# INTEGRATION OF FUZZY LOGIC METHOD AND COCOMO II ALGORITHM TO IMPROVE PREDICTION TIMELINESS AND SOFTWARE DEVELOPMENT COST

**Neneng Rachmalia Feta[1*]; Deki Satria[2]; Fitria[3]**

Information Systems and Technology[1,2,3]
Bank Rakyat Indonesia Institute of Technology and Business
http://bri-institute.ac.id/
nenengrachmaliafeta@gmail.com[1*], deki.satria@bri-institute.ac.id[2], fitria.fitrib@gmail.com[3]

***Abstract***—*This study discusses improving the prediction of timeliness and cost of software development using the Constructive Cost Model II (COCOMO II) method and the application of Fuzzy Logic. And aims to obtain accurate time and cost prediction estimates on software development projects to obtain maximum cost results for a software development project. This study utilizes an adaptive fuzzy logic model to improve the timeliness of software development and cost estimates. Using the advantages of fuzzy set logic and producing accurate software attributes to increase the prediction of the time and price of software development. The fuzzy model uses the Two-D Gaussian Membership Function (2-D GMF) to make the software attributes more detailed in terms of the range of values. In COCOMO I, NASA98 data set; and four data projects from software companies in Indonesia were used to evaluate the proposed Fuzzy Logic COCOMO II, commonly known as FL-COCOMO II. Using the Mean of Magnitude of Relative Error (MMRE) evaluation technique, after experimenting using MF1, MF2, and MF3 treatments, the results obtained that MF1 with a value of 65.51 is a better treatment than MF2 with a value of 92.74 and MF3 with a value of 163.36 because the MF1 value has the smallest MMRE value among other treatments. While at the minimum MRE points, MF2 has the smallest value, namely 0%, compared to MF1 with a value of 0.02% and MF3 with a value of 70.12%.*

**Keywords**: COCOMO II, Fuzzy Logic, Software Cost Estimation, Gaussian Membership Function (2-D GMF).

**Abstrak**—*Penelitian ini membahas mengenai peningkatan prediksi ketepatan waktu dan biaya pengembangan perangkat lunak dengan menggunakan metode Constructive Cost Model II (COCOMO II) dan penerapan Logika Fuzzy. Serta bertujuan untuk memperoleh perkiraan prediksi ketepatan waktu dan biaya yang akurat pada proyek pengembangan perangkat lunak, sehingga dapat memperoleh hasil biaya yang maksimal untuk sebuah proyek pengembangan perangkat lunak. Penelitian ini menggunakan model logika fuzzy adaptif untuk meningkatkan ketepatan waktu pengembangan perangkat lunak dan perkiraan biaya. Menggunakan keuntungan dari logika himpunan set fuzzy serta menghasilkan atribut perangkat lunak yang akurat untuk meningkatkan prediksi ketepatan waktu dan biaya pengembangan perangkat lunak. Dua-Dimensi Gaussian Membership Function (2-D GMF) digunakan dalam model fuzzy untuk membuat atribut perangkat lunak lebih rinci dalam hal rentang nilai. Pada COCOMO I, NASA98 set data; dan empat proyek data dari perusahaan perangkat lunak di Indonesia digunakan dalam evaluasi yang diusulkan Fuzzy Logic COCOMO II atau yang biasa disebut FL-COCOMO II. Dengan menggunakan teknik evaluasi Mean of Magnitude of Relative Error (MMRE), setelah dilakukan percobaan menggunakan perlakuan MF1, MF2, dan MF3 didapatkan hasil bahwa MF1 dengan nilai 65,51 merupakan perlakuan yang lebih baik dibandingkan dengan MF2 dengan nilai 92,74 dan MF3 dengan nilai a nilai 163,36 karena nilai MF1 memiliki nilai MMRE paling kecil diantara perlakuan lainnya. Sedangkan pada titik MRE minimum, MF2 memiliki nilai terkecil yaitu 0%, dibandingkan dengan MF1 dengan nilai 0,02% dan MF3 dengan nilai 70,12%.*

***Kata Kunci***: *COCOMO II, Fuzzy Logic, Estimasi Biaya Perangkat Lunak, Gaussian Membership Function (2-D GMF).*

## INTRODUCTION

Many software development project failures occur because the project is completed more than the planned cost and schedule and is a significant problem for software project managers. Poor forecasting causes projects to exceed budget and schedule and, in many cases, causes software development projects to fail. Between 30% and 40% of software, projects are ultimately completed outside budget or schedule, and more projects are canceled or fail (Pospieszny et al., 2018). Among the many reasons for failure, inaccuracy in

Vol. 19, No. 1 March 2022 | DOI https://doi.org/10.33480/techno.v19i1.3037

software estimates has been identified as the root cause of a high percentage of losses in software development (Singal et al., 2020).

Software cost prediction defines organizations' techniques and procedures to estimate bid proposals, project planning, and probability estimation (Christina & Banumathy, 2019). Thus, prediction accuracy is an important and significant issue for executives, managers, technical staff, and practitioners who perform or rely on cost prediction (Parwita et al., 2017). Unfortunately, accurate estimation of software development costs also challenges software engineering researchers due to the continued lack of precise estimates (Singal et al., 2020). The reasons that software cost estimation is complex and error-prone include (Christina & Banumathy, 2019):

a. Software cost estimation requires a significant amount of cost to be done correctly;
b. This process is often carried out in a hurry, with no cost calculations required to make estimates;
c. Experience is required to develop estimates, especially for large projects, and;
d. Human prediction

Some software development cost prediction models can be classified as algorithmic and non-algorithmic models. The model algorithm is based on a statistical analysis of historical data (Indra & Aqlani, 2018) (Subandri & Sarno, 2017), for example, Software Life Cycle Management (SLIM) (Kholed Langsari et al., 2018) and Constructive Cost Model (COCOMO) (Yadav, 2017). Non-algorithmic techniques based on new approaches such as Parkinson's, Expert Judgment, Price to-Win, and Machine Learning Approaches (K Langsari & Sarno, 2017b) (Sinha & Gora, 2021). Machine Learning Approaches represent facts from the human mind (K Langsari & Sarno, 2017a), for example, rule induction, fuzzy systems, genetic algorithms, artificial neural networks, bayesian networks, and evolutionary computing. These five approaches are classified into Soft Computing Groups. The importance of approximation from algorithmic and non-algorithmic techniques is discussed further in the following sections.

The well-known cost estimation algorithmic models are Boehm's COCOMO I and II (Singal et al., 2020), Albrecht's Function Point; and Putnam's SLIM (Kholed Langsari et al., 2018). This model requires input and accurate estimation of specific attributes, such as Source Line Of Code (SLOC), number of users, interface, complexity, etc. This is not easy to obtain in the early stages of software development. However, the formulas and calculations of this model are easy to understand and can also provide fast estimates compared to non-algorithmic models. In addition, the attributes and relationships used to estimate software development costs may change over time, and/or differ in the software development environment (Yadav, 2017). The limitations of algorithmic models lead to the exploration of non-algorithmic models.

In 1990, a non-algorithmic model was born and was proposed for software cost estimation. Software researchers have turned their attention to new approaches based on soft computing approaches such as artificial neural networks, fuzzy logic, and genetic algorithms. Fuzzy Logic (FL) offers a robust linguistic representation representing inaccuracies in input and output models and provides a more knowledge-based approach to constructing an effective model. Research shows that using FL can lead to better performance in reducing the inaccuracy of input and output parameters (K Langsari & Sarno, 2017b).

Gray and Macdonnell, in the study of Singal et al., compare popular techniques in software cost estimation, such as regression techniques, function point analysis (FPA), fuzzy logic, and artificial neural networks. Their research shows that the fuzzy logic model has produced better performance than other models (Singal et al., 2020). They introduced an application of fuzzy logic for cost estimation, which was used as a development tool, FUzzy logic SOftware MEasuring (FULSOME) (Iqbal & Sang, 2021), to assist software managers in decision making. The FULSOME model selects two critical variables: the complexity adjustment factor and the point function mismatch. Then triangular membership functions are defined for small, medium, and large interval sizes, complexity, and software effort.

The research of Raza, Fei, and Liu attempted to apply fuzzy logic to the algorithmic cost estimation model in dealing with uncertainty and imprecision problems in the model (Raza, 2019). They proposed a software fuzzy size model for COCOMO I. This study found an unusual input fuzzy rule linguistic variable when setting the determinate value for the software attribute measure in COCOMO I because an accurate estimate of Kilo Delivered Source Instructions (KDSI) could not do before starting the project.

Riyanarto applies fuzzy modelling techniques to COCOMO I and the Point Function Model (Sarno et al., 2015). Murad and Goyal et al. investigated the application of fuzzy logic to Effort Multipliers (EM) among COCOMO I (Murad et al., 2021) (Goyal et al., 2015). Also, fuzzification of COCOMO I without considering the adjustment factor, so they introduced F-COCOMO. They gave the software a new measure of COCOMO I, and the coefficients associated with the development mode

were assigned to a fuzzy set. In another study, Kumar et al. in Bedi's research applied fuzzy logic to the Manpower Buildup Index (MBI) of the Putnam estimation model based on 64 different rules. Bedi's research results show that fuzzy logic can be effectively applied to software cost estimation (Bedi & Singh, 2017).

Fuzzy logic has also been applied to non-algorithmic models to overcome model uncertainty. For example, Indra and Aqlani proposed a combination of estimation from the fuzzy logic model with the analogy technique (Indra & Aqlani, 2018). Analogy estimation is one of the expert-based classification techniques, and this is a type of Case-based Reasoning (CBR) method (Zhang, 2019). In addition, a fuzzy analogy for software cost estimation has also been applied to web-based software.

In summary, fuzzy logic has been applied to algorithmic and non-algorithmic cost estimation models to achieve better estimation results. However, there is still a lot of uncertainty about what technique is used to look at the type of estimation problem (Raza, 2019). Therefore, choosing between different approaches is a difficult decision that requires the support of a well-defined evaluation method to demonstrate each estimation technique and apply it to estimation problems (Singal et al., 2020).

For decades, accurate software cost prediction has been essential for software development projects. However, inaccurate estimates in leading the project can exceed the budget and schedule, and even in many cases, the project can be stopped completely (Indra & Aqlani, 2018). The ability to accurately estimate development time, costs, labour, and new methodological changes can replace older software development cost prediction models. Therefore, an accurate software development cost prediction model will be needed in software development project management.

This study proposes an effective Fuzzy Decision Tree model for embedding in COCOMO II to overcome the ambiguity and uncertainty of software attributes, resulting in more accurate estimation results. The steps are taken to apply the estimate: strategic planning, feasibility studies, system specifications, evaluation of supplier proposals, and software development project planning (Baiquni et al., 2017).

## MATERIALS AND METHODS

The research method describes the approach to calculating software cost predictions with COCOMO II and the Fuzzy Logic approach and will then discuss the effect of Fuzzy Logic on COCOMO II or what we call FL-COCOMO II.

### A. Model COCOMO II

In Yadav's research, the COCOMO I model is a regression-based estimation software cost prediction model developed by Boehm in 1981 (Yadav, 2017) and is considered the best known and the most reasonable model among all traditional cost estimation models. The COCOMO I was the most stable model at the time. One of the problems with using COCOMO I at the time was that it was incompatible with the development environment of the late 1990s. Therefore, in 1997, Boehm developed it into COCOMO II to solve most of the problems of COCOMO I (Baiquni et al., 2017). Figure 1 shows the software schedule, cost, and labour estimation formulas and processes in COCOMO II. Equation (1) shows the exact formula regarding Efforts or efforts that can be made. Equation (2) discusses the formula regarding the software scheduling process. Equation (3) shows the formula for estimating labour staff in software development. Equation (4) shows the formula for calculating software development costs (K Langsari & Sarno, 2017b). COCOMO II includes several software attributes such as 17 Effort Multipliers (EMs), 5 Scale Factors (SFs), Software Size (Software Size), and effort estimates used in the Post Architecture Model COCOMO II (Putri et al., 2017). Descriptions of the 17 EM and 5 SFS based on their numerical values and productivity ranges are shown in Table 1 (Baiquni et al., 2017).

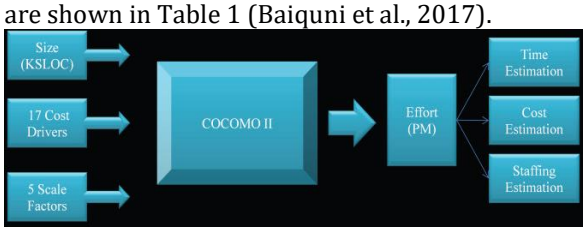


Figure 1. Software schedule, cost, and labour estimation process in COCOMO II (Putri et al., 2017)

$$Effort_{PM} = Ax[Size]^{B+0.01x\sum_{j=1}^{5} SF_j} x[i = 1] \; 17 \; \Pi \; EM \quad \text{.....(1)}$$

$$Schedule_{Months} = Cx\,(Effort)^{D+0.2x\,0.01x \sum_{j=1}^{5} SF_j} \quad \text{.....(2)}$$

$$Average\ staffing_{People} = \frac{Effort}{Schedule} \quad \text{.....(3)}$$

$$\text{COST} = Effort\ x\ (Payment_{Month}) \quad \text{.....(4)}$$

A = 2.94; B = 0.91; C = 3.67; D = 0.28

*Size: Software Size* (SLOC) (K Langsari & Sarno, 2017b)

Techno Nusa Mandiri: Journal of Computing and Information Technology
Vol. 19, No. 1 March 2022 | DOI https://doi.org/10.33480/techno.v19i1.3037

The ambiguity and uncertainty of software attributes can impact software estimates. Thus, accurate software attributes produce special software estimates that software project managers and organizations most desire.

Table 1. COCOMO II EMs Range

| No | Effort Multiplier | Range |
|---|---|---|
| 1 | Required software reliability (RELY) | 0.82-1.26 |
| 2 | Database size (DATA) | 0.90-1.28 |
| 3 | Product complexity (CPLX) | 0.73-1.74 |
| 4 | Developed for reusability (RUSE) | 0.95-1.24 |
| 5 | Documentation match to life-cycle needs (DOCU) | 0.81-1.23 |
| 6 | Execution time constraint (TIME) | 1.00-1.63 |
| 7 | Main storage constraint (STOR) | 1.00-1.46 |
| 8 | Platform volatility (PVOL) | 0.87-1.30 |
| 9 | Analyst capability (ACAP) | 1.42-0.71 |
| 10 | Programmer capability (PCAP) | 1.34-0.76 |
| 11 | Personnel continuity (PCON) | 1.29-0.81 |
| 12 | Applications experience (APEX) | 1.22-0.81 |
| 13 | Platform experience (PLEX) | 1.19-0.85 |
| 14 | Language and tool experience (LTEX) | 1.20-0.84 |
| 15 | Use of software tools (TOOL) | 1.17-0.78 |
| 16 | Multisite development (SITE) | 1.22-0.80 |
| 17 | Required development schedule (SCED) | 1.43-1.00 |

Table 1 represents the variables or factors that affect software estimates. In determining the factors that influence software attributes, it can be done by reviewing the existing literature based on previous studies related to software estimation. For example, regarding the COCOMO II Architecture model, these parameters were determined based on the use of COCOMO'81 and the experience of a group of senior software cost analysts.

The value of SFs is based on the premise that they are a significant source of exponential variation in project effort and product variation.

B. Fuzzy Logic (FL)

In 1965, in Kholed's research, Lotfi Zadeh officially developed the multi-value theory and introduced the term fuzzy into the engineering literature (Kholed Langsari et al., 2018). Fuzzy Logic (FL) started using the concept of fuzzy set theory. Fuzzy theory is a class theory with unclear boundaries and will be seen as an extension of the set of a classical theory (Kaur et al., 2018). The membership $\mu A(x)$ of element X of the classical set A, as part of the universe X, is defined by equation (5), as follows:

$$\mu A(x) = \begin{cases} 1 \text{ if } x \in A \\ 2 \text{ if } x \notin A \end{cases} \quad \text{..... (5)}$$

A system based on FL has a direct relationship with fuzzy concepts (such as fuzzy sets, linguistic variables, etc.) and fuzzy logic. The popular fuzzy logic systems can be categorized into three types: Pure fuzzy logic systems, Takagi and Sugeno fuzzy systems, and fuzzy logic systems with fuzzified and defuzzifiers (Kaur et al., 2018). Since most engineering applications produce crisp data as input and expect crisp data as output, the last type is the most widely used type of fuzzy logic system. The first fuzzy logic system with a fuzzifier and defuzzifier was proposed by Mamdani and has been successfully applied to various industrial processes and consumer products (Kaur et al., 2018). The three main steps of using fuzzy logic in a model are:

Step 1: Fuzzification: convert input crips to fuzzy set.

Step 2: Fuzzy Rule-Based System: Fuzzy logic system using fuzzy IF-THEN rules; Fuzzy Inference Engine: After all the input crips values are fuzzified into the respective linguistic values, the inference engine accesses the fuzzy rule base to obtain linguistic values for intermediate and output linguistic variables.

Step 3: Defuzzification: converts the fuzzy output into crisp output (Raza, 2019).

C. Fuzzy Logic COCOMO II (FL-COCOMO II)

FL-COCOMO II is based on COCOMO II and Fuzzy Logic. COCOMO II includes a set of software attribute inputs: 17 EM, 5 SFS, 1 SS, and one output, estimated effort. The FL-COCOMO II architecture is shown in Figure 2.

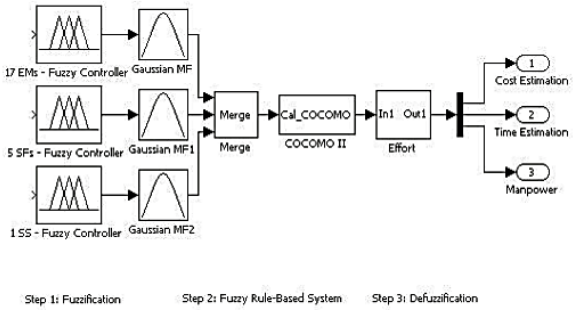


Figure 2. FL-COCOMO II architecture includes: 17 EMs, 5 SFs, 1 SS, and estimated effort

In COCOMO II, the effort is expressed as Person Months (PM). Determination of the effort required by a software development project based on the size of the software project is Kilo Source Lines of Code (KSLOC). Traditionally, software effort estimation problems have relied on single (numeric) values of EMs, SFs, and software sizes given as software attributes to estimate effort. However, software size can be calculated based on previously developed software similar to the current one (especially at the start of the project). Although the truth and accuracy of such estimates are minimal, it is fundamentally important to recognize the situation and with a technique that can evaluate the associated imprecision that is in the final cost estimate.

The software size can be determined using fuzzy sets in EMs, SFs, and attributes by dividing possible values rather than using fixed values. Generally, this distribution form is represented in the form of a fuzzy set. It is essential to clarify that ambiguity and uncertainty at the input level of COCOMO II produce uncertainty at the output level (Singal et al., 2020). Converting software attributes for each fuzzy set can increase software attributes' accuracy, resulting in very accurate estimates. On the other hand, inaccurate input estimates can lead to less detailed effort estimates. The overlapping membership functions of bell-shaped cost drivers and trapezoidal cost drivers change the fuzzy model to a more precise model.

In addition, it is possible that when using the membership function bell-shaped cost drivers and trapezoidal cost drivers, there are several attributes assigned to the maximum level of compatibility instead of being transferred to the lowest degree. To avoid this linearity, it is proposed to use better functions, membership functions of bell-shaped cost drivers and trapezoidal cost drivers, to represent the model's inputs. 2-D GMF is represented by equation (6) as follows:

$$\mu A_i(x) = Gaussian\,(x, c_i, \sigma_i) = e^{\frac{-(x-c_i)^2}{2\,\sigma_i 2}} \text{.....(6)}$$

Where $c_i$ is the midpoint of $i^{th}$ of the fuzzy set and $\sigma_i$ is the width of $i^{th}$ fuzzy set (Baiquni et al., 2017).

Applying fuzzy logic to COCOMO II to construct FL-COCOMO II is described as follows. The three main processes in FL-COCOMO II are Fuzzification, Fuzzy Rule-Based/Fuzzy Inference Engine, and Defuzzification. Software attributes in COCOMO II are converted to fuzzy variables based on the fuzzification process with terms and conditions Extra Low (XL), Very Low (VL), Low (L), Nominal (N), High (H), Very High (VH) ), and Extra High (XH) were defined for the 11 software attributes (17 EMs, 5 SFs, and 1 SS) assigned to each software attribute. The fuzzy sets corresponding to various language-related values for each software attribute are defined using 2-D GMF. For example, fuzzification of Applications Experience (AEXP) EM is based on the membership function bell-shaped cost drivers and trapezoidal cost drivers function, using the Fuzzy Inference System tool in the MATLAB software; the definitions are shown in Table 2 and Figure 3. Fuzzy Inference System (FIS) is a fuzzy tool in the MATLAB software used in the fuzzification, fuzzy calculations, fuzzy rules, and defuzzification process of FL-COCOMO II. FIS supports the Mamdani fuzzy method and the Sugeno fuzzy method. FLCOCOMO II is based on the Sugeno fuzzy system, which is more accurate than the FIS Mandani method.

Table 2. Applications Experience (AEXP) EM Description

| Effort Multiplier: Applications Experience (AEXP) | | | | | | |
|---|---|---|---|---|---|---|
| Descriptors | 2 months | 6 months | 1 year | 3 years | 6 years | - |
| Effort Multipliers | 1.42 | 1.19 | 1 | 0.85 | 0.71 | - |
| Rating | VL | L | N | H | VH | EX |

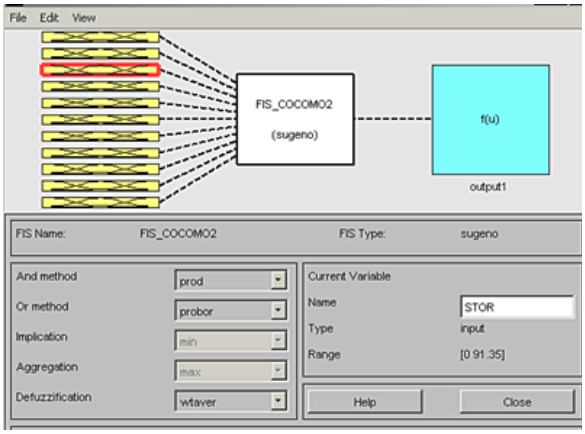

Figure 3. Fuzzification Process in MATLAB

In the first step, all software attributes of COCOMO II are converted in response to the fuzzy set and its variables (***FuzzyEMij***) instead of using fixed values of ***EMij***. ***FuzzyEMij*** is calculated using equation (7). The original ***EMij*** value and the membership function bell-shaped cost drivers and trapezoidal cost drivers $\mu$ are defined for various fuzzy sets related to EMs, SFs, and SS. This process helps to reduce the ambiguity and uncertainty of software attributes at this level.

$$\boldsymbol{FuzzyEMij} = \boldsymbol{F}\left(\mu_{A1}^{V1}, \ldots\ \mu_{Ai}^{Vi}, \boldsymbol{EM}_{i1} \ldots \boldsymbol{EM}_{ij}\right) \text{.....(7)}$$

For convenience, F is taken as a linear function, where $\mu_{A1}^{V1}$ the membership function of the fuzzy set $Aj$ is related to the controlling value $Vi$, as shown in equation (8).

$$\boldsymbol{FuzzyEMij} = [\boldsymbol{j} = 1]\, k_i \sum \mu_{A1}^{V1} * \boldsymbol{EM}_{ij} \text{ .....(8)}$$

The fuzzy rules for FL-COCOMO II are defined through linguistic variables in the fuzzification process. It is important to note that the fuzzy rules are adapted to all functions of the degree of precision, according to the test and the characteristics of the project. Fuzzy rules are defined based on the "AND" and "OR" relationships or their combination between the input variables, as shown below:

Aturan Fuzzy:
IF TOOL is Low THEN effort is Low
IF PCAP is Very Low THEN effort is Very High
IF RUSE is Nominal THEN effort is Nominal

Techno Nusa Mandiri: Journal of Computing and Information Technology
Vol. 19, No. 1 March 2022 | DOI https://doi.org/10.33480/techno.v19i1.3037

IF DATA is Very High THEN effort is Very High
. . . (Bedi & Singh, 2017)

The number of rules defined for FL-COCOMO II is more than 193 based on input variables. In applying fuzzy rules from FL-COCOMO II, the FIS tool in the MATLAB software is used, as shown in Figure 4. The process carried out is to input the fuzzy rules. Then, the fuzzy input rules are adjusted to The Range of COCOMO II Ems in Table 1.

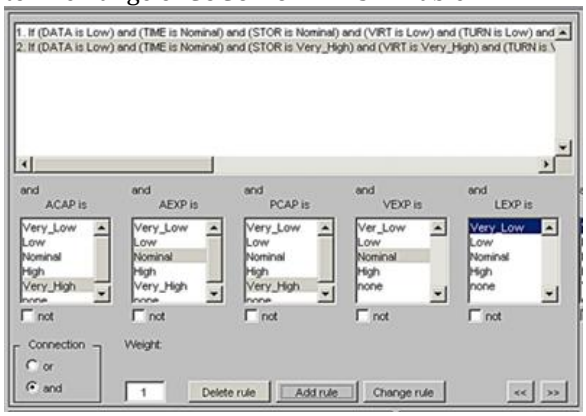

Figure 4. The process of applying fuzzy rules in MATLAB

The last step is the defuzzification of the effort variable using defuzzification techniques such as Mean of Maximum (MOM), Center of Area (COA), and First of Maximum (FOM). Defuzzification of the "Effort" output is carried out using the Mean of Maximum (MOM) technique because the results obtained are more accurate when compared to other defuzzification techniques.

## RESULTS AND DISCUSSION

Data set is a collection of data used in software development project data (Bedi & Singh, 2017). Two data sets are used in the research: NASA data consisting of 93 project data collected from 2000 to 2016 and project data as many as 60 data collected from 2008 to 2018. But from 2008 to 2018 data, some data overlap. Before calculation, the data set is converted into an Excel file for easy analysis.

FL-COCOMO II was evaluated through two sources; namely, the data used for testing the fuzzy method is the NASA data group which is 102 project data from 2000 to 2018, and data for 15 types of COCOMO cost drivers, number of lines of program code, project type, and actual effort. Project development.

A. Dataset 1

COCOMO data set I-Boehm (Singal et al., 2020) was the first researcher to look at software engineering from an economics point of view and its relationship to model cost estimation from the dataset. The COCOMO I dataset includes 63 project history data. Data is available at http://openscience.us/.

B. Dataset 2

NASA93 data set NASA93 data set includes 93 project data from NASA, which is the data element measurement benchmarking manager. This data is available at http://openscience.us/

C. Metode Evaluasi

Enterprises in the COCOMO model are described as Person Months (PM). PM is the effort required by a person or group to complete a project. This study uses the Magnitude of Relative Error (MRE) as shown in equation (9) and the Mean Magnitude of Relative Error (MMRE) formula shown in equation 10 to evaluate research results (Suherman et al., 2020). In addition to the two formulas, the study also looked at the maximum, minimum, median, and standard deviation values of the MRE.

$$MRE = \frac{Actual\ Effort - Predictive\ Effort}{Actual\ Effort} \quad \text{.....(9)}$$

$$MMRE = \frac{1}{N}\sum_{i}^{N} \mathrm{MRE}_i \quad \text{.....(10)}$$

The 11 variables with effort multipliers each have a value from 0.70 to 1.46, multiplied by the cost drivers. Effort multipliers were studied by Boehm in 1981 after project regression analysis in the COCOMO I data set. However, not all cost drivers are defined in fuzzy sets because the cost drivers are only an ordinary description (Tahir & Adil, 2018). For example, the cost drivers are RELY, CPLX, MODP, and TOOL (Murad et al., 2021). As for the number of lines of program code in this study using the Kilo Size Line Of Code (KSLOC).

Next, determine the membership function for the fuzzification process. The membership function used is taken from the journal Cost Model Using Fuzzy Logic (Goyal et al., 2015). The Fuzzy Logic Toolbox assists the creation of this membership function in MATLAB. The membership function in the journal is a trapezoidal membership function which can be seen in Figure 5. The membership function graph depicts the range of each cost driver. Membership function points are different for each cost driver. The treatment with the trapezoidal membership function is the MF1 treatment.

This study will compare the use of the trapezoidal membership function with the bell-shaped membership function. For the treatment of MF2 using a bell-shaped membership function. The bell-shaped membership function is obtained with the help of MATLAB by changing the trapezoidal so that the bell-shaped points and function graphs are obtained. The bell-shaped membership function is

made by looking at the interval approach, similar to the trapezoidal membership function. However, the number of points used in the bell-shaped is different from the trapezoidal one. The trapezoidal membership function uses four points to create a graph, while the bell-shaped one only requires three points.

The last treatment, namely MF3, combines trapezoidal and bell-shaped membership functions. This treatment aims to find the best model. A bell-shaped membership function is used for the 'nominal' category, while the 'very low', 'low', 'high', and 'very high' types use a trapezoidal membership function. For MF3, the 'nominal' category was chosen for the cost drivers to be converted into a bell-shaped function, considering that the 'nominal' type has a wide interval. The data entered is processed with fuzzy logic and will be calculated using COCOMO II calculations. The results of COCOMO II calculations can be seen in Table 3. After the results of the estimated effort are obtained for each treatment, the accuracy will be measured.

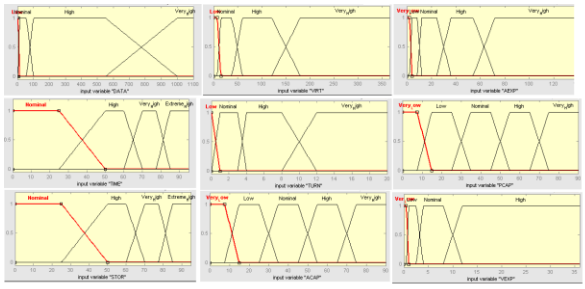

Figure 5. Trapezoidal membership function cost drivers

Table 3. COCOMO II calculation results

| No | Cost Driver | Kategori | Effort Multipliers |
|---|---|---|---|
| 1 | RELY | High | 1.15 |
| 2 | DATA | Low | 0.94 |
| 3 | CPLX | High | 1.15 |
| 4 | TIME | Nominal | 1.00 |
| 5 | STOR | Nominal | 1.00 |
| 6 | VIRT | Low | 0,87 |
| 7 | TURN | Low | 0,87 |
| 8 | ACAP | Nominal | 1,00 |
| 9 | AEXP | Nominal | 1,00 |
| 10 | PCAP | Nominal | 1,00 |
| 11 | VEXP | Nominal | 1,00 |
| 12 | LEXP | Nominal | 1,00 |
| 13 | MODP | High | 0.91 |
| 14 | TOOL | Nominal | 1,00 |
| 15 | SCED | Low | 1.08 |

The estimated results of the project development effort will be compared with the actual project effort. To measure the accuracy of the proposed business estimates, this study uses MRE. MRE looks at how close the estimated business results are to the actual effort. The smaller the MRE value, the better the business forecast results. The graph of the comparison of the MRE values for each treatment can be seen in Figure 6. This study used a threshold value of 20%. From the graph of the MRE value in the MF1 treatment, only 32 projects met the threshold value. While in MF2, there are 17 projects, and in MF3, there are no projects that meet the threshold. But most of the projects do not meet the specified threshold.

The smaller the MRE value, the closer the software effort estimate is to the actual business value. To measure the accuracy of the data set, the average MRE (MMRE) value was used for the three treatments. The results of the three treatments can be seen in Table 5. The MMRE value of the MF1 treatment is the most minor compared to other treatments. Then the MF1 treatment has a better predictive result. The maximum value and standard deviation of MRE in the MF1 treatment were also better than in MF2 and MF3 treatments. The project belongs to the semi-detached type from the table above, with the coefficient values shown in Table 4.

Number of Program Code Lines (KLSOC) = 25.9
Actual effort of project development = 117.6

For example, from the sample project data in Table 3 above, the business value will be calculated using the COCOMO II model as follows:

Table 4. Coefficient Score

| No | Coefficient | Score |
|---|---|---|
| 1 | Coefficient A | 3 |
| 2 | Coefficient B | 1.12 |

Effort Adjustment Factor (EAF) =
1.15 * 0.94 * 1.15 * 1.00 * 1.00 * 0.87 * 0.87 * 1.00 * 1.00 * 1.00 * 0.91 * 1.00 * 1.08 = 0.925.

Effort Adjustment Factor (Effort) =
EAF * Coefficient A * KSLOC $^{\text{Coefficient B}}$ =
$0.925 * 3 * 25.9^{1.12}$ =
106.208
(Person−Months)

Table 5. Evaluation Results on Treatment of MF1, MF2, MF3

| No | Treatment | MMRE (%) | Min MRE (%) | Max MRE (%) | MRE Standard Deviation (%) |
|---|---|---|---|---|---|
| 1 | MF1 | 65.51 | 0.02 | 1014.53 | 132.30 |
| 2 | MF2 | 92.74 | 0.00 | 1039.62 | 139.55 |
| 3 | MF3 | 163.36 | 70.12 | 5316.87 | 526.91 |

In comparing treatments from the MMRE value, it was found that MF1 has the smallest value, then MF2 with a difference of 27.23%, and MF3 with the most significant MMRE value can be seen in Figure 6. The graph depicts the accuracy level of

MF1, which is higher than the other two treatments. Because if the MMRE value is taller, the difference in the estimated results will be further away from the actual value of the business.

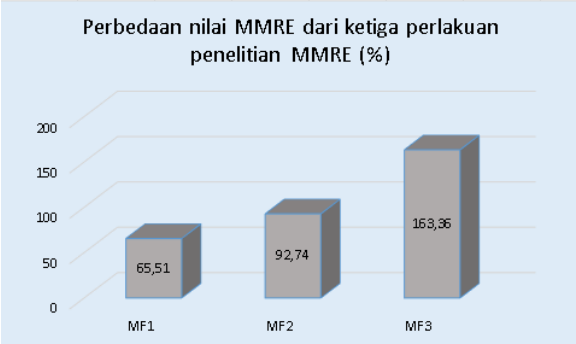


Figure 6. Graph of differences in MMRE values of the three treatments

The MF1 treatment had a minimum MRE value of 0.02%, but the MF2 treatment had a smaller MRE minimum value of 0%. The minimum MRE value percentage comparison shows that MF2 can reach the data better, even though MF1 has good results in the overall MRE calculation.

## CONCLUSION

After experimenting using MF1, MF2, and MF3 treatments, the results obtained that MF1 with a value of 65.51 is a better treatment than MF2 with a value of 92.74 and MF3 with a value of 163.36 because the MF1 value has the smallest MMRE value among other treatments. While at the minimum MRE points, MF2 has the smallest value, namely 0%, compared to MF1 with a value of 0.02% and MF3 with a value of 70.12%. It can be concluded that the trapezoidal membership function provides better accuracy than using a bell-shaped. But the bell-shaped has the smallest MRE value. The rapid growth of technology causes more and more new methods and logic that effectively solve a problem. Future research is expected to collaborate with NASA data-providing institutions to test the business forecast model. The new study is also likely to provide more recent data.

## REFERENCES

Baiquni, M., Sarno, R., Sarwosri, & Sholiq. (2017). Improving the accuracy of COCOMO II using fuzzy logic and local calibration method. *2017 3rd International Conference on Science in Information Technology (ICSITech)*, 284–289. https://doi.org/10.1109/ICSITech.2017.8257126

Bedi, R. P. S., & Singh, A. (2017). Software Cost Estimation using Fuzzy Logic Technique. *Indian Journal of Science and Technology*, *10*(3). https://doi.org/10.17485/ijst/2017/v10i3/109997

Christina, M. A., & Banumathy, C. (2019). Software cost estimation using neuro fuzzy logic Framework. *International Journal of Research in Engineering, Science and Management*, *2*(1), 219–224.

Goyal, M. V, Satapathy, S. M., & Rath, S. K. (2015). Software project risk assessment based on cost drivers and Neuro-Fuzzy technique. *International Conference on Computing, Communication & Automation*, 823–827. https://doi.org/10.1109/CCAA.2015.7148487

Indra, M., & Aqlani, Z. (2018). Comparative Analyisis of Software Cost Estimation Project using Algorithmic Method. *Engineering Software Requirements*, *1*(1), 17–27.

Iqbal, N., & Sang, J. (2021). Fuzzy Logic Testing Approach for Measuring Software Completeness. *Symmetry*, *13*, 604. https://doi.org/10.3390/sym13040604

Kaur, I., Narula, G. S., Wason, R., Jain, V., & Baliyan, A. (2018). Neuro fuzzy—COCOMO II model for software cost estimation. *International Journal of Information Technology*, *10*(2), 181–187. https://doi.org/10.1007/s41870-018-0083-6

Langsari, K, & Sarno, R. (2017a). Optimizing COCOMO II parameters using particle swarm method. *2017 3rd International Conference on Science in Information Technology (ICSITech)*, 29–34. https://doi.org/10.1109/ICSITech.2017.8257081

Langsari, K, & Sarno, R. (2017b). Optimizing effort and time parameters of COCOMO II estimation using fuzzy multi-objective PSO. *2017 4th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI)*, 1–6. https://doi.org/10.1109/EECSI.2017.8239157

Langsari, Kholed, Sarno, R., & Sholiq. (2018). Optimizing time and effort parameters of COCOMO II using fuzzy Multi-objective Particle Swarm Optimization. *Telkomnika (Telecommunication Computing Electronics and Control)*, *16*(5), 2199–2207. https://doi.org/10.12928/TELKOMNIKA.v16i5.9698

Murad, M. A., Abdullah, N. A. S., & Rosli, M. M. (2021). Software Cost Estimation for Mobile Application Development-A Comparative Study of COCOMO Models. *2021 IEEE 11th International Conference on System Engineering and Technology, ICSET 2021 - Proceedings*.

https://doi.org/10.1109/ICSET53708.2021.9612528

Parwita, I. M. M., Sarno, R., & Puspaningrum, A. (2017). Optimization of COCOMO II coefficients using Cuckoo optimization algorithm to improve the accuracy of effort estimation. *2017 11th International Conference on Information & Communication Technology and System (ICTS)*, 99–104. https://doi.org/10.1109/ICTS.2017.8265653

Pospieszny, P., Czarnacka-Chrobot, B., & Kobylinski, A. (2018). An effective approach for software project effort and duration estimation with machine learning algorithms. *Journal of Systems and Software*. https://doi.org/10.1016/j.jss.2017.11.066

Putri, R. R., Sarno, R., Siahaan, D., Ahmadiyah, A., & Rochimah, S. (2017). Accuracy Improvement of the Estimations Effort in Constructive Cost Model II Based on Logic Model of Fuzzy. *Advanced Science Letters*, *23*, 2478–2480. https://doi.org/10.1166/asl.2017.8767

Raza, K. (2019). Fuzzy logic based approaches for gene regulatory network inference. *Artificial Intelligence in Medicine*, *97*, 189–203. https://doi.org/10.1016/j.artmed.2018.12.004

Sarno, R., Sidabutar, J., & Sarwosri. (2015). Improving the accuracy of COCOMO's effort estimation based on neural networks and fuzzy logic model. *2015 International Conference on Information & Communication Technology and Systems (ICTS)*, 197–202. https://doi.org/10.1109/ICTS.2015.7379898

Singal, P., Kumari, A. C., & Sharma, P. (2020). Estimation of Software Development Effort: A Differential Evolution Approach. *Procedia Computer Science*, *167*(2019), 2643–2652. https://doi.org/10.1016/j.procs.2020.03.343

Sinha, R. R., & Gora, R. K. (2021). Software effort estimation using machine learning techniques. In *Lecture Notes in Networks and Systems*. https://doi.org/10.1007/978-981-15-5421-6_8

Subandri, M. A., & Sarno, R. (2017). Cyclomatic Complexity for Determining Product Complexity Level in COCOMO II. *Procedia Computer Science*, *124*, 478–486. https://doi.org/10.1016/j.procs.2017.12.180

Suherman, I. C., Sarno, R., & Sholiq. (2020). Implementation of Random Forest Regression for COCOMO II Effort Estimation. *2020 International Seminar on Application for Technology of Information and Communication (ISemantic)*, 476–481. https://doi.org/10.1109/iSemantic50169.2020.9234269

Tahir, F., & Adil, M. (2018). An Empirical Analysis of Cost Estimation Models on Undergraduate Projects Using COCOMO II. *2018 International Conference on Smart Computing and Electronic Enterprise (ICSCEE)*, 1–5. https://doi.org/10.1109/ICSCEE.2018.8538361

Yadav, R. (2017). OPTIMIZED MODEL FOR SOFTWARE EFFORT ESTIMATION USING COCOMO-2 METRICS WITH FUZZY LOGIC. *International Journal of Advanced Research in Computer Science*, *8*, 121–125. https://doi.org/10.26483/ijarcs.v8i7.4113

Zhang, L. (2019). The Research on General Case-Based Reasoning Method Based on TF-IDF. *2019 2nd International Conference on Safety Produce Informatization (IICSPI)*, 670–673. https://doi.org/10.1109/IICSPI48186.2019.9095927