Improving the Accuracy of COCOMO's Effort Estimation Based on Neural Networks and Fuzzy Logic Model

Riyanarto Sarno[1], Johannes Sidabutar[2], and Sarwosri[3]
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
[1]riyanarto@if.its.ac.id, [2]jo.chris93@gmail.com, [3]sri@its-sby.edu

***Abstract*— Constructive Cost Model II (COCOMO II) is investigated as the most popular model for software cost estimation. COCOMO II depends on several variables or Cost Drivers (CD). This research investigates the role of Effort Multiplier (EM) and Line of Code (LOC) to improve the accuracy of cost estimation. Fuzzy Logic has been implemented to the COCOMO II to represent the EM. Furthermore, in order to produce better estimation, this research uses Gaussian Membership Function to redesign the Effort Multiplier by studying the behavior of COCOMO II. This research also applies Neural Network (NN) approach to increase the accuracy of software effort estimation by training the software development datasets. The result is proposed model gives contribution to decrease error significantly.**

***Keywords*—** COCOMO, neural networks, fuzzy, software cost estimation, cost driver, Gaussian membership.

I. INTRODUCTION

Software cost estimation is one of the major issues in software project management and development. The accuracy of software cost important is important to guide software companies for making good management to develop software. Moreover, good management of software development could estimates the cost and resources of software precisely. It could handle both overestimates and underestimates of software effort and cost. This also could manage the quick configuration of application [15]. This accuracy derives from some variables or cost drivers. So, obtaining an accurate software cost estimation needs accurate cost drivers too.

In last few decades, software industry has been introduced with some model that estimate software cost. Among all the software cost estimation model, Constructive Cost Model (COCOMO) is the most famous to calculate the software cost. COCOMO was published by Barry Boehm in 1981 [1]. The model was observed from a datasets that consisted of 63 data points. Each data point was divided to 16 variables. COCOMO divided cost driver into 3 aspects such as Effort Multiplier (EM), Line of Code (LOC) and Scale Factors (SF). All of the cost drivers will be calculated with an equation to produce the number of effort in person-months (PM). In 2000, Barry Boehm [2] introduced COCOMO II which has been proved more accurate with some improvement in several cost drivers.

The most common method to improve software cost and effort estimation is Artificial Neural Network (ANN). This research specifically uses feed forward to get the estimated effort, back propagation as learning algorithm and sigmoid activation. Feed forward multilayer perceptron sends inputs to hidden layers. Each connection from input to hidden layer has weight. From the hidden layer, inputs will be activated using sigmoid activation and pass the result to the output. When output receives the result, it will count the error by comparing output with the actual effort. Back propagation algorithm is applied to train dataset by updating every weights that involve in the architecture. By updating the weights, it seems that output will be more accurate than without back propagation learning algorithm. This because the weights have been improved based on the errors.

The second method that will be implemented in this research is Fuzzy Logic Model. Fuzzy logic is a decision making process based on several rules which aim to solve imprecision and approximation problems. This model is an interpretation of two exceptional human abilities. First, human has ability to generate rational decisions based on imprecision, uncertainty, and incompleteness of information. Second, human has ability to execute physical and mental tasks without any computational logic [3, 4]. There are several categories of Fuzzy Membership Functions such as Triangular, Trapezoidal and Gaussian Membership Functions. In other research [5], the triangular and trapezoidal membership functions have been used to create a more accurate estimation of COCOMO's effort. So, this paper uses Gaussian Membership Functions (GMF) to achieve better accuracy by attaining more gentle transition in the membership function. By applying GMF in COCOMO's cost drivers, the result is expected nearer to the real effort.

978-1-5090-0096-8/15/$31.00 © 2015 IEEE

The remaining paper is classified as follows: section 2 presents the COCOMO and ideas that have been implemented in this paper. Section 3 defines several approaches that has been done for software cost estimation. Section 4 presents the proposed methodology which is used feed forward feed forward to get the estimated effort, back propagation as learning algorithm and sigmoid activation. Section 5 describes experimental results based on proposed methodology. Last section concludes proposed model could be implemented to estimate software cost more accurate than current model.

II. LITERATURE REVIEW

A. Software Cost Estimation

Software cost estimation is related to techniques that are proposed to estimate the number of time, budget, effort, and staff needed to create a software [6]. Cost estimation would be very useful if it is observed at an early stage during a project by predicting the amount of effort, time and staff. However, estimating at the beginning of software development are such a complicated thing to do because of some uncertainties during software development projects which affect amount of effort that is used for software development [7]. There are several common method to improve software cost estimation such as fuzzy, neural network and analogy method [16].

B. COCOMO Model

The COCOMO model is a technique that is designed to estimate software effort by collecting data from vast application projects. This information was considered to observe equation that closed to actual effort. This equation consists of several factors that affect the estimation. All of the estimated effort are expressed as Person Months (PM) [7]. Barry Boehm introduced an algorithmic cost model in 1981 which is known as the COCOMO [1]. This model was observed from 63 application projects. The model divides into Basic, Intermediate and Detailed sub models. [8].

The COCOMO II consists of several application aspects such as: 17 Effort Multipliers, 5 Scale Factors, Software Size. All of these aspects are combined to produce effort estimation that are applied in the Post Architecture Model of the COCOMO II. The effort multipliers grouped into four types:

1) Product attributes
   a) Required software reliability (RELY)
   b) Database size (DATA)
   c) Product complexity (CPLX)
   d) Developed for Reusability (RUSE)
   e) Documentation Match to Life-Cycle Needs (DOCU)
2) Computer attributes
   a) Execution Time Constraint (TIME)
   b) Main Storage Constraint (STOR)
   c) Platform Volatility (PVOL)
3) Personnel attributes
   a) Analyst Capability (ACAP)
   b) Programmer Capability (PCAP)
   c) Personnel Continuity (PCON)
   d) Application Experience (APEX)
   e) Platform Experience (PLEX)
   f) Language and Tool Experience (LTEX)
4) Project attributes
   a) Use of Software Tools (TOOL)
   b) Multisite Development (SITE)
   c) Required Development Schedule (SCED)

COCOMO II formula for the process is given by:

$$\text{Effort (PM)} = A * \text{Size}^{B*} \prod_{i=1}^{17} \text{Effort Multiplier}_i \quad (1)$$

Where A = 2, 94; B = 0, 91

More details about the Model can be found in [2].

C. Artificial Neural Network

Artificial Neural Network (ANN) is an interconnected group of nodes which is inspired by the architecture of human neural networks. Commonly ANN is divided in three. The connection between the nodes are weighted and the values of these weights determine the function of the network. Each node calculates a weight from its inputs and delivers to output by using a certain activation function. This output will be used to train the network. This process will continue until a certain threshold has been reached [13].

There are many different algorithms that apply ANN model. The most frequent algorithm is Feed-forward Multilayer Perceptrons. This algorithm uses back-propagation algorithm to train the architecture by calculating error of each data. The ANN is initialized with random weights for each connection between nodes. Then, back-propagation learning algorithm gradually adjusts weights which use some techniques based on the error.

D. Fuzzy Logic

In 1965, Fuzzy logic (FL) was first introduced by Lofti A. Zadeh. It was inspired from human brain's unique technique of processing words. Zadeh was motivated to solve imprecision problem for measurement process. He explains that ''*As complexity rises, precise statements lose meaning and meaningful statements lose precision*" [9]. This model combines both quantitative and qualitative information to solve imprecision. The model uses fuzzy sets which have degrees of membership [10]. There are several forms of membership function such as triangular, trapezoidal, Gaussian etc.

Fuzzy Logic System implements fuzzy approach such as fuzzy sets and fuzzy logic. There are several widely known fuzzy logic system. They are grouped into three categories [11]: pure fuzzy logic systems, Takagi and Sugeno's fuzzy system, and fuzzy logic system with fuzzifier and defuzzifier. Fuzzifier and deffuzifier is the most famous fuzzy logic systems. The fuzzifier convert crisp inputs into fuzzy sets and the defuzzifier convert fuzzy sets into crisp outputs. Fig 1 shows form of logic system that is proposed by Mamdani [12].

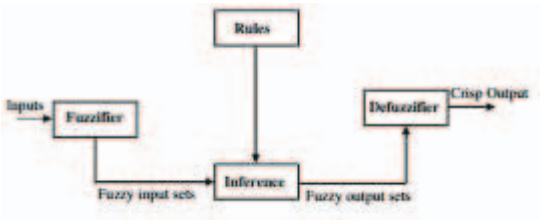

Figure 1. Fuzzy logic system with fuzzifier and deffuzifier

III. RELATED WORK

Reddy and Raju [7] tried to improve accuracy by fuzzifying COCOMO cost drivers. They studied the behavior of COCOMO cost drivers, and then proposed to characterize the use of Gaussian Membership Function (GMF) for cost drivers to represent the linguistic values. Their experimental result shows a more accurate effort estimation rather than COCOMO and Trapezoidal Membership Function (TMF).

Kaushik and Soni [8] investigated the use of back-propagation neural networks for estimating software cost. They improved the accuracy of COCOMO by training the ANN for every dataset. The test result from the trained neural network was compared to actual effort. The result shows that a more accurate effort estimation could be reached by their proposed model.

IV. PROPOSED WORK

First of all, this research tries to learn the every effort multipliers in COCOMO. Each effort multiplier (EM) uses linguistic values to represent the character of each EM. From that linguistic values which is ranged from Very Low to Extra High, this research divides EM to two categories such as qualitative EM and quantitative EM. The qualitative effort multipliers are RELY, CPLX, DOCU, RUSE, PVOL, PCON, TOOL, SITE and the other effort multipliers are quantitative. Fuzzy Model is used to redesign the quantitative EM because quantitative EM description can be translated into Fuzzy Logic. For example, TIME effort multiplier has range from nominal to extra high. The difference for every levels is percentage use of available execution time. This research uses Gaussian Membership Function (GMF) for Fuzzy Logic. GMF creates a smoother transition from one level to another level. Fuzzy Logic was applied using fuzzy logic tool box in MATLAB software. There were several types of Fuzzy Logic such as GMF, Triangular Membership Function, Trapezoidal Membership Function and other types in the tool box. The tool box is named as Fuzzy Inference System (FIS) Editor. This FIS Editor GUI helps us to create input and output with any range and any number of membership function that we need. Also, FIS Editor allows us to create rules from input to output. In this research, the threshold would be similar to the following:

R_1 : IF Input DATA is low THEN Output data is decreased
R_2 : IF Input DATA is nominal THEN Output data is unchanged
and so on...

Fig 2 shows Input Membership Function of DATA effort multiplier. The DATA description of every levels is translated into GMF. For example, low level of DATA has description that is stated database size in bytes of characters divided by software size in SLOC must be less than ten so we draw the low interval to less than ten and so on.

Fig 3 show Output Membership Function of DATA effort multiplier. The value of GMF is taken from the value of every level in DATA EM. For example, low level has value of 0.90 so we draw the decreased interval to 0.90. After creating input and output to GMF, we set the rules based on rules that have been stated before. And then the new value of each level is resulted. And we apply this to other qualitative EM.

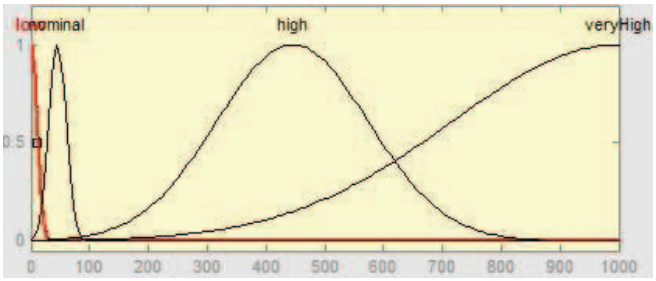

Figure 2. Representation of Input DATA EM using Gaussian Membership Function

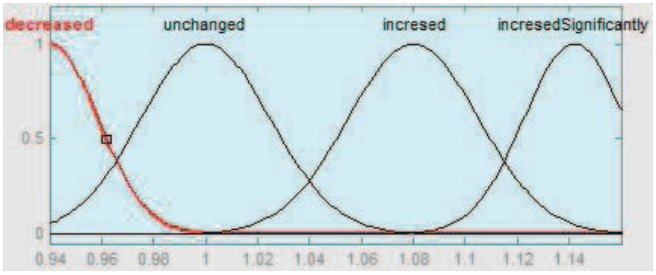

Figure 3. Illustration of Output DATA EM using GMF

After getting new value of EM, we replace the value in datasets with Fuzzy COCOMO values. And then the data is used to create ANN.

The proposed structure of ANN is used to adjust the Fuzzy COCOMO effort estimation. This structure require 17 Effort Multipliers, 5 Scale Factors, 3 biases, 2 hidden layer and 1 output. This research transform every input into a linear model using natural logarithms as shown in Eq.2.

$$\ln(PM) = \ln(A) + \ln(EM_1) + \ln(EM_2) + \cdots + \ln(EM_{17}) + [1.01 + SF_1 + \cdots + SF_5] * \ln(size)] \quad (2)$$

The above equation becomes:

$$e = [b_1 + w_1 * X_1 + w_2 * X_2 + \cdots + w_3 * X_3] + [b_2 + Y_1 + \cdots + Y] * [v_i + \ln(size)] \quad (3)$$

Where
e = ln(PM)
X_1 = ln(EM_1);...;X_{17}=ln(EM_{17})
Y_1 = SF_1;...;Y_5=SF5

Fig 4. shows the proposed architecture of neural network. The 22 inputs and 2 biases are divided to 2 hidden layer. This architecture count the contribution of Effort Multipliers and Scale Factors separately. Each node is connected with weight which is initialized as w, v, wb or wh.

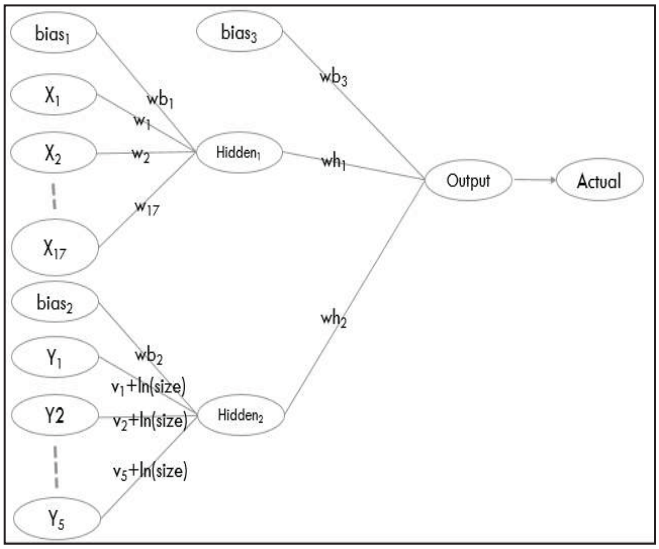

Figure 4. Architecture of the proposed neural network model.

These are the steps to calculate new set of weight and to improve error:

Step 1 : Initialize the inputs as ln(input)
Step 2 : Initialize the weights and biases
$w_i=wb_i=wh_i=1$; $v_i=0$; $bias_i=1$
Step 3 : Set learning rate η ($0 < \eta \leq 1$)
Step 4 : Test stopping condition for false,
Repeat the steps 5 to 13
Step 5 : For each training data,
Repeat the steps 6 to 12
Step 6 : Compute the hidden layers
$Hidden_1 = b_1 + \sum X_i * w_i$ for i=1 to 17
$Hidden_2 = b_2 + \sum Y_i * (v_i + \ln(size))$ for i=1 to 5
Step 7 : Activate the hidden layers

$$Hidden_i = \frac{1}{1 + e^{-Hidden_i}} \quad \text{for i = i to 2}$$

Step 8 : Compute the output layer
$e = bias_3 + Hidden_1 * wh_1 + Hidden_2 * wh_2$
Step 9 : Count error
$err = \ln(\text{Actual Effort}) - e$
Step 10 : Count Δw, hidden layer errors and Δv
$\Delta wh_i = \eta * err * hidden_i$ for i = 1 to 2
$hidden_i\_err = err * wh_i * hidden_i * (1-hidden_i)$ for i = 1 to 2
$\Delta w_i = hidden_1\_err * \eta * X_i$ for i = 1 to 17
$\Delta v_i = hidden_2\_err * \eta * Y_i$ for i = 1 to 5
$\Delta wb_i = hidden_i\_err * \eta$ for i = 1 to 2
$\Delta wb_3 = \eta * err$
Step 11 : Update the weights
$wh_i = wh_i + \Delta wh_i$ for i = 1 to 2
$wi = wi + \Delta w_i$ for i = 1 to 17
$vi = vi + \Delta v_i$ for i = 1 to 5
$wb_i = wb_i + \Delta wb_i$ for i = 1 to 3
Step 12 : Count the test data using new weights
Step 13 : Test stopping condition
If the error between estimated effort and actual effort in test data is smaller than a specific threshold or the number of iteration exceeds a specific number, stop: else continue.

Using this approach, we improve the accuracy of effort estimation by send input forward and learn backward to update weights. The variable η applied in above formula is the learning rate, a constant value which helped to train the weight.

V. EXPERIMENTAL RESULTS

This section shows the results achieved by applying the proposed fuzzy and neural networks model to dataset. Fuzzy Logic model is implemented in Matlab while Neural Network model is executed by creating console application uses C++ language.

NASA dataset consists of 93 data points. Each data point presents NASA project from different centers in several years. Furthermore, it consists of 24 attributes. The attributes is divided into four categories such as seven attributes describe the project, fifteen attributes show COCOMO Effort Multiplier in the range from Very Low to Extra High, line of code (KLOC), and actual effort in person months. This research divides dataset into 80% for training data and 20 % for testing data.

This research tries to compare the estimated effort with real effort. To evaluate the accuracy of estimated effort, this research uses a most common techniques such as Magnitude Relative Error (MRE) and Mean Magnitude Relative Error (MMRE) [14], which is defined as in "(4)".

$$MRE = \frac{|Actual\ Effort - Estimated\ Effort|}{Actual\ Effort} * 100 \quad (4)$$

The MRE values are performed for each project, while mean magnitude relative error computes the average of MRE from N projects.

$$MMRE = \frac{1}{N}\sum_{x=1}^{N} MRE \quad (5)$$

From these equations, the smaller value of MRE or MMRE is the closer to actual effort. For example, Project ID 4 has 44.57% error by using COCOMO II Model, 24.89% error by using Fuzzy model and 6.15% error using proposed model. It means that this proposed model can reduce 38.42 % error from COCOMO II Model.

TABLE 1.
COMPARISON OF EFFORT ESTIMATION RESULTS IN MRE

Project ID	MRE(%) using COCOMO II Model	MRE(%) using proposed Fuzzy Model	MRE(%) using proposed Fuzzy + NN Model
4	44.57	24.89	6.15
9	61.45	56.15	21.18
10	69	62.83	3.77
14	24.13	39.14	51.32

Project ID	MRE(%) using COCOMO II Model	MRE(%) using proposed Fuzzy Model	MRE(%) using proposed Fuzzy + NN Model
20	26.14	46.15	43.58
27	37.94	0.6	0
30	41.76	5.6	2.94
37	25.47	13.34	8.73
39	34.48	23.82	6.79
43	57.64	51.29	32.87
44	31.65	20.71	14.34
55	58.11	50.15	13.9
59	74.18	90.26	45.16
63	19.28	6.47	7.14
64	61.26	55.11	8.78
70	30.63	7.82	21.5
73	26.90	2.86	24.52
79	63.37	58.19	4.22
90	81.25	79.3	30.26
MMRE(%)	**45.74**	**36.56**	**18.27**

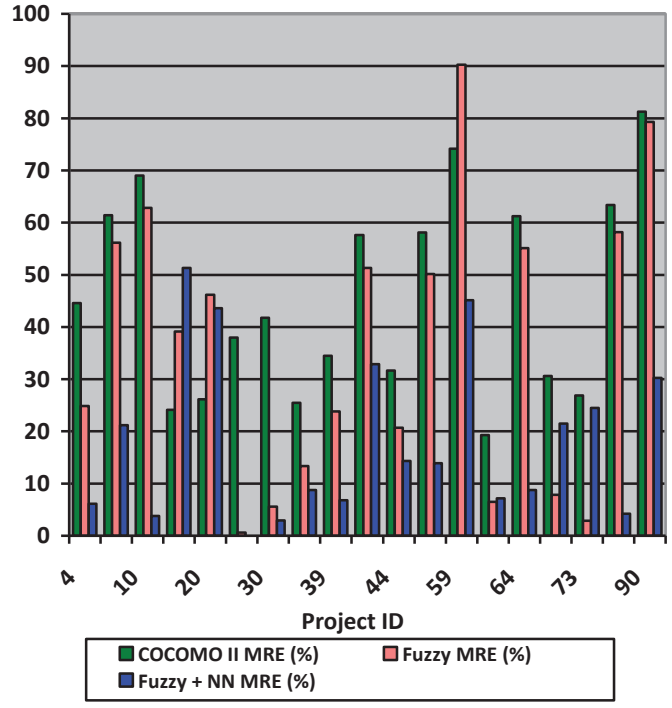

Figure 5. Bar chart showing effort estimation accuracy in MRE

VI. CONCLUSION

Predicting an accurate software cost estimation has always been a challenge not only for application industry but also academic communities. The more accurate software cost estimation could handle the software development resources efficiently. There are some effort estimation model that can be applied to forecast application estimation cost.

In this paper, both Fuzzy Logic and Neural Network are used to improve the accuracy of COCOMO model. Fuzzy Logic with Gaussian Membership Function (GMF) could redesign the COCOMO Effort Multipliers because GMF could make a smoother transition which means a more accurate Effort Multipliers.

The neural network that we have used in this research is multi-layer feed-forward neural network with back-propagation learning algorithm. Sigmoidal activation function is applied only in hidden layer while output layer uses linear function. This proposed model has been applied in NASA dataset. The result is proposed model creates a major improvement rather than Fuzzy model or COCOMO Model. Furthermore, the other architecture of neural networks could also be applied for improving software cost estimation.

VII. REFERENCES

[1] B. Boehm, Software Engineering Economics, New Jersey, USA: Prentice-Hall, Englewood Cliffs, 1994.

[2] B. Boehm, Software Cost Estimation with COCOMO II, New Jersey, USA: Prentice Hall, 2000.

[3] L. Zadeh, "From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions," *IEEE,* pp. 105-119, 1999.

[4] L. Zadeh, "A new direction in AI – toward a computational theory of perceptions," *AI Magazine,* pp. 73-84, 2011.

[5] V. Chandra, "Software effort estimation: a fuzzy logic approach," *International Journal of Computer Applications,* vol. 103, pp. 39-43, 2014.

[6] I. Attarzadeh and S. Ow, "Improving the accuracy of software cost estimation model on a new fuzzy logic model," *IDOSI,* pp. 177-184, 2010.

[7] C. S. Reddy and K. Raju, "An improved fuzzy approach for COCOMO's effort estimation using gaussian membership function," *Journal of Software,* vol. 4, pp. 452-459, 2009.

[8] A. Kaushik, A. Soni and R. Soni, "A simple neural network approach to software cost estimation," *Global Journal of Computer Science and Technology,* vol. 13, no. 1, pp. 23 - 30, 2013.

[9] L. Zadeh, "Fuzzy sets, Information and Contro," pp. 338-353, 1965.

[10] Z. Liu and H. Li, "A Probabilistic Fuzzy Logic System for Modeling and Control," *IEEE,* pp. 848-859, 2005.

[11] L. Wang, Adaptive Fuzzy System and Control: Design and Stability Analysis, New Jersey: Prentice-Hall, 1994.

[12] E. Mamdani, "Applications of fuzzy algorithms for simple dynamic plant," in *Proceedings of the IEEE*, 1974.

[13] R. Hughes, "An evaluation of machine learning techniques for software effort estimation," *University of Brighton,* 1996.

[14] L. Briand, K. Emam, D. Surmann and I. Wieczorek, "An assessment and comparison of common software cost estimation modeling techniques," *ISERN,* 1998.

[15] R. Sarno, C. Djeni, I. Mukhlash and D. Sunaryono, "Developing a workflow management system for enterprise resource planning," *Journal of Theoretical and Applied Information Technology,* vol. 72, no. 3, pp. 412-421, 2015.

[16] R. Sarno, J. L. Buliali and S. Maimunah, "Pengembangan metode analogy untuk estimasi biaya rancang bangun perangkat lunak," *MAKARA Journal of Technology Series,* vol. 6, no. 2, pp. 46 - 55, 2002.