Int. j. inf. tecnol. (June 2018) 10(2):181–187
https://doi.org/10.1007/s41870-018-0083-6

ORIGINAL RESEARCH

Neuro fuzzy—COCOMO II model for software cost estimation

Ishleen Kaur[1] · Gagandeep Singh Narula[2] · Ritika Wason[3] · Vishal Jain[3] · Anupam Baliyan[3]

Received: 19 September 2017 / Accepted: 10 January 2018 / Published online: 18 January 2018

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. Also, the proposed model has been validated with traditional COCOMO model (COCOMO 81) by using NASA software project data. The experimental results suggest that the proposed model gives better SCE as compared to its traditional counterpart.

Keywords COCOMO model · Neuro-fuzzy approach · Costar · Soft computing · Effort estimation

✉ Gagandeep Singh Narula
gagan.narula87@gmail.com

Ishleen Kaur
kaur.ishleen20@gmail.com

Ritika Wason
rit_2282@yahoo.co.in

Vishal Jain
vishaljain83@ymail.com

Anupam Baliyan
anupam_hod1976@yahoo.co.in

[1] Department of Computer Engineering, Ambedkar Institute of Technology, Delhi, India

[2] CSIR-NISTADS, New Delhi, India

[3] Bharati Vidyapeeth's Institute of Computer Applications (BVICAM), New Delhi, India

1 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. Section 4 details proposed neuro fuzzy model. Section 5 provides validation of given model followed by conclusion and references.

2 Literature review

Whenever an estimate is generated, all requirements and business rules are verified with previous estimations taken into account. As per Standish report [5], cost estimation survey states that 30% of projects never complete, average project exceeds cost by 90%, schedule by 67% and 15% of projects do not deliver anything.

Cai et al. [6] replaced probabilistic software models (PSRM) by fuzzy software reliability modeling. It is said that world is full of fuzziness and partial knowledge beliefs that can only be handled by fuzzy model. Karunanithi et al. [7] proposed connectionist models to predict software reliability and cost. Sitte [8] analyzed neural networks to conclude that these networks are simpler to use and prove best in nonlinear models. Tian and Noore [9] worked on evolutionary neural network approach founded on multi input and single output design. This approach performs better in case of failure time prediction. Su and Huang [10] suggested working and development of neural network approach to build dynamic weighted combinational model (DWCM) that predicts software maintainability. Madsen [11] designed soft computing framework to apply intelligent techniques for monitoring software reliability engineering. Kumar et al. [12] designed a framework for measuring software complexity, adaptability, security and usability but the model failed to measure interaction among various metrics. Wason et al. [13] proposed automata-based model to control runtime software reliability.

A plethora of studies led by researchers in context of measuring software quality to estimate cost of software have been conducted. A comparative study on algorithms used in NN model was conducted by Aggarwal et al. [14]. It includes quasi network method, levenberg model etc. A fuzzy model has also been suggested by Aggarwal et al. [15] for estimating software maintainability and usability. It is based on fuzzy logic approach that identifies error prone software components. Yang et al. [16] devised a model on fuzzy neural network in order to forecast software quality. The model produces empirical results by taking primary data and knowledge gained from human experiences into consideration. Despite, such large number of significant research to improve the process of software development and design, gap within estimated and predicted software costs does not seem to reduce. To achieve the same we propose an innovative, hybrid approach using COCOMO II to estimate software cost factors and NFNN model to calculate cost. The details of the proposed model are discussed in the next sections.

3 Using costar to estimate cost in COCOMO'81 and COCOMO II models

The traditional COCOMO'81 model uses metric in terms of Delivered Source Instructions (DSI), which is similar to SLOC while COCOMO II model defines metric in terms of SLOC. The major difference between both metrics is that SLOC consist of various physical lines.

3.1 Scale drivers

COCOMO II considers five scale drivers as follows:

- Precedentedness.
- Development flexibility.
- Architecture/risk resolution.
- Team cohesion.
- Process maturity.

The above Scale Drivers replace the COCOMO'81 Development Mode (Organic, Semidetached, or Embedded).

3.2 Cost drivers

COCOMO II has 22 cost drivers consisting of 17 Effort multipliers and five scalar factors. For example, if our plan needs to build up software on airline reservation system in that case we have to place the Required Software Reliability (RELY) expenditure driver to very high.

A. Results produced by costar

Figures 1 and 2 below describe the process of cost estimation factors—effort and schedule in both models. They also display that cost of software project is less in COCOMO II as compared to basic COCOMO model (COCOMO'81).

4 Proposed neuro-fuzzy model

COCOMO II model has 22 cost estimators with five scale factors (SF) and 17 effort multipliers (EM) that acts as input to proposed model. The intended outcome is effort estimated using COCOMO II post architectural model equation as [17]:

$$Effort = A \times (Size)^{B+0.01\times\sum_{i=1}^{S} SF_i} \times \prod_{i=1}^{17} EM_i,$$

where A, B are calibration constants.

Rating of the cost estimators can be continuous or linguistic terms (very low, low, nominal, high, very high and extra high). Every rating value of cost estimator is related to value used in COCOMO model because each cost

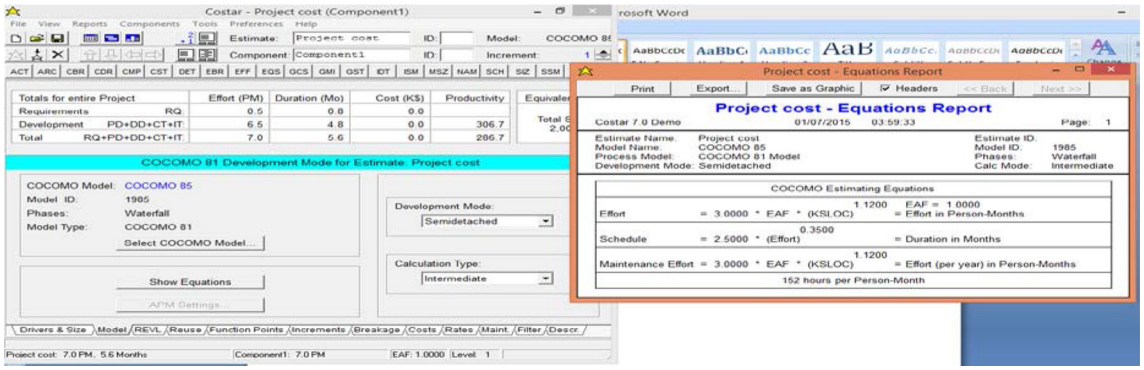

Fig. 1 COCOMO 81 intermediate results in semi-detached mode with SLOC 2000 as total size

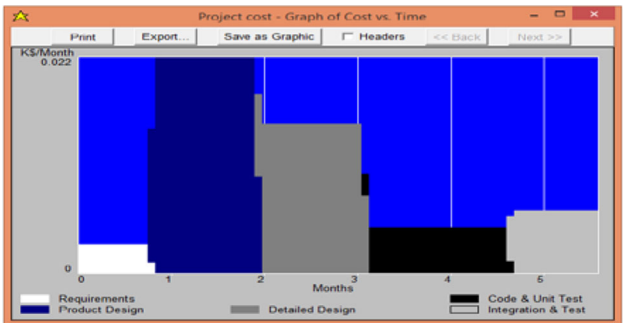

Fig. 2 Cost in COCOMO 81 = 0.022$

estimator represents one factor that is linked to effort. The proposed model is divided into two parts:

a. Set of Neuro fuzzy sub models (NF's).
b. One cost driver to each sub model.

Rating value of cost driver acts as input to each NF and output of each NF is their effort multiplier corresponding to each cost driver. Each NF sub model translates linguistic terms of cost drivers into qualitative multipliervalues by using fuzzy sets. They are described as membership functions. Each NF is represented by ANFIS (Adaptive Neuro fuzzy inference system) [18].

ANFIS technique is a combination of Neural Network NN and Fuzzy Inference System FIS that considers input nodes as fuzzy sets and provides learning process for fuzzy modeling to deal with various data sets. It calculates membership function attributes that apply fuzzy inference system components (membership editor, rule editor) to follow the provided input or output data. It permits fuzzy system to adapt and study latest information to reach at some conclusion.

FIS is based on three strategies-

- Outlier identification: it includes model type, rules to be applied and interval values.
- Parameter estimation: it deals with association rules. Represented as: "If Then" rules. If is called Antecedent, Then is called consequent [19]. They are used to show relationship among various data items.
- Validating model: the model is validated on MATLAB simulation tool to produce membership functions (Fig. 3).

The complete working of the NFNN COCOMO II model is depicted below in Fig. 4.

A. Layout of proposed model.

B. Results of ANFIS using simulation tool—MATLAB. (Figs. 5, 6).

4.1 Validation by NASA project data

This section validates the modeling performance and accuracy of the proposed neuro fuzzy COCOMO II model. NASA93 publicly available data from original COCOMO'81 database [20] was used for validation purpose. A comparative analysis was made between proposed model and COCOMO'81 model (traditional model) because COCOMO II does not have trained data present and also cost drivers of COCOMO'81 are compatible for validation. The dataset consists of two independent variables Lines of Code LoC and ME and one dependent variable Effort. The dataset of NASA93 project consists of 20 projects out of which few are outlined in table below.

The validation results of experiment were assessed using the concept of Mean Magnitude Relative Error (MMRE) to estimate accuracy of proposed model (Table 1). MMRE is calibrated using following equation:

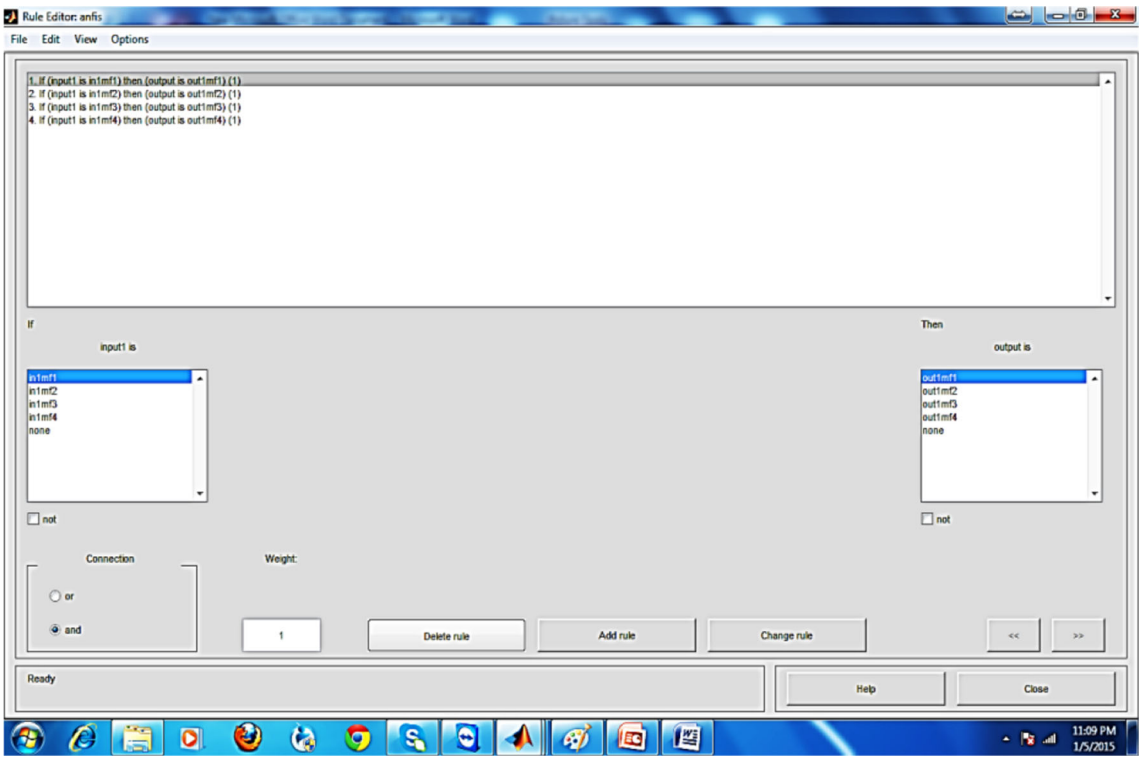

Fig. 3 Fuzzy rules IF THEN applied on data set

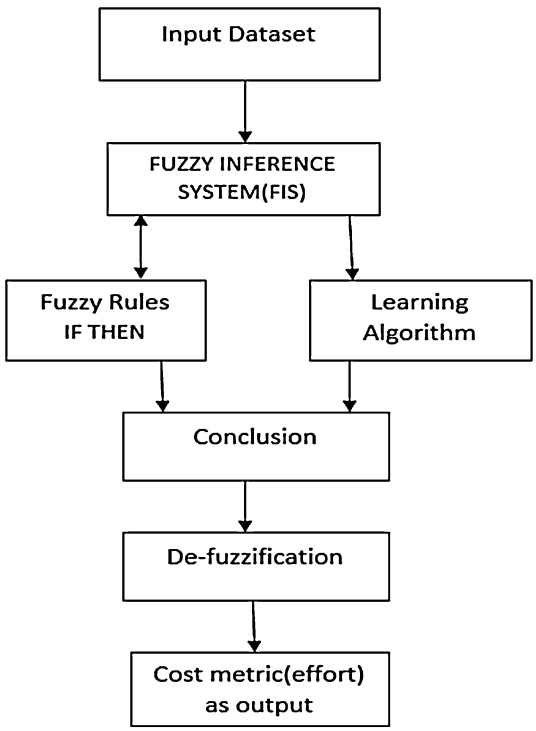

Fig. 4 NFNN COCOMO II model framework

$$MMRE = O\,(I = 1\ to\ n)$$
$$[(Estimated\ effort - Actual\ effort)\ /\ Actual\ effort].$$

The results obtained after calculation are reported in Table 2 below.

4.2 Conclusions and future work

The paper proposes 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. The validation of proposed model is done by taking data from NASA project of original database under consideration and results are compared with COCOMO'81 model.

As future enhancement, the same neuro fuzzy technique can be applied to object oriented metrics to predict software quality attributes. This approach can further be extended to predict quality of tools like SLIM (Putnam), CA estimates and others.

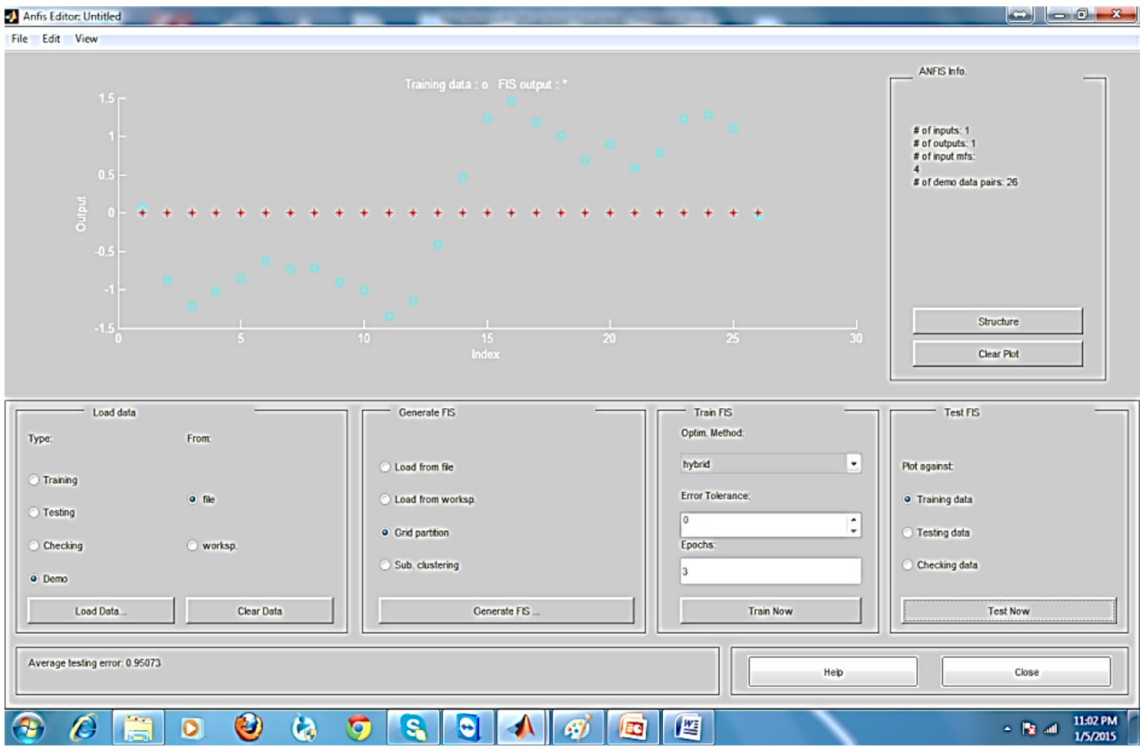

Fig. 5 FIS output produced on trained set of data (fuzzification)

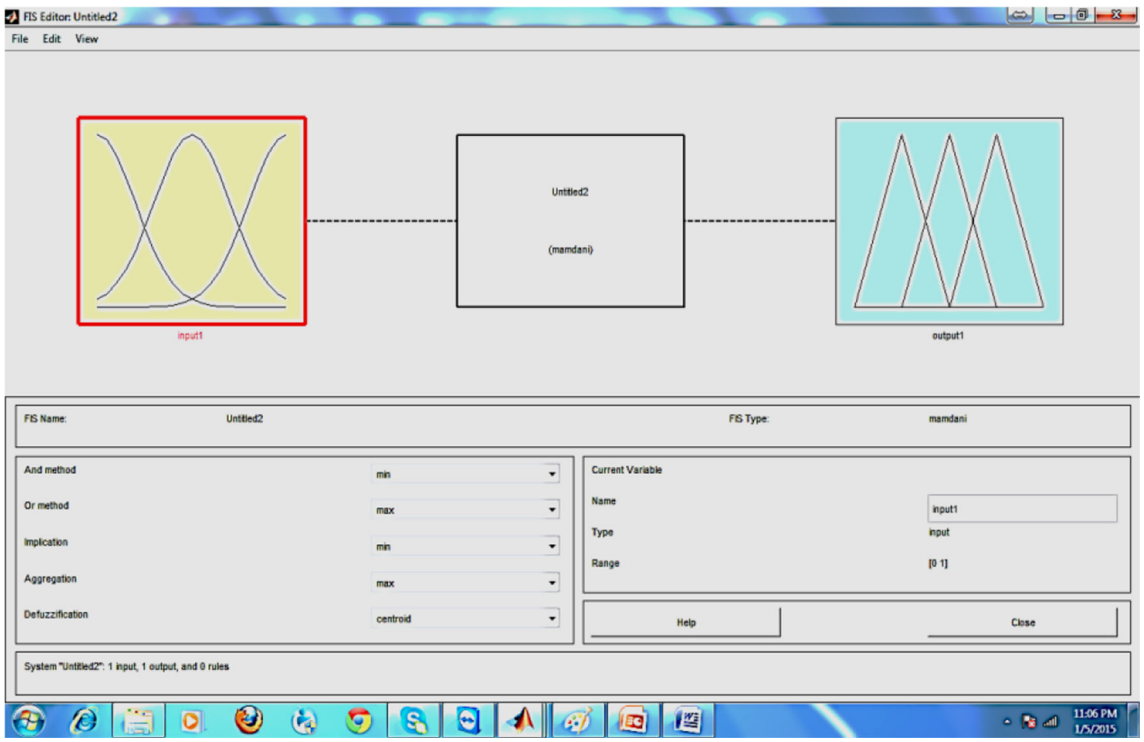

Fig. 6 Output produced after defuzzyfication of Fig. 5

Table 1 NASA software project dataset [19]

Project no.	Lines of code (KLOC)	Methodology (ME)	Actual effort (P-M)
1	91.1	31	117.2
2	52.5	19	87.5
3	32.9	36	40.1
4	2.9	25	6
5	77.4	33	95.7
6	100	33	140
7	12.5	27	23.9

Table 2 Effort estimation

S. no.	Actual effort	COCOMO'81 estimated	Error (%)	Proposed model estimated	Error (%)
1	117.2	185.5	58.3	134.8	15.02
2	87.5	119	36	102	16.57
3	40.1	75	87.03	53	32.17
4	6	5	− 16.6	8	33.3
5	95.7	119	24.3	101	5.53
6	140	173	23.57	149	6.43
7	24.1	27	12.03	23	4.56

Appendix A

Costar cost drivers

	Personnel factors	COCOMO II post architecture	COCOMO 81, COCOMO 85
ACAP	Analyst capability	Yes	Yes
APEX AEXP	Applications experience	Yes	Yes
PCAP	Programmer capability	Yes	Yes
LEXP	Programming language experience		Yes
VEXP	Virtual machine experience		Yes
PERS	Personnel capability		
LTEX	Language and tool experience	Yes	
Product factors			
RELY	Required software reliability	Yes	Yes
DATA	Database size	Yes	Yes
CPLX	Software product complexity	Yes	Yes
RUSE	Required reusability	Yes	
Platform factors			
TIME	Execution time constraint	Yes	Yes
STOR	Main storage constraint	Yes	Yes
VIRT	Virtual machine volatility		Yes
PVOL	Platform volatility	Yes	
PDIF	Platform difficulty		
Project factors			
TOOL	Use of software tools	Yes	Yes
FCIL	Facilities		
RVOL	Requirements volatility		

References

1. Boehm B et al (2000) Software cost estimation with COCOMO II. Prentice Hall PTR, New Jersey, p 07458
2. Boehm B (1981) Software engineering economics. Prentice- Hall Inc, New Jersey, p 07632
3. MacDonell S, Gray A (1997) A comparison of modeling techniques for software development effort prediction. In: proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems, Springer-Verlag. 869–872
4. Chulani S (1999) Bayesian analysis of software cost and quality models, Ph.D. Dissertation, University of Southern California
5. Standish Report-1998, 1999, 2000, Chaos Report
6. Cai KY, Wen CY, Zhang ML (1991) A critical review on software reliability modeling. Reliab Engi Syst Saf 32(3):357–371

7. Karunanithi N, Whitley D, Maliya YK (1992) Prediction of software reliability using connectionist models. IEEE Trans Softw Eng 18:563–574
8. Sitte R (1999) Comparison of software-reliability-growth predictions: neural networks vs parametric recalibration. IEEE Trans Reliab 48(3):285–291
9. Tian L, Noore A (2005) Evolutionary neural network modeling for software cumulative failure time prediction. Reliab Eng Syst Saf 87:45–51
10. Su Y-S, Huang C-Y (2006) Neural-network-based approaches for software reliability estimation using dynamic weighted combinational models. J Syst Softw 80(4):606–615
11. Madsen H, Thyregod P, Burtschy B, Albeanu G, Popentiu F (2006) On using soft computing techniques in software reliability engineering. Int J Reliab Qual Saf Eng 13(1):61–72
12. Kumar R, Grover PS, Kumar A (2010) A fuzzy logic approach to measure complexity of generic aspect-oriented systems. J Object Technol 9(3):43–57
13. Wason R, Ahmed P, Rafiq MQ (2013) A tool for runtime reliability estimation and control using automata-based software reliability model. In Proceedings of the second WSEAS International Conference on Computers, Digital Communications and Computing (ICDCC 2013). 51–56
14. Aggarwal KK, Singh Y, Chandra P, Puri M (2005) Evaluation of various training algorithm in a neural network model for software engineering application. ACMSIGSOFT Softw Eng Notes 30(4):1–4
15. Aggarwal KK, Singh Y, Chandra P, Puri M (2005) Measurement of software maintainability using a fuzzy model. J Comp Sci 1(4):538–542
16. Yang B, Yao L, Huang H (2007) Early software quality prediction based on a fuzzy neural network model, Third International Conference on Natural Computation (ICNC 2007), IEEE 1: 760–764
17. Center for Software Engineering (1997) "COCOMO II Model Definition Manual," Computer Science Department, University of Southern California, Los Angeles, Ca. 90089. http://csse.usc.edu/csse/research/cocomoii/cocomo2000.0/cii_modelman2000.0.pdf. Accessed 13 Apr 2000
18. Khoshgaftaar TM, Allen ED, Hudepohl JP, Aid SJ (1997) Application of neural networks to software quality modeling of a very large telecommunications system. IEEE Trans Neural Netw 8(4):902–909
19. Narula Gagandeep Singh, Jain Vishal (2013) Implementation of data mining in online shopping system using TANAGRA tool. Int J Comp Sci Eng 2(1):47–58 **(ISSN 2278–9960)**
20. Sheta AF (2006) Estimation of the COCOMO model parameters using genetic algorithms for NASA software project. J Comp Sci 2(2):118–123
21. Shivakumar et al (2016) A neuro fuzzy algorithm to compute software effort estimation. Glob J Comp Sci Technol C Soft Data Eng 16(1), ISSN 0975-4172
22. Saka et al (2016) "Modeling software effort estimation using hybrid PSO-ANFIS", Intelligent Technology and its applications (ISTIA) IEEE. 28–30. https://doi.org/10.1109/isitia.2016.7828661