International Journal of Computer Applications (0975 – 8887)
Volume 87 – No.16, February 2014

Develop Efficient Technique of Cost Estimation Model for Software Applications

Lalit V. Patil
Research Scholar (Comp), Bharath University, Chennai, India

Sagar K Badjate
ME IT Dept, Smt. Kashibai Navale Navale College of Engineering, Pune, India

S.D.Joshi, Ph. D
Prof. Dept. of Comp BVDUCOE, Pune,India

ABSTRACT

Software cost estimation predicts the amount of effort and development time required to build the system. Instead of just putting values into giving equation to calculate the cost and effort, we require more work on a scale and cost drivers to increase the accuracy of software cost estimation. The Software cost estimation process depends on the attributes such as peoples working in teams, programming language used and software tools used, salaries and overhead costs associated with the development team, database size used, training cost, accidental rework, a policy used in an organization, cost of shared facilities such as a library, restaurant, resources used such as light, network etc, which gives a clear idea about software cost estimation. There are so many models available categorized into algorithmic and non-algorithmic model each of their strengths and weakness. We propose a hybrid approach, which consists of Functional Link Artificial Neural Network (FLANN) and COCOMO-II with training algorithm. FLANN reduces the computational complexity in multilayer neural network. It does not have any hidden layer, and it has fast learning ability.

Keywords

Functional link artificial neural network (FLANN), software cost estimation, COCOMO-II.

1. INTRODUCTION

Before 1972, software cost estimation process was performed manually using estimating algorithms. Some of the methods developed on a trial basis. The methods can be derived from actual projects can be utilized for similar size projects. During estimation of projects, project failures have been an important issue. Software projects usually fail during planning and estimation steps. Failure reasons such as incomplete requirements, insufficient planning, and estimation at early stages. Hence, the software estimation process depends on several conditions, parameters. For cost estimation, there are many methods and guideline available, but there is still a need for improvement. We can reduce the cost estimation error through analysis of cost estimation error, for example; we can identify different cost estimation processes, which lead to more accurate estimate. Software cost estimation is the process to determine staff allocation and schedule for a software project because human effort is the important cost driver in a software project. The effort estimate determines the budget of the project. Software cost estimation is divided into two methods: Algorithmic and Non-algorithmic models [1]. These two models are used for accurate estimation. The following sections describe existing methodologies their advantages and disadvantages, area of improvement.

2. EXISTING METHODS

2.1 Algorithmic models

These models work based on the algorithm. They usually need data and make results by using the mathematical relations. Algorithmic Models are sorted out into some different examples. Each algorithmic model uses an equation to do the estimation:

$$Effort = f(x1, x2, x3, \ldots., x_n)$$

Where $(x1, x2, \ldots., x_n)$ are the cost factors.
The All cost factors using in these models are:
Product factors: required reliability, product complexity, database size used, required reusability, documentation match to life-cycle needs.
Computer factors: execution time constraint, main storage constraint, computer turnaround constraints, platform volatility.
Personal factors: analyst capability, application experience, programming capability, platform experience, language and tool experience, personnel continuity.
Project factors: multisite development, use of software tools, required development schedule.

a) Advantages of algorithmic model:
- Repeatable estimations are possible.
- Easy to modify input data, refine and customize formulas.

b) Disadvantages algorithmic model:
- Poor sizing inputs and inaccurate cost driver rating will result in inaccurate estimation.

2.1.1 Source line of code

SLOC includes commands and data definition, but, it does not include instructions such as comments, blanks, and continuation lines. After computing the SLOC for software, its amount is compared with other projects, which their SLOC has been computed before, and the size of the project is estimated. SLOC is the base of the estimation models in many complicated software estimation methods. SLOC usually is computed by considering SL as the lowest, SH as the highest and SM as the most probable size [2].

$$S = (SL + 4SM + SH)/6$$

The expected SLOC for the entire software system E is simply the sum of the expected SLOC of each piece

$$S = \sum_{i=0}^{n} \text{Si}$$

Where n is the total number of pieces.

a) Advantages
- SLOC is an easy to compute line of code. Manual counting effort can be removed by automatic counting process. Hence, it has a scope for automation of counting the lines of code.

b) Disadvantages:

- SLOC Measuring seems very difficult at the early stages of the project because of the lack of information about requirements.

2.1.2 *Function Point Size Estimates*

Albrecht (1983) presented Function Point metric to measure the functionality of the project [2]. In this method, estimation includes:

- User-input: data or control user-input types.
- User-output: output data types to the user.
- Inquiry types: interactive inputs requiring a response.
- Internal files: files (logical groups of information) that are used and shared inside the system.
- External files: files that are passed or shared between the system and other systems.

a) Advantages:

- It can be estimated from requirements specifications.
- It is possible to estimate development costs in the early phases of development.
- Function points are independent of the language, tools, or methodologies used for implementation.

b) Disadvantages:

- It requires manual work, as it can be counted manually and which require more time.
- It requires detailed knowledge of requirement for estimation of software size using function points.
- New developer cannot easily estimate the size as it requires experience with function point.

2.1.3 *COCOMO*

The COCOMO (Constructive Cost Model) model is a regression based software cost estimation model. Barry Boehm developed it in 1981[3]. It is the most popular method, which is categorized in algorithmic models.
The basic COCOMO model has a very simple form:

$$MAN - MONTHS = K1 * (Thousands\ of\ Deliverd\ Source\ Instruction)k2$$

Where K1 and K2 are two parameters dependent on the application and development environment. Estimation of COCOMO model can accurately, which considers qualification and experience of the development team.

a) Advantages:

- Simple to estimate cost and effort.

b) Disadvantages:

- Estimation at the early stage of software development leads to failure of estimation.

2.1.4 *COCOMO-II*

Boehm developed and calibrated the popular algorithmic methods used for software cost estimation called COCOMO II, which has three different models, but the most detailed one is the post architecture model [4]. It is used after the project's overall architecture has been developed. It determines the efforts (in Person-Months) required for a project based on software project's size in KSLOC (Kilo Source Line Of Code) as well as other cost factors known as scale factors and effort multipliers by:

$$Effort = A * [Size]E * \prod_{i=1}^{17} Em$$

$$Where\ E = B + 0.01 * \sum_{i=1}^{5} SFi$$
$$A = 2.94, B = 0.9$$
$$Time = C * Effort^{F}$$
$$Where\ F = D + 0.2 * 0.01 * \sum_{i=1}^{5} SFi$$
$$C = 3.67, D = 0.28$$
$$COST = Effort * (Payment\ per\ Month)$$

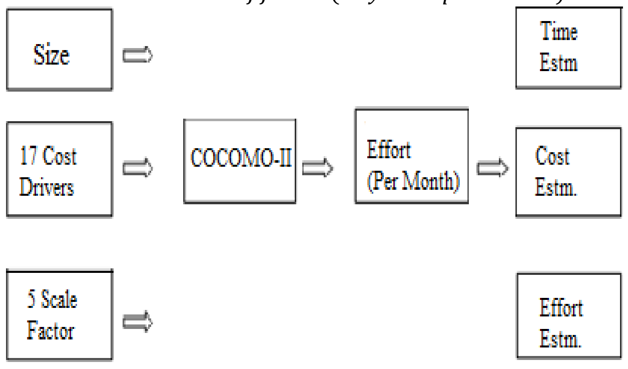

Fig 1: COCOMO II Cost & Effort Estimation Model

- Five scale factors are:
  Precedentedness, Developing Flexibility,
  Risk Resolution, Team Cohesion, and Process maturity.
- Seventeen Cost Factor Based on
  Product attributes; Computer attributes; Personal attributes; Project attributes.

a) Advantages:

- COCOMOII is an industry standard.
- Very profound information is easily available.
- Clear and effective calibration process.

b) Disadvantages:

- The duration calculation for small projects is unreasonable [5].

2.2 Non -algorithmic Models

In non-algorithmic models, it is necessary to have the enough information about the previous projects because these methods perform the estimation by analysis of the historical data. Non-algorithmic methods are easy to learn because all of them follow the human behavior. The non-algorithmic method includes:

2.2.1 *Expert Judgment*

Expert judgment techniques involve expert or group of expert consulting with software cost estimation. They share experiences and understanding of the proposed project to arrive at an estimate of its cost [5].
It involves following steps:

- The coordinator provides to each expert with a specification and estimation form.
- The coordinator calls meeting with experts and discusses estimation issues.
- Coordinator prepares and distributes a summary of a estimation in the form.
- Coordinator again calls meeting with experts discuss points where their estimates varied widely.
- Experts fill the form and above steps iterate for many times as accurate estimate occurs.

a) Advantages:

- Availability of experts makes a difference between the past project experience and the requirements of the proposed project.
- Experts can tell impacts caused by a new technique, architecture, language involved in future projects.

b) Disadvantages :

- It is hard to document the factors used by the experts or experts-group.
- The expert may be some biased, optimistic, and pessimistic, even though they have been decreased by the group consensus.

2.2.2 Estimation by Analogy

In this method, comparison of the proposed project to previously completed a similar project where the project development information known. Input data of completed projects are taken into the estimate the proposed project [6]. It is straightforward method. It can act similar with expert judgment method experts often search for analogous situations to inform their opinion.

The steps using estimating by analogy are:

- Selecting the most similar completed projects whose characteristics have been stored in the historical database.
- Deriving the estimate for the proposed project from the most similar completed projects by analogy.

a) Advantages:

- The estimator's past experience and knowledge can be used which is not easy to be quantified.
- Easy to find differences between the completed and the proposed project can be identified and impacts estimated.

b) Disadvantages:

- They are restricted to information that is available at the point that the prediction required.

2.2.3 Top-down estimation method

It is also called Macro Model. In this method, cost estimation for the project is derived from the global requirement of the project. The requirement of the project is divided into low-level components. This method useful in the early phase of the software development, it is very useful because there is no detailed information available.

a) Advantages:

- It works on system-level activities such as integration, documentation, configuration management.
- It is easier to implement, due less requirement.

b) Disadvantages:

- It does not identify the detailed level problems; due to this the cost estimate will be wrong.

2.2.4 Bottom-up cost estimating Method

In this method cost of software components is estimated and then combines the results to arrive at an estimated cost of the overall project. The leading method using this approach is COCOMO's detailed model.

a) Advantages

- It is easy to handle estimate of components of software for which the group has a feel.
- Minimize errors in estimation method.

b) Disadvantage

- It is a time consuming process.
- It may estimate inaccurate results because; necessary information may not available in the early phase.

2.2.5 Neural Network

An Artificial Neural Network (ANN), often just called a neural network, it is the node like structure which act as a mathematical model inspired by biological neural networks. A neural network consists of interconnected nodes of artificial neurons, and it processes information using a connectionist approach to computation. It forms a network as it consists of nodes which are interconnected to each other. It has following factors:

- Interconnection nodes.
- The learning process for updating the weights of nodes
- Activation function which converts the input to output.

For estimation, it can be used with COCOMO-II. Inputs to COCOMO-II are cost and scale drivers, which are then passed to the neural network [7]. So many neural network models available feed forward neural network, multilayer neural network, they can be used different training algorithms as feed forward algorithm and back propagation algorithm and gives output.

a) Advantages:

- Estimation process can give accurate results due to training of network process.

b) Disadvantage:

- Complexity increases due to learning of parameters.
- It is a tedious task to train the network.

2.2.6 Fuzzy Logic

This method consists of three main components such as fuzzification process, inference from fuzzy rules and defuzzification process. There are four steps in fuzzy approach.

- Fuzzification: It converts a crisp input into a fuzzy set.
- Fuzzy rule base: Fuzzy logic systems use fuzzy IF-THEN rules.
- Fuzzy Inference Engine: Access fuzzy rule base to generate the output.
- Defuzzifier: It converts the fuzzy output into crisp output.

Fuzzy logic used with algorithmic model COCOMO-II for software cost estimation [8]. Cost and scale drivers are inputs to fuzzy rule, and it produces accurate estimates. A combination of neural network and fuzzy logic forms Neuro-fuzzy model. This model can use with COCOMO-II model [9].

a) Advantages:

- Gives an accurate estimate with fuzzy rules.

b) Disadvantage:

- Complexity increases due to the complex process of fuzzy logic.

3. AREA OF IMROVEMENT

Commonly used software cost estimation methods are SLOC, function point size estimate, COCOMO, analogy, expert judgment, neural networks and fuzzy logic. These methods are very popular for estimation.

COCOMO is the most commonly used technique because it is simple for estimating the efforts for the project at any stage. COCOMO is taken as a base model for the software cost estimation with Machine Learning by using the neural networks. Researchers are constantly working on developing new software cost estimation techniques using neural networks.

COCOMO-II used with radial basis function (RBF) neural networks, multi-layer neural networks for software development effort. Some works have demonstrated that the level of accuracy in software effort estimates strongly depends on the values of the parameters of these methods [10]. The wavelet neural network is also used with COCOMO-II for improvements in efforts and cost [11]. Here, we mainly focus on problems associated with Multilayer neural networks.

Multilayer neural network has a multi-layer architecture with one or more hidden layers between its input and output layers. The node output from each layer is directly input to the successive layer nodes. Each node layer it consists of nonlinear function. The training of MLP starts with random initial weight, and its process consists of a forward pass input vector through the network layer by layer. In forward pass, the sum of input passed to nonlinear function with a training algorithm of FLANN to produce output.

3.1 Drawbacks of multilayer neural network

- It depends on the complexity of the problems; number of layers and number of neurons in the hidden layer need to be changed.
- As the number of layers and the number of neurons in the hidden layers increases, training the model becomes more complex.

4. PROPOSED SYSTEM

To overcome the complexities associated with multi-layer neural network, we have used a single layer neural network as an alternative approach, but the single layer neural network being linear in nature. It cannot map nonlinear problems. To bridge the gap between the linearity in the single layer neural network and the highly complex multilayer neural network, we use the Functional Link Artificial Neural Network (FLANN) is proposed by PAO [12].

We propose a hybrid model as we used COCOMO-II as a base model with FLANN. FLANN utilizes a higher combination of its inputs. FLANN is used for prediction it has a single-layer network compared to the MLP. It is a flat network with no hidden layer that makes it easier.

The Proposed system predicting software effort is a single-layer feed forward neural network, which has one input layer and an output layer. It generates output by providing input as cost drivers and scale drivers and then processing to the final output layer. The proposed model as shown in fig 2.

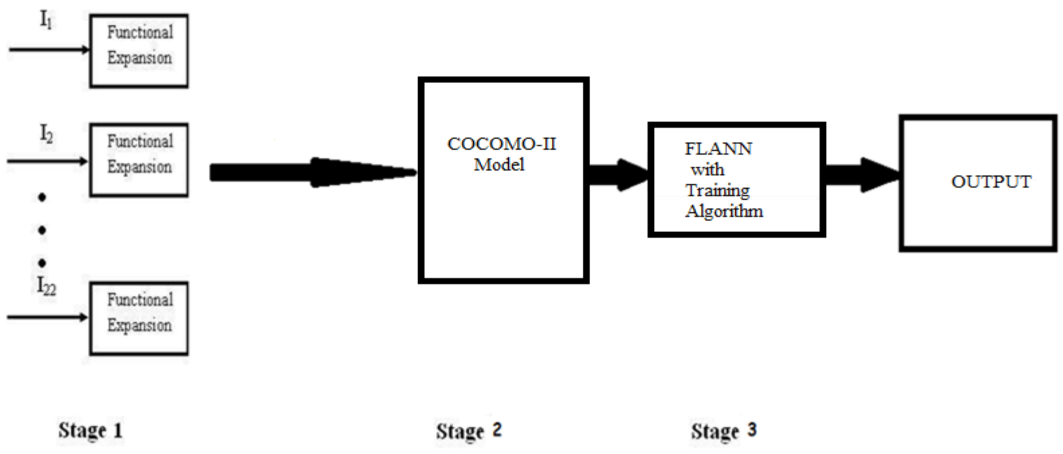

Fig2: Proposed System using COCOMO-II with FLANN

4.1 Proposed Implementation

Step1: Determine the size of source code:

Determine the tentative size of the project. This may include specifications, source code, manuals, technical documents etc. They can be calculated as function point or SLOC Method.

Step 2: Input five scale factor and seventeen efforts Multiplier cost drivers.

Product factors: required reliability, product complexity, database size used, required reusability, documentation match to life-cycle needs.

Computer factors: execution time constraint, main storage constraint, computer turnaround constraints, platform volatility.

Personal factors: analyst capability, application experience, programming capability, platform experience, language and tool experience, personnel continuity.

Project factors: multisite development, use of software tools, required development schedule.

They include very low, low, nominal, high & very high. According to that rating value, we can calculate the complexity of the project & how many person/month required for developing the project.

Step 3: FLANN

The twenty-two cost factors of the validated dataset are contained as the input of the network. These factors are then expanded functionally by using the learning algorithm.

Step 4: Implementation for Effort & Time
A use COCOMOII model for calculating the effort & Time.

$$Effort = A * [Size]E * \prod_{i=1}^{17} Em$$

$$Where\ E = B + 0.01 * \sum_{i=1}^{5} SFi$$
$$A = 2.94, B = 0.9$$
$$Time = C * Effort^{F}$$
$$Where\ F = D + 0.2 * 0.01 * \sum_{i=1}^{5} SFi$$
$$C = 3.67, D = 0.28$$

Step 5: Estimation of cost:

$$COST = Effort * (Payment\ per\ Month)$$

5. CONCLUSION

There are many software cost estimation methods available, including algorithmic methods and non-algorithmic methods. We have seen their advantages and disadvantages.

To increase accuracy, we need to improve existing models. By utilizing a combination of algorithmic and non-algorithmic model with FLANN, as it is a fast learning network. Which can significantly improvement in its total computational efficiency of the software cost estimation.

For the future, work the accuracy of FLANN increase by applying particle swarm optimization methods such as an artificial bee colony algorithm.

6. REFERENCES

[1] Vahid Khatibi, Dayang N.A.Jawawi "Software Cost Estimation Methods: A Review", CIS Journal 2011.

[2] Albrecht. A.J. and J. E. Gaffney, "Software function, source lines of codes, and development effort prediction: a software science validation", IEEE Trans Software Eng. SE, pp.639-648, 1983.

[3] Boehm B. W. "*Software Engineering Economics*", Englewood Cliffs, NJ, Prentice-Hall, 1981.

[4] Musilek A. "On the Sensitivity of COCOMO II Software Cost Estimation Model" IEEE 2002.

[5] Jorgensen, M. "Practical guidelines for expert-judgment-based software effort estimation", IEEE Software, 22(3), 57-63. Doi: 10.1109/MS. 73, 2005.

[6] Shepperd M. "Estimating Software Project Effort Using Analogies" IEEE NOV. 1997.

[7] K. Srinivasan and D. Fisher, "Machine Learning Approaches to Estimating Software Development Effort", IEEE Transactions on Software Engineering, 21 (2), 1995.

[8] Iman Attarzadeh, Siew Hock Ow, "Improving Estimation Accuracy of the COCOMO II Using an Adaptive Fuzzy Logic Model" 2011 IEEE International Conference on Fuzzy Systems June 27-30, 2011, Taipei, Taiwan.

[9] Xishi Huang, Luiz F. Capretz, Jing Ren, Danny Ho, "A Neuro-Fuzzy Model for Software Cost Estimation" IEEE Proceedings of the Third International Conference on Quality Software (QSIC'03).

[10] Adriano L.I. Oliveira , Petronio L. Braga, Ricardo M.F. Lima, Márcio L. Cornélio "GA-based method for feature selection and parameters optimization for machine learning regression applied to software effort estimation" Journal of Information and Software Technology 52 (2010) 1155–1166.

[11] K. Vinay Kumar, V. Ravi, Mahil Carr, N. Raj Kiran- "Software development cost estimation using wavelet neural networks" The Journal of Systems and Software 81 (2008) 1853–1867.

[12] Y. H. Pao, Adaptive Pattern Recognition and Neural Networks, Reading, MA: Addison-Wesley, 1989.