AN EFFICIENT APPROACH OF SOFTWARE COST ESTIMATION USING SOFT COMPUTING TECHNIQUES

Puja Sinha
Research Scholar, M.Tech. Research, Computer Science Engineering, Jharkhand Rai University, Ranchi

Dr. Prakash Kumar
Assistant Professor, Jharkhand Rai University, Ranchi

ABSTRACT

Software development cost estimation is an important activity in the early software design phases. The input datasets are primarily taken from the promise repository. Data mining and soft computing techniques are used to assess the software development cost estimation. Each feature in the input dataset is divided, the linguistic terms along with the membership are identified using trapezoidal membership functions, and associative classification is adopted for generating rules. The large number of rules is filtered with respect to the support and confidence. Genetic algorithm is employed as an optimization tool for selecting the best rules. The example presented demonstrates the improvement in accuracy. The crisp costs are presented after defuzzification of the output. This paper aims to utilize an adaptive fuzzy logic model to improve the accuracy of software time and cost estimation. Using advantages of fuzzy set and fuzzy logic can produce accurate software attributes which result in precise software estimates. 63 Historic projects of NASA dataset having COCOMO format is used in the evaluation of the proposed Fuzzy Logic COCOMO II.

Key words: Software Cost; Cost Estimation; Fuzzy Logic; Soft Computing.

http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=9&IType=4

1. INTRODUCTION

The term "software development" may be used to refer to the activity of coding, which is the process of programming & maintaining the source code, but in other words it includes all that is involved between the conception of the desired software.

Software development cost estimation is the process of predicting the most realistic use of cost required to develop or maintain software based on incomplete, uncertain and/or noisy
input. The expert estimation of cost is the dominant strategy when estimating software development cost. Construction of formal software cost estimation models has been the main focus in this area of research. The early models were typically based on regression analysis or mathematically derived from theories from other domains.

2. ESTIMATION MODELS

During the last two decades, many researchers have developed the models like case-based reasoning [1], classification and regression trees [2], simulation [3], neural networks [4], Bayesian statistics [5], lexical analysis of requirement specifications [6], genetic algorithm [7,8], linear programming [6], economic production models [9], soft computing technique [10], fuzzy logic modeling [11], statistical method of bootstrapping [5], and combinations of two or more of these models have been developed to evaluate the models.

It was stated that the accuracy from the group estimate is more than that of the individual cost estimation, and the analysis of code further increases the quality. Since the planning poker is mainly used in agile project, the efficient use of expert cost estimation is the major disadvantage in this approach.

To improve the accuracy of analogy software cost estimation model, fuzzy logic based feature subset selection algorithms were developed [17]. The similarity degree between two given fuzzy clusters, and consequently the similarity between all clusters for all selected features were assessed. Feature weighting heuristics for analogy based cost estimation models were proposed by ranking the features with respect to weights [18]. Heuristic was applied to modify standard process of PCA, and the ordering of the features corresponds to the eigenvalues and eigenvectors. Though the accuracy is improved for specific projects, it does not include all type of projects for highlighting the accuracy. Association COCOMO computes software development cost as function of program size and a set of "cost drivers" that include subjective assessment of product, hardware, personnel, project attributes, etc. This extension considers a set of four "cost drivers" that are Product attributes, Hardware attributes, Personnel attributes and Project attributes.

The detailed model uses different costs multipliers for each cost drivers attribute these Phase Sensitive cost multipliers are each to determine the amount of cost required to complete each phase. In detailed COCOMO, the cost is calculated as function of program size and a set of cost drivers given according to each phase of software life cycle. The five phases of detailed COCOMO are:-

- Plan and requirement.
- System design
- Detailed design
- Module code and test

3. PUTNAM MODEL

The Putnam model [5] is an empirical software effort estimation model. As a group, empirical models work by rule mining and associative classification were also commonly used for cost prediction [19, 20]. Several parameters were used for filtering the rules since the number of rules generated was of huge volume.

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_0, 1 \]

(1) The parameters used were support, confidence, and information gain. Rule ranking algorithm, rule priority and rule sorting methods were used for predicting the results [2123]. Hill
Climbing method was used to find the threshold value for support and confidence which is used for filtering the association rules [24].

Fuzzy logic based methods like fuzzy membership function [25], fuzzy clustering [26, 27], fuzzy partitioning [28], fuzzy with maximum entropy [29] and fuzzy feature selection [30] are also available in the literature. In a classification system based on Michigan and Pittsburgh approaches, a fitness value is assigned to each.

If $x \leq 1$, $x$ is definitely a member of the set $x$, whereas if $x > 0$, $x$ is definitely not a member of the set $x$. For all $x$ with $0 < x < 1$, the membership is not certain.

These grades or values of membership are either obtained subjectively or as the values of a function for those particular events or patterns. This membership function enables us to perform quantitative calculations in fuzzy decision making. Its choice and justification are important for the success of applications. The most frequently applied membership function shapes are the triangular, trapezoidal and Gaussian. Fig. 1 shows a triangular membership function described by fuzzy if-then rule [31]. In the extraction of association rules using genetic algorithms, comprehensibility, J-measure and accuracy were taken as multi objective function [32]. In Michigan approach, each individual in GA encodes a single prediction rule. For transactional items, the combination of fuzzy, clustering and GA was adopted [33, 34].

A system based on FL has a direct relationship with fuzzy is the concepts (such as fuzzy sets, linguistic variables, etc.) and fuzzy logic. The popular fuzzy logic systems can be divided into three types i.e., Pure fuzzy logic systems, Takagi and Sugeno’s fuzzy use system, and fuzzy logic system with fuzzifier and defuzzifier. Since most of the engineering applications produce crisp data (cross-industry standard process) as input and expects crisp data (i.e., cross-industry standard process for data mining) as output, the last type is the most widely used type of fuzzy logic systems. Fuzzy logic system with fuzzifier and defuzzifier, first, suggested by Mamdani and it has been successfully applied to a variety of industrial processes and consumer products.

\[
(x) \quad \begin{array}{c}
0 \\
1.0 \\
(x-a) / (b-a) \\
(c-x) / (c-b) \\
x-a \\
a x b b x c x c
\end{array}
\]

Fig. 1. Triangular membership function
4. FUZZY LOGIC

Fuzzy logic is a mathematical tool for dealing with uncertainty and imprecision. It is a theory of unsharp boundaries and is used to solve problems that are too complex to be understood qualitatively. It consists of four main components:

1. **Components**:
   - **Fuzzifier** converts the crisp input into a fuzzy set. Membership Functions are used to graphically describe a situation.
   - **Fuzzy Rule Base** uses if-then rules.
   - **Fuzzy Inference Engine** has a collection of if-then rules stored in fuzzy rule base. It performs two operations i.e. aggregation and composition.
   - **Defuzzification** is the process to translate the fuzzy output into crisp output.

![Trapezoidal membership function](image)

**Figure 2** Trapezoidal membership function
A. Membership Function

A fuzzy set is characterized by grades of membership.

For\( x \) a fuzzy set\( x \), the membership function \( \mu_x \) describes the grade of membership to the fuzzy set for each element \( x \) in the domain \( X \):

Similarly Fig. 3 shows a Gaussian membership function

Described by

\[
x \exp \left( \frac{1}{(x-m)^2} \right)
\]

![Figure 3 Gaussian membership function](image)

5. GENETIC ALGORITHM

Genetic Algorithm (GA) is a heuristic search that imitates the process of natural evolution. This heuristic is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the huge class of evolutionary algorithms, which generate solutions to optimization problems using techniques inspired by natural evolution, like inheritance, mutation, selection, and crossover. The most important aspects of using GA are definition of objective function, implementation of genetic representation, and implementation of genetic operators. The implementation of GA is as follows.

Initialization: Initially many individual solutions are randomly

B. Fuzzy Rules

Representation of knowledge such as rules are the most popular forms. If \( X \) is A then \( Y \) is B (where A and B are linguistic values defined by fuzzy sets on cosmos of discourse \( X \) and \( Y \)). A rule is also called a fuzzy implication generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Conventionally, the population is caused randomly, allowing the entire range of possible solutions (the search space).

Selection: There are many selection procedures in GA like Roulette wheel selection, Boltzman selection, Tournament selection, Rank selection, and Steady-state selection. “X is A” is called the predecessor or premise

Fitness function: Fitness function \( F(X) \) is first derived from “Y is B” is called the outcome or conclusion
C. Fuzzy Inference

Using fuzzy logic principles, the fuzzy IF-THEN rules in the objective function and used in successive genetic operations. The transformations used are as follows.

\[ F(X)f(X) \] for maximization problem; the fuzzy rule base are combined to map a fuzzy set \( A' \) in \( U \) to \( F(X)1 \)

\( f(X) \) for minimization problem, if \( f(X) \) a fuzzy set \( B' \) in \( V \). It is the brain of fuzzy system. Composition based inference and individual-rule based inference are the two methods to infer the rules.

\[
F(X) = \begin{cases} 
1 & \text{if } f(X) = 0 \\
\frac{1}{f(X)} & \text{otherwise}
\end{cases}
\]

D. Defuzzification

The fuzzy output of the inference engine can be converted into crisp (i.e., cross industry standard process) using membership functions analogous to the ones used by the fuzzifier. The commonly used defuzzifying methods are Smallest of maximum (SOM), Mean of maximum (MOM), Bisector of area (BOA), Centroid of area (COA), and Largest of maximum (LOM), which are respectively shown in Fig. 4 as

\[ z^a, z^b, z^c, z^d \text{ and } z^e. \]

![Figure 4 Defuzzification](image)

Crossover: Crossover also called recombination operator is a function wherein, two individuals are crossed to form an offspring. Types of crossover operations used in genetic algorithms are Single-site Cross over, Two-point Cross over, Multi-point Cross over, Uniform Cross over, and Matrix Cross over.

Mutation: After cross over the strings are subdued to mutation. Mutation of a bit involves capsizing it, changing 0 to 1 and vice-versa with a small mutation probability \( P_m \).

6. PROPOSED SYSTEM

The interests of the software developers are to know the cost estimation of software tasks. It could be done by comparing similar tasks that have already been developed. Software schedule and cost estimation assert the planning and tracking of software projects. Effectively controlling the expensive investment of software development is highly important. The reliable and accurate cost estimation in software engineering is an ongoing challenge owing to financial and strategic planning.

However the choice of prediction technique is uncertain to various problems. Hence it requires the support of a well-defined evaluation scheme to rank each prediction achieved using each model is different for different evaluation criterion. To improve the accuracy and to avoid the dependency of cost and cost estimation on line of code (LOC), we propose a hybrid...
computational intelligence approach that combines fuzzy, association rule mining and evolutionary computation concepts.

The main objective of the proposed work is to design a model for estimating the cost for a software development project. The cost estimated should be more accurate, which implies the deviation is minimum. The proposed system combines the machine learning techniques like fuzzy set theory, association rule mining and genetic algorithms. Since machine learning technique is used for cost prediction, the dataset available in the PROMISE repository has been used for training and testing. First, the training dataset is used to design the cost estimation model. The model thus developed can be used for predicting the cost for software projects that needs to be developed. Desharnais dataset is taken as a sample dataset for this proposed work会员 functions are identified. The entire dataset can be used as training set for generating the model. Based on the fuzzy sets and the boundaries defined for the membership functions, the membership values for each attribute are calculated.

C. Rule generation

The relationships among the variables are represented by means of rules. The rules are of the following form. “Antecedent proposition” “consequent proposition”

These rules are generated using the associative classifier. The antecedent proposition is always a fuzzy proposition of the type “$X_1$ is Nominal” “$X_2$ is High” where $X_1$ is a linguistic variable and Nominal is a linguistic constant (term), $X_2$ is the consequent variable and High is a consequent class.

D. Rule Pruning

The number of rules thus generated will be so large, since the dataset holds large number of attributes as well as the membership functions for each attribute is also more. To avoid this, we extract rules which frequently occur in the dataset based on the rule quality metrics like support and confidence.

Where $\mu$ is the membership function of antecedent while $\lambda$. Data Pre-processing

Data pre-processing involves analysis on the dataset and the following steps:
(i) Identifying the fuzzy sets (i.e., Generation of fuzzy partitions for the numerical attributes).
(ii) Defining the type of membership function to be used for each attribute.
(iii) Defining the boundaries for the membership functions.

The conventional way to discover the fuzzy sets needed for a definite data set is to consult a domain expert who will define the sets and also their membership functions. The number of fuzzy partitions for each attribute depends on the nature and properties of the attribute itself. The membership functions used can be triangular, trapezoidal, Gaussian, two-sided Gaussian, etc. Trapezoidal Membership function is used for all attributes in the proposed system. The boundaries for the membership functions are defined based on the nature of the attributes.

B. Fuzzification

The training dataset is fed into the fuzzifier to convert the numeric values to linguistic terms, and the degree to which they belong to each of the appropriate fuzzy sets via is the membership function of consequent, and $N$ is the total number of fuzzified records, $t$ is the total number of matched records in the dataset.

The threshold values for support and confidence values are defined. Once a rule has passed the $\text{MinSupp}$ threshold, the code then checks whether or not the rule passes the $\text{MinConf}$ threshold. If the item confidence is huge as compared to $\text{MinConf}$, then it will be caused as a candidate rule in the classifier. Otherwise, the item will be discarded. So, all items that survive
MinConf are generated as candidate rules in the classifier. Hence, the rules that have support and confidence greater than the threshold values are regarded as interesting. Frequent sets that don’t include any fascinating rules do not have to be considered anymore. At the end, all the discovered rules can be presented to the user with their support as well as confidence values.

E. Rule Selection

The best co-operative rules are selected by means of Genetic Algorithm using accuracy as the parameter. Thus, the output of this module provides best set of rules that helps in prediction. The rules discovered in the previous module are treated as the initial population of solutions. The solutions are binary encoded. The binary encoded solution is then used for evaluating the accuracy.

(5) Since the goal of the project is to maximize the accuracy, accuracy is used as the objective function. The formula for computing the accuracy is as

$$\text{Accuracy} = \frac{TP}{TP + TN}$$

where, TP - True Positive (number of positive cases correctly classified as belonging to the positive class); TN - True Negative (number of negative cases correctly classified as belonging to the negative class); FP - False Positive (number of negative cases misclassified as belonging to the positive class); FN - False Negative (number of positive cases misclassified as belonging to the negative class).

(6) defuzz $Q_{i,j} = \frac{W_i m W_i}{1,j \in j_{\text{rule confidence}}}$

where, $m W_i$ is the centre value of expected interval of target Negative (number of negative cases correctly classified as belonging to the positive class).

7. RESULTS AND DISCUSSION

FN - False Negative (number of positive cases misclassified as belonging to the negative class).

Reproduction or Selection is the first genetic operator applied on population. Chromosomes are selected from the population to be parents to cross over and produce offspring. Roulette wheel selection is used as the selection strategy for selecting individuals. This method selects individual that can be used in the next generation of solutions with a probability proportional to the fitness. The more accurate rules have higher probabilities of being moved to the next generation.

Mutation is carried out in case the following two constraints are not satisfied.

(i) A rule is framed without antecedent but with consequent.

(ii) When an already existing individual is reproduced again by crossover or mutation operation.

The solutions are then decoded. The rules with accuracy value greater than the threshold value defined are noted. The process repeats with the rules framed after crossover and mutation. The rule selection module produces set of rules that are more accurate with regard to the training dataset.
F. Defuzzification

The test set consisting of records for which cost needs to be predicted is fed into the defuzzifier. Using the set of rules obtained from the rule selection module, the output (cost) is identified. The output so obtained is a linguistic term which is defuzzified to get a numeric value which gives the Cost in Person-hours.

This module is used for testing the model developed against the test set. The test set is fed as input to the defuzzifier. It makes use of the set of rules produced by the rule selection module to predict the output. If n number of rules satisfies the test set record, then those n rules are considered for predicting the output. The formulae for defuzzification and cost calculation are as follows:

It is observed that the error values are oscillating during the initial phase of the generations, and thereby it steadies along with the generations as shown in Fig. 6.

![Figure 6](image)

**Figure 6** Error propagation against generations

It is also observed that the error values are oscillating during the initial phase of the accuracy, but they do not exhibit the monotonic nature as shown in Fig. 7.

![Figure 7](image)

**Figure 7** Error propagation against accuracy

8. SUMMARY AND CONCLUSION

Estimation of cost required for developing the software product involves many stages. Importantly the cost estimation plays a vital role in the early software design phases. The time and budget requirement for the model will depend on the accuracy. In this proposed work, the input datasets are primarily taken from the promise repository. Data mining and soft computing techniques are used to assess the software development cost estimation. The trapezoidal membership functions are selected to represent the uncertain linguistic terms and the rules are generated by associative classification. In order to optimize the best rules, genetic algorithm is employed. The defuzzified results are presented, and the variations of errors with respect to the generations and accuracy are tabulated. The variations of the errors are also plotted.
REFERENCES


