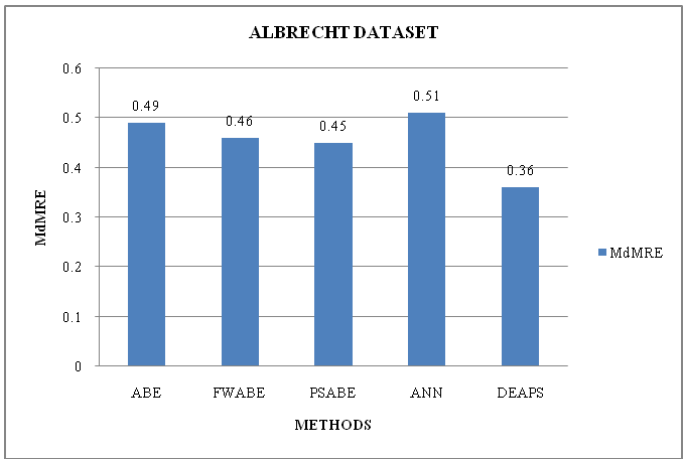

Figure 3: Comparison of MdMRE values

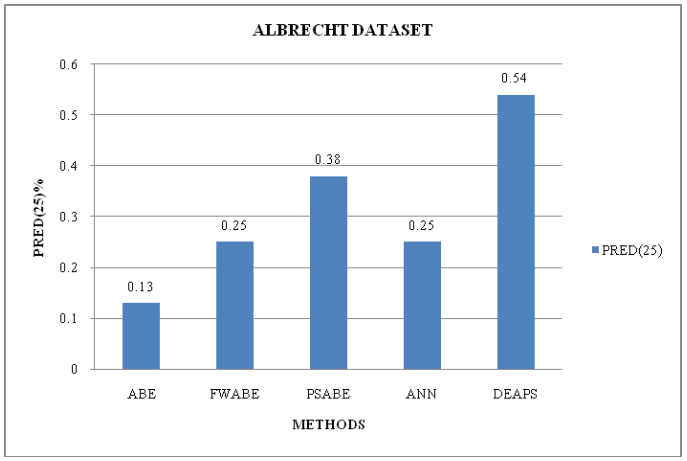

Figure 4. Comparison of Pred(0.25) values

The following Figure 5 shows a combined illustration of the test results

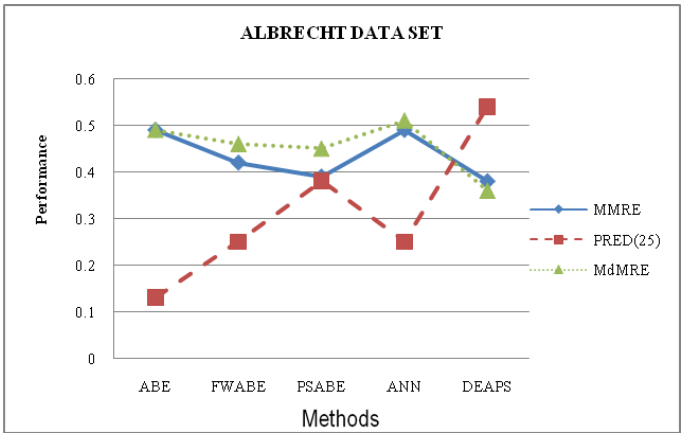

Figure 5: Results of Proposed model DEAPS on Albrechdt Dataset

The results shows that the application of the proposed Model DEAPS for the selection of relevant project has the best performance among all methods (0.26 for MMRE, 57% for Pred(0.25)and 0.22 for MdMRE). The MMRE and MdMRE are lesser than the other methods and also the probability of a project having MRE<=0.25 is also very high when compared with other models.

5.0 Conclusion and Future work:

This paper gives a detailed study of how Evolutionary Computation Algorithm has been used in the Software Effort Estimation models. Also a new approach has been proposed to simplify the Analogy based estimation. In our proposed model DEAPS, Differential Evolution Algorithm is used to select the most relevant project from set of historical projects that matches with the new project. The proposed method is implemented in JAVA platform. The experimental results are given and the observation of results clearly indicates that this model is better than existing methods. The metrics used are MMRE, MdMRE and pred(25%). As the search space is big, the Evolutionary Computation method is used which has been proved to be useful. Future work is to analyzes the performance of the model with few more real datasets and to prove efficiency of this method.

REFERENCES

[1] Magne Jorgensen, Tanja M.Grusehke and R. Gupta, “The Impact of Lessons- Learned sessions on Effort Estimation and Uncertainity Assesments”, IEEE Trans. on Software Engg., pp. 368-383, 2009.

[2] Ning Nan and Donald E.Harter, “Impact of Budget and Schedule Pressure on Software Development Cycle time and Effort”, IEEE Trans. on Software Engg., pp. 624-637, 2009

[3] Linda M Laird, “The Limitations of Estimation”, IT Pro, pp. 40-45, 2006.

[4] Magne Jorgensen and Martin Sheppard, "A Systematic review of Software Development Cost Estimation Studies", IEEE Trans. on Software Engg., pp. 33-53, 2007

[5] Karel Dejaeger, Wouter Verbeke, David Martens, Bart Baesens, "Data mining techniques for Software Effort Estimation: A Comparative study", IEEE Trans. on Software Engg., vol. 38, pp. 375-397, 2012

[6] Tim Menzies, Andrew Butcher, David Cok, Lucas layman, Forrest Shull, Burak Turhan, "Local vs Global lessons for defect Prediction and Effort Estimation", IEEE Transactions on Software Engg., Vol.39, pp. 822-834, 2013

[7] Ekram kocaguneli, Tim Menzies, Ayse Basar Bener and Jacky W Keung, "Exploiting the essential assumptions of Analogy based Effort Estimation", IEEE Transactions on Software Engg., pp. 425-437, 2012

[8] Nikolos Mittas and Lefteris Angelis, "Ranking and clustering Software Cost Estimation Model through a multiple Comparison Algorithm", IEEE Transactions on Software Engg., pp. 537-551, 2013

[9] Mark Harman, Afshin Mausouri, "Search based Software Engineering: Introduction to special issue of IEEE Trans. on Software Engg.", IEEE Transactions on Software Engg., pp. 737-741, 2010

[10] Ekrem Kocaguneli, Tim Menzies, Jacky Keung, David Cok, Ray Madachy, "Active Learning and Effort Estimation: finding the essential content of Software Effort Estimation data", IEEE Transactions on Software Engg., pp. 1039-1053, 2013

[11] Ray Ashman, "Project Estimation: A Simple Use Case based Model", IT Pro, pp. 40-44, 2004

[12] Magne Jorgensen, "Evidence based guidelines for assessment of Software development Cost Uncertainty", IEEE Transactions on Software Engg., pp. 942-954, 2005

[13] Chao Jung Hsu, Nancy Urbina Rodas, Chin Yu Huang and Kuan- Li Peng, "A Study of improving the Accuracy of Software Effort Estimation using Linear Weighted Combination", Proc. of Annual IEEE Comp. Software Applications Conference, pp. 98-103, 2010

[14] Magne Jorgensen and Stein Grimstad, "The impact of irrelevant and misleading Info. on Software development Effort Estimates: A Randomized Controlled Field Experiment", IEEE Transactions On Software Engg., pp. 695-707, 2011

[15] C. Mair, G. Kadoda, M. Lefley, K. Phalp, C. Schofield, S, Shepperd, S. Webster, "An Investigation of Machine Learning based Prediction Systems", Journal of Software Systems, pp. 23-29, 2000

[16] Gavin R. Finnie, Gerhard E.Wittig, "AI tools for Software development Effort Estimation", Proc. of International Conference on Software Engg. education and practice, pp. 83-92, 1996

[17] Ruchika Malhotra, Ankita Jain, "Software Effort Prediction using statistical and Machine Learning Methods", International Journal of Adv. Comp. science application, pp. 45-52, 2011

[18] Y.F.LI, M.Xie, T.N.Goh, "A Study of Genetic Algorithm for Project Selection for Analogy based Software Cost Estimation", Proc. of IEEE IEEM, pp. 1256-1260, 2010

[19] Klaus Krogmaun, Michael Kuperberg, Ralf Reussner, "Using Genetic Search for Reverse Engeering of Parametric Behavior Models for Performance Prediction", IEEE Transactions On Software Engg. pp. 865-877, 2007

[20] Colin J.Burgess and Martin Lefley, "Can Genetic Programming improve Software Effort Estimation? A Comparative Evaluation", Elsevier, pp. 863-873, 2001

[21] I.Thamarai, Dr.S. Murugavalli, "Using Differential Evolution in the Prediction of Software Effort", Proc. of Fourth International Conference on Advanced Computing, pp. 1-3, 2012

[22] Y.F. Li, M. Xie, T.N. Goh, "A study of project selection and feature weighting for analogy based software cost estimation", The Journal of Systems and Software – Elsevier, pp. 241-252, 2009