Comparative Analyisis of Software Cost Estimation Project using Algorithmic Method

Muhammad Indra Zul Aqlani
@muhammadindra99@gmail.com

Nuning Septiana
nuning.s123@gmail.com

Abstrak
Software Cost Estimation has become an important factor to determine the efficiency of software development. There are many model of cost estimation like algorithmic model, top-down, and expert judgement. From all those models, Development in Algorithmic model is higher than the others. In this paper we present a comparative analysis of software cost project using algorithmic methods.

1. Introduction

Software cost estimation is one of the most important parts in designing a software. Software cost estimation becomes one of the considerations in determining the efficiency of software development. Unfortunately the cost estimation to develop the software has a weakness on the level of accuracy. Apart from the many methods, procedures, tools are still needed a lot of improvement. The approximate cost of software is related to how long and how many people are required to complete a project. The estimated cost of the software begins on the proposal of manufacture and will continue throughout the life of the project. The cost estimation process includes size estimates, business estimates, development of preliminary project schedules and ultimately estimates overall project cost.

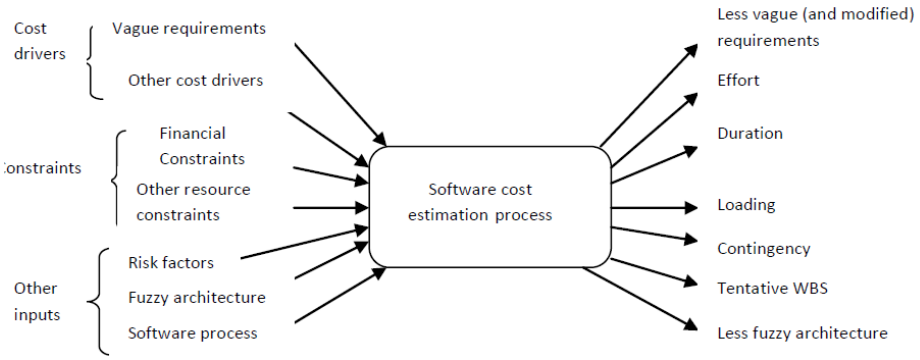
The development of cost estimation becomes very important, as it becomes one of the keys to the success of a software development project.Therefore, to manage budget and software project schedules [1], various software cost estimation models have been developed. Accurate software cost estimates are essential for developers and customers [2].

Development cost estimation is very much from hybrid research methodology to design morphological-rank-linear (MRL) perceptrons in the problem of software development cost estimation (SDCE) [13] and research about an evolutionary morphological approach to solve the software development cost estimation (SDCE) problem [14] both discussed how to develop SDCE with hybrid sytem. Another study also discusses a novel Constructive Cost Model (COCOMO) based on soft computing approach is proposed for software cost estimation. This model carries some of the desirable features of neural networks, such as learning ability and good interpretability, while maintaining the merits of the COCOMO model [15]. This proves that the cost estimation can still be developed further.

Many factors affect the accuracy of cost estimation in software development. This is important to discuss, here are some reasons why accuracy in cost estimates is important:

- Helps classify and prioritize the overall business plan development project.
- Used to determine what resources should be given to the project and how well these resources will be used.
- Projects can be more easily managed and controlled when resources are more suited to real needs.
- Customers expect actual development costs to be consistent with cost estimates.

Other factors that also affect cost is the ability of the programmer, the experience of the developer area, the complexity of the project. Pic1 describes the input and output of the software cost estimation process [3].



Pic1. Input & Output to the Estimation Process

Thus it can be concluded that the production of Software timely and commensurate directly depends on the initial estimate of the required costs, effort and resources. All cost prediction models are divided into one of these two categories[11]:

a) Parametric Model or Algorithm.
b) Non-Algorithm Method.

The Algorithm model is based on a mathematical formula and depends on the measurement and processing of a particular attribute project. The non-Algorithm model is a proposed heuristic it requires reasoning, logic and considerable knowledge of basing. In non algorithmic methods we find while in the Algorithmic method we calculate The algorithm method is good because it has been improved and well defined step by step procedure to provide estimation. The algorithm m odel is good because it does not require an outside parameter that requires tuning or calibration according to the measurement environment. In this study we made a comparison of the algorithmic method in software cost estimation which will provide knowledge of suitable algorithmic model applied to various projects.

2. Software Cost Estimation Overview

Cost estimation has been developed since 1960. There are many methods of cost estimation but basically there are six main methods which consist of:

1. Algorithmic (Parametric) Model
2. Expert Judgment (Expertise Based)
3. Top - Down
4. Bottom - Up
5. Estimation by Analogy
6. Price to Win Estimation

In this paper, we only analyze cost estimation using Algorithmic model. This technique use mathematical equation to perform the software estimation. The mathematical equations are based on historical data or theory.

Generally, there are 4 methods of Algorithmic-based cost estimation which will described below:

1. COCOMO Method (Constructive Cost Model)

COCOMO stands for Constructive Cost Model. It was first introduced by Barry Bohem in 1981. COCOMO computes software development effort and cost as a function of program size expressed in estimated lines of code. The basic steps of COCOMO are described below.

First, COCOMO obtain initial estimate using KLOC. Then, they determine a set of multiple factors from different attributes of the project. Finally, they adjust the effort estimate by multiplying the initial estimate with all multiplying factor.

2. Function Point Analysis
The FPA model was proposed by Albrecht in 1979. Its estimation based on the functionalities of the project. The main advantage

Their basic steps for calculating function point metric is:

1. Count total is calculated using information domain and the weighting factor
2. The value added factor is based on the response to the determined characteristics, each involving a scale and the empirical constands.
3. Function point is the product of count total and the value added factor

Besides estimating the cost, FPA can be used to predict the number of error during testing and forecast the number of component from the projects. However, counting function point is sometimes subjective and complicated.

G. Karner overcome this limitation by develope it as use case point in 1993, modify the function point with use case point. [16]

3. Putnam Model
Larry Putnam introduced the quantitative software measurement by 1970s. It is based on the analysis of the life-cycle in terms of Rayleigh distribution of project personel level versus time. SLIM used productivity level to derive the software equation where.

4. Walson Felix
Walston and Felix developed their effort model in 1977. It provides a relationship between delivered ines of source code. This model constitutes participation, customer-oriened changes, memory constraints. Etc.

3. Software Cost Estimation Paper

COCOMO with Neural Network
COCOMO model is a regression based software cost estimation. The first model was released in 1981 called COCOMO 81. The problem using COCOMO81 is it does not match the development of the late 1990s. Therefore they released the second version called COCOMO II in 1997. There are three models of COCOMO II:

- Application Composition Model - suitable for pojects built with GUI-builder tools.
- Early Design Model - to get rough estimates of a project. It use a small set of new cost drivers.

- Post-Architectural Model - The most detailed model. It could use fuction points or LOC as size estimates.

COCOMO II describes 17 cost drivers that are used in the Post-Architecture model. The cost drivers are rated on a scale from Very Low to Extra High. The post architecture model is given as:

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

$$\text{where } B = 1.01 + 0.01 \times \sum_{j=1}^{5} \text{Scale Factor}_j$$

In "1":

A: Multiplicative Constant

Size: Size of the software project measured in terms of KSLOC (thousands of Source Lines of Code, Function Points or Object Points)

Meanwhile, the use of neural network is to estimate PM (person-month). It requires 24 input nodes in the input layer in network that corresponds to all EM and SF. To accomplish this, a specific hidden layer and a sigmoid activation function with some pre-processing of data for input layer is considered.[4]

Based on the evaluation using MMRE, COCOMO II with neural networks model has better performance than the older COCOMO with 12.7% improvements. Further research is how to optimize the neural network itself in COCOMO II.

Fuzzy Emotional COCOMO II using Multi-Agent System (FECSE)

COCOMO II only considered the project characteristic. But in reality, considering the characteristic of team member is important because having an agile team is a significat element for the success of the project. Hence, a fuzzy agents and multi-agent system have been used to simulate personal characteristics and interactions in team. The personal characteristics and interaction will be the factors of personality and emotional in the productivity of team member.

The main goal of the FECSE model is considering team characteristic to make COCOMO II more accurate.There are two kinds of agent in FECSCE: "Team Member Agent" (TMA) and "Simulator Agent" (SA). TMA is a fuzzy agent for the simulation of team member. Multi-TMA simulates the team and the communication of TMA reflects the intracommunication of team. In this paper, it only considered the direction, not the quality of communication. A mesage from SA initializes the internal variables of each TMA.
The communication of TMA is displayed in the figure below

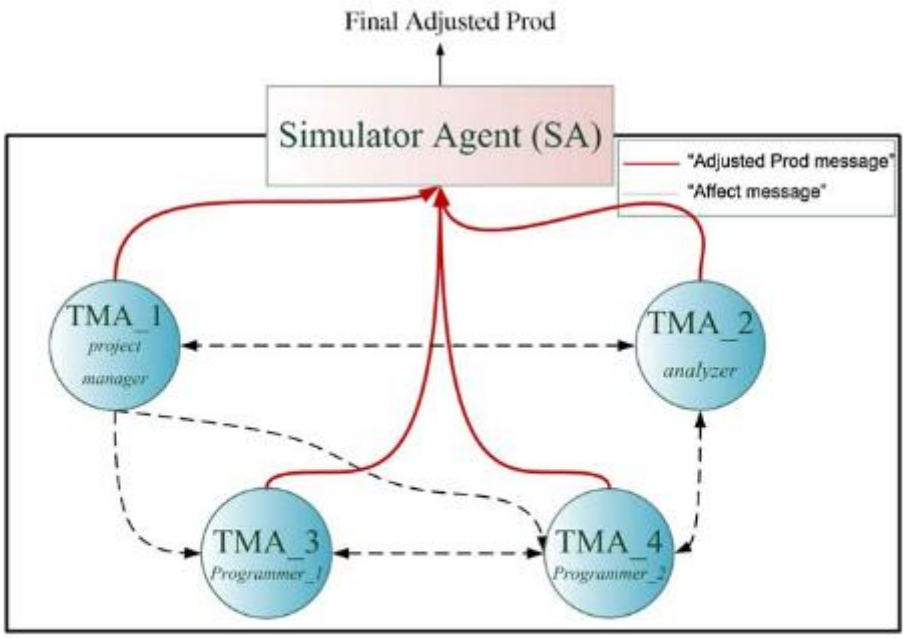


Fig. 4. The communication schema of FECSCE shows the communication between TMAs and SA. The team contains four members: a project manager, an analyzer, and two programmers. They are simulated by TMAs. The thick lines indicate the communications between TMAs and SA; the broken lines indicate the communications between the TMAs.

The model was evaluated with three projects gained from the companies. the result showed improvement of cost estimation. Future works can be considering the quality of communication and use more personality factor to gain better accuracy.

HOD-COCOMO Algorithm

HOD-COCOMO stands for Human Opinion Dynamic-Constructive Cost Model. It optimize the coefficient of COCOMO so that MMRE can be reduced. Generally, effort of COCOMO can be calculated by equation below

Effort = a* (size) b +EAF+C

In this equation, size is measured in terms of LOC or KLOC. Effort multiplier is defined by EAF. The initial value of a and b are fixed from COCOMO, but it can change from the organization requirement. So a and b parameter should be assigned to improve the accuracy and reduce the error. Below is the algorithm of HOD-COCOMO.

Fitness function is used for finding the best value and result opinion. Every objective contain some weight for combining two objective in to a single objective. The total weight will be equal to 1. The design fitness function in here will involved MMRE and prediction error

The proposed model is compared with the traditional COCOMO. The result is HOD-COCOMO has lower MMRE than the other. Further works will apply Meta-Heuristic algorithm to get better result of HOD-COCOMO.

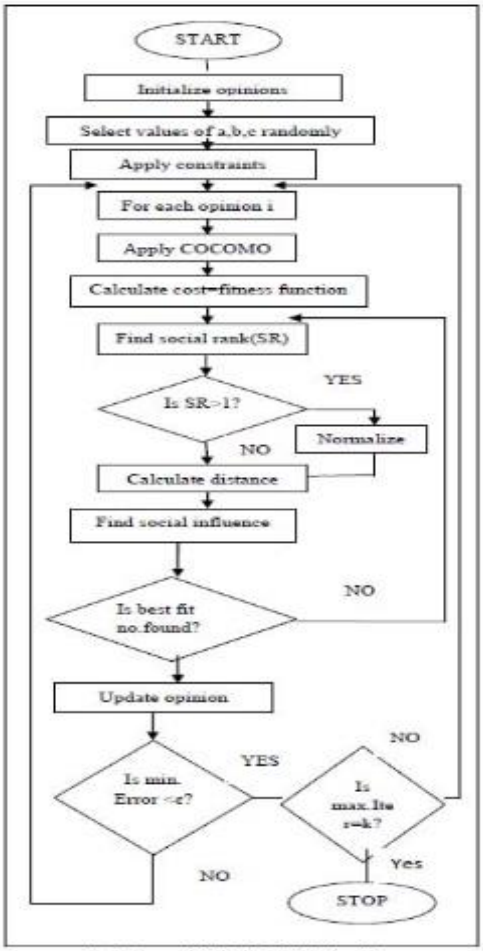


Fig 2. Proposed HOD-COCOMO Algorithm

Use Case Point Method with Modified set of Environmental Factors

UCP method is the extension of Function Point method with the benefit of requirement analysis in object-oriented process. It starts with measuring the functionality of the system based on the use case model. UCP has 3 factors: They are UUCP, Technical Factor, and Environmental Factors.The environmental were modified in order to gain more accurate estimation.[6] To estimate the cost using this methods, first we should identify the complexity of each use case. After that, we can assign a weight factor for each use case based on the level of complexity. Then we calculate the total weight of all factor. This research add new factor of environmental which are:

Client type,New Technology, Team Co-ordination, Growth Rate of Organization, Team Composition, Organization Library Availability.

The new factors may give more accurate result, but the method to assign the weight for new factor must be well-defined. If not, it will lead to catastrophicresults.
Future works is how to assign the weight of new factor

Use Case Point Method with Tree Boost Model
Treebost Model is a method to increase the accuracy of a predictive function by applying the function frequently in a series and combining the output of each funcions.[7] This model consist of a series of trees.

Below are the treeboost algorithm

$$F(x) = F_0 + A1 * T1(x) + A2 * T2(x) + ... + AM * TM(x).$$

Where F(x) is predicted target, F0 is the starting value and is the mean of the target variable (Software Effort), x is a vector which represent pseudo-residuals, T1(x) is the first tree of the series that fits the pseudo-residuals and A1,A2,etc are coefficient of nodes. The variable that were used for Treeboost model are independent variables which include software size, productivity, and complexity.

Based on experiment using four criteria, Tree Boost Model has better result than using multiple linear regression model but it works more complicated, and not recommended to use for project with size more than 2500. Future works will be calibrating treeboost model when new dataset available

Enhancing UCP Estimation Method using Soft Computing Techniques
UCP present some limitations that affect the accuracy. the limitation is the abence of the graduation when classifiying the complexity of the use case. This paper presents techniques using fuzzy logic and neural network to improve the accuracy of the use case point method.

First the use case will be classified as ux, suc as x e [1,10] where x represents number of transactions. There will be ten degree complexity of use case. The approach will be be implemented in two stages.

First, a fuzzy logic is applied to determine the complexity factor of ux. Second, neural network model takes ux (10 vector) as an input, in addition to 3 vector which represent the 3 types of actor (simple, average, or complex). The output will be the size of the software.The proposed fuzzy model compared with the Karner's model and the result is an improvement by 22% in MMRE and 9% in MMER. While the proposed Neural Network model has improvement by 20%.

Future work will focus on revamping the use case model. The largest use case should contain at least 20 transactions. Second, the complexity weight of use case will be calibtrated using neuro-fuzzy. 'Extend' and 'include' use case should be considered.

Putnam Model

Putnam's model and SLIM: Putnam's model is proposed according to manpower distribution and examination of many software projects. Software equation for puntam's model is as follows[12]:

$$S = E * Effort^{1/3} * td^{4/3}$$

where td is the software delivery time; E is the environment factor that reflects the development capability, which can be derived from historical data using the software equation. The size S is in LOC and the Effort is in person-year. Another important relation found by Putnam is

$$Effort = D0 * td^{3}$$

where D0, is a manpower build-up factor, which ranges from 8 (new software) to 27 (rebuilt software). Com- bining above 2 equations, final equation is obtained as :

$$Ef\ f\ ort = D0^{4/7} * E^{-9/7} * S^{9/7} \text{ and}$$
$$td = D0^{-1/7} * E^{-3/7} * S^{3/7}$$

Putnam's model is also widely used in practice and SLIM is a software tool based on this model for cost estimation and manpower scheduling. One significant problem with the Putnam model is that it is based on knowing, or being able to estimate accurately, the size (in lines of code) of the software to be developed. There is often great uncertainty in the software size. It may result in the inaccuracy of cost estimation.(Kumari & Pushkar, 2013)[10]

Walston-Felix Model

Walston and Felix (1977) developed their effort model from a various aspects of the software development environment such as user database of sixty projects collected in IBM's Federal Systems division. It provides a relationship between delivered lines of source code. This model constitutes participation, customer-oriented changes, memory constraints etc. According to Walston and Felix model, effort is computed by [Kumari & Pushkar, 2013]:

EFFORT = 5.2 (KLOC) $^{0.91}$, Duration D = 4.1 (KLOC)

4. Analysis Comparison Advantage and Disadvantage

Analysis comparison COCOMO

Name	Method	Advantage	Disadvantage
COCOMO with Neural Network	Utilize Neural Network in post architecture model of COCOMO II	Overcome the characteristic of uncertainty and vagueness in COCOMO	The performance depends on the size of training data. Need to spesific the model of Neural Network
FECSE	Considering team characteristic such as emotion and personality factor, and the communication of team member inside COCOMO II. Two kinds agent used in here are Team Member Agent and Simulator Agent	Give more accuracy thanks to the new factor 'team characteristic'	Too many factors. No standard method to get the value of 'team characteristic.
HOD-COCOMO	Human Opinion Dynamic-Constructive Cost Model. It optimize the coefficient of COCOMO so that MMRE can be reduced.	Give better exact results because of the optimized coefficient.	Unclear how to obtain the opinion of the project

Analysis comparison Use Case Point

Name	Method	Advantage	Disadvantage

Use Case Point Method with Modified set of Environmental Factors	adding the following six environmental factors for increase the result of the UCP method more accurate and precise	real time Systems, obtain precise and accurate result.	have not tried on some other data (can not be a reference)
Use Case Point Method with Tree Boost Model	put forward a novel Treeboost model to predict software effort from use case diagramsThe inputs of our reeboost model include Software size, team productivity and project complexity.	improve the accuracy of decision trees Models for predict software effort from use case diagrams	calibrating model Treeboost when new datasets are available
UCP Estimation Method using Soft Computing Techniques	enhancement model using fuzzy logic and neural network	Fuzzy : approach presents ten degrees of complexity of the use cases, approach provides graduation among the complexity weight (in UCP havent) Neural network : approach was used as a black box to map the input vectors of the use case model to software size. The results showed that the UCP software estimation can be improved up to 22% in some projects	Can not handle the largest use case should contain at least twenty transactions.

Name	Type	Advantage	Disadvantage
Cocomo	Algoritmic	Clear results, very common	Much data is required, It ‘s not suitable for any project,
UCP	Algoritmic	obtain precise and accurate result.	Can not handle the largest use case should contain at least twenty transactions.
Function Poin	Algoritmic	Language free, Its results are better than SLOC	Mechanization is hard to do , quality of output are not considered

Putnam	Algoritmic	Fast and easy. Fit for large scale project.	Lack of some details required to estimate Parts of the software.
Walston & Felix	Algoritmic	Based on historical statistical data.	Haven't been practiced a lot because of Statistical problems.

5. CONCLUSION

We present a comprehensive analysis of software cost estimation using parametric methods. Each methods has advantage and disadvantage. But out of four general parametric methods we explained before, COCOMO and FPA (UCP) has more development or optimization than Putnam model and Waltson-Felix. Based on that, We recommend reader to use COCOMO or UCP. A question about which optimization COCOMO or UCP we should use is depend with our needs. Some methods like FCSE are good for big project which consist of many workers. The enhancement using Neural Network or Fuzzy in COCOMO or UCP can derive more accurate result but not suitable for large project.

Referensi

[1] L.H. Putnam, A general empirical solution to the macro software sizing and estimation problem,. IEEE Transactions on Software Engineering, pp. 345.361, , July 1978.

[2] Caper Jones, Estimating software cost. tata Mc- Graw -Hill Edition 2007.

[3] S. Kumari and S. Pushkar, “Performance Analysis of the Software Cost Estimation Methods : A Review,” *Int. J. Adv. Res. Comput. Sci. Softw. Eng. Res.*, vol. 3, no. 7, pp. 229–238, 2013.

[4] Attarzadeh, I., & Ow, S. H. O. S. H. (2010). A novel soft computing model to increase the accuracy of softwAttarzadeh, I., & Ow, S. H. O. S. H. (2010). A novel soft computing model to increase the accuracy of software development cost estimation. Computer and Automation Engineering (ICCAE), 2010 The. *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, *3*, 603–607. https://doi.org/10.1109/ICCAE.2010.5451810

[5] Kazemifard, M., Zaeri, A., Ghasem-Aghaee, N., Nematbakhsh, M. A., & Mardukhi, F. (2011). Fuzzy Emotional COCOMO II Software Cost Estimation (FECSCE) using Multi-Agent Systems. *Applied Soft Computing Journal*, *11*(2), 2260–2270. https://doi.org/10.1016/j.asoc.2010.08.006

[6] Jena, P. P., Jha, P., Jena, P. P., & Malu, R. K. (2014). Estimating Software Development Effort using UML Use Case Point (UCP) Method with a Modified set of Environmental Factors Estimating Software Development Effort using UML Use Case Point (UCP) Method with a Modified set of Environmental Factors, *5*(April), 9–12.

[7] Nassif, A. B., Capretz, L. F., Ho, D., & Azzeh, M. (2012). A Treeboost Model for Software Effort Estimation Based on Use Case Points. *2012 11th International Conference on Machine Learning and Applications*, 314–319. https://doi.org/10.1109/ICMLA.2012.155

[8] Nassif, A. B., Capretz, L. F., & Ho, D. (2010). Enhancing Use Case Points Estimation Method Using Soft Computing Techniques. *Journal of Global Research in Computer Science*, *1*(4), 12–21.

[9] Jain, R., Sharma, V. K., & Hiranwal, S. (2017). Reduce mean magnitude relative error in software cost estimation by HOD-COCOMO algorithm. *2016 International Conference on Control Instrumentation Communication and Computational Technologies, ICCICCT 2016*, 708–712. https://doi.org/10.1109/ICCICCT.2016.7988044

[10] Suelmann, H. (2014). Putnam's effort-duration trade-off law: Is the software estimation problem really solved? *Proceedings - 2014 Joint Conference of the International Workshop on Software Measurement, IWSM 2014 and the International Conference on Software Process and Product Measurement, Mensura 2014*, 79–84. https://doi.org/10.1109/IWSM.Mensura.2014.25

[11] Kashyap, D., & Misra, A. K. (2013). Software development cost estimation using similarity difference between software attributes. *Proceedings of the 2013 International Conference on Information Systems and Design of Communication - ISDOC '13*, 1. https://doi.org/10.1145/2503859.2503860

[12] Borade, J. G., & Khalkar, V. R. (2013). Software Project Effort and Cost Estimation Techniques. *International Journal of Advanced Research in Computer Science and Software Engineering*, *3*(8), 730–739.

[13] R. D. A. Araújo, S. Soares, and A. L. I. Oliveira, "Expert Systems with Applications Hybrid morphological methodology for software development cost estimation," vol. 39, pp. 6129–6139, 2012.

[14] R. D. A. Araújo, A. L. I. Oliveira, S. Soares, and S. Meira, "An evolutionary morphological approach for software development cost estimation," *Neural Networks*, vol. 32, pp. 285–291, 2012.

[15] I. Attarzadeh and S. H. Ow, "A Novel Soft Computing Model to Increase the Accuracy of Software Development Cost Estimation," pp. 603–607.

[16] Karner, G. (1993). Resource Estimation for Objectory Projects, 1–9.