AN APPROACH FOR SOFTWARE EFFORT ESTIMATION USING FUZZY NUMBERS AND GENETIC ALGORITHM TO DEAL WITH UNCERTAINTY

Divya Kashyap and A. K. Misra

Department of Computer Science and Engineering
Motilal Nehru National Institute of Technology Allahabad, India
{div.kashyap@gmail.com and akm@mnnit.ac.in}

ABSTRACT

One of the most critical tasks during the software development life cycle is that of estimating the effort and time involved in the development of the software product. Estimation may be performed by many ways such as: Expert judgments, Algorithmic effort estimation, Machine learning and Analogy-based estimation. In which Analogy-based software effort estimation is the process of identifying one or more historical projects that are similar to the project being developed and then using the estimates from them. Analogy-based estimation is integrated with Fuzzy numbers in order to improve the performance of software project effort estimation during the early stages of a software development lifecycle. Because of uncertainty associated with attribute measurement and data availability, fuzzy logic is introduced in the proposed model. But hardly a historical project is exactly same as the project being estimated due to some distance associated in similarity distance. This means that the most similar project still has a similarity distance with the project being estimated in most of the cases. Therefore, the effort needs to be adjusted when the most similar project has a similarity distance with the project being estimated. To adjust the reused effort, we build an adjustment mechanism whose algorithm can derive the optimal adjustment on the reused effort using Genetic Algorithm. The proposed model Combine the fuzzy logic to estimate software effort in early stages with Genetic algorithm based adjustment mechanism may result to near the correct effort estimation.

KEYWORDS

Fuzzy-logic, Genetic algorithm, Similarity difference, Analogy based estimation, etc.

1. INTRODUCTION

One of the most critical activities during the software life cycle is that of estimating the effort and time involved in the development of the software product under consideration. This task is known as Software Cost Estimation. Estimations may be performed before, during and after the development of software. The cost and time estimates are necessary during the first phases of the software life cycle, in order to decide whether to proceed or not (feasibility study). After completion, these estimates may be useful for project productivity assessment. The principal components[3] of project costs are: Hardware costs, Travel and training costs, Effort costs (the costs of paying software engineers). The dominant cost is the effort cost. This is the most difficult to estimate and control, and has the most significant effect on overall costs. Software cost estimation is a continuing activity which starts at the proposal stage and continues throughout the lifetime of a project. Projects normally have a budget, and continual cost estimation is necessary to ensure that spending is in line with the budget. Effort can be measured in staff-hours or staff-months.

2. BACKGROUND

Software cost estimation methods fall in three main categories Algorithmic method, Expert judgment and Analogy based method[1]. In Algorithmic cost modeling: model is developed using historical cost information which relates some software metric (usually its size) to the project cost. An estimate is made of that metric and the model predicts the effort required. Algorithmic estimation involves the application of mathematical models, such as COCOMO.

Expert judgment [11] relies on the experience of experts. The accuracy of expert based prediction is low. They each estimate the project cost and the final cost estimate is arrived at by consensus. And third one is Estimation by analogy this technique is applicable when other projects in the same application domain have been completed. The cost of a new project is estimated by analogy with these completed projects.

Each technique has advantages and disadvantages. For large projects, several cost estimation techniques should be used in parallel and their results compared. Comparing with other models, Estimation by Analogy (EbA) [1] appears to be well suited to effort estimation, especially when the software product is poorly understood.

2.1 *Analogy based Estimation (EbA)*

Estimation by analogy is categorized as a composite method [17] or as a machine learning method. Estimation by analogy is mainly a data-driven[1,2] method. It compares the project under consideration (target project) with similar historical projects through their common attributes. The idea of analogy-based estimation is to determine the effort of the target project as a function of the known efforts from similar historical projects. Analogy-based estimation can be applied in the very early phase of a software project when detailed information about the project is not yet available, and can be later improved when more detailed information is accessible. Analogy Based Software Estimation is based on the principle that actual values achieved within the organization in an earlier and similar project are better indicators and predict the future project performance much better than an estimate developed using other techniques. It also facilitates bringing the organizational experience to bear on the new projects.

Analogy Based Methods are based on actual values achieved within the organization in an earlier project and hence are more reliable than other methods of estimation. This method is easy to learn and very quick to come out with a good estimate value. This technique facilitates use of organizational expertise and experience to be brought forth for the current project like no other technique of software estimation. It can be applied in the very early phase of a software project when detailed information about the project is not yet available and can be improved later.

2.2 *Fuzzy Numbers*

A fuzzy number is a quantity whose value is imprecise, rather than exact as is the case with “ordinary” (single-valued) numbers. It is an extension of a regular number in the sense that it does not refer to one single value but rather to a connected set of possible values, where each possible value has its own weight between 0 and 1. This weight is called the membership function. In fuzzy logic, a membership function provides a measure of the degree of similarity of an element to a fuzzy set. A membership function represents the degree of truth as an extension of valuation. The membership function fully defines the fuzzy set. A generalized Fuzzy number A is a Fuzzy subset of the real line R and represented as A= [a, b, c, d; w], where 0 ¡ w ¡= 1 represent decision maker. The elements a, b, c, d are real numbers. The Trapezoidal membership of this Fuzzy number A should satisfy the following conditions:

(a) A is a continuous mapping from R to the closed interval in [0, 1].
(b) $A(x) = 0$, where infinite $\leq x \leq a$ and $d \leq x \leq$ infinite.
(c) $A(x)$ is monotonically increasing in [a, b].
(d) $A(x) = w$ where $b \leq x \leq c$.
(e) $A(x)$ is monotonically decreasing in [c, d].

The membership function of a fuzzy number could be represented also by Triangular or Gaussian function.

2.3 *Genetic Algorithm*

In the computer science field of artificial intelligence[15], a genetic algorithm (GA)[18] is a search 4 heuristic that mimics the process of natural evolution. This heuristic[6] is routinely used to generate useful solutions to optimization and search problems.

1. Randomly generate a initial value population $X(0):=(x_1,x_2,\ldots..,x_N)$;

2. Compute the fitness $F(x_i)$ of each of the chromosome xi in the current population $X(t)$;

3. Create new chromosomes $X_r(t)$ by mating current chromosomes, playing mutation and recombination as the parent chromosome mate;

4. Delete numbers of the population to make room for new chromosomes;

5. Compute the fitness of $X_r(t)$ and insert these into population;

6. t :=t+1, if not (end-test) go to step 3, or else stop and return the best chromosome.

3. Proposed Work

Analogy[10] operates with one, or perhaps two past projects selected on the basis of their close similarity to the proposed project. Estimating software project effort by analogy involves a number of steps:

- Establish the attributes of our planned project, (e.g. size, language type, etc.)

- Measure or estimate the values of those project attributes. Search the Dataset for a project that closely matches the attributes of our planned project.
- Compare each of the chosen attributes[16], (size, platform etc.,).
- Use the known development effort from the selected project as an initial estimate for the target project
- Establish or adjust the initial effort estimate in light of the differences between the analogue and our planned project.
- A linear equation is adopted to reduce the estimation error caused by the similarity measures between pair of projects in the adjustment mechanism.

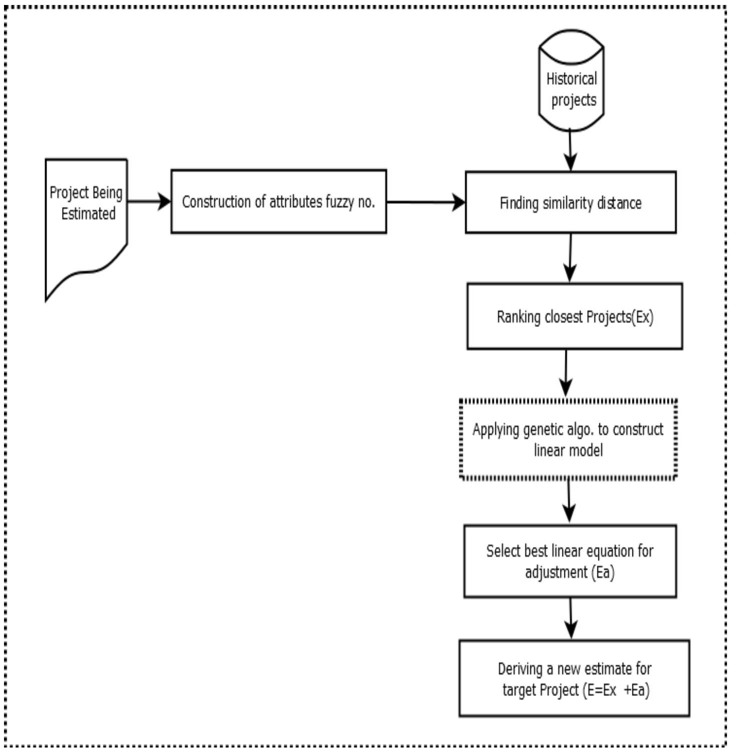
Historical projects
Project Being Estimated
Construction of attributes fuzzy no.
Finding similarity distance
Ranking closest Projects(Ex)
Applying genetic algo. to construct linear model
Select best linear equation for adjustment (Ea)
Deriving a new estimate for target Project (E=Ex +Ea)

Figure 1: Framework of adjusted fuzzy analogy-based estimation

The proposed model as shown in figure.1 is considered as a form of EA (Estimation by Analogy) and comprise of three main Stages.

- Construction of fuzzy number of attributes
- Finding Similarity distance between planed project and historical project Figure 1:
- Retrieving the closest project.
- Adjusting the reused effort using genetic algorithm

Construction of fuzzy number of attributes:

Since we choose COCOMO81 Dataset that describe the project using 17 attributes, first of all each real number of each attribute should be replaced by its corresponding Fuzzy number. Fuzzy numbers can be constructed from either expert opinion or from data [13]. Expert opinion technique is totally subjective and depends on identifying pessimistic, optimistic and most likely values for each Fuzzy number[5], where construction of using data is based on the structure of data only.

Therefore based on DATA, and using Fuzzy modeling [9], we have built the membership functions for each attributes. The numeric value of attributes in Fuzzy Number will be represented in the following sequence:

Planned Project (P): [a1, a2, a3, a4; wa]
Historical Project (H): [b1, b2, b3, b4; wb]

Finding Similarity between planned project and historical project:

To calculate the similarity distance between two fuzzy numbers we use the method proposed by Azzeh et. al[9]. This method combines the concept of geometric distance. Center of gravity (COG) and height of generalized fuzzy numbers. The degree of similarity S(P,H) between two generalized Fuzzy numbers is a composition of three elements, Height of Adjustment Ratio (HAR), Geometric Distance (GD) and Shape of Adjustment Ratio (SAF).

$$S(P,H) = HRA * (1 - GD) \div SAF \qquad (1)$$

HAR is used to assess the degree of difference in height between two generalized Fuzzy numbers.

$$HAR = \sqrt{}(\min(\frac{wa}{wb}, \frac{wb}{wa})) \qquad (2)$$

The GD is used to measure the geometric distance between two generalized Fuzzy numbers including the distance between their x-axis centroid.

$$GD = \frac{1}{5} * \left((a1 - b1) + (a2 - b2) + (a3 - b3) + (a4 - b4) + (xa - xb)\right) \qquad (3)$$

SAF is used to adjust the geometric distance, for example, if the two Fuzzy numbers have different shapes.

$$SAF = 1 + abs(ya - yb) \qquad (4)$$

Substitute all these values in equation 1, will result the similarity distance between one attribute. Similarly we can calculate similarity distance for all the attributes.

Let it is S1, S2, S3, S4….S (no. of attributes) Collectively it will give the similarity distance between two projects.

S= S1 + S2 + S3 + S4 +......S (no. of attributes)

Similarly, we will calculate similarity distance between our planned project and each of historical project to get the most similar project.

Ranking the Closest Project

After calculating the aggregated similarity measure between the planned project and each individual historical project, the similarity results are sorted according to their value. We choose the project which has highest similarity because it has highest opportunity to contribute in the final estimate. Let the effort of the closest project is E.

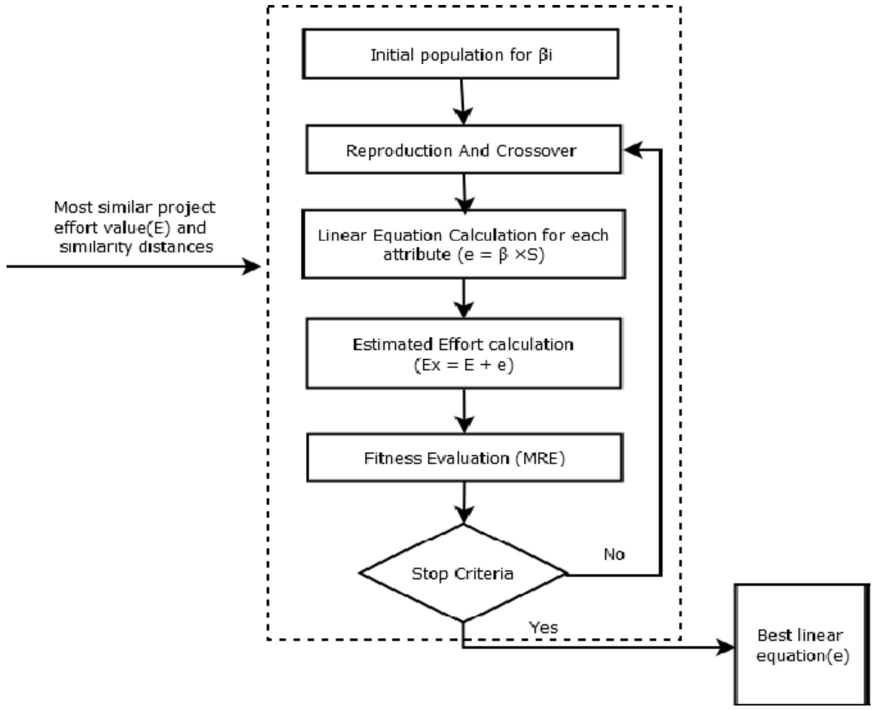
Initial population for βi
Reproduction And Crossover
Most similar project effort value(E) and similarity distances
Linear Equation Calculation for each attribute (e = β ×S)
Estimated Effort calculation (Ex = E + e)
Fitness Evaluation (MRE)
Stop Criteria
No
Yes
Best linear equation(e)

Figure 2: A framework of adjusted fuzzy analogy-based estimation

Adjusting the reused effort:

In most cases, using only the closest project to derive the new estimate is not sufficient [12]. It may lead to bad estimation accuracy. Therefore to adjust the reused effort generated using Fuzzy number we use genetic algorithm method. The framework of adjustment model using GA is shown in fig. 2.

Here, we evaluated the potential benefits of applying GA to analogy-based software effort estimation models based on Fuzzy Numbers. GA is adopted to derive a suitable linear model from the similarity distances between pairs of projects for adjusting the reused effort. GA operators include the processes of reproduction, crossover and mutation. It can be thought of as an evolutionary process where in the fitness of each chromosome is calculated in each generation.
These iterative processes optimize the coefficients of this linear equation and the combination of the best linear equation, and the combination of best linear equation with the Estimation by Analogy using Fuzzy[13] Number is the software effort estimation model For n software projects, e is the adjustment value for the reused effort in terms of the expected error between the reused effort and the actual effort of the project being estimated:

Among m effort drivers, a general form of one such linear equation is

$$e = Bi * Si$$

Here “Bi” is the coefficient of effort driver i.

Similarly we can calculate e for all effort drivers and summations of all give us the linear equation for adjustment. GA explores a linear adjustment[6,8] equation using GATOOL. Among the n projects, the coefficients “Bi” of this linear equation are treated as variables to be examined and the MRE is the value to be minimized. The number of effort drivers for a software project is the number of coefficients to be examined for GA.

Initial Population: The coefficients of linear equation are encoded in a chromosome and groups of chromosomes make up a population. Initial population enables us to specify an initial population for the genetic algorithm. Initial range specifies lower and upper bounds for the entries of the vectors in the initial population. We use the Default value of initial population and the initial range is set to [0,2].

Selection: The selection function chooses parents for the next generation based on their scaled values from the fitness scaling function. We use Tournament Selection function which selects each parent by choosing individuals at random, the number of which you can specify by Tournament size, and then choosing the best individual out of that set to be a parent.

Reproduction: Reproduction options determine how the genetic algorithm creates children at each new generation. We prefer Elite count method for reproduction. Elite count specifies the number of individuals that are guaranteed to survive to the next generation.

Crossover: Crossover combines two individuals, or parents, to form a new individual, or child, for the next generation. To get accurate result, we choose Single point Crossover function. Single point chooses a random integer n between 1 and Number of variables, and selects the vector entries numbered less than or equal to n from the first parent, selects genes numbered greater than n from the second parent, and concatenates these entries to form the child.

Stopping Criteria: Stopping criteria determine the causes to terminate algorithm. Generations specifies the maximum number of iterations the genetic algorithm performs. We choose 100 generation to terminate the algorithm. Time limit specifies the maximum time in seconds the genetic algorithm runs before stopping. Fitness limit specifies the best fitness value is less than or equal to the value of Fitness limit, the algorithm stops. We use the default value for fitness limit.

Fitness Function: Fitness Function: In our fitness function, we calculate adjusted effort by putting the value of coefficient of effort drivers.

$$Ex = E + e$$

Once we get the adjusted effort, we can calculate the Magnitude of Relative Error (MRE).

$$MRE = (Ea - Ex)/Ea$$

Where Ea is the actual effort value of the planned project.

Using the MRE as a fitness function to which we want to minimize, the coefficients of linear equation are examined. Finally we get the best linear equation (e) to adjust the Effort calculated by Estimation by Analogy using Fuzzy numbers method.

4. EXPERIMENTAL STUDY

To validate the proposed model, we use the COCOMO81 dataset. Cocomo81 dataset describe 63 projects and 17 attributes that define software projects. We left the Line of code (LOC) attribute in our experiment, since it is not available in early stage of software development. Each attribute is measured on the scale of "very low", "low", "Nominal", "High", "Very High" and "High".

Once deciding the attributes of project, we construct the fuzzy numbers of all above attributes with the help of genfis3 function (MATLAB(R2011a))[17], and calculated the similarity distances with all other historical project using equations explained in chapter 4 and selected the most similar project.

Using the effort of most similar project and calculated similarity distances between each attribute, we find the coefficients of all effort drivers in OPTIMTOOL (MATLAB (R2011a)) [17] that result the significant less error.

There is no proof on software cost estimation models to perform consistently accurate within 25. Applying genetic algorithm [14] we obtained the value of coefficients of effort drivers (shown in Table 2) that aim to minimize the MRE[7] of projects. For some of the effort drivers the coefficient value is negligible. It may be conclude that these drivers does not play important role in the adjustment of effort.

For the proposed adjusted analogy-based estimations, Fig. 3 shows the improvement of MRE based on analogy-based estimations without adjustment on effort. Fig. 4 compares the MMRE of different models with the proposed model and observe that analogy based estimation using Fuzzy numbers can be further improve after adjusting its effort using GA.

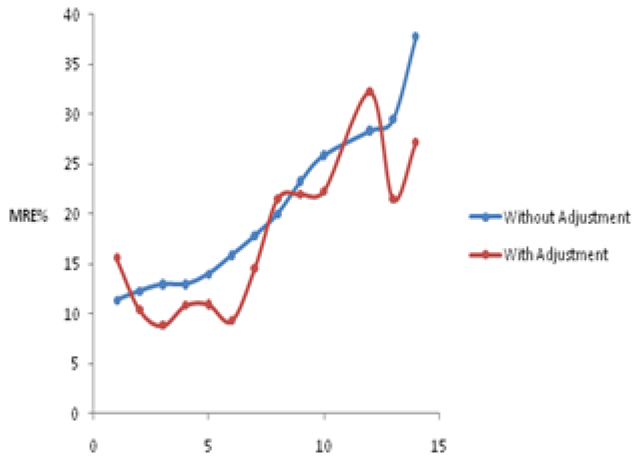

Figure 3: MRE comparison between Without adjusted EbA and Proposed model

5. Conclusion And Future Work

Delivering accurate software estimation has been a research challenge for a long time, where none of the existing estimation models has proved to consistently deliver accurate estimate. Expert based estimation is still the widely used model in industry which is based on past experience. Thus, these estimates are subjective and require high levels of expensive expertise. In this model, we consider two main problems associated with Estimation by Analogy. The first one is uncertain data in the early stage of software effort estimation and second one is that the most similar project still has similarity distance with the project being estimated in most of the cases.

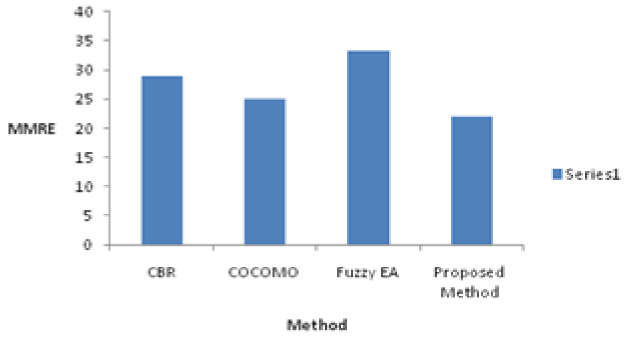

Figure 4: MMRE results of different models

Cost Driver	coefficients of drivers(Bi)
ACAP	1.235
AEXP	1.074
CPLX	0.7286
DATA	0.728
LEXP	5.71
MODP	-
PCAP	0.12
RELY	0.86
SCED	0.63
STOR	6.21
TIME	-
TOOL	-
TURN	1.19
VEXP	0.684
VIRT	0.6

Table 2: Attribute Coefficients value for COCOMO dataset

To solve the problem of uncertain, missing and vague values of attributes related to project, we used the concept of fuzzy numbers. The fuzzy model is employed in Estimation by Analogy to reduce uncertainty and improving the way to handle both numerical and categorical data in similarity measurement. Converting each attribute (real number) in to fuzzy number solve the problem of ambiguous data. After conversion, calculating the similarity distance of the target project with all historical projects results the most similar project. But the most similar project still has similarity distance with the project being estimated. To adjust the effort, we use the concept of Genetic algorithm and obtain a linear equation in terms of attributes associated with projects and coefficients of scale factors. The value of coefficients is obtained using GATOOL under the condition of low value of Magnitude of Relative Error.

Combining the concept of Fuzzy logic and Genetic algorithm in a model to estimate the software effort improve the accuracy of estimation techniques. The experiments conducted applying this model to COCOMO dataset showed significant improvement under various accuracy measures.

The coefficients () will not be same for all datasets. Therefore, to find the best value of for each dataset is the challenge for this model. Another limitation results from using fuzzy model that needs sufficient number of observations in order to construct fuzzy sets. In future, we will use feature subset selection technique to focus on important attributes that affects the effort and extends the work in such a way so that it can work for any dataset.

REFERENCES

[1] Fiona Walkerden, Ross Jeffery, “An empirical Study of Analogy-based Software Effort Estimation”, Kluwer Academic Publishers, Boston, 1999.

[2] L. Angelis, I. Stamelos, “A Simulation Tool for Efficient Analogy Based Cost Estimation”, Kluwer Academic Publishers, 2000.

[3] Practical Software Engineering , http://ksi.cpsc.ucalgary.ca/courses/45196/mildred/451/CostEffort.html

[4] Prasad Reddy, Sudha K,. R Rama Sree P, “Application of Fuzzy Logic Approach to Software Effort Estimation”, International Journal of Advanced Computer Science and Applications, Vol. 2, o. 5, 2011.

[5] B.W. Boehm, “Software Engineering Economics”, Prentice-Hall, Englewood Cliffs, NJ,1981.

[6] C.J. Burgess, M. Lefley, “Can genetic programming improve software effort estimation? A comparative evaluation”, Information and Software Technology 43, pp. 863-873, 2001.

[7] T. Foss, E. Stensrud, B. Kitchenham, I. Myrtveit, “A simulation study of the model evaluation criterion MMRE”, IEEE Transactions on Software Engineering 29 (11), pp.985-995, 2003.

[8] S.J. Huang, N.H. Chiu, “Optimization of analogy weights by genetic algorithm for software effort estimation”, Publishing in Information and Software Technology (in press).

[9] Mohammad Azzeh, Daniel Neagu, Peter I.Cowling, “Analogy-based Software effort estimation using Fuzzy numbers”, The journal of Systems and Software 84, 2010.

[10] Nikolaos Mittas,Marinos Athanasiades, Lefteris Angelis, “Improving analogy-based software cost estimation by a resampling method”, Information and Software Technology, 2007.

[11] M. Jorgenson, “A review of studies on expert estimation of software development effort”, The Journal of Systems and Software 70, pp. 37-60, 2004.

[12] A. Idri, A. Abran, T. Khoshgoftaa, “Fuzzy analogy: a new approach for software effort estimation”, 11th International Workshop in Software Measurements, 2001.

[13] M. Jorgensen, “Realim in assessment of effort estimation uncertainty: it matter how you ask.”, IEEE Transaction on software Engineering 30, 2004.

[14] K.K. Shukla, “Neuro-genetic prediction of software development effort”, Information and Software Technology 42, 2000.

[15] J. Holland, “Adaptation in Natural and Artificial Systems”, University of Michigan Press, Ann Arbor, MI, 1975.

[16] LC Brian and I Wieczorek,Resource modeling in software engineering, in Encyclopedia of Software Enginerring, 2001.

[17] Introduction of Matlab, http://www.mathworks.in/products/matlab/

[18] K.S. Tang, K.F.Man, S.Kwong and Q. He, “Genetic Algorithms and their Applications”, IEEE SIGNAL PROCESSING MAGAZINE, November 1996.