International Journal of Hybrid Information Technology
Vol.8, No.10 (2015), pp.73-80
http://dx.doi.org/10.14257/ijhit.2015.8.10.07

A Proposed Framework for Software Effort Estimation Using the Combinational Approach of Fuzzy Logic and Neural Networks

Pawandeep Kaur and Rupinder Singh

CSE Department, Chandigarh University
jassal.pawan23@gmail.com, rupipanjgotra1989@gmail.com

Abstract

Software effort and cost estimation has turned out to be the major challenge in IT industries. In this paper, different software effort and cost estimation techniques like algorithmic methods, expert judgment, analogy based estimation and soft computing methods and their various aspects are discussed. Software effort estimation is followed by cost assessment that is useful for both customers and developers. In this paper comparison between different approaches are discussed and proposed a hybrid technique for effort estimation using fuzzy logic and neural network. This will be useful to make better estimation results.

Keywords: *Software Effort Estimation Techniques, Fuzzy logic, Neural Networks, Accurate estimation*

1. Introduction

Software effort estimation is the process of predicting the realistic amount of effort/cost required for the completion of software projects. Effective software project estimation is one of the most demanding and essential activities in software development. Software effort estimation holds good at early stages of software development to meet competitive demands of today's industry. Now the days, people are looking for new and good quality software but at low cost and in lesser time [1]. So the cost-benefit analysis is to be done by client or developer [2]. Analyzed Estimation is translated into dollars. As the demand of software effort estimation has increased in industry it is becoming a most important task that has to be done at early stages of development so that feasibility can be evaluated. Accuracy in effort estimation plays a vital role in the success of software project management [3]. Resource allocation is the biggest issue in estimation as if excess amount of resources are allocated then it is called overestimation and if lesser then it is called underestimation. Accurate effort prediction is useful in reducing the impact of risks. Good software estimation is difficult task at early stages as having a lot of uncertainties in inputs like change in requirement, platform change, size of project, budget constraints, complexity *etc.*

1.1. Importance of Accurate Estimation

- To determine what amount of resources and manpower required to complete the project and how well they will be used.
- To find effort in terms of time, cost and technical resources.
- To assess the impact of changes in project and support re-planning.

1.2. Ingredients to Accurate Estimation

- **Working Environment**: The environment in which project is to developed or executed is very important.

ISSN: 1738-9968 IJHIT

- **Tools**: Tools and techniques that are used to execute or evaluate the project play an important role in software development.
- **Consistency**: The historical data is useful to estimate similar projects which were developed earlier. On occasion new observations, obstacles, issues will occur so it will make the get experience to tackle the situation.

2. Related Work

A Magne Jorgensen, *et al.,* [4, 5] focused on the reason of occurrence of error in software effort estimation by interviewing the employees in different labs in software organizations. Then they focused on the expert judgment process with some useful, easily implemented practices. They found that by mixing planning, estimation and bidding leads to less accurate results. They also define that by combining the estimation methods having different weakness and strengths increase the estimation accuracy because estimation increases with more number of sources. They define the guidelines for the selection of experts and estimation methods, and to provide learning opportunities.

Iman Attarzadeh, *et al.,* [6] described the use of fuzzy logic modeling for effort estimation. They proposed a new fuzzy logic model and provide training for better estimation. After training they compare it with COCOMO to get estimation results.

K.K.Aggarwal, *et al.,* [7], explored the use of neural network for estimating the lines of code (LOC) of the software projects. They are using the International Software Benchmarking Standards Group (ISBSG) Repository Data (release 9) for the software projects having record of LOC. They perform experiments on various algorithms and find out the best one, having minimum Mean Absolute Percentage Error (MAPE).

Magne Jorgensen, *et al.,* [8], discussed a lesions learning technique for effort estimation and uncertainty assessment. They explained that this method is based on learning process; they create two groups-control group and learning group. They found that uncertainty assessment results better in case of control group. So lesions learning technique will be better if designed carefully and having minimum bias.

Ning Nan, *et al.,* [9] described the affect of pressure due to budget and schedule compression on cycle time and cost of software development either positive or negative. They proposed a nonlinear relationship between budget pressure and schedule pressure on software development. They proposed four hypotheses and collect the cross-sectional data from international technology firm to test those hypotheses. They assess the possibility of achieving effort and improved cycle time, minimum estimation bias with proper management of pressure in software development. Analysis suggests that if software clients cooperate with development teams in dealing with schedule compression then organization has beneficial effects.

Ali Idri, *et al.,* [10] described the architecture of neural network. They explained the usage of neural networks in software effort estimation and the easiness with which neural networks were used in estimation purposes.

E. Praynlin, *et al.,* [11] proposed ANFIS for software effort estimation and using Gaussian Membership Function, Trapezoidal Function and Triangular Function and analyzed the results in the view of MRE. Lotfi Zadeh [12] gave the idea about fuzzy values and the usage of fuzzy systems in software effort estimation.

Ashish Sharma, *et al.,* [13], proposed a systematic and accurate method for software development effort estimation on the basis of requirement based complexity of the new developing software *i.e.,* requirement based software development effort estimation (RBDEE) from IRBC(Improved Requirement Based Complexity) using requirements as input from SRS. Firstly they develop RBDEE method and then compare it with other existing estimation methods to check its robustness. As this technique is used at early stage of software development, so it will give accurate results than the techniques using function point as input, based on expert judgment (may be biased).

Elham Khatibi, *et al.,* [14], proposed a systematic framework for the effort estimation in the organization. Basically indictors are divided into four groups. The proposed method has indicators which are helpful to discover the strength and weakness of the organization in respect to effort estimation.

Chander Diwaker, *et al.,* [15] explained the various cost estimation techniques/methods developed like algorithmic methods, analogy based estimation, expert judgment method, price to

win method, top-down method, and bottom-up method. They also discussed the metrics and techniques for estimation. The Strengths and weakness of different approaches were explained. They discussed that for known projects, we should use the expert judgment method or analogy based method and for large and lesser known projects, use algorithmic model like COCOMO II.

Jyoti G. Borade, *et al.*, [16] explained the various software metrics- size oriented metrics (SLOC, DSI), function oriented metrics (FP, UAFP), object points, test points, use case points. After that they discussed several estimation techniques and they also explained recent trends in software estimation like Bayesian Belief Network and Test execution effort. They enlighten the usefulness of BBN when required information is uncertain, vague, incomplete and conflicting.

3. Techniques for Software Effort Estimation

Software estimation techniques are broadly classified into two categories: Algorithmic models and Non- algorithmic models.

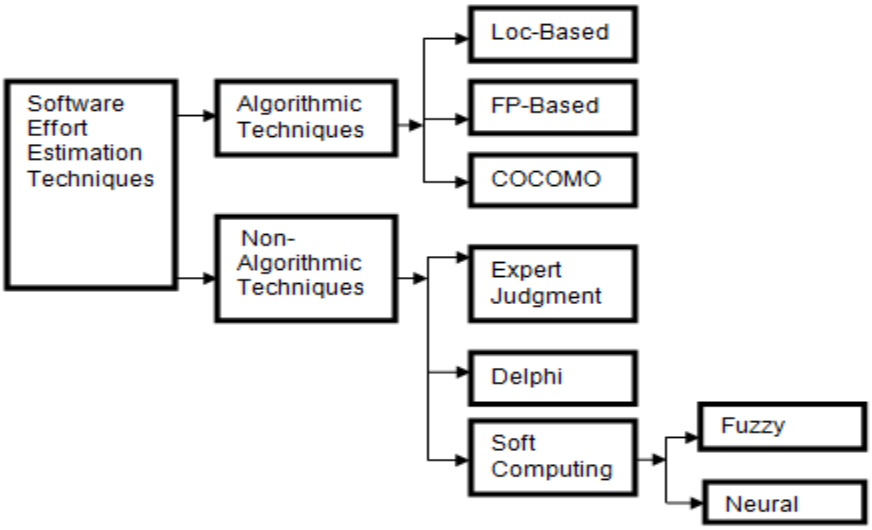


Figure 1.1 Software Effort Estimation Techniques

3.1. Algorithmic Techniques

The algorithmic techniques are based on some mathematical models and formulas which are used to calculate effort estimate. Algorithmic methods are using historical data and inputs like LOC (Lines of Code), number of functions to be preformed, various cost drivers on which estimation have done. Different algorithmic techniques are:

3.1.1. LOC Based: LOC based estimation is the simplest among all. This method measures the size of project simply by counting number of source instructions (excluding comment lines and header lines).but it is not helpful at the early stages of software development.

3.1.2. FP Based: Function Point metric is defined by Albrecht (1983) to measure the functionality of project. It overcomes the limitations of LOC based. Estimation is done by calculating following factors

- Number of User Inputs
- Number of User Outputs
- Number of Logic Files
- Number of Interfaces
- Number of Inquiries

By calculating complexity degree indicator we can evaluated UFP (Unadjusted Function Point) and Technical Complexity Factor (TCF) we can calculate FP

$$FP = UFP \times TCF \quad (1)$$

Function Points are free of languages, tools, construction methods. But FP is complex method.

3.1.3. COCOMO (Cost Constructive Model): The Constructive Cost Model (COCOMO) is an algorithmic software cost estimation model developed by Barry W. Boehm in 1981 (also called COCOMO'81). This model is based on a regression formula with parameters using historical data and current characteristics of projects. This is the first single-value, static model for effort estimation. This model is based on a nonlinear equation, estimating effort in PM (person months) and schedule in months.

$$\text{Effort} = a * (\text{KLOC})^b \quad (2)$$

Here KLOC is kilo lines of code and a, b are complexity factors. COCOMO is set of three models: Basic COCOMO, Intermediate COCOMO, and Detached COCOMO having three different modes each.

As this model have been experiencing difficulties in estimating new life cycle processes, reused projects, etc. So a new technique has been developed i.e. COCOMO II. COCOMO II is the successor of COCOMO'81 and developed in 1997 by Barry W. Boehm and his colleagues. It has 17 cost drivers, 5 scale drivers and one Line of Code (LOC). Now COCOMO model is useful for projects having code- reusability, off the shelf components etc. This model consists of application composition model, early design model, post architecture model having equations for effort estimation. The cost drivers of COCOMO II are rated on scale from very low to extra high as in the COCOMO 81[9]. The post architecture model is given as:

$$\text{Effort} = A \times [\text{Size}]^B \times \prod_{i=1}^{17} \text{Effort Multiplier} \quad (3)$$

Where B=1.01+0.01×j, A is Multiplicative Constant, Size is line of code (KLOC).

3.1.4. Putnam's Model: In 1978 Lawrence Putnam created a model for software estimation. Putnam's model is an empirical model and describes the time and effort required to finish the software project. Software equation for Putnam model:

$$S = E * \text{Effort}^{1/3} * t_d^{1/3} \quad (4)$$

Where td is time required for development, E is the environmental factor which effect the development, S is the size in KLOC and Effort is in Person months. SLIM is the name give to the tools of Putnam's model for cost estimation and manpower scheduling.

3.2. Non-Algorithmic Models

In 1990's non-algorithmic models was discovered and have been used in effort and cost estimation of projects.

3.2.1. Expert Judgment: Expert Judgment the most frequently applied technique for effort estimation of software projects. This model is used when there is limitation in finding and collecting accurate information. As it is based on experts, they must be experienced and there is probability of biasness in results. Process will be done by taking suggestions from the experts who had experience in similar projects.

3.2.2. Delphi: Delphi method basically combined the expert's opinion and reduces the biasness. In this technique a meeting is held with the experts and one coordinator. They will give their suggestions without discussing it with others. Coordinator will gather all the forms from experts and sums up them, process will repeat until result is approved by everyone.

3.2.3. Soft Computing Methods: In 1990's, soft computing techniques came up and are widely used in IT industry. Soft computing techniques are used in researches to handle uncertainty, imprecision due to its inherent nature. Generally used soft computing techniques are Fuzzy logics, Neural Networks for software effort prediction.

- **Fuzzy Logic**: Now the days, fuzzy logic has becoming the most active technology to deal with uncertainty and vagueness in software project estimation as it bear a resemblance to human's decision making ability. Fuzzy logics provide the linguistic representation of input and output data and also provide the knowledge based approach. In 1965, Lofti Zadeh developed a multi-valued set theory called fuzzy set theory [12].

- **Neural Networks**: In 1943 Warren McCulloch and Walter Pitts created a computational model for neural networks based on mathematics and algorithms [6]. Neural networks are group of interconnected nodes (neurons) which are capable of computation and machine learning. Neural networks are used in software engineering due to its learning ability from previous data. The Neural Network is initialized with random weights and gradually learns the relationships implicit in a training data set by adjusting its weights when presented with these data. An Artificial Neural network can be single layer or multilayer perceptron [10].

4. Obstacles in Software Effort Estimation

Software developer's most important and tough job is to estimate effort required for software development. Although estimation task is uncertain in nature, it depends upon its various factors which are not clear properly.

- Frequent changes in requirements
- Lack of proper tools for effort estimation
- Incorrect data gathering
- Lack of risk analysis and management
- Lack of and techniques effort estimation

5. Comparison of Software Effort Estimation Techniques

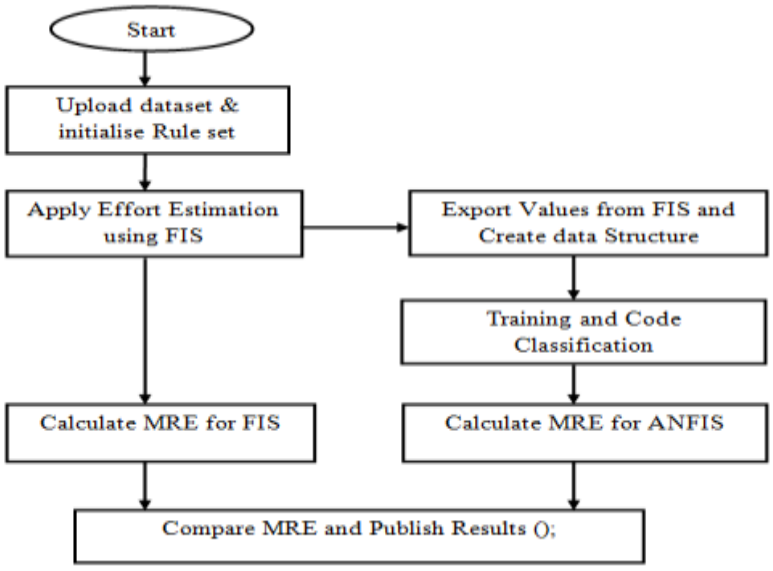
Table1. Comparison of Different Software Effort Estimation Techniques

Techniques	Type-Algo/Non-Algo	Input	Output	Biasness	Pros	Cons
LOC	Algorithmic	LOC	LOC	---	Simple and easy	Not useful in early stages
FP	Algorithmic	User inputs	FP	---	Language free, results are better than LOC	Mechanism is hard to do, quality of output is not so good
COCOMO	Algorithmic	KLOC	Effort(PM)	---	Clear results, common method	Much data is required
Putnam	Algorithmic	SLOC	Effort, Time	---	Results based on curve	Users input may affect results

Expert Judgment	Non-Algorithmic	Past project data	Suggestions from experts	Yes	Fast prediction, Adapt to especial projects	Success depends upon experts
Delphi	Non-Algorithmic	Previously used data	Refined ideas	Yes	Less biasness than expert judgment	Experts may be biased, usually is done incomplete
Fuzzy	Non-Algorithmic	Linguistic values	Fuzzified data	No	Flexibility, reduces uncertainty	Maintaining the meaningfulness is difficult
Neural	Non-Algorithmic	Neurons	Trained data	No	Inconsistent with unlike databases, provide training	No guideline, performance depends upon large dataset

6. Proposed Methodology

The proposed framework is evaluation of software effort using adaptive neural fuzzy model and fuzzy model. Environment used for proposed technique is MATLAB. The Proposed framework measures the effort required for software development using FIS (Fuzzy Inference System) and with ANFIS (Artificial Neural Fuzzy Inference System). Then make the comparison using MRE (Mean Relative Error) and MMRE (Magnitude of Mean Relative Error) measures and Figureure out the best technique for effort estimation having lowest value of MRE and MMRE.



Figureure 1. Proposed Methodology

Evaluation Criteria: There are a large number of evaluation criteria introduced for software effort estimation; among them we have selected MRE (Magnitude of Relative Error). The evaluation consists of comparison of the estimated effort (using proposed model) with actual effort

$$MRE = \frac{Actual\ effort - Estimated\ Effort}{Actual\ Effort}$$

The acceptable level of MRE is 0.25.

7. Conclusion and Future Scope

This paper presented an overview on various software effort estimation techniques presently available in IT industry. Software effort prediction is very important task in software development because the future of the project depends upon estimation task. The techniques discussed are LOC-Based estimation, FP-Based, COCOMO, Expert judgment, Delphi, Analogy-Based, Soft Computing techniques. We concluded that an earlier developed technique doesn't deal with uncertainty. In this paper we proposed a new model for software effort estimation that performs better in achieving high accuracy because fuzzy logic handles ambiguity, obscurity and neural networks deals with training dataset.

References

[1] S. G. Macdonell and A. R. Grey, "The viability of Fuzzy logic modeling in software development effort estimation: opinions and expectations of project managers", International Journal of Software Engineering and Knowledge Engineering, **(2005)**.
[2] J. N. V. R S. Kumar, T. G. Rao, Y. N. Babu, S. Chaitanya and K. Subrahmanyam, "A Novel Method for Software Effort Estimation Using Inverse Regression as firing Interval in fuzzy logic", IEEE, **(2011)**.
[3] E. Kocaguneli, "Exploiting the Essential Assumptions of Analogy-Based Effort Estimation", IEEE transactions on software engineering, vol. 38, no. 2, **(2012)** March/April.
[4] M. Jorgensen and K. Molokken-Ostvold, "Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method", IEEE transactions on software engineering, vol. 30, no. 12, **(2004)** December.
[5] M. Jorgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation", IEEE, **(2005)** May/June.
[6] I. Attarzadeh and S. H. Ow, "Proposing a New High Performance Model for Software cost Estimation", International Conference On Computer And Electrical Engineering IEEE, **(2009)** November.
[7] K. K. Aggarwal, Y. Singh, P. Chandra and M. Puri, "Bayesian Regularization in a Neural Network Model to Estimate Lines of Code Using Function Points", Journal of Computer Sciences, **(2005)**.
[8] M. Jorgensen and T. M. Gruschke, "The Impact of Lessons-Learned Sessions on Effort Estimation and Uncertainty Assessments", IEEE Transactions on Software Engineering, vol. 35, no. 3, **(2009)** May/June.
[9] N. Nan and D. E. Harter, "Impact of Budget and Schedule Pressure on Software Development Cycle Time and Effort", IEEE transactions on software engineering, vol. 35, no. 5, **(2009)** September/October.
[10] A. Idri and T. M. Khoshgoftaar and A. Abran, "Can Neural Networks be easily Interpreted in Software Cost Estimation", IEEE Transaction, **(2002)**.
[11] E. Pranylin and P. Latha, "Estimating Development Effort of Software Projects using ANFIS", International Journal of Computer Applications, **(2005)**.
[12] L. A. Zadeh, "Fuzzy Logic, Neural Networks, and Soft Computing", *Communication of ACM*, vol. 37, no. 3, **(1994)** March, pp.77-84.
[13] A. Sharma and D. S. Kushwaha, "Estimation of Software Development Effort from Requirements Based Complexity" Procedia Technology, vol. 4, **(2012)**.
[14] E. Khatibi and R. Ibrahim, "Efficient Indicators to Evaluate the Status of Software Development Effort Estimation inside the Organizations", International Journal of Managing Information Technology (IJMIT), vol. 4, no. 3, **(2012)** August.
[15] C. Diwaker and A. Dhiman, "Size and Effort Estimation Techniques for Software Development", IJSWS, **(2013)**.
[16] J. G. Borade and V. R. Khalkar, "Software Project Effort and Cost Estimation Techniques", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, Issue 8, **(2013)** August.

