An Effective Approach to Software Cost Estimation Based on Soft Computing Techniques

Marappagounder Shanker[1], Jayabalan Jaya [2], and Keppanagounder Thanushkodi[2]
[1] Information and Communication Engineering, Anna University, Chennai
[2]Akshaya College of Engineering and Technology, Coimbatore

Abstract: *Employing estimation models in software engineering help in envisaging some essential traits of future entities like software development effort, software reliability and programmers productivity. Of these models, the one that supports the estimation of software effort has drawn substantial attention currently to carry out researches. Estimation by analogy is one among the interesting techniques used for estimating the software effort. But, the process of estimating by analogy is unable to handle categorical data accurately. A novel technique that relies on reasoning by analogy, fuzzy logic and linguistic quantifiers is being proposed here for estimating effort, provided that the software project is represented either by categorical or numerical data. Use of fuzzy logic-based cost estimation models is more suitable if unclear or inaccurate information are considered. Fuzzy systems attempt to imitate the processes of the brain through a rule base. The proposed method utilizes Fuzzy logic based analogy approach to estimate the cost and the effort. The performance analysis of the proposed scheme is made using Mean Absolute Relative Error (MARE) and Mean Magnitude of Relative Error (MMRE) which is validated with other existing techniques.*

Keywords: *Cost estimation, effort estimation, analogy, fuzzy logic, MARE, cost constructive model.*

Received October 18, 2012; accepted June 30, 2014; published online August 9, 2015

1. Introduction

The goal of software engineering is to develop the techniques and tools needed to develop high-quality applications that are more stable and maintainable. In order to assess and improve the quality of an application during the development process, developers and managers use several metrics [1]. Various business and technical motives such as shorter development cycles, lower development costs, improved product quality and access to source code, more and more software developers and companies are basing their software products on open source components [13]. Structural organization of software has a major influence on locality of changes during software evolution. One of the important types of such changes is those concerned with extending and modifying the implemented functionality [12].

Structural organization of software has a main power on locality of changes during software development. One of the significant kinds of such changes is those distressed with widening and adapting the executed functionality [12]. The capability of software quality models to precisely recognize critical elements permits for the application of spotlighted certification activities ranging from physical inspection to testing, static and dynamic analysis, and automated formal analysis techniques. Consequently, Software quality models assist makes certain the dependability of the delivered products. To forecast fault-proneness of program modules in software engineering, various statistical methods have been suggested [6].

Estimating the work-effort and the schedule required to develop and/or maintain a software system is one of the most critical activities in managing software projects. The task is known as software cost estimation [25]. Software size estimates are important to determine the software project effort. However, according to the last research reported by the Brazilian Ministry of Science and Technology-MCT, in 2001, only 29% of the companies accomplished size estimates and 45.7% accomplished software effort estimate. There is not a specific study that identifies the causes of the effort low estimates index, but the reliability level of the models can be a possible cause [22]. Cost estimation is an essential component of infrastructure projects. Accurate estimation will assist project managers to choose adequate alternatives and to avoid misjudging of technical and economic solutions [24]. The accuracy of cost estimation increases toward the end of the project due to detailed and precise information [21].

Software cost estimation by analogy is one of the most conspicuous machine learning techniques and is basically a form of case based reasoning. Estimation by analogy is based on the assumption that similar software projects have similar costs. However, the technique needs improvement especially while handling the categorical variables [11]. The subjects of estimation in the area of software development are size, effort invested, development time, technology used and quality. Particularly, development effort is the most important issue. So far, several effort models have been developed and most of them include software size as an important parameter. Function point is a measure of software size that uses logical functional terms business owners and users more readily understand. Since, it measures the functional requirements, the measured size stays constant despite the programming language, design technology, or development skills involved [5].

Some major problems encountered during cost estimation process are:

- Factor that affect the cost are non-linear in nature so, it is very difficult to establish a mapping between these factors and output metrics.
- Data present at the starting phases are incomplete and imprecise so measurement of metrics are difficult.
- There are many models available for estimation purpose but the problem is how to determine which model is useful in which situation.
- Difficulty to use algorithmic model and non-algorithmic model together [7].

The modern society is continuously becoming more and more dependent on software systems and information technology. Thus, a proper estimation method of projects affects the cost and quality of a project. Many estimation models have come into existence. This has brought researchers attention to develop good software project estimation models [10].

The novel method introduced in this paper for estimating the software effort employs fuzzy analogy method to overcome the shortcomings of using fuzzy and analogy in a separate manner. A set of candidate measures for software projects similarity have been developed and validated and these measures have their basis on fuzzy sets, fuzzy reasoning and linguistic quantifiers. Though the software projects are represented either by numerical or categorical data, this novel scheme has the potential to estimate the software effort. The usual analogy procedure does not work on categorical data. Hence, estimation by analogy that is based on fuzzification is dealt in this work. The rest of the paper is organized as follows: Section 2 deals with some of the recent research works related to the proposed technique. Section 3 explains the fuzzy analogy technique. Section 4 describes the proposed technique for cost estimation with all necessary mathematical formulations and figures. Section 5 discusses about the experimentation and evaluation results with necessary tables and graphs. Section 6 concludes the paper.

2. Related Works

The proposed technique aims to estimate the software cost in an efficient way. Numerous techniques have been developed so far to estimate the effort and cost associated with the software projects. In this segment, we have presented few of the significant researches for iris recognition.

Many researchers have been focused on software cost estimation based on soft computing techniques. Here, we are presenting some of the methodology proposed by various researchers.

Rao *et al.* [16] have proposed a computationally efficient Functional Link Artificial Neural Network (FLANN) for cost estimation and to reduce the computational complexity so that, the neural net becomes suitable for on-line applications. FLANN do not have any hidden layer; the architecture becomes simple and training does not involve full back propagation. In the course of adversity in neural networks, this dynamic neural network excellently works which would initially use COCOMO (Cost Constructive Model) approach to predict the cost of software and uses FLANN technology with backward propagation. The proposed network processes each and every neuron crystal clear so that, the entire network was completely “white box”. This method gives much more accurate value when compared with others because their method involves proper training of data using back propagation algorithm which was used to train the network, becomes very simple because of absence of any hidden layer.

Reddy and Raju [17] have proposed a software effort estimation model based on artificial neural networks. The model was designed accordingly to improve the performance of the network that suits to the COCOMO model. The proposed method uses multi layer feed forward neural network to accommodate the model and its parameters to estimate software development effort. The network was trained with back propagation learning algorithm by iteratively processing a set of training samples and comparing the network’s prediction with the actual effort. COCOMO dataset was used to train and to test the network and it was observed that proposed neural network model improves the estimation accuracy of the model.

Attarzadeh and Ow [2] have proposed a fuzzy logic realistic model to achieve more accuracy in software effort estimation. The main objective of this research was to investigate the role of fuzzy logic technique in improving the effort estimation accuracy by characterizing inputs parameters using two-side Gaussian function which gave superior transition from one interval to another. The methodology adopted in this study was use of fuzzy logic approach rather than classical intervals in the COCOMO II. Using advantages of fuzzy logic such as fuzzy sets, inputs parameters could be specified by distribution of its possible values and these fuzzy sets were represented by membership functions.

Prasad *et al.* [15] have presented a method concerned with developing software effort estimation models based on artificial neural networks. The models were designed to improve the performance of the network that suits to the COCOMO Model. Artificial neural network models were created using radial basis and generalized regression. A case study based on the COCOMO81 database compares the proposed neural network models with the Intermediate COCOMO.

Software cost estimation is an important phase in software development. It predicts the amount of effort and development time required to build a software system. It is one of the most critical tasks and an accurate estimate provides a strong base to the development procedure. Kaushik *et al*. [8] have discussed the most widely used software cost estimation model, the COCOMO. The model was implemented with the help of artificial neural networks and trained using the perceptron learning algorithm. The COCOMO dataset was used to train and to test the network.

The effort invested in a software project was one of the most challenging task and most analyzed variables in recent years in the process of project management. Software cost estimation predicts the amount of effort and development time required to build a software system. It was one of the most critical tasks and it helps the software industries to effectively manage their software development process. There were a number of cost estimation models. Each of these models has their own pros and cons in estimating the development cost and effort. Kaushik *et al*. [9] have investigated the use of back-propagation neural networks for software cost estimation. The model was designed in such a manner that accommodates the widely used COCOMO model and improves its performance. It deals effectively with imprecise and uncertain input and enhances the reliability of software cost estimates. The model was tested using three publicly available software development datasets.

Software cost estimation was a challenging and onerous task. Estimation by analogy was one of the expedient techniques in software effort estimation field. However, the methodology utilized for the estimation of software effort by analogy was not able to handle the categorical data in an explicit and precise manner. Early software estimation models are based on regression analysis or mathematical derivations. Today's models are based on simulation, neural network, genetic algorithm, soft computing, fuzzy logic modelling etc., Ziauddin *et al*. [25] have utilized a fuzzy logic model to improve the accuracy of software effort estimation. In this approach fuzzy logic was used to fuzzify input parameters of COCOMO II model and the result was defuzzified to get the resultant Effort. Triangular fuzzy numbers are used to represent the linguistic terms in COCOMO II model.

3. An Efficient Software Cost Estimation System Based on Fuzzy Analogy

Prediction of work effort and plan needed for the development and/or maintenance of software system plays a vital role in software project management and this process is termed as software cost estimation. The estimation of time and cost help in coarse validation and observation of the project's advancements at the time of development process. Once, the development process is finished, these estimates assist in the evaluation of project productivity. Software cost estimation can be defined as the method of forecasting the effort needed to develop a software system. Majority of the cost estimation models try to construct an effort estimate, which can then be changed into the project period and expenditure. Even when effort and cost have close association, they are not essentially connected by a simple transformation function. Effort is generally measured in terms of person months of the programmers, analysts and project managers. This effort estimate can be then be converted into a dollar cost figure through the computation of an average salary per unit time of the staff engaged and then, multiplying the resultant by the estimated effort required.

3.1. Analyze Software Functional and Programmatic Requirements

Software estimation involves the investigation and filtering of the software operational requirements in addition to the identification of technical and programmatic constraints and necessities. This allows the work elements of the project-specific Work Breakdown Structure (WBS) to be divergent and necessitates the prediction of software size and effort.

Three steps are available for making analysis and filtering of the requirements:

1. The software functional requirements are to be examined and filtered to the minimum level of detail that is achievable. Proper risk adjustments can be made only if the unclear requirements are accurately detected. These unclear requirements make the estimation of software size to be difficult and reproduce larger uncertainty. If an incremental development strategy is employed, the filtering will be relying on the requirements that are specific for each increment.
2. The physical architecture hierarchies of the software that depend on the functional requirements are analyzed and refined. The architecture is described by the software segments to be created. This is then followed by the decomposition of each segment into the lowest level function that is feasible.
3. Investigates the project and software plans for discovering the programmatic constraints and requirements, which incorporates imposed budgets, schedules, margins and make/buy decisions.

3.2. Work Elements and Procurements

This step aims to define the work elements and procurements of the software project, which will be used up in the software estimate. The work elements and procurements will naturally come under the following categories of a project-specific WBS such as: Software management, software development test bed, software development environment, software systems engineering, software test engineering etc.

These WBS categories contain actions across the software life-cycle from requirements analysis through end of system test. The software operations and support (including maintenance) does not belong to the range of these estimates. Work elements such as SQA and IV and V are not mostly the piece of software manager's budget, but are listed here to make the software managers understand that these services are being offered by the project.

3.3. Estimate Software Size

This process is done to predict the size of software because software size is another criterion that influences the expense of software. The two main software size metrics are the Source Lines of Code (SLOC) and the function points. SLOC is a natural artifact that computes the physical size of the software. However, it remains unavailable until the coding phase and cannot be defined in a similar way across various programming languages. Function Points is a perfect software size metric for price prediction.

3.4. Effort Estimation

- *Fuzzy Logic*: is a methodology that has the basis on fuzzy set theory for providing solution to issues, which are more complicated for quantitative recognition. Fuzzy logic comprises of three steps as follows:

  1. Fuzzification.
  2. Inference Engine.
  3. Defuzzification.

  The fuzzifier converts the input into linguistic terms using membership functions. The membership functions specify the extent to which a given numerical value of a particular variable fits the linguistic term being addressed. The fuzzy inference engine does the mapping between the input membership functions and the output membership functions with the help of fuzzy rules. These fuzzy rules emerge from expert's idea the relationships being modeled. A defuzzifier executes the Defuzzification process, which combines the output into a single label or numerical value as per the requirement.

- *Fuzzy Membership Function*: A mathematical definition for a fuzzy set can be produced by assigning to each possible individual in the universe of discourse, a value describing its grade of membership in the fuzzy set to a bigger or smaller extent as denoted by a larger or smaller membership grade. The universe of discourse refers to the input space. A membership function is a curve, which describes the way each point in the input space is mapped to a membership value or degree of membership between 0 and 1. A membership function is used to characterize the fuzziness in a fuzzy set. A membership function categorizes the element in the set into discrete or continuous. Various shapes are utilized for graphical representations. Hence, the selection of shape of the membership function is vital. Several types of member functions are available. But here, Triangular Membership Function (TMF) is used.

- *TMF*: It is a three point function given by a lower limit *p*, an upper limit *q* and the modal value such that *p*<*m*<*q*. The value *q-m* is called margin when it is equal to the value *m-p*. It can be symmetrical and asymmetrical.

$$f(x) = \begin{cases} 0 & if\ x \le p\, or\, x \ge q \\ (x-p)/(m-p) & if\ x \in (p,m) \\ (q-x)/(q-m) & if\ x \in (m,q) \end{cases} \tag{1}$$

- *Fuzziness*: Fuzziness of a TMF is defined by Equation 2.

$$fuziness\ of\ TMF = \frac{\lambda - \mu}{2m}, \quad 0 < TMF < 1 \tag{2}$$

Where *m* indicates the model value, μ and λ represent the right and left boundaries respectively. Higher value of fuzziness reveals that the TMF is fuzzier.

- *Effort Estimation by Fuzzy*: In fuzzification, the triangular fuzzy number is utilized and is defined by Equation 3.

$$T(S) = \begin{cases} 0 & if\ (S \le a) \\ (S-\mu)/(m-\mu) & if\ \mu \le S \le m \\ (\lambda - S)/(\lambda - m) & if\ m \le S \le \lambda \\ 0 & if\ S \ge \lambda \end{cases} \tag{3}$$

Where *S* is the size as input, *E* the effort as output, μ, *m* and λ are the parameters of membership function T(S), m is the model value, μ and λ are the right and left boundaries respectively.

Let (*m*, 0) split the base of the triangle in ratio *k*: 1 internally, where *k* is a real positive number. Thus, the value of m is given by Equation 4.

$$m = \frac{\mu + k\lambda}{k+1} \tag{4}$$

Fuzziness can be now defined as in Equation 5.

$$F = \frac{\lambda - \mu}{2m} \quad \text{so, approximately} \tag{5}$$

$$\mu = \left(1 - \frac{2kF}{k+1}\right) * m \tag{6}$$

$$\lambda = \left(1 + \frac{2F}{k+1}\right) * m \tag{7}$$

Hence, the TMF $\partial(E)$ is represented by Equation 8.

$$\delta(E)=\begin{cases}0 & if\ E\le a\mu^{b}\\ \dfrac{(E/a)^{1/b}-\mu}{m-\dfrac{\mu}{2}} & if\ a\mu^{b}\le E\le am^{b}\\ \dfrac{\lambda-(E/a)^{1/b}}{\lambda-\dfrac{m}{2}} & if\ am^{b}\le E\le a\lambda^{b}\\ 0 & if\ E\ge a\lambda^{b}\end{cases} \qquad (8)$$

- *Defuzzification*: The output fuzzy estimate of E can be calculated as a weighted average of the optimistic ($a\alpha^b$), most likely (am^b) and pessimistic estimate ($a\beta^b$). Fuzzy effort estimate (E) is given by the formula in Equation 9.

$$E=\frac{w_1(a\alpha^b)+w_2(am^b)+w_3(a\beta^b)}{w_1+w_2+w_3}+\coprod_{i=1}^{15}EM_i \qquad (9)$$

  Where w_1, w_2 and w_3 are the weights of the optimistic, most likely and pessimistic estimate respectively and EM_i is the 15 effort multipliers from COCOMO. Maximum weight should be provided to the most expected estimate. Here, the value of m indicates the size in KLOC. The values of α and β, k, F, w_1, w_2 and w_3 are arbitrary constants. The effort is obtained in terms of persons per month.

- *Fuzzy Rules*: The fundamental element of the COCOMO model is utilized for the generation of the fuzzy rules to estimate nominal effort, free of cost drivers. In this way, by dividing input and output spaces into fuzzy regions, the correspondence between mode, size and resulting effort can be produced [14, 23]. The parameters of the effort MFs were established for the given mode, size pair. 3 MFs representing effort were obtained for a random size and 3 modes respectively. Rules formulated depending on the fuzzy sets of modes, sizes and efforts appear in the following form:

If mode is organic and size is s_1 **then** effort is e_{11}
If mode is semi-detached and size is s_1 **then** effort is e_{21}
If mode is embedded and size is s_1 **then** effort is e_{31}
If mode is organic and size is s_2 **then** effort is e_{12}
If mode is semi-detached and size is s_2 **then** effort is e_{22}
If mode is embedded and size is s_2 **then** effort is e_{32}
⋮
If mode is m_j and size is s_i **then** effort is e_{ji}

Figure 1. Fuzzy rules.

In this work, an optimized fuzzy logic based framework is proposed for managing the imprecision and uncertainty associated with the data at earlier phases of the project and to accurately predict the software effort as well. This framework is constructed upon the existing cost estimation model, called the COCOMO. The COCOMO model is an empirical model that was formed by gathering data from a large number of software projects. Making analysis on these data can discover formulae that were the best fit to the observations. These formulae give the relation between the size of the system and product, project and team factors and the effort required to develop the system. In COCOMO, effort is stated by Person Months (PM). Cost drivers have up to six levels of rating and they are: Very low, low, nominal, high, very high, and extra high. Each rating has an equivalent real number Effort multiplier (EF), based upon the factor and the level to which the factor can control productivity.

Product attributes are related to the necessary characteristics of the software product being developed. Platform attributes are the constraints enforced on the software by the hardware platform. Personnel attributes are multipliers, which consider the knowledge and abilities of the people functioning on the project. Project attributes are connected with the specific features of the software development project.

Scale Factors (SF) are acceptable product objectives, flexibility, team coherence, etc., EF indicate software reliability, database size, reusability, complexity, etc., the imprecise nature of the cost drivers have an important effect on the accuracy of the effort estimates derived from software effort estimation models. The vagueness and uncertainty of software effort drivers cannot be prevented from occurring. Hence, a fuzzy model that adopts fuzzy sets can be beneficial for verifying the cost drivers in an easier way.

3.5. Cost Estimation

This step aids in estimating the size of the software product. The two main types of cost estimation methods are the algorithmic and non-algorithmic cost estimation methods. Algorithmic models change extensively in mathematical complexity. Few of them rely on simple arithmetic formulas that use summary statistics like means and standard deviations. Others rely on regression models and differential equations. For the purpose of enhancing the accuracy of algorithmic models, the model has to be adjusted or calibrated to local conditions. These models cannot be used off-the-shelf. Calibration can also result in inaccuracy. The initial step to predict cost is to decide the cost of procurements. The cost of procurements involves the determination of cost of support and services such as workstations, test-bed boards and simulators, ground support equipment, network charges and phone charges.

This is particularly correct in situations, where the expense is made to fit into the budget imposed on the software project. Consequently, it may be essential to repeat the estimates of other steps several times, decrease the effort and procurements or guess more risk to fit into the imposed budget. If the schedule becomes too large, costs will get elevated because effort moves out to more expensive years. The goal of software costing is to precisely forecast the cost involved in developing the software. If the project cost has been computed as a piece of a project bid to customer, then a choice has to be made about the price cited to the customer. Typically, price is simply cost plus profit. Figure 2 depicts the entire cost estimation process.

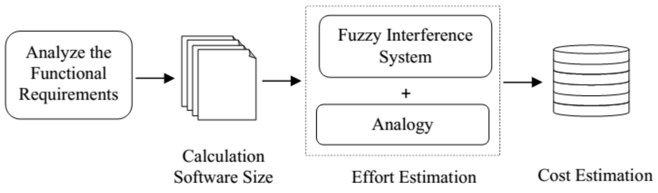
Analyze the Functional Requirements
Calculation Software Size
Fuzzy Interference System
+
Analogy
Effort Estimation
Cost Estimation

Figure 2. Block diagram for the entire proposed methodology.

4. Results and Discussion

The datasets used for this work are the Desharnais dataset [19], NASA 93 [20] and COCOMO NASA dataset.

- *Dataset Description*: The original version of the Desharnais dataset includes 81 projects, of which 4 were excluded owing to incomplete values. The dataset owns 9 independent variables and 1 dependant variable. Actual Effort in person hours is considered as the 10th variable for the matrix B.

The NASA 93 dataset contains 93 complete projects with 17 independent variables, of which 15 are categorical. This dataset is in COCOMO 81 format collected from NASA centers published in Predictor Models In Software Engineering (PROMISE). Here, the DevEffort variable is taken into account for generating matrix B.

Table 1 represents the difference of both actual and estimated efforts for all the 3 datasets. Here, we represented the effort of only 5 projects from each data set and the performance was calculated for the whole database.

Table 1. Effort comparison of different datasets.

Projects No.	Effort					
	COCOMO NASA		NASA 93		Desharnais	
	Actual Effort	Estimated Effort	Actual Effort	Estimated Effort	Actual Effort	Estimated Effort
1	278	268.22	117.6	109.90	5152	5036.45
2	1181	1112.06	117.6	113.93	5635	5432.06
3	1248	1191.16	31.2	27.76	805	946.24
4	480	420.35	36	35.01	3829	3709.38
5	120	103.14	25.2	23.74	2149	2134.32

- *Performance Analysis*: Different investigators have utilized different error measurements. The most accepted error measure is the Mean Absolute Relative Error (MARE).

$$MARE = \sum_{i=1}^{n} \left(\left| \left(est_i - acl_i \right) / acl_i \right| \right) / n \tag{10}$$

Where est_i is the estimated effort, acl_i is the actual effort and n is the number of projects in the model.

In the proposed method, 3 types of dataset are utilized. The comparison of MARE values of all the three datasets is described in Figure 3. From the graph, it can be noted that the NASA 93 data set has low MARE value when compared against the other 2 datasets.

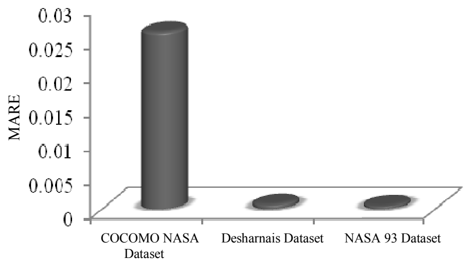

Figure 3. The MARE measure for all the three data sets.

The MARE measures for all the datasets used in effort estimation process is given in Table 2.

Table 2. MARE measures of all the datasets.

Datasets	MARE
Desharnais Dataset	0.000724
NASA 93 Dataset	0.000507
COCOMO NASA Dataset	0.026174718

The value of MARE for effort can be compared with other existing methods stated in [4]. The comparison between the proposed method and the existing method is shown in Table 3. The proposed method utilizing Desharnais dataset is compared against the existing method.

Table 3. MMRE comparison for Desharnais dataset.

Methods	MMRE
Proposed Method	0.000713
Fuzzy Set Method [18]	0.00138

The MRE and MMRE can be computed using the formulas in Equations 11 and 12 respectively.

$$MRE = \left| \left(acl_i - est_i \right) \right| / acl_i \tag{11}$$

$$MMRE = \frac{1}{N} \sum_{i}^{N} MRE_i \tag{12}$$

The comparison of proposed method with the existing method based on MRE and MMRE measure is expressed in Figure 4.

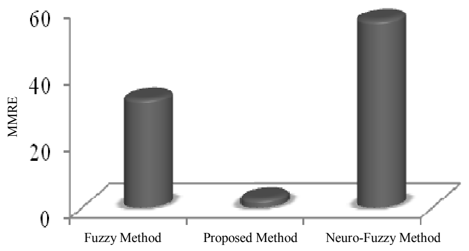

Figure 4. MMRE comparison for COCOMO NASA dataset.

Here, the MMRE measure of the proposed method, making use of COCOMO NASA dataset is compared to the existing method cited in [18]. The MMRE measure is estimated in %.

The measurements are displayed in Table 4. From Figure 4 and Table 4, it is evident that our proposed method works efficiently with COCOMO NASA dataset, when compared to the existing methods mentioned in [3, 18].

Table 4. MMRE comparison for COCOMO NASA dataset

Methods	MMRE (%)
Proposed Method	2. 6
Fuzzy Method	32.651
Neuro-Fuzzy Method	56.46

Thus, it is apparent that the proposed method of cost estimation system, which relies on soft computing techniques, can effectively estimate the effort and cost of the software project models.

5. Conclusions

In this paper, a novel method for estimating the software project effort is proposed. This technique depends on reasoning by analogy, fuzzy logic and linguistic quantifiers. This kind of approach holds well, when the software projects are expressed by categorical and/or numerical data. Therefore, this proposed approach enhances the classical analogy process that does not consider categorical data. In the fuzzy analogy approach, fuzzy sets are used to characterize both the categorical and the numerical data. The proposed methodology is cross validated with 3 datasets such as COCOMO NASA, Desharnais, and Nasa 93. The NASA 93 Dataset outperforms the other two datasets and all the datasets are compared with the existing effort estimation techniques. The results of experimentation reveal that the proposed method is capable of effectively estimating both effort and cost of the software project models.

Reference

[1] Al Dallal J., “Mathematical Validation of Object-Oriented Class Cohesion Metrics,” *International Journal of Computers,* vol. 4, no. 2, pp 45- 52, 2010.

[2] Attarzadeh I. and Ow S., “A Novel Algorithmic Cost Estimation Model Based on Soft Computing Technique,” *Journal of Computer Science*, vol. 6, no. 2, pp. 117-125, 2010.

[3] Du W., Ho D., and Capretz L., “Improving Software Effort Estimation Using Neuro-Fuzzy Model with SEER-SEM,” *Global Journal of Computer Science and Technology*, vol. 10, no. 12, pp. 52-64, 2010

[4] Idri A., Abran A., Khosgoftaar T., “Fuzzy Analogy: A New Approach for Software Cost Estimation,” *in Proceedings of the 11th International Workshop in Software Measurements*, Montréal, Canada, pp. 93-101, 2001.

[5] Iraji M. and Motameni H., “Object Oriented Software Effort Estimate with Adaptive Neuro Fuzzy use Case Size Point (ANFUSP),” *International Journal of Intelligent Systems and Applications,* vol. 4, no. 6, pp. 14-24, 2012.

[6] Jiang Y., Cukic B., and Ma Y., “Techniques for Evaluating Fault Prediction Models,” *Empirical Software Engineering*, vol. 13, no. 5, pp. 561-595, 2008.

[7] Kashyap D., Tripathi A., and Misra A., “Software Development Effort and Cost Estimation: Neuro-Fuzzy Model,” *Journal of Computer Engineering*, vol. 2, no. 4, pp. 12-14, 2012.

[8] Kaushik A., Chauhan A., Mittal D., and Gupta S., “COCOMO Estimates Using Neural Networks,” *International Journal of Intelligent Systems and Applications,* vol. 4, no. 9, pp. 22-28, 2012.

[9] Kaushik A., Soni A., and Soni R., “A Simple Neural Network Approach to Software Cost Estimation,” *Global Journal of Computer Science and Technology Neural & Artificial Intelligence*, vol. 13, no. 1, pp. 1-10, 2013.

[10] Khan I. and Alam M., “Software Cost Estimation using a Neuro-Fuzzy Algorithmic Approach,” *International Journal of Computer Science and Management Research*, vol. 2, no. 7, pp. 3140–3147, 2013.

[11] Malathi S. and Sridhar S., “Estimation of Effort in Software Cost Analysis for Heterogenous Dataset Using Fuzzy Analogy,” *International Journal of Computer Science and Information Security*, vol. 10, no. 10, pp. 1-5, 2012.

[12] Olszak A., Bouwers E., Jrgensen B., and Joost V., “Detection of Seed Methods for Quantification of Feature Confinement,” *in Proceedings of the 50th International Conference on Objects, Models, Components*, Patterns, pp. 552-268, 2012.

[13] Orsila H., Geldenhuys J., Ruokonen A., and Hammouda I., “Update Propagation Practices in Highly Reusable Open Source Components,” available at: http://www.zakalwe.fi/~shd/publications/orsila_update_propagation_practices_2008.pdf, last visited 2008.

[14] Pedrycz W., Peters H., and Ramanna S., “A Fuzzy Set Approach to Cost Estimation of Software Projects,” *in Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, Alberta, Canada, pp. 1068-1073, 1999.

[15] Prasad P., Sudha K., Rama S., and Ramesh S., “Software Effort Estimation using Radial Basis and Generalized Regression Neural Networks,” *Journal of Computing*, vol. 2, no. 5, 2010.

[16] Rao B., Sameet B., Swathi K., Gupta K., RaviTeja C., and Sumana S., “A Novel Neural Network Approach for Software Cost Estimation Using Functional Link Artificial Neural Network (FLANN),” *International Journal of Computer*

Science and Network Security, vol. 9 no. 6, pp. 126-131, 2009.

[17] Reddy C. and Raju K., "A Concise Neural Network Model for Estimating Software Effort," *International Journal of Recent Trends in Engineering*, vol. 1, no. 1, pp. 188-193, 2009.

[18] Reddy C. and Raju K., "Improving the Accuracy of Effort Estimation through Fuzzy Set Representation of Size," *Journal of Computer Science*, vol. 5, no. 6, pp. 451-455, 2009.

[19] Shepperd M., Schofield C., and Kitchenham B., "Estimating Software Project Effort Using Analogies," *IEEE Transaction on Software Engineering*, vol. 23, no. 11, pp. 736-743, 1997.

[20] Shirabad S. and Menzies T., "The PROMISE Repository of Software Engineering Repository. Repository," available at: http://promise.site.uottawa.ca/SERepository, last visited 2014.

[21] Sodikov J., "Cost Estimation of Highway Projects in Developing Countries: Artificial Neural Network Approach," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 6, pp.1036-1047, 2005.

[22] Tronto I., Silva J., and Anna N., "The Artificial Neural Networks Model for Software Effort Estimation," *INPE ePrint*, vol. 2006, 2006.

[23] Wang L. and Mendel J., "Generating Fuzzy Rules by Learning from Examples," *IEEE Transactions on System, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414-1427, 1992.

[24] Yahya A., Ahmad R., and Lee S., "Impact of CMMI Based Software Process Maturity on COCOMO II's Effort Estimation," *the International Arab Journal of Information Technology*, vol. 7, no. 2, pp. 129-138, 2010.

[25] Ziauddin., Kamal S., khan S., and Nasir J., "A Fuzzy Logic Based Software Cost Estimation Model," *International Journal of Software Engineering and Its Applications*, vol. 7, no. 2, pp. 7-18, 2013.

Marappagounder Shanker Marappagounder Shanker received his BTech degree in the stream of Information Technology from V.L.B Janakiammal College of Engineering and Technology, Anna University, Chennai, and ME degree in computer science engineering from Kongu Engineering College, Anna University, Chennai. He is also an SAP Certified Business Intelligence (BI) Consultant. He has worked with HCL Technologies as Software Engineer for 2 years.He is currently working with ATOS. His specializations include software engineering, ERP(SAP)-data warehousing. His current research interests include:Cost and effort estimation techniques in software engineering.

Jayabalan Jaya received her Ph.D in Information and Communication Engineering from Anna University, Chennai. She completed her M.Tech in Advanced communication systems (University second rank holder) from SASTRA University, Tanjore, Tamilnadu and BE in Electronics Communication Engineering from Sri Ramakrishna Engineering College, Coimbatore, Tamilnadu. She has published two books, Digital Signal Processing and Digital Image Processing and has published 20 research papers in International level and 34 research papers in National level. She is a proud member of various chapters like IEEE, Indian Society for Technical Education (ISTE) Computer Society of India (CSI), Indian Women Network IWN, International Association of Computer Science & Inf. Tech. (IACSIT), ICTACT, International Association of Engineers (IAENG) and IETE.

Keppanagounder Thanushkodi received his Bsc degree in electrical and electronics engineering from College of Engineering, Chennai, University of Madras in 1972, MSC degree (Engg) in power systems engineering from University of Madras in PSG College of Technology, Coimbatore and in 1991, and PhD in power electronics from Bharathiyar University, Coimbatore. He has highly professionalized teaching experience for 43 years. He has published more than 150 Papers in various Journals and 200 Papers in National and International Conferences. Currently he is The Director, Akshaya College of Engineering and Technology, Coimbatore. His research interests include the areas of power system, power electronics, computer networking, image processing, software engineering and virtual instrumentation.