International Journal of Research and Development in Applied Science and Engineering (IJRDASE)
ISSN: 2454-6844

Software Cost Estimation Model Development and Parameter Optimization using Genetic Algorithm

Gaurav Vats
Information Technology Dept
IMS Engineering College, India
gaurav.vats@gmail.com

Rohit Agarwal
Computer Science & Engg. Dept
B.I.E.T., Lucknow (U.P.)
rohitagarwal202@gmail.com

Abstract: Software cost estimation is one of important activity of software development. Software cost estimation plays an important role in software engineering practice, often determining the success or failure of contract negotiation and project execution. The purpose of this paper is to discover answers to questions through the application of scientific procedures. The main aim of research is to find out the truth which is hidden and which has not been discovered as yet. Though each research study has its own specific purpose, we may think of research objectives as development of a cost estimation framework for software project. My aim is to develop effective approaches in the cost estimation framework for object oriented design phase. The proposed work is to develop cost estimation framework in design phase using object oriented perspective as well as to develop a predictive model using a combination of Regression and Genetic Algorithm optimization techniques.

Keywords: Cost Estimation, GA, OOPs, Regression Analysis.

1. Introduction:

It has been surveyed that nearly one-third projects overrun their budget and late delivered and two-thirds of all major projects substantially overrun their original estimates. The accurate prediction of software development costs is a critical issue to make the good management decisions and accurately determining how much effort and time a project required for project managers as well as system analysts and developers. Without reasonably accurate cost estimation capability, project managers cannot determine how much time and manpower cost the project should take and that means the software portion of the project is out of control from its beginning; system analysts cannot make realistic hardware-software trade-off analyses during the system design phase; software project personnel cannot tell managers and customers that their proposed budget and schedule are unrealistic. This may lead to optimistic over promising on software development and the inevitable overruns and performance compromises as a consequence. But, actually huge overruns resulting from inaccurate estimates are believed to occur frequently.

The overall process of developing a cost estimate for software is not different from the process for estimating any other element of cost. There are, however, aspects of the process that are peculiar to software estimating. Some of the unique aspects of software estimating are driven by the nature of software as a product. Other problems are created by the nature of the estimating methodologies. Software cost estimation is a continuing activity which starts at the proposal stage and continues through the lift time of a project. Continual cost estimation is to ensure that the spending is in line with the budget.

Cost estimation is one of the most challenging tasks in project management. It is to accurately estimate needed resources and required schedules for software development projects. The software estimation process includes estimating the size of the software product to be produced, estimating the effort required, developing preliminary project schedules, and finally, estimating overall cost of the project.

It is very difficult to estimate the cost of software development. Many of the problems that plague the development effort itself are responsible for the difficulty encountered in estimating that effort. One of the first steps in any estimate is to understand and define the system to be estimated. Software, however, is intangible, invisible, and intractable. It is inherently more difficult to understand and estimate a product or process that cannot be seen and touched. Software grows and changes as it is written. When hardware design has been inadequate, or when hardware fails to perform as expected, the "solution" is often attempted through changes to the software. This change may occur late in the development process, and sometimes results in unanticipated software growth.

After 20 years research, there are many software cost estimation methods available including algorithmic methods, estimating by analogy, expert judgment method, price to win method, top-down method, and bottom-up method. No one method is necessarily better or worse than the other, in fact, their strengths and weaknesses are often complimentary to each other. To understand their strengths and weaknesses is very important when you want to estimate your projects.

Software cost estimation is one of important activity of software development. Software cost estimation plays an important role in software engineering practice, often determining the success or failure of contract negotiation and project execution. Cost estimation's deliverables such as effort, schedule, and staff requirements are valuable information for project formation and execution. The more software becomes important in almost every human activity,

the more it becomes complex and difficult to implement. Even if modern software technologies render easier the development of certain types of software products, increased user demands and new application domains produce additional problems. It is not surprising that software project management activities are becoming increasingly important.

2. Related Work:

SCE is a process used in software development industry to estimate or predict the resource, efforts, cost of any development process, furthermore to the management controlling and monitoring process over the software development process, before inventing the techniques of estimation in the beginning of 1970 the software estimation process was mainly depend on a rules of thumb and some simple algorithms to estimate the size, efforts and cost after that the idea of function points was introduced by Albrecht [1] which till now gain a high reliability in the estimation process, since that time many studies carried on function points to calibrate its weights, modify its parameters and its adjustment factors to reflect the current improvement in technology and software development industry.

The history of SCE begin early in 1960 when Frank Freiman develop the concept of parametric estimation models that's leads to the development of PRICE model for hardware, in 1970 the researchers in this field analyze many projects using statistical techniques attempting to identify the factors which affect the cost of software development using correlation and regression techniques, by the end of this decade the Constructive Cost Model

COCOMO is being formulated by Barry W. Boehm and C. Abts, PRICE software cost estimation parametric model was developed by the end of this decade also by Frank Freiman and Rbert park, Allan Albrecht and John Gaffney of IBM developed function point analysis FPA to estimate the size and effort of information systems, Halasted define another measure for software size depends on number of operands and operators which did not last in use too long.

In 1980 many evolutions occurred to the previous techniques and models, Barry Boehm and W. Royce defined a revised model to adopt ADA programming language named Ada COCOMO, Capres Jones [4] improve the functionality of FPA via including the effect of complex algorithm in the computation of function point, Charles Symons [5] propose another Version of FPA to handle the problem of subjectivity in FPA called Mark II Function Points.

In 1990 the leader of software cost estimation researcher Barry Boehm et.al reformulates his model into COCOMO II which consists of three main sub models Called Application Composition, Early Design and Post architecture models where he used many software sizing models like Object Points, Function Points and source line of code.

In the last decade until this year 2012 researches in SCE become numerous side to side with the rapid exponential improvement in software and information technology industry, mainly the researches related to handle the new phenomena in software engineering such as Object Oriented environment, agile projects, reusable component, SPI, Component Based Programming, real time systems etc via new sizing techniques, inventing and improving and calibrating the previous models and techniques to be applicable to the new current environment.

Currently, SCE has gain more attention to produce a reliable trusted models which able to predict the cost and efforts to develop any software system hence the estimation process yields important results for the managers, developers and users that's allocate and determine the cost, resources and the schedule for any proposed system.

3. Methodology:

The genetic algorithm is a method for solving both constrained and unconstrained optimization problems that is based on natural selection, the process that drives biological evolution. The genetic algorithm repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an optimal solution. You can apply the genetic algorithm to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non differentiable, stochastic, or highly nonlinear.

The genetic algorithm uses three main types of rules at each step to create the next generation from the current population:

- Selection rules select the individuals, called parents that contribute to the population at the next generation.
- Crossover rules combine two parents to form children for the next generation.
- Mutation rules apply random changes to individual parents to form children.

The following outline summarizes how the genetic algorithm works:

- The algorithm begins by creating a random initial population.
- The algorithm then creates a sequence of new populations. At each step, the algorithm uses the individuals in the current generation to create the next population.

To create the new population, the algorithm performs the following steps:

1. Scores each member of the current population by computing its fitness value.
2. Scales the raw fitness scores to convert them into a more usable range of values.
3. Selects members, called parents, based on their fitness.
4. Some of the individuals in the current population that have lower fitness are chosen as elite. These elite individuals are passed to the next population.

5. Produces children from the parents. Children are produced either by making random changes to a single parent mutation or by combining the vector entries of a pair of parents crossover.
6. Replaces the current population with the children to form the next generation.
7. The algorithm stops when one of the stopping criteria is met.

Preliminary Considerations

1. Determine how a feasible solution should be represented

(a) Choice of Alphabet. This should be the smallest alphabet that permits a natural expression of the problem.

(b) The String Length. A string is a chromosome and each symbol in the string is a gene.

2. Determine the Population Size.

This will remain constant throughout the algorithm. Choosing a population size too small increases the risk of converging prematurely to a local optimum, since the population does not sufficiently cover the problem space. A larger population has a greater chance of finding the global optimum at the expense of more CPU time.

3. Determine the Objective Function to be used in the algorithm.

A Genetic Algorithm

1. Determine an Initial Population.
   (a) Random or
   (b) By some Heuristic
2. *REPEAT*

A. Determine the fitness of each member of the population. (Perform the objective function on each population member) Fitness Scaling can be applied at this point. Fitness Scaling adjusts down the fitness values of the super performers and adjusts up the lower performers, promoting competition among the strings. As the population matures, the really bad strings will drop out. Linear Scaling is an example.

B. Reproduction (Selection)

Determine which strings are "copied" or "selected" for the mating pool and how many times a string will be "selected" for the mating pool. Higher performers will be copied more often than lower performers. Example: the probability of selecting a string with a fitness value of f is f/ft, where fit is the sum of all of the fitness values in the population.

C. Crossover

1. Mate each string randomly using some crossover technique
2. For each mating, randomly select the crossover position(s).

(Note one mating of two strings produces two strings. Thus the population size is preserved).

D. Mutation

Mutation is performed randomly on a gene of a chromosome. Mutation is rare, but extremely important. As an example, perform a mutation on a gene with probability .005. If the population has g total genes (g = string length * population size) the probability of a mutation on any one gene is 0.005g, for example. This step is a no-op most of the time. Mutation insures that every region of the problem space can be reached. When a gene is mutated it is randomly selected and randomly replaced with another symbol from the alphabet. UNTIL Maximum number of generation is reached.

The present work deals with the development of a GA based optimization model for the prediction of the software cost estimates for which the object oriented dataset from forty Java systems derived during two successive semesters of graduate courses on Software Engineering is going to be used. For this a two stage data analysis will be done on MATLAB platform. Initially the dataset will be analyzed using Linear Regression Technique. The linear regression model so obtained will be later on analyzed for the development of the Fitness Function in the GA based model for the optimization of the model parameters so as to arrive at better software cost estimation prediction accuracy. The general framework for the present work is given below. The complete details about the proposed framework and its implementation using genetic algorithm will be dealt in the next section.

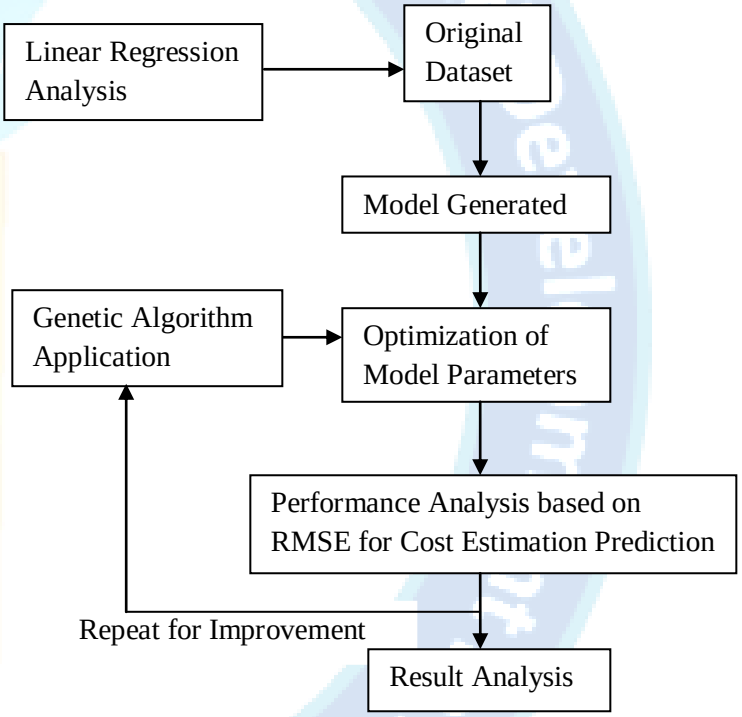

Fig. 1: The general framework for the present work

Genetic Algorithms (GA) are direct, parallel, stochastic method for global search and optimization, which imitates the evolution of the living beings, described by Charles Darwin. GA is part of the group of Evolutionary Algorithms (EA). The evolutionary algorithms use the three main principles of the natural evolution: reproduction, natural selection and diversity of the species, maintained by the differences of each generation with the previous. Genetic Algorithms work with a set of individuals, representing possible solutions of the task. The selection principle is applied by using a criterion, giving an evaluation for the individual with respect to the desired solution. The best-suited individuals create the next generation. The large variety of problems in the engineering

sphere, as well as in other fields, requires the usage of algorithms from different type, with different characteristics and settings.

4. Result and Discussion:

After optimization of the fitness function using MATLAB command, the optimized function value and the optimal parameter values are obtained. Using different parameter options for GA algorithm functions solutions obtained are as follows:

$S_{estimated} = S_{observed} = \alpha + b*x(1) + c*x(2) + d*x(3);$

Where

α = 143.654790554716, a constant term

b = 0.802930099303848

c = 0.212133114686023,

d = 0.0811574301282941

The lower and upper bound values used for the parameters are as follows:

lb = [0.75 0.15 0.07];
ub = [0.85 0.25 0.10] ;

Further the figure 2 below shows optimized parameter values of all the datasets using GA optimization.

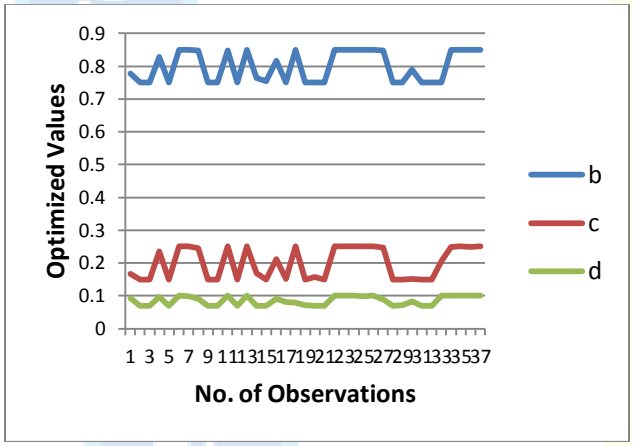

Fig. 2: Plot of optimized parameters "b", "c" and "d" for datasets

On further analysis of the above equations (2) and (3), it was seen that equation (3) for the model was found to be the best developed model, resulting in low RMSE value of 61.66 as compared to that of regression model for the same datasets having RMSE value of 96.31. Also, MAE values for RR and GA based models are 0.17188 and 0.098818 respectively, which again demonstrates the superiority of GA over other techniques (Table 1 and Fig. 4 & 5 below). Further, it clearly demonstrates that genetic algorithm optimization techniques have been successful in developing a better prediction model by lowering the RMSE value. It is shown in figure 3 below. The MATLAB plot of the various functions used in the optimization of the model has been shown in figure 4 below. Further, the various fitness function values of parameters which are to be optimized have been plotted, as shown in figure 6 below.

Table 1: RMSE and MAE values using RR and GA

	Using Regression	Genetic Algorithm
RMSE	**96.31**	**61.66**
MAE	**0.17188**	**0.098818**

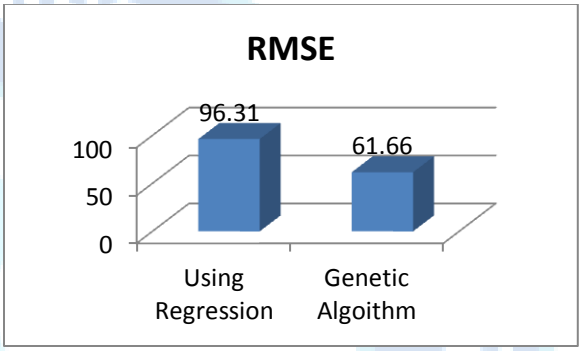

Fig. 3: Plot of RMSE Value using Regression and GA techniques

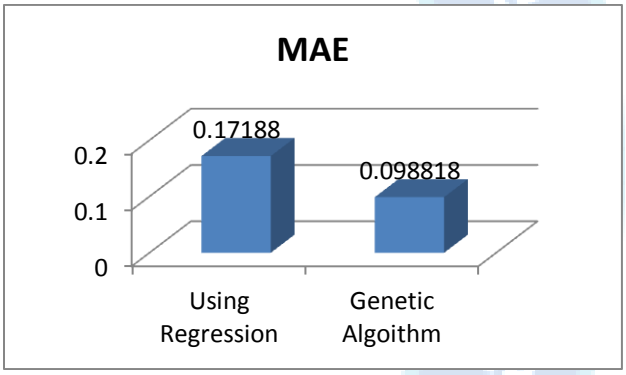

Fig. 4: Plot of MAE Value using Regression and GA techniques

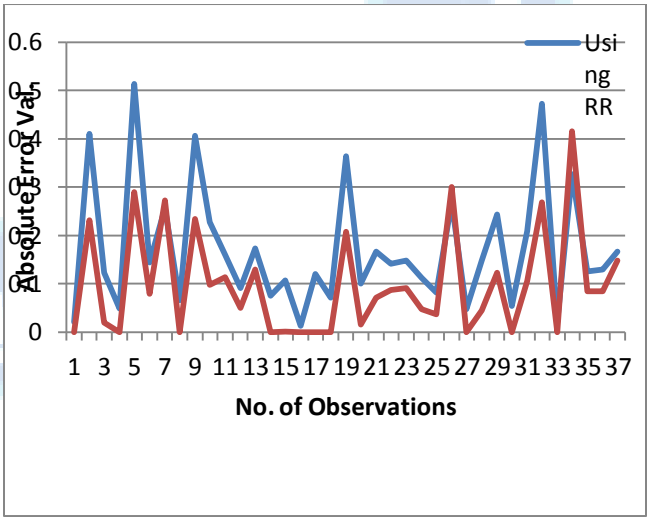

Fig. 5: Plot of Absolute Error Values using Regression and GA techniques for all datasets

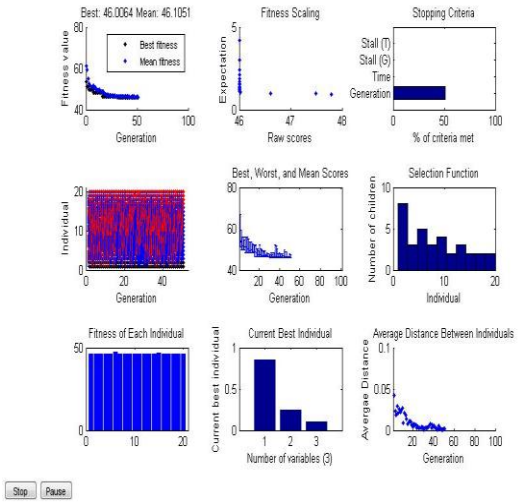

Fig. 6: Output Plot of various GA plot functions in MATLAB

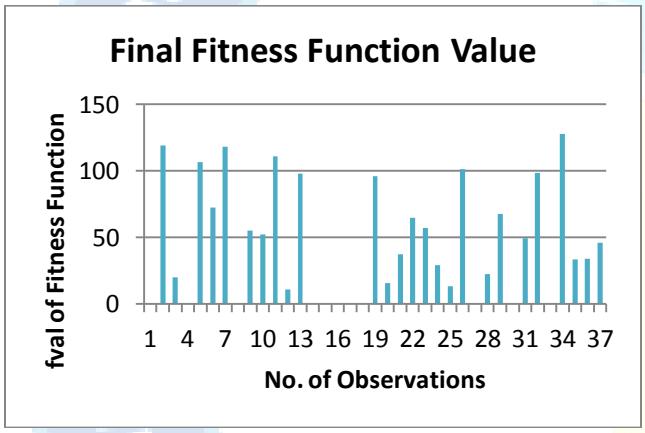

Fig. 7: Plot of the final value of the Fitness Functions

Further, from the perusal of comparative plots of observed and predicted effort values as given in Fig. 7 & 8, both for RR and GA based, it is seen that for both RR and GA based model the predicted values closely follows the observed trend, but still GA based trend is almost superimposed over one another.

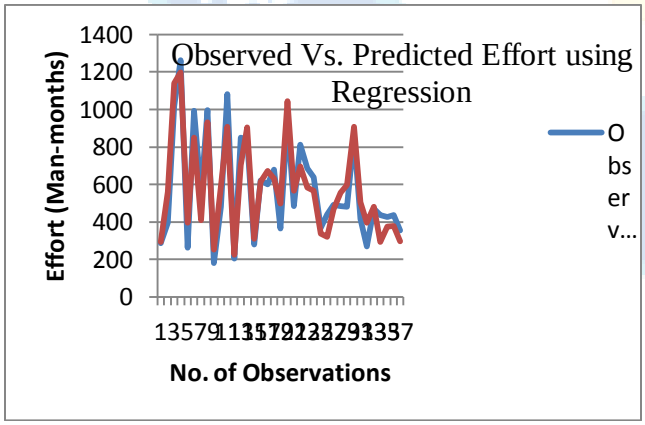

Fig. 8: Plot of Observed Vs. Predicted Effort using RR

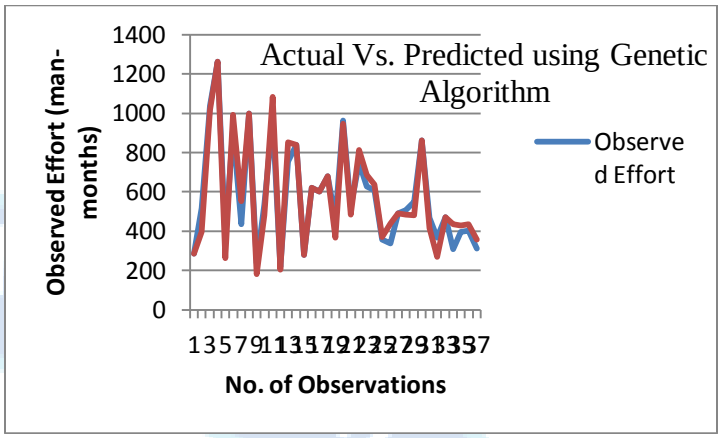

Fig. 9: Plot of Observed Vs. Predicted Effort using GA

5. Conclusion:

In this study, applicability and capability of Genetic Algorithm techniques for application in software cost estimation as a predictive tool has been investigated. It is seen that GA models are very robust, characterized by fast computation, capable of handling the noisy and approximate data that are typical of data used here for the present study. From the analysis of the results given earlier it is seen that GA has been able to perform well for the prediction of effort estimation. Due to the presence of non-linearity in the data, it is an efficient quantitative tool. The studies have been carried out using MATLAB simulation environment.

References:

[1]. N. Veeranjaneyulu, S.Suresh, Sk.Salamuddin3 and Hye-jin Kim, (2014), " Software Cost Estimation on e-Learning Technique using A Classical Fuzzy Approach", International Journal of Software Engineering and Its Applications Vol. 8, No. 11 (2014), pp. 217-222

[2]. Stein Grimstad, et. al, (2006), "A Framework for the Analysis of Software Cost Estimation Accuracy", ISESE'06, ACM 1-59593-218-6/06/0009

[3]. Lalit V. Patil, et. Al., (2014), " Develop Efficient Technique of Cost Estimation Model for Software Applications", International Journal of Computer Applications (0975 – 8887) Volume 87 – No.16, February 2014

[4]. Jyoti G. Borade, (2013), " Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 8, pp 730-739.

[5]. K. Subba Rao, et. al. (2013), "Software Cost Estimation in Multilayer Feed forward Network using Random Holdback Method", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 10, pp 1309-1328.

[6]. Vahid Khatibi, Dayang N. A. Jawawi, (2011), " Software Cost Estimation Methods: A Review", Journal of Emerging Trends in Computing and Information Sciences, Volume 2 No. 1, pp 21-29.

[7]. Swati Waghmode, Dr.Kishor Kolhe, (2014), "A Novel Way of Cost Estimation in Software Project Development Based on Clustering Techniques", International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, Issue 4, PP. 3892-3899.
[8]. Isa Maleki, Laya Ebrahimi, Saman Jodati, Iraj Ramesh,(2014), "Analysis Of Software Cost Estimation Using Fuzzy Logic", International Journal in Foundations of Computer Science & Technology (IJFCST), Vol.4, No.3,, PP. 27-41.
[9]. Hasan Al-Sakran, (2006), "Software Cost Estimation Model Based on Integration of Multi-agent and Case-Based Reasoning", Journal of Computer Science 2 (3): 276-282.
[10]. Soumyabrata Mukherjee, Bishnubrata Bhattacharya, Suvajit Mandal, (2013), "A Survey On Metrics, Models & Tools Of Software Cost Estimation", International Journal of Advanced Research in Computer Engineering &Technology (IJARCET) Volume 2, Issue 9,, PP. 2620-2625.
[11]. Geetika Batra, Kuntal Barua, (2013), "A Review on Cost and Effort Estimation Approach for Software Development", International Journal of Engineering and Innovative Technology (IJEIT) Volume 3, Issue 4,, PP. 290-293.
[12]. Narendra Sharma, Ratnesh Litoriya, (2012), " Incorporating Data Mining Techniques on Software Cost Estimation: Validation and Improvement", International Journal of Emerging Technology and Advanced Engineering, Volume 2, Issue 3,, pp. 301-309.
[13]. Zulkefli Mansor, Saadiah Yahya, Noor Habibah Hj Arshad , (2011), " Review on Traditional and Agile Cost Estimation Success Factor in Software Development Project", International Journal on New Computer Architectures and Their Applications (IJNCAA) 1(3): 942-952.
[14]. Martin Shepperd and Chris Schofield, (1997), "Estimating Software Project Effort Using Analogies", IEEE Transactions On Software Engineering, VOL. 23, NO. 12,, PP. 736-743.