Establishing the Optimal Software Cost Equation Using Cost Affecting Factors and Elitist Gene Expression Programming

Divya Kashyap and A. K. Misra
1 CSED, Motilal Nehru National Institute of Technology, Allahabad

Abstract- Accurate estimation of software’s cost is one of the most intricate activity during the software development lifecycle. There are many factors that directly or indirectly affect the development cost of any software and these factors range from the size of the software to the experience of project manager. Calculating and interpreting these factors would be much valuable in software cost estimation if an ideal relationship between cost and these factors is found out. In this paper we have tried to identify some of the critical factors that affect the cost and we have also attempted to create a software cost equation that institutes the relationship between these factors and the software cost, using Gene Expression Programming. Our aim is to create an optimized model for cost estimation that includes least number of most acute factors for more accurate predictions. To settle this relationship, we have used Gene Expression Programming, a variant of Genetic Programming, because of its proven potential in evolving mathematical equations.

Keywords: Software Development Cost, Cost Affecting Factors, Genetic Expression Programming, Elitism, Function Points and Cost Estimation.

I. INTRODUCTION

Software cost estimation is one of the widely researched areas related to software engineering and software project management. It is the most onerous task which is perilous for its customers, developers and users because it severely affects the total software project management process, including contract negotiations, scheduling, resource allocation and project planning [1, 2]. Overestimation of cost may result in the failure to win the project contract or over employment of the resources while underestimating it may result in undeveloped functions, low quality and late deliveries. So it is a considerably important activity to envisage the software cost and its component units as early as possible in the development process so as to shape the future development plans [3]. One can also claim that the timely production of quality software also depends on this estimation.

Various cost estimation models have been described in the past and Bhoem in [4] classifies these models into six categories: Parametric models, Expertise based, Learning oriented, Dynamic based, Regression based and Composite Bayesian. Out of these prototypes parametric or factor based models are the most prominent and precise. These are also easy to understand and use [2, 3, 4]. Some of the factor based models are Putnam’s SLIM [5], Function Point [6], Checkpoint [7], Price-S [8], Estimacs [9], Seer-Sem [10], SELECT-estimator [11] and COCOMO II [12, 13]. The traditional parametric model is envisioned in figure 1.

Laird & Brennan [15] have shown that notwithstanding the presence of several cost estimation models, only 28% of the total completed projects are delivered on time and each of these delivered projects can averagely overrun its budget by 45% or even more. Cost estimation is typical and critical because there are many factors that affect the software cost. Some of these factors are: Function size, duration, implementation language, development technique, developer’s skills, project schedules, project manager’s discretions and many more [6, 16, 17, 18]. Calculations that connect these parameters are hard because these parameters are ambiguous, uncertain and there is dearth of information about these factors during the initial phases of software development when we have to determine the cost. Also the rapid changes in user requirements make this process even more typical.

To tackle this imprecision and uncertainty we have used Gene Expression Programming (GEP) [19, 20] to successfully find an optimal relationship between the software cost factors. GEP is a special form of Genetic Programming [21, 22] in which each individual is a calculation unit or a mathematical equation. It is an optimization procedure used to optimize a set of mathematical expressions based on the fitness behavior of the expression, determined by its ability to perform a given computational task. In our paper the software cost equation is modeled as a mathematical expression and its accuracy in estimating the cost would be considered as its fitness. To improve the accuracy, we have combined Genetic Expression Programming with Elitist strategy [23] which makes it computationally fast and more accurate. The flow of rest of the paper goes in the following manner: In section two we have described the various factors that affect the software cost in section one. Then in third section we briefly discussed elitist strategy and gene expression programming. Finally, in section four we have framed the problem and tried to approach it using elitist gene expression programming with results and conclusions.
II. FACTORS AFFECTING SOFTWARE DEVELOPMENT COST

Finding the software cost function is the most arduous task in the field of software engineering, since there are many factors that simultaneously and synchronously affect the software development activities during the whole development life cycle. We have analyzed some hundred projects from ISBSG data repository [33] to find out the most prominent factors, so as to quantify the effect of these parameters on the software cost, using Gene Expression Programming. The following factors are the prerequisite for cost estimation:

A. Function Points (real number)

Function Points (FP), founded by Albrecht in [24], is the measurement of the functionality of any program. FP can be calculated from the detailed software requirements. The five categories of functions to be count are: external inputs, external outputs, external inquiries, external interfaces and internal files. After the functions of all categories ($x_i$) have been found out, with their complexity, we calculate the Unadjusted Function Points (UFP) [25], using the following table 1 and equation 1.

$$UFP = \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \cdot x_{ij}$$ (1)

$w_{ij}$ and $x_{ij}$ are the weighting factor and function count respectively, in the $i^{th}$ row and $j^{th}$ column.

B. Software Quality Factor (real number)

The software quality is the degree to which the software meets the client’s expectations. To calculate the Software Quality Factor ($SQF$), we have used the following quality attributes, described by McCall’s in[26]: Correctness, Usability, Efficiency, Reliability, Integrity, Portability, Reusability, Maintainability, Flexibility and Testability. Although it is very difficult to directly quantify the quality attributes of any system, $SQF$ can be estimated by combining the ratings for the individual quality factor [27] as described in following equation 2.

$$SQF = c_1m_1 + c_2m_2 + \ldots + c_nm_n$$ (2)

c$_n$ are regression coefficients, $m_n$ are the metrics that affect the quality factor calculated on a grading scheme ranging from 0 (low) to 10 (high).

C. Project Duration (natural number)

It is the total number of workdays for which the project has been developed. Cost is directly proportional to project duration.

D. Team Size (natural number)

Total number of developers who functioned conjointly throughout the entire development process, for the successful completion of the project. Cost increases with the size of team.

E. Number of Consultants (natural number)

It is the number of advisers or experts associated with the project. Project cost increases with the number of counselors.

F. Risk Classification

The overall risks and contingencies associated with the development process as well as the project itself.

G. Explicit Constraints

The explicit constraints levied by the client or developers, such as, schedule constraints, execution time limitations, resource constraints or storage constraints.

H. Team’s Experience and Ability

The assessment of skills, expertise and capabilities of the programmers in handling the project of particular type.
the expertise and ability, the lesser will be the cost. This factor also includes the project team’s cohesion.

I. Development Platform
This is the primary platform for the development of the software. Usually, each software project is developed for one of the following platforms: multi-platform, mainframe, midrange or personal computer. The cost decreases from multi-platform to personal computers.

J. Use of Modern Programming Practices
The degree to which modern software tools and programming practices are used in the development of software project. This factor also specifies the type of development language use in the process. Usually using the 4GL language and other development tools such as IDEs increases the implementation efficiency. It may increase or decrease the cost. An increase in the cost would be noticed if development tools are very costly or cost may decrease if the use of tools or particular programming language saves time and effort.

The factors from \( H \) to \( L \) are measured at a scale of 1 to 5, ranging from 1 (very low) to 5 (very high). As one common characteristic of all parametric models is that the quality of outputs rest on the quality of inputs i.e. accuracy of cost estimation directly depends on the preciseness of input parameters and the relationship settled between there discrete parameters and the software cost [14], so care must be taken to be more and more precise while calculating these parameters.

III. GENE EXPRESSION PROGRAMMING AND ELITISM

A. Gene Expression Programming
Gene Expression Programming (GEP) is a variant of Genetic Programming [28]. It is a subset of Evolutionary Computation [29, 30] that automatically evolves and creates, computer ably programmable units. These units can be conventional mathematical expressions or complex polynomial structures [31]. In GEP each computational unit or mathematical expression is encoded as a linear chromosome (genotype) which can be translated into expression trees (phenotypes). There is a one-to-one relationship between the symbols used in the linear chromosome and the nodes of the corresponding expression tree.

B. GEP Chromosome Structure
A GEP chromosome is made up of functions and terminals and has two parts, head and tail. Head is made of both terminals and functions, while tail contains only terminals. The tail length \( t \) is calculated through the following formulae:

\[ t = h(2^n - 1) + 1 \] (3)

Where \( h \) is the head length and \( n \) is the number of distinct terminals. The tail, randomly formed, is a noncoding region which has nothing related to the expression but it plays a crucial role in the evolution. To understand, how GEP can be effectively utilized to code for mathematical equations and how genetic operators can be applied over them, let us considering the structural organization of GEP using two expressions, \( exp_1 \) and \( exp_2 \).

\[
exp_1: a + ((b - d) * c) (4)
\]

\[
exp_2: (a + (b^2))^2 - c (5)
\]

In context to \( exp_1 \) and \( exp_2 \):

- Set of functions, \( F = \{+, -, *, ^\} \),
- Set of terminals, \( T = \{a, b, c, d\} \),
- \( h_1 = h_2 = 7 \) and \( t_1 = t_2 = (7(4-1) + 1) = 22 \).

\[
exp_2 = + a * c - b d (6)
\]

\[
exp_2 = - c a ^ b d (7)
\]

These expressions, \( exp_3 \) and \( exp_4 \) are known as Karva notation or the K-expression of the corresponding expressions: \( exp_2 \) and \( exp_3 \). K-expressions are straight forwardly formed by reading the expression tree from left to right and top to bottom. The trees corresponding to \( exp_1 \) and \( exp_2 \) are shown in figure 2.

![Expression trees corresponding to exp1 and exp2](image)

Figure 2: Expression trees corresponding to \( exp_1 \) and \( exp_2 \)

GEP is simple and elegant. The structures of genotypes allow the genetic operators to be applied over it very easily. Secondly they have a perfect and one-to-one mapping from genotype to phenotype which makes it easier and faster to quickly determine the fitness of any chromosome. Thus GEP efficiently completes the prerequisites of any evolutionary system.

C. GEP Genetic Operations
Due to the simplicity of GEP any selection scheme, roulette-wheel selection or tournament selection can be applied on GEP chromosomes. The three kinds of recombination operators for GEP are: single point, two point and gene recombination for any GEP chromosome. Chromosomes can be replicated through mutation also. For more detail on GEP genetic operators we redirect the reader to [19]. Sample recombination and mutation for \( exp_1 \) and \( exp_2 \) are shown in
predicts the cost, combining all input parameters. This considerably reduces the loss of good solutions.

Elitism or elite preservation is a technique to retain the best solutions during subsequent generation of any evolutionary algorithm. There are two general approaches to inculcate elitism in the evolutionary process. One is to combine the parents and offspring of the previous generation before forming the mating pool for the current one. The other way is to maintain an archive to which the promising solutions at each generation can be copied [32]. In our paper also, we have applied the elitist strategy by maintaining an archive. Since the memory resources are limited, for both the variants, we select and hold only a specific number of elites. Elitism makes any evolutionary algorithm fast and efficient because it considerably reduces the loss of good solutions.

D. Elitism

Elitism or elite preservation is a technique to retain the best solutions during subsequent generation of any evolutionary algorithm. There are two general approaches to inculcate elitism in the evolutionary process. One is to combine the parents and offspring of the previous generation before forming the mating pool for the current one. The other way is to maintain an archive to which the promising solutions at each generation can be copied [32]. In our paper also, we have applied the elitist strategy by maintaining an archive. Since the memory resources are limited, for both the variants, we select and hold only a specific number of elites. Elitism makes any evolutionary algorithm fast and efficient because it considerably reduces the loss of good solutions.

IV. PROBLEM DESCRIPTION AND SOLUTION APPROACH

Hitherto, various cost estimation models are present in the software industry that either calculates cost as the function of cost drivers or they simply model the cost estimation as regression problem to construct the expression which best predicts the cost, combining all input parameters. This approach is known as metric based software cost estimation since we consider various metrics/factors to predict the software cost. The three major disadvantages of using this terminology are:

1) Because of inherent dynamism in the cost affecting factors, it is difficult to select the appropriate factors for cost estimation of any particular project. Once we are able to locate them, it is even more difficult to find their values in the early phases of development.

2) We have to analyze a lot of test cases to achieve a certain level of accuracy in the constructed expression or otherwise our predictions would be far from the reality.

3) The rapid changes in the Information Technology and software development methodology and processes have made it difficult to stabilize a particular estimation model.

Although, we are also succeeding in the same direction (metric based) for cost estimation, we are unlike from the previously proposed models, in our approach. We have tried to assuage these three problems.

First we have found out the least set of most affecting factors. From the available pool of factors we have selected only those factors which are not only the most dominating ones but these factors are easy to calculate also. This set of factors is briefly discussed in section 2 of this paper. Thus we have condensed the problem one.

Secondly, instead of using the conventional regression tools or simple genetic programming, we have used elitist Gene Expression Programing (GEP), described in section 3 of this paper. GEP has a proven potential for setting and optimizing the complex mathematical equations. It is simple and has directly mapped genotypes and phenotypes. Its operators are fast and effective. Also a very little knowledge is required by this heuristic method. Thus we overcome the second problem.

Thirdly, in our factor set we have considered quality factors, explicit constraints, development platform and use of modern programming practices. All these factors will clearly indicate the characteristics of the development methodology. So the third problem is also tapered.

In the proposed methodology, we have considered ten parameters to set up a cost estimating equation. One of the most prominent factor is function point, it itself requires some pre-calculations on the basis of user requirements, so as to be used in the estimation equation directly. The others factors can also be easily calculated from the users requirements and project manager’s strategies. To connect these parameters we have used eight simple mathematical operators, namely, addition, subtraction, multiplication, division, exponent, log, square and cube. Now we have to establish the relationship
between the factors using these mathematical operations, so as to form the cost equation and we have used elitist GEP for this purpose. To start the elitist GEP iterations we randomly form a pool of five hundred initial equations. These initial and the consequent equations get processed through thousand iterations. This processing includes binary tournament selection, single point head-tail crossover and random mutation. This procedure eventually results in final selection, single point head-tail crossover and random algorithm 1 of this paper.

$$N$$ is the total number of fitness cases. actual\_effort\_i and estimated\_effort\_i represents the value of actual effort and model calculated effort respectively for the \( i^{th} \) project. Both actual\_effort and estimated\_effort\_are measured in the person-months. The cost in INR or dollars is directly proportional to this effort unit, i.e. the more person-months the more would be the cost. This whole optimization procedure, based on elitist Gene Expression Programming is briefly described in algorithm 1 of this paper.

$$\text{MMRE} = \frac{1}{N} \sum_{i=1}^{N} \left| \frac{\text{actual\_effort\_i} - \text{estimated\_effort\_i}}{\text{actual\_effort\_i}} \right|$$  \(8\)

Figure 3: MMRE for different models

Algorithm 1: Elitist Genetic Expression Programming (GEP)

**Inputs:**

**Parameters:**
- Terminals: All factors discussed in section 2.
- Functions: \{+, -, *, /, ^, log(x), x^2, x^3\}
- Fitness function: Mean Square Error (MSE).
- Population: 500.
- Size of elite archive (s): 50.
- Total generations (m): 1000.
- Selection operator: Binary tournament selection.
- Crossover operator: Single point.
- Mutation operator: Random single point.
- Mutation probability: 10%

**Output:**
Set of 50 best cost equations.

**Procedure:**

**Step 1.** Generate an initial population \( P_0 \) and create the empty archive \( A_0 \). Set \( t=0 \).

**Step 2.** Calculate the fitness of the individuals in \( P_t \) and \( A_n \) using genotype (k-expression) to phenotype (expression tree) conversion and equation 8.

**Step 3.** Copy the best \( s \) solutions in \( P_t \) and \( A_n \) to \( A_{n+1} \).

**Step 4.** If \( t > m \) return \( A_{n+1} \).

**Step 5.** Apply selection operator to \( P_t \) to form the mating pool.

**Step 6.** Apply crossover and mutation operator on the mating pool to form \( P_{t+1} \). Set \( t = t+1 \) and go to Step 2.
VII. CONCLUSIONS

Cost estimation is a hard-hitting topic related to software industry. Although, many cost estimation models have been proposed till date, there is a continuous requirement of new techniques to predict the cost because of rapid change in the domains of information technology and software development methodology itself. We always necessitate a model that absorbs all the factors associated with the project and fallouts with a precise development cost estimate. In the light of this issue, our research in this paper gives an overview of the state of art of software cost estimation using principles of evolutionary computation.

We have tried to find out the most dominating parameters that potentially affect the cost of any software project and then we have proposed a methodology to set up a cost equation containing these parameters. Gene expression programming, which is a special case genetic programming, is used to establish the software cost equations between the software cost and the factors that affect it. From our experiment we have found that the equations found from Elitist Gene Expression Programming are the most accurate ones when compared to other traditional techniques.

VIII. REFERENCES


