JURNAL INFOTEL
Informatics - Telecommunication - Electronics

Website Journal: http://ejournal.st3telkom.ac.id/index.php/infotel
ISSN: 2085-3688; e-ISSN: 2460-0997

Effort Estimation For Software Development On Mobile Application of 'Tangkap Reptil'

Condro Kartiko*[1], Novanda Alim Setya Nugraha[2], Dery Sudrajat[3]
[1,3] Department of Software Engineering, Institut Teknologi Telkom Purwokerto
[2] Department of Informatics, Institut Teknologi Telkom Purwokerto
[1,2,3] Jl. D.I Panjaitan, No.128, Purwokerto 53147, Indonesia
*Corresponding email: condro.kartiko@ittelkom-pwt.ac.id

Received 26 November 2019, Revised 16 December 2019, Accepted 19 December 2019

Abstract — An essential aspect of planning and management of software design projects is to estimate work time, costs, and human resources. The calculation solution made in this study aims to assist in calculating the estimated time of developing a reptile capture application using the Use Case Point (UCP) method. The UCP method is a software effort estimation method that shows better performance compared to other methods. The result of this research is the risk of software development on the mobile application of 'Tangkap Reptil' has a small chance, can be done in a relatively short time, and does not require a lot of resources. The development of mobile-based "Tangkap Reptil" applications in the case study used in this study took approximately 3,233,608 hours of work, with a relatively low risk of project development. If it is will assume in one day worked by one person has 8 hours of work, so the total duration of time required for 404,201 days or 13.47 months or 1.12 years.

Keywords – Use Case Point, Effort Estimation, Aplikasi Android

I. INTRODUCTION

One exciting task in the software development life cycle is the estimations of effort. Software effort estimation is one important aspect that determines the success of a software project. Many software projects fail due to poor project management [1], including the calculation of bad software estimates. Appropriate estimation is critical to the success of a software project. If the estimate is too low than it should be, it will result in incomplete software projects due to time and cost. But if the estimate is too high, it will cause the software project to be unrealistic to be carried out.

Although project effort estimation plays a very crucial role in the success of a project, there are still many problems that exist in software effort estimation. For example, lack of information on a software project at the beginning of the project, making it difficult to estimate the software project. Another problem is the sustainability of the project, which is sometimes still not certain, thus requiring the decision holder to make resource efficiency.

In recent years, software effort estimation has received a considerable amount of attention from researchers and became a challenge for the software industry. In the last two decades, many researchers and practitioners proposed statistical and machine learning-based models for software effort estimation [2]. Software effort estimation can use machine learning models such as Linear Regression (LR), Support Vector Machine (SVM), K-Nearest Neighbor (KNN), Multi-Layer Perceptron Neural Network (MLPNN), Probabilistic Neural Network (PNN), Recurrent Neural Network (RNN) [3]. Some cost model methods for estimating projects such as COCOMO, Function Point Analysis (FPA), Mark II Function Points, and COSMIC [4] are widely applied in the industry. These models use parametric estimation models in their implementation. The model still has a weakness that is needed by people who have experience or are very familiar with calculating points for each function of the estimated software [1]. The development of methods for evaluating software that can overcome these weaknesses includes Use Case Point (UCP) estimation. The UCP was designed to carry out estimations at the

start of a software project [5]. UCP is also a development of Function Point Analysis for object-oriented applications [6] [7]. The UCP method is an initial estimation based on a use case diagram, where UCP can be used to understand problems that might occur in software, project size estimates, and general system architecture. Software effort estimation based on UCP methods provides only fixed estimation value, which cannot deal with the uncertain and ambiguous conditions of the particular software [8]. This method estimates the number of attempts based on the complexity of the use case [6].

Several previous studies on UCP have reported the following results. In [9], a comparison between estimated efforts using UCP and actual efforts has a deviation rate of 19%. In contrast, estimates by experts have a deviation rate of 20%. Two other studies reported that UCP had a deviation rate of 6% [10] and 9% [11] compared to actual efforts. UCP can also be used to estimate the costs involved in developing large and medium scale software projects [12] and enterprise resource planning software [13]. Research [14] makes an estimated inventory system of goods using the UCP estimation method. Thus, it seems that UCP can be used reliably for effort estimation.

The purpose of this study is to estimate reptile encyclopedia applications using the UCP estimation method. The case study used is in the development project of mobile devices based on reptile capture applications. The results of this study are expected to be one of the examples and references for developing other mobile-based applications. So that people who will develop a mobile-based application have a clear general picture of the estimated size and general architecture of the mobile-based application.

II. RESEARCH METHOD

Estimation in the UCP method starts with measuring the complexity of the actor (unadjusted actor weight) and use case (unadjusted use case weight) in the system. The next step is to measure the technical complexity (technical complexity factor) and environmental (environmental factor) system used in the development of software projects. After that, measurements on the UCP, productivity factor, and effort estimation. The details of the UCP process flow are as follows.

A. Actor Complexity

The division of actor complexity in the UCP measurement method into three categories, among others

a) Simple: represent systems that communicate with other systems using the Application Programming Interface (API)
b) Medium: represent systems that communicate with other actors using special protocols such as HTTP and FTP, or humans who communicate with the system using the terminal console.
c) Complex: humans who use the Graphical User Interface (GUI) to communicate with the system.

For each class described previously, an Unadjusted Actor Weight (UAW) calculation is calculated for the weight of each actor, where the weighting rules are three for complex actors, two for medium actors and one for simple actors. For the calculation of the UAW formula is as follows:

$$\text{UAW: } \sum \text{weight (c)} \times \text{number of actors (c)} \quad (1)$$

wherein:

c : Class of the actors concerned (simple, medium, complex)

B. Use Case Complexity

The next step is to measure the complexity of the use case. In contrast to actor complexity, the use case complexity is estimated based on the transaction complexity of each use case. Transactions in the use case are activities/scenarios contained in each use case. The complexity of the transaction from the use case is divided into three categories, *i.e.*

$$\text{use case complexity} \begin{cases} \textit{simple} \text{ if transaction} < 4 \\ \textit{average} \text{ if} \leq 4 \text{ transaction} \leq 7 \\ \textit{complex} \text{ if transaction} > 7 \end{cases}$$

For each class described the use case complexity calculation is done by calculating the Unadjusted Use Case Weight (UUCW), where the weighting rule is five for the simple use case, 10 for the average use case and 15 for the complex use case. For the calculation of the UUCW formula is as follows

$$\text{UUCW} = \sum \text{weight (c)} \times \text{number of use case (c)} \quad (2)$$

wherein:

c : The class of the use case in question (simple, medium, complex)

C. Technical Factors and Software Development Environment

The UCP method uses 21 measurements to measure the technical factors and system development environment, which is divided into 13 parameters to measure the system technical (technical complexity factors) and eight parameters to measure the system development environment (environmental factors). Details of the technical complexity factor are shown in Table 1, and the environmental factor (EF) is shown in Table 2.

Table 1. Technical Complexity Factors

Technical Complexity Factors		
Factor	**Description**	**Weight**
T1	*Distributed System*	2
T2	*Performance*	1

Technical Complexity Factors		
Factor	**Description**	**Weight**
T3	*End-User Efficiency*	1
T4	*Complex processing*	1
T5	*Reusable code*	1
T6	*Easy to install*	0,5
T7	*Easy to use*	0,5
T8	*Portable*	2
T9	*Easy to change*	1
T10	*Concurrent*	1
T11	*Security features*	1
T12	*Access for third parties*	1
T13	*Special training required*	1

Table 2. Environmental Complexity Factors

Environmental Complexity Factors		
Factor	**Description**	**Weight**
F1	*Distributed System*	1,5
F2	*Performance*	0,5
F3	*End-User Efficiency*	1
F4	*Complex processing*	0,5
F5	*Reusable code*	1
F6	*Easy to install*	2
F7	*Easy to use*	-1
F8	*Portable*	-1

The influence of Technical Complexity Factor (TCF) on the UCP measurement method is in the range from zero (0) to five (5), where the higher the influence of the factors concerned in the project work, the higher the estimated value of the effect. TCF calculation is multiplying the weight of the influence assessment, with the weighting provisions in Table 1. The formula for UCF calculation is as follows.

$$\text{TCF} = 0.6 + (0.01 \times \sum_{i=1}^{13} \text{factor weight}_i \times \text{value of influence}_i) \quad (3)$$

While the influence of EF is done the same calculation with TCF, which gives an estimated value of each factor in accordance with its influence in the range of zero (0) to five (5) then multiplied by the weight of each factor according to Table 2. The formula for EF calculation is as follows.

$$\text{EF} = 1.4 + (-0.03 \times \sum_{i=1}^{8} \text{factor weight}_i \times \text{value of influence}_i) \quad (4)$$

D. UCP

To calculate the UCP, an Unadjusted Use Case Point (UUCP) calculation must be added by adding an UAW with an UUCW. The formula for calculating UUCP is shown in (5).

$$\text{UUCP} = \text{UAW} + \text{UUCW} \quad (5)$$

After UUCW is obtained, a UCP calculation is performed using the formula shown in (6).

$$\text{UCP: UUCP} \times \text{TCF} \times \text{EF} \quad (6)$$

wherein:

UUCP : Unadjusted Use Case Points
TCF : Technical Complexity Factors
EF : Environment Factor

E. Productivity Factor dan Effort Estimation

The results of the effort estimation value from the multiplication between the productivity factor and the UCP value. To get the benefit of productivity factor according to Schneider and Winters calculations, if the EF value is ≤ two, the productivity factor value is 20 for each UCP, if the EF value is in the range of three and four, the productivity factor value is 28 for each UCP, if the EF value is more than 4, the productivity factor value used is 36 for each UCP [15].

III. RESULT

In this study, the use of UCP techniques to study in general, the efforts and risks in mobile-based "Tangkap Reptil" application software projects. Fig. 1 shows the flow diagram in this study regarding the implementation of the UCP technique.

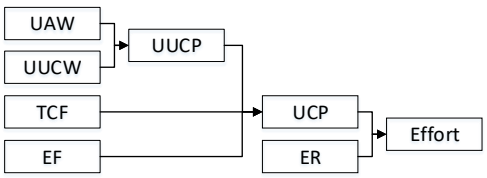

Fig 1. The Methodology For Determining Software Effort Estimation

Use the mobile-based reptile capture application case in this research case study in Fig. 2. Based on the mobile-based "Tangkap Reptil", the application has 25 use case cases with two main actors, namely general and admin. General user actors have ten use cases. Use cases for public users, among others see the application in two languages, look at the main menu, see the type of reptile, see a detailed description of reptiles, see the news, see FAQ, see the event, see the video, see the reptile community, and access the help reptile. The admin user actor has 15 use cases. Use cases for admin users, among others enter reptile type, change reptile type, removing reptile type, enter news, change news, delete news, enter the reptile community, change the reptile community, delete the reptile community, enter event, change event, delete event, enter FAQ, change FAQ, and delete FAQ. The steps in the UCP estimation process in mobile-based reptile capture applications are as follows:

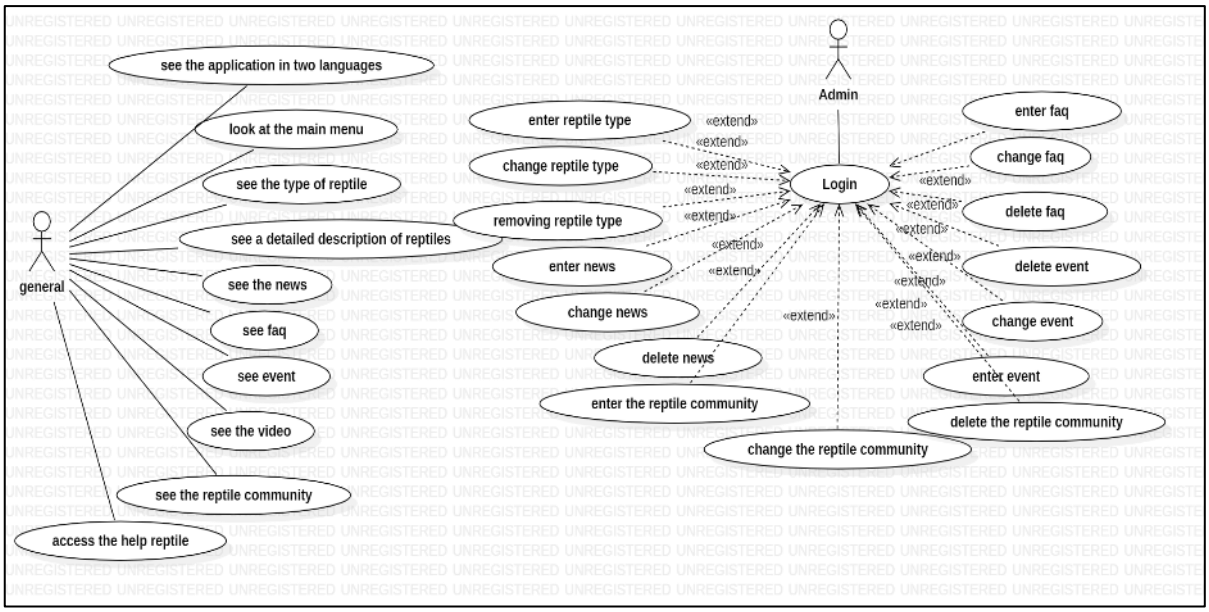

Fig 2. Use case "Tangkap Reptil" Application Based On Mobile

A. Actor Complexity

Calculation of the extended UCP method starts with determining the value of the Unadjusted Actor Weight. Actors in this system are divided into two (2), where the two actors in mobile-based "Tangkap Reptil" applications are human, so they are classified into complex actors that weight three (3), so the value of UAW is 6. Details of the calculation of UAW in "Tangkap Reptil" applications are listed in Table 3.

Table 3. Unadjusted Actor Weight

UAW			
Class	**Actor Amount**	**Weight**	**Number of Actors × Weight**
Simple	0	1	0
Medium	0	2	0
Complex	2	3	6
Total UAW			6

B. Use Case Complexity

The mobile-based "Tangkap Reptil" application in this study has 25 use cases which are looking at the main menu, seeing reptile types, seeing detailed reptile descriptions, viewing news, viewing FAQ, viewing events, viewing videos, viewing reptile communities, accessing reptile help, viewing applications in two language, enter reptile types, change reptile types, delete reptile types, enter news, change news, delete press, enter reptile communities, change reptile communities, delete reptile communities, enter events, change events, delete occurrences, insert FAQ, change FAQ, and delete the FAQ. The 25 use cases will be divided into three classes, namely simple, average, and complex, based on the number of transactions contained in it to get an UUCW. Details of the use case distribution based on its complexity are shown in Table 4, and UUCW results are shown in Table 5.

Table 4. Distribution of Use Case Complexity

Use Case	**Transaction**	**Class**
see the application in two languages	7	*average*
look at the main menu	2	*simple*
see the type of reptile	2	*simple*
see a detailed description of reptiles	2	*simple*
see the news	4	*average*
see FAQ	1	*simple*
see event	1	*simple*
see the video	1	*simple*
see the reptile community	1	*simple*
access the help reptile	7	*average*
enter reptile type	3	*simple*
change reptile type	3	*simple*
removing reptile type	3	*simple*
enter news	3	*simple*
change news	3	*simple*
delete news	3	*simple*
enter the reptile community	3	*simple*
change the reptile community	3	*simple*
delete the reptile community	3	*simple*
enter event	3	*simple*
change event	3	*simple*
delete event	3	*simple*
enter FAQ	3	*simple*
change FAQ	3	*simple*
delete FAQ	3	*simple*

Next, determine the value of UUCW obtained by multiplying the weight by the number of actors according to the Use Case type. UUCW calculations yield a total of 140, presented in Table 5.

Table 5. Unadjusted Use Case Weight

UUCW			
Class	**Number of Use Cases**	**Weight**	**Number of Use Case × Weight**
Simple	22	5	110
Average	3	10	30
Complex	0	15	0
Total UUCW			140

C. Technical Factors and Software Development Environment

The next process is to calculate the TCF and EF. The weighting of the factor influence estimation from TCF and EF is carried out in the range of zero (0) to five (5), where the higher the influence/impact of these factors, the higher the weight of the estimation. Details of TCF weighting can be seen in Table 6.

Table 6. Technical Complexity Factors

Technical Complexity Factors				
Factor	**Description**	**Weight**	**Impact**	**Weight × impact**
T1	Distributed System	2	2	4
T2	Performance	1	5	5
T3	End-User Efficiency	1	3	3
T4	Complex processing	1	1	1
T5	Reusable code	1	4	4
T6	Easy to install	0,5	4	2
T7	Easy to use	0,5	4	2
T8	Portable	2	3	6
T9	Easy to change	1	2	2
T10	Concurrent	1	2	2
T11	Security features	1	3	3
T12	Access for third parties	1	2	2
T13	Special training required	1	2	2
Total Factor				38

From Table 6, the TCF value is obtained

$$TCF = 0.6 + (0.01 \times 38) = 0.98$$

Whereas the details of EF weighting are shown in Table 7.

Table 7. Environmental Complexity Factors

Environmental Complexity Factors				
Factor	**Description**	**Weight**	**Impact**	**Weight × impact**
F1	Familiarity with standart process	1,5	3	4,5
F2	Application experience	0,5	2	1
F3	Object-oriented experience	1	3	3
F4	Lead analyst capability	0,5	3	1,5
F5	Motivation	1	1	1
F6	Stable requirements	2	3	6
F7	Part-time workers	-1	4	-4
F8	Difficult programming language	-1	4	-4
Total Factor				9

From Table 7, the EF value is obtained

$$EF = 1.4 + (-0.03 \times 9) = 1.13$$

D. UCP

The next stage is to calculate the UCP by multiplying UUCW with TCF and EF; the results of the UUCP calculation are as follows.

$$UUCP = UAW + UUCW =: 6 + 140 =: 146$$

So, the UCP value is as follows

$$UCP = UUCP \times TCF \times EF = 146 \times 0{,}98 \times 1{,}13 = 161{,}6804$$

E. Productivity Factor dan Effort Estimation

Based on the calculation of the EF value, the productivity value used is 20 hours for each UCP, where the risk of project work is relatively low, and the effort estimation of work is 3,233,608 hours of work.

IV. DISCUSSION

Fig. 1 shows that the proposed methodology for determining software effort estimation. This methodology is for determining efforts using UCP. Also, to get software development time, efforts made using UCP will be distributed to each feature. The process aims to get effort into each feature.

The first step in this research is to get the complexity of the actors or UAW stated in Section II.A. There are

several categories for actors of each use case, including simple, medium, or complex. The grouping is useful for categorizing and managing each actor; the presentation in Table 3 shows the detailed criteria.

The second step in this research is to get the use case or UUCW complexity stated in Section II.B. UUCW expresses the complexity of the use case as measured by the number of transactions in the use case. Each use case in a categorized system is simple, average, or complex. Details of these criteria are in Table 4. The UUCW calculation results are the sum of the weights of each use case presented in Table 5.

The third step in this research is to obtain the TCF stated in Section II.C. TCF is used to estimate software size to consider system technical considerations. TCF determination by assigning weights between 0 (irrelevant factors) to 5 (essential factors) for each of the 13 technical factors listed in Table 1. Acquisition of TCF as a result of the calculation of the weight of each technical factor listed in Table 6.

The fourth step in this research is to obtain the UC stated in Section II.D. UCP is obtained by multiplying UUCP, TCF, and ECF. The final step in this research is effort calculation. Getting the value of effort by multiplying the value of UCP and ER constant in staff / UCP hours stated in Section II.E. The amount of effort needed to develop Reptile Capture mobile application software is in Section III.E.

V. CONCLUSION

Based on the process and steps in this research, we conclude that the acquisition of estimates for small-scale software development projects can use UCP. They were starting from the acquisition of UAW value of 6 and UUCW of 140. The two values are to calculate UUCP, which results are 146. Furthermore, calculation of the correlation between UUCP values, TCF values of 0.98, and EF values of 1.13 are added to produce UCP amounted to 161.6804. The development of mobile-based "Tangkap Reptil" applications in the case study used in this study took approximately 3,233,608 hours of work, with a relatively low risk of project development. If it is will assume in one day worked by one person has 8 hours of work, so the total duration of time required for 404,201 days or 13.47 months or 1.12 years. If it is assumed to work with a team of three people working hours of 30 hours per week, then this project should be completed in 36 workweeks.

In general, the risk of developing a mobile-based reptile capture application is not high, the processing time is relatively short, and the team required is not much. This research can be improved by measuring the actual results of making the whole system and using the UCP method for estimating other mobile-based application models in Indonesia so that it can be estimated the average effort needed in making a mobile application.

ACKNOWLEDGMENT

Thank you to the Directorate of Research and Community Service of the Directorate General of Research and Development Strengthening Ministry of Research Technology and Higher Education of the Republic of Indonesia for providing financial support in the Beginner Lecturer Research scheme so that this research can be carried out.

REFERENCES

[1] P. P. Jena and S. Mishra, "Survey report on software cost estimation using use case point method," *Int. J. Comput. Sci. Eng. Technol.*, vol. 5, no. 4, pp. 280–287, 2014.

[2] N. Ghatasheh, H. Faris, I. Aljarah, and R. M. H. Al-Sayyed, "Optimizing software effort Estimation models using firefly algorithm," *Journal of Software Engineering and Applications*, vol. 8, pp. 133-142, 2015.

[3] S. Shukla and S. Kumar, "Applicability of neural network based models for software effort estimation," in *2019 IEEE World Congress on Services (SERVICES)*, Milan, 2019.

[4] A. Abran, S. Oligny, C. Symons, D. S. Pierre, and J. M. Desharnais, "Functional size measurement methods - COSMIC-FFP: design and field trials," in *FESMA-AEMES Software Measurement Conference 2000*, 2000.

[5] M. M. Kirmani and A. Wahid, "Use case point method of software effort estimation: a review," *International Journal of Computer Applications*, vol. 116, no. 15, pp. 43-47, 2015.

[6] G. Karner, "Resource estimation for objectory projects," *Objective Systems SF AB*, 1993.

[7] G. I. Ibarra and P. Vilain, "Software estimation based on use case size," in *Brazilian Symposium on Software Engineering*, 2010, pp. 178-187.

[8] J. F. Vijay, "Knowledge based non-functional software distinguishing quality effort estimation using fuzzy use case point approach," *International Journal of Knowledge Management Studies*, vol. 10, no. 1, pp. 21-32, 2019.

[9] B. Anda, "Comparing effort estimates based on use cases with expert estimates," in *Empirical Assessment in Software Engineering(EASE)*, Keele UK, 2002, p. 13.

[10] S. Nageswaran. (2001, June) Test_Effort_Estimation.pdf.[Online].

[11] E. R Carrol, "Estimating software based on use case points," in *Object Oriented Programming Systems Languages and Applications (OOPSLA) Conference*, San Diego, 2005, pp. 257–265.

[12] K. Shaleh, "Effort and cost allocation in medium to large software development project," *International Journal of Computer*, vol. 5, no. 1, pp. 74-79, 2011.

[13] G. F. Prassida, A. H. N. Ali, and Sholiq, "Estimasi biaya pembuatan modul enterprise resource planning (erp) untuk unit bisnis pabrik gula di pt. perkebunan xyz dengan metode use case point," *Jurnal Teknik Pomits*, vol. 1, no. 1, pp. 1-6, 2012.

[14] R. Adhitama and C. Kartiko," Effort estimation menggunakan metode use case point untuk pengembangan perangkat lunak," *Journal of Informatics, Information System, Software Engineering and Applications (INISTA)*, vol. 1, no. 1, pp. 55-62, 2018.
[15] G. Schneider and J. P. Winters, Applying Use Cases: A Practical Guide (2nd Edition), Boston, Massachusetts: Addison-Wesley Professional, 2001.