IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-6, June 2019 E-ISSN: 2347-2693

# An Analysis of Software Reliability Estimation Using Fuzzy Logic Function With Cocomo II Model

**Ritu[1*], Kamna Solanki[2], Amita Dhankhar[3], Sandeep Dalal[4]**

[1,2,3] Dept. of Computer Science and Engineering, UIET, MDU, Rohtak, India
[4]Dept. of Computer Science and Engineering, MDU, Rohtak, India

*Corresponding Author: Roseritu5@gmail.com*

DOI: https://doi.org/10.26438/ijcse/v7i6.623626 | Available online at: www.ijcseonline.org

Accepted: 10/Jun/2019, Published: 30/Jun/2019

***Abstract***— Software cost estimation SCE is directly related to quality of software. The paper presents a hybrid approach that is an amalgamation of algorithmic (parametric models) and non-algorithmic (expert estimation) models. Algorithmic model uses COCOMO II while non algorithmic utilizes Neuro-Fuzzy technique that can be further used to estimate accuracy in irregular functions. For generalization of the model, Neuro-fuzzy membership functions have been used and simulated using mathematical tool MATLAB. The main objective of this research is to investigate the role of fuzzy logic technique in improving the effort estimation accuracy using COCOMO II by characterizing inputs parameters using Gaussian, trapezoidal and triangular membership functions and comparing their results. NASA (93) dataset is used in the evaluation of the proposed Fuzzy Logic COCOMO II. After analyzing the results it had been found that effort estimation using Gaussian member function yields better results for maximum criterions when compared with the other methods

***Keywords***— *COCOMO II, Estimation, Neuro-Fuzzy, Reliability, Membership function, Soft Computing, Software Effort Estimation, Gaussian Membership Function.*

## I. Introduction

Software development is becoming a necessity at a grandiose rate among all types and size of organizations. Software practitioners have become more and more apprehensive about their software cost and development. Varied software cost estimation models have been proposed over the past few years. However, they are unable to cope with the realistic realities of software engineering like handling imprecise information, dealing with uncertainty and many more [1–4]. The model proposed in this manuscript has been validated for its accuracy and estimation by using publicly available NASA93 software project data consisting of 20 projects with their values allocated to each cost driver. Results have been tabulated after comparing basic COCOMO and proposed fuzzy model. Results prove that the proposed model is more accurate and precise due to machine learning algorithm application that discovers knowledge and produces expertise results. Basic COCOMO model generates assumption-based results using historical data without applying any algorithms or sets.

The rest of this paper is categorized as follows: Sect. 2 reviews available literature and work done in field of software cost estimation. Section 3 describes easy and efficient way of estimating software cost parameters by using Costar software estimation tool based on COCOMO II model. It depicts how parameters like effort, schedule are estimated using pre-defined COCOMO equations.

An expert model that is combination of algorithmic approach namely COCOMO II and machine learning algorithm namely Neuro Fuzzy (NF) approach. The Size of Project and Output of sub models Neuro Fuzzy acts as input to COCOMO II model that is amalgamated with neuro fuzzy technique and produces final cost metric.

## II. SOFTWARE ESTIMATION

Software engineering is a type of engineering used for the development of software product. It requires top-most degree of analysis, hard work and the supervision of the two. With expanding size and complexity of the software product, its development has become a more difficult task which needs to be taken care off. Hence there will be no perception between the simple activity and complicated activity, both can be equally taken care for an efficient software product. Various difficulties which are being faced in the software development process are quality degradation, cost over-run and schedule over-run [1]. Apart from there difficulties faced

in the software development product there is another problem which is poor estimation. If the estimation is not accurate it will result into an ambiguity in the development process. Effective estimation is beneficial for appropriate project planning and control, although it is the most crucial and difficult task in the development process. During under-estimating project leads to quality degradation. However, over-estimating is even worse than under-estimating as more resources are allocated for the software development process without any scope. Precise planning of the project and tracing the on-going development process is the second fundamental task which certify for the success of the project. As estimates are accessible, next job is to allocate task to the individuals. There will be constant evaluation of the development process which is beneficial in determining status of the developed project. Trailing the process will provide possibility to the project manager to handle any unexpected situation during the software development process [2]. Proper management of any project will initiate with proper estimation. An effective estimation is the foundation for an effective software development process. Without an effective estimation both project planning and tracking of the development process are not possible. If the estimates are depressed then project management will employ more people in order to boost up the development process otherwise it will lead in poor results of the development process and employee disappointment [3].
Basic software estimations are:
Estimation of the cost
Estimation of the effort.
Estimation of the schedule.
Estimation of the size.

## III. Estimation Methods In Software Engineering

These are methods used for estimating cost, effort, schedule and size. This project is supported on COCOMO 2 model for cost and effort estimation. There is an integration of all the three models of the COCOMO 2 like application composition, early design and Post architecture for measuring various parameters. However, for size estimation, function point analysis (FPA) is used.

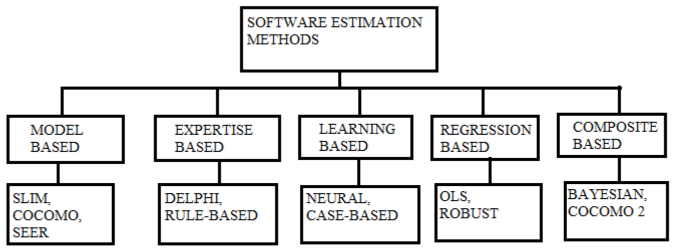


Figure 1: Software Estimation Methods

In past numerous estimation techniques like COCOMO (Constructive Cost Model), SEER (System Evaluation and Estimation of Resources) and SLIM (Software life cycle model) are developed which will opt mathematical model for estimation. The methodology of these techniques will consider related data of the project as their input and past data of the projects is used for marking of the models [1].

In case when past data of the projects in not available then expert knowledge is a criterion used for the estimation of the software development process. There are two techniques which will come under this class are Delphi and Rule-based. Delphi technique is purely based on verdict of the expert whereas rule-based technique is embraced from the Artificial Intelligence (AI) in which mixture of rules will work together to get the desired output [2].

A lot of research is done for the evolution of learning-based techniques used for the software estimation. First, Neural network which is outlined by three entities namely interconnected Structure, neurons and learning algorithm is one of the wide spread learning-based technique. Secondly, Case-based technique which itself a type leaning based technique in which database of the developed projects are preserved and when developing a new project, it expected development cost is compared with the database projects by which there will be prediction on the cost measure of the newly product which is going to be developed [3].

Regression methods like Ordinary least squares (OLS) method and robust method are used in estimation of the software product. Where robust method of regression will solve the most familiar problem of outliers in the field of software engineering data [4].

Model based techniques are commonly used in the industry due to their independency with regard to previous information and there is a perception that they will work on fixed parameters relevant for the model therefore, being used in the estimation of the software development product. In this methodology, values of different standard parameters are retrieved with reference to the project expectation. The estimation of the software is calculated using the equations defined in the model. Numerous tools are accessible in the market for the process of software estimation [5].

There are three models of COCOMO II explained as follows:
i. Application Composition – This model is suitable for those applications which can be fabricated by merging prepackaged outcomes but can't be developed by application developers. This model will utilize object points methodology for size valuation. It will measure size of any tool on the basis of record and 3GL elements. Example - GUI builders, Query browsers, Database managers etc.

ii. Early Design – This model can be utilized for application creators, structure consolidation and framework growth segments. It utilizes unadjusted function points for the size assessment.
iii. Post Architecture – This model has identical methodology as of COCOMO 81 and utilizes entirely 17 cost drivers for software valuation and utilizes unadjusted function point and source lines of code for size valuation
COCOMO 2
Various formula are used in the COCOMO 2 model are-
Estimated effort per months = A * (size) E where,
Value (constant set) = 2.94
Size = KLOC provided by the company
E = Estimated effort based on the 17 effort multipliers which are grouped in 4 category which are explained as follows –
i. Product Attributes – Software Reusability, Database, and complexity of the product.
ii. Computer Attributes – Execution time constant, Storage and Total predicted time to complete software.
iii. Personal Attributes – Tool cost, Programmer capability and Analyst capability.
iv. Project Attributes – Development time and Developed version of software tool cost.
Formula used in the software reliability estimation –
i. Unadjusted Function Points = Σ (External Input) + Σ (External Output) + Σ (External File) + Σ (External Inquiries) + Σ (Internal File).
ii. Degree of Influence= Σ General Application Characteristics [i] where i = 1 to 14.
iii. Technical Complexity Factor = (0.65 + 0.01 * Degree of Influence).
iv. Function Points = (Unadjusted Function Points * Technical Complexity Factor)/100.
v. Person Month = New Object Points / Developer experience and capability.

## IV. CHARACTERISTICS OF SOFTWARE RELIABILITY

Failure occurs primarily due to design faults:
For detecting the error, Design is modified for repairs to make it powerful against conditions
4.2 There is no wear-out phenomenon:
1) Software bugs occur without any warning.
2) While doing reforms, "Old" code can result in more number of failure rate because of errors.
3) External environment conditions generally not affect the reliability of the software.
4) Internal environment related conditions, such as inappropriate clock speeds or insufficient memory affect software reliability.
4.3 Reliability is not time dependent
1) Failure happens due to the error prone execution
2) The growth of the reliability is observed as errors are detected and corrected.

5. Software Reliability Activities
The software reliability process includes software development, operations, and maintenance. A software reliability process includes faults, defects, corrections, errors, updating, and expenses on the resource, such as manpower effort. Some of the
Reliability activities are as follows:
5.1 Construction: Generation of new documentation and code artifacts
5.2 Combination: It forces on reusability of old documents and code components with the new one.
5.3 Correction: Analyzing and removing document and code related defects by analyzing the test items.
5.4 Preparation: Generating of different test items.
5.5 Testing: Test cases are executed, to know the trigger points where failure occurs frequently.
5.6 Identification: Categorized each error or bug whether new or previously
5.7 Repair: Faults are removed which possibly introduces new faults for which regression testing is done.
5.8 Validation: Perform checks to make sure that repairs are effective and have not affected other parts of the software.
5.9 Retest: implementation of the cases to check for specified repair's completion. If it is incomplete, new test cases may be needed to repair them further.

## V. RESULTS AND DISCUSSION

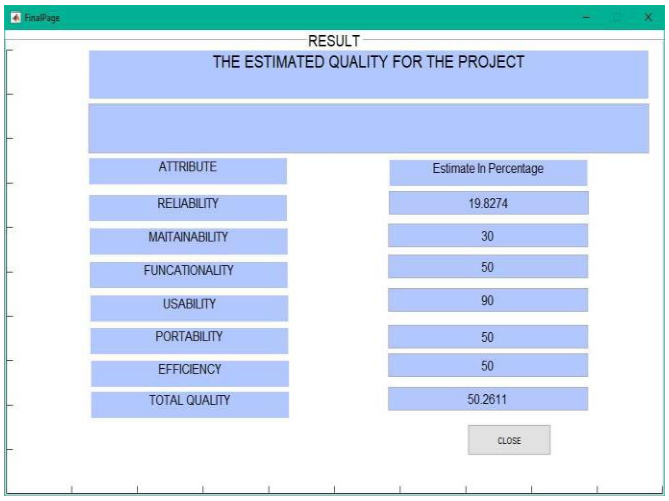
This paper will consider three different company with their values like line of code, function points and actual efforts and by applying formulae, predicted value of the estimation is achieved. This company data is collected from google open source and perform operations with the help of COCOMO 2 model and Fuzzy tools set.

*Example set:*

| COMPANY NAME | C1 | C2 | C3 |
|---|---|---|---|
| LINE OF CODE | 14000 | 12000 | 11000 |
| FUNCTION POINTS | 218 | 187 | 185 |
| ACTUAL EFFORTS | 38 | 32 | 30 |

| COMPANY NAME | COCOMO PREDICTED ESTIMATION |
|---|---|
| C1 | 33.54 |
| C2 | 27.89 |
| C3 | 18.32 |

Final Result of the estimates quality of the project, here total quality will be around 50.26 % performed on various reliability factors.



RESULT

THE ESTIMATED QUALITY FOR THE PROJECT

| ATTRIBUTE | Estimate In Percentage |
|---|---|
| RELIABILITY | 19.8274 |
| MAITAINABILITY | 30 |
| FUNCATIONALITY | 50 |
| USABILITY | 90 |
| PORTABILITY | 50 |
| EFFICIENCY | 50 |
| TOTAL QUALITY | 50.2611 |

## VI. CONCUSION AND FUTURE WORK

Conclusion

To ensure the best quality of the software product, software testing is an integral part of the software development process. The main objective of this thesis was to design and implement a testing process which is executed to check whether the product is efficient or not. During the development process, developer will perform various operations which will check the reliability of the software based on following parameters like efficiency, maintainability, reliability, portability and usability. If the developer finds that the product is not meeting the requirements for the efficient system it will make changes to the code with the help of COCOMO 2 model and neuro fuzzy set. Various operations are performed by passing data set values to check out the reliability in a pre- developed stage.

Parameters like function points, KLOC and actual efforts of various companies are compared and reliability estimation is predicted. This illustrates that 17 cost drivers are used along with various programming languages with their UFP values. Data sets are provided by the owner of the company on which tester will perform operations to decide the reliability estimation. Finally, after performing data set operations on different parameters it will provide result in measure of the overall quality of the software product.

FUTURE WORK

Another attempt can be added to future implementation for representing the results more rigorously. This work can be further enhanced by including more testing tools for comparison so that, it could find more suitable testing tools for testing the software testing. Further, some different metrics can be used for performance evaluation so that results could be more realistic and reliable. New testing issues can arise which can be taken care off for an efficient software product.

## REFERENCES

[1] J. Gaffney (Jnr) and E. John, "Software Function Source Lines of Code and Development Effort Prediction: A Software Science Validation", IEEE Transactions on Software Engineering, vol. 9, issue-6, pp. 639-647, 1983.

[2] R. Rombach and H. Dieter, "The TAME Project: Towards Improvements Oriented Software Environments", IEEE Transactions on Software Engineering, vol. 14, issue-6, pp. 758-773, 1988.

[3] Symons and Charles R., "Function Point Analysis: Difficulties and Improvements", IEEE Transactions on Software Engineering, vol. 14, issue-1, pp. 2-10, 1988.

[4] Vahid, Khatibi, Dayang and N. A. Jawawi, "Software Cost Estimation Methods: A Review", Journal of Emerging Trends in Computing and Information Sciences, vol. 2, issue-1, pp. 21-29, 2010.

[5] Randy K. Smith, Joanne E. Hale and Allen S. Parrish, "An Empirical Study Using Task Assignment Patterns to Improve the Accuracy of Software Effort Estimation", IEEE Transactions on Software Engineering, vol. 27, issue-3, pp. 264-267, 2011.

[6] Shubhangi Mahesh Potdar, Manimala Puri and Mahesh P. Potdar, "Literature Survey on Algorithmic Methods for Software Development Cost Estimation", International Journal of Computer Technology & Applications, vol. 5, issue-1, ISSN: 2229-6093, pp. 183-188, 2014.

[7] Chemuturi K.M, "Software Estimation Best Practices, Tools and Techniques: A Complete Guide for Software Project Estimators", J. Ross Publishing Inc, pp. 49-65, 2009.

[8] Magne Jorgensen, "Practical Guidelines for Expert-Judgment-Based Software Effort Estimation", Simula Research Laboratory, IEEE, pp.57-63, 2005.

[9] Vahid Khatibi, Dayang N. A. Jawawi "Software Cost Estimation Methods: Review", Journal of Emerging Trends in Computing and Information Sciences, vol. 2, issue- 1, 2011.

[10] Matson J., Barrett B. and Mellichamp J., "Software Development Cost Estimation Using Function Points", IEEE Transactions on Software Engineering, vol. 20, issue-4, pp. 275-287,1994.

[11] M. Shepperd and C. Schofield, "Estimating Software Project Effort Using Analogies", IEEE Transaction on software engineering, vol. 23, pp. 736-743, 1997.

[12] C. S. Reddy and K. Raju, "A Concise Neural Network Model for Estimating Software Effort", International Journal of Recent Trends in Engineering, vol. 1, pp. 188-193, 2009.

[13] F. J. Heemstra, "Software cost estimation, Information and Software Technology", vol. 34, pp. 627-639, 1992.

[14] L. Lederer and J. Prasad, "Causes of Inaccurate Software Development Cost Estimates", Journal of Systems and Software, vol. 31, pp. 125-134, 1995.

[15] Chetan Nagar, "Software efforts estimation using Use Case Point approach by increasing technical complexity and experience factors", International Journal of Computer Sciences and Engineering, ISSN:0975-3397, vol.3, issue-10, pp. 3337-3345, 2011.

[16] N. Karunanitthi, D. Whitley and Y.K Malaiya, "Using Neural Network in Reliability Prediction", IEEE Transaction on software engineering, vol. 9, issue-4, pp. 53-59, 1992.

[17] T. J. Mc Cabe, "A complexity measure", IEEE Transaction on software engineering vol. 2, issue-4, pp. 308-320, 1976.

[18] A.J. Albrecht and J. E. Gaffney, "Software function, source lines of code and development effort prediction: A software science validation", IEEE Transaction on Software Engineering, vol. 9, issue-6, pp. 639-647, 1983.