Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

JURNAL TEKNIK INFORMATIKA

Homepage : http://journal.uinjkt.ac.id/index.php/ti

Use Case Point Activity-Based Costing and Adjusted Function Point for Software Cost Estimation

Ahmad Fauzan Haryono[1], Andria Farhan[2], Dewi Khairani[3*], Supardi Razak[4], Fitri Mintarsih[5]

[1,4]Mining Engineering, Faculty of Science and Technology, Syarif Hidayatullah State Islamic University Jakarta
[2,3,5]Informatics Engineering, Faculty of Science and Technology, Syarif Hidayatullah State Islamic University Jakarta
[1,2,3,4,5]Jl. Ir. H. Juanda No. 95, Ciputat, Tangerang Selatan, Banten, Indonesia

Article:
Accepted: July 14, 2025
Revised: June 16, 2025
Issued: October 30, 2025

***Correspondence Address:**
dewi.khairani@uinjkt.ac.id

ABSTRACT

Effective software development planning is vital across various industries, and inadequate planning can lead to project failures. A key part of this planning is accurately estimating effort and costs, which is crucial for staying within budget and meeting deadlines. This research compares two methods – Use Case Point Activity Based Costing (with 21 complexity factors) and Adjusted Function Point (with 16 complexity factors) – for estimating costs versus actual values. The analysis reveals that the Use Case Point method had a 23.52% deviation from actual costs, while the Adjusted Function Point method had a 33.35% deviation. These findings provide essential reference points during software project planning, ensuring estimates closely align with actual values based on project-specific attributes. This deviation underscores the importance of precision and selecting appropriate methodologies tailored to each project's unique characteristics. Ultimately, this research equips businesses and project managers with a robust financial prudence framework and enhances the likelihood of project success.

Keywords : *activity-based costing; adjusted function point; cost estimation; function points; software estimation.*

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

1. INTRODUCTION

Software Project Management (SPM) is a management process from software development that leads to planning, designing, implementing, testing, measuring, monitoring, and controlling software [1]. The correct planning stage is the initial stage in the success of a software project. Many business people in all fields need to understand this because poor planning is usually the cause of project failure. This study also states that the Function Point (FP) method is considered faster because it does not require system analysis, resulting in a reporting scheme using narrative drafts and databases. Some software project planning activities include resource estimation, cost estimation, and project schedule [2]. Cost estimation in software is the process of predicting the effort required to develop a software system [3]. Accurate estimates help us complete projects within the allotted time and budget. Incorrect software project planning will fail with a percentage of 66% due to exceeding the budget allocation 33% due to delays from the specified schedule [4]. Various techniques, estimation models, and tools for software estimation are used to provide accurate and reliable predictions of the effort, time, and cost required to complete a software development project. These techniques may include algorithmic models, expert judgment, historical data analysis, and simulation-based methods. Estimation tools may range from simple spreadsheets to sophisticated software applications that use artificial intelligence and machine learning algorithms to improve accuracy and automate estimation. Ultimately, software estimation aims to help project managers make informed decisions about resource allocation, risk management, and project scheduling to ensure successful project completion. [5].

Measuring the software that must be built or made can be a starting point for estimation and planning in software projects [6]. Software measurement is very important for software creation and the software life cycle. Software measurement must be carried out in the Software Development Life Cycle (SDLC) planning stage. SDLC is the most popular software development method in the technology industry. This method provides a structured and systematic approach to developing software, starting from the planning phase all the way through implementation and maintenance, many applications are developed using SDLC in example [7][8]. SDLC involves a series of well-organized phases, including requirements analysis, system design, code implementation, testing, and maintenance. Each phase has clear objectives and tasks, ensuring that the resulting software meets expected quality and functionality standards. The sequential approach in SDLC helps mitigate the risk of errors and enables developers to identify issues early on.

Software size is one of the main inputs to software development efforts; if the size is correctly estimated, the effort estimates will be realistic and translate to realistic cost estimates. After determining the software size, estimate the effort needed to build the required software product [9]. Software metrics and measurements are often used to determine how future software versions will look, such as by removing or adding features. According to previous research conducted by Adhitama [10], Use Case Point (UCP) performs better than other estimation techniques. In previous studies [11]–[14] only software measurements were carried out to get the size of the software under study without calculating the estimated cost. Whereas in research [15]–[18] a software measurement test was carried out, which then calculated the estimated cost of the software that had been measured, but in that study, no deviation value was calculated from the estimated cost obtained.

In previous studies, software measurements have also been carried out, the results of which are used as a reference for cost estimation. After obtaining the estimated cost value, the deviation value is calculated from the estimated cost results obtained. However, the calculation of the deviation value is calculated from the cost estimation results using the Adjusted Function Point and ordinary Function Point methods [4]. This study calculates two software measurement models, the UCPabc method and the Adjusted Function Point method, to estimate the cost of software development projects.

2. METHODS

This study uses a quantitative approach with measurement methods and estimation software as the main measurement methods. This is to produce correct predictions, i.e., the

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

project ends without delay and succeeds on budget; however, any part of the error in the overall effort and cost estimates can result in project failure in terms of delivery time, budget or options [19]. Rough Order of Magnitude (ROM) estimates the project's cost. This type of estimation is done very early in a project or even before a project has officially started. Project managers and top management use these estimates to help make project selection decisions. The accuracy of ROM estimation is usually -50% to +100%, which means the actual project cost can be 50% below the ROM estimate or 100% above [20].

Function Point Analysis (FPA), Use Case Point (UCP), Use Case Point Activity-Based Costing (UCP-ABC), and Adjusted Function Point (AFP) are used to provide accurate predictions of the effort, time, and cost required to complete a project. These techniques are typically applied after project planning and data collection. FPA involves identifying and classifying different software components, while UCP estimates the effort required by analyzing the system's use cases. UCP-ABC incorporates activity-based costing to estimate the cost of software development, while AFP adjusts function points based on non-functional requirements. The estimation results are evaluated, and the most suitable approach is selected based on the project's needs. This systematic approach to software estimation helps project managers make informed decisions about resource allocation, risk management, and project scheduling to ensure successful project completion.

2.1. Function Points Estimation

Function Point consists of 5 components that can be assessed as follows[2].External Input type is an interface that enters data into the application. The second is the External Output type, which is the output generated by the application for the user which can be in the form of a printed report or displayed on the screen. Furthermore, there is the External Inquiries type, which is a function related to transferring stored data. Then there is the Internal Logical Files Type, which is a function related to logical data storage in the form of a file or some relational database. Moreover, the last one is the External Interface Files type, which is a function related to data communication on other devices/machines. To get the Function Point value, the steps are as follows:

a. Calculating Crude Function Points (CFP)

Crude Function Points is a software complexity and measurement tool used to estimate the cost and effort required to develop a software system. CFP was developed as a Function Point (FP) variant, which measures the functionality provided by a software system. Table 1 is a reference for calculating CFP.

Table 1. *Complexity weight [21]*

Type	Complexity Level		
	Simple	Average	Complex
External Input (EI)	3	4	6
External Output (EO)	4	5	7
Internal Logical File (ILF)	7	10	15
External Interface File (EIF)	5	7	10
External Inquiry (EQ)	3	4	6

b. Calculating Relative Complexity Adjustment Factor (RCAF)

RCAF measures the degree of interrelationship between different functions or modules in a software system. Linkage refers to the level of dependence or influence of a module or function on other modules or functions. RCAF has typically used software based on 14 characteristics to evaluate the design of a software system and identify potential problems or areas for improvement. The Complexity Rating scale ranges from 0 to 5, where 0 indicates that the software's characteristics are unaffected by complexity, and 5 indicates that complexity has a significant impact. Table 2 is one of the table references that can be used as a guide in calculating RCAF.

Table 2. *Function point characteristic [21]*

ID	Characteristic
C1	Data communications
C2	Distributed function
C3	Performance objectives
C4	Heavily used configuration
C5	Transaction rate
C6	Online data entry
C7	End-user efficiency
C8	Online update
C9	Complex processing
C10	Reusability
C11	Installation ease
C12	Operational ease
C13	Multiple sites
C14	Facilitate change

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

c. Calculating Function Point (FP)

Next is calculating FP, which uses the formula (1), where the numbers 0.65 and 0.01 are determined by the International Function Point Users Group (IFPUG).

$$FP = CFP \times (0.65 + 0.01 \times RCAF) \quad (1)$$

2.2. Use Case Points Estimation

The steps for calculating use case points are carried out according to those formulated by G. Karner are as follows[22]:

a. Calculate Unadjusted Use Case Points (UUCP)

The first step at this stage is to determine the Unadjusted Actor Weight (UAW) as a Simple, Average, or Complex category, according to Table 3.

Table 3. *Actor weight*

Complexity	Weight	Description
Simple	1	Actors in use cases interact through the Application Programming Interface (API)
Average	2	Actors interact via protocols, such as TCP/IP
Complex	3	Actors interact through a Graphical User Interface (GUI)

Total Unadjusted Actor Weights (UAW) are obtained by calculating the number of actors based on each type (level of complexity) and multiplying by the total weight for each factor according to the formula (2).

$$\text{UAW} = \sum(weight \times number\ of\ Actors) \quad (2)$$

The next step in calculating the UUCP is to determine the Unadjusted Use Case Weight (UUCW). how to calculate UUCW is the same as calculating UAW, that is, each use case is divided into three groups, simple or average or complex, depending on the number of transactions made. For a more detailed explanation of the use case description, see Table 4.

Table 4. *Use case weight*

Complexity	Weight	Description
Simple	5	Using ≤ 3 transactions
Average	10	Using 4 to 7 transactions
Complex	15	Using > 7 transactions

Total Unadjusted Use Case Weights (UUCW) is obtained from calculating the number of use-cases from each level of complexity multiplied by the total factor of each use case. The formula for calculating UUCW is as in formula (3):

$$\text{UUCW} = \sum(weight \times number\ of\ use\ case) \quad (3)$$

Then add up the UAW and UUCW to get the Unadjusted Use Case Point (UUCP), as in the formula (4):

$$UUCP = UAW + UUCW \quad (4)$$

b. Calculate Use Case Point (UCP)

There is a complexity factor value in calculating the UCP value. What is meant by complexity factors is factors that directly affect the process of working on a software project. The complexity factor is divided into two groups: Technical Complexity Factor (TCF) and Environmental Factor (EF). The following explanation of these complexity factors is presented in Table 5 and Table 6.

Table 5. *Technical complexity factor*

Factor	Description	Weight
T1	Distributed System	2
T2	Performance	1
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 6 Environmental Complexity Factor

Factor	Description	Weight
F1	Familiarity with standard process	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2
F7	Part-time workers	-1
F8	Difficult programming language	-1

Multiply the TCF value by the corresponding weighting value. The weighting value from 0 to 5 is assigned to each factor based on the magnitude of its influence. 0 indicates no effect, 3 indicates average, and 5 indicates a significant impact. The outcome of multiplying the numbers and weights is the Technical Factor (TF), which is then used to get the Technical Complexity Factor (TCF) stated in formula (5).

$$\text{TCF} = C_1 + C_2 \times \sum_{i=1}^{13} weight\ factor_i \times influence_i \quad (5)$$

Where:
$C_1 = 0.6$ and $C_2 = 0.01$

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

The value on the environmental factor is multiplied by the respective weighting value. The weighting of the values given from 0 to 5 for each factor depends on how much influence the factor has. 0 means no effect, 3 means average, and 5 means a big influence. The results of multiplying the values and weights are then added up to get the total ECF, the formula represented in Formula (6).

$$\text{ECF} = C_1 + C_2 \times \sum_{i=1}^{8} weight\ factor_i \times influence_i \quad (6)$$

Where:
C_1 = 1.4 and C_2 = -0.03

The Use Case Point (UCP) value is obtained by multiplying the UUCP value with TCF and EF as stated in Formula (7).

$$UCP = UUCP \times TCF \times ECF \quad (7)$$

Where:
UUCP = *Unadjusted Use Case Points*
TCF = *Technical Complexity Factors*
ECF = *Environment Complexity Factor*

2.3. Use Case Point Activity-Based Costing

UCPabc is an integration model between Use Case Point (UCP) software measurement methods and Activity-Based Costing (ABC) financing techniques. This model estimates the total effort, which is the output of the UCP method added to the three main components in the ABC technique, namely: identification of resources and activities, cost rate, and relative product weight. Integrating Use Case Point and Activity-Based Costing (later called the UCPabc model) can produce software cost estimates, especially using a resource-sharing system [15].

The UCPabc model considers the unique characteristics of software development projects, such as the dynamic nature of the development process and the need for specialized resources. By combining the UCP method, which measures the functional requirements of the software, with the ABC technique, which identifies the resources and activities required to develop the software, the UCPabc model provides a comprehensive approach to estimating software development costs. This approach enables organizations to make informed decisions about resource allocation and project planning and to manage their software development projects better to ensure successful outcomes.

2.4. Adjusted Function Point

The Adjusted Function Point approach is the same as the Function Point method, it is just that there is a change in the calculation of the Relative Complexity Adjustment Factor (RCAF) at the Function Point to become Modification Complexity Adjustment Factor (MCAF) which is a comparative study of the complexity factor in the well-known cost estimation method of FP and Use Case Points (UCP) are then compared with the determined non-functional requirements of the project [11]. Table 7 lists the Modification Complexity Adjustment Factor needed to measure the AFP.

Table 7. *Calculation of modification complexity adjustment factor*

No.	Modification Complexity Adjustment Factor
1	Level of reliability for recovery
2	Level of data communications
3	Level of distributed data processing
4	Level of performance needs
5	Level of environment configuration
6	Level of transaction rate
7	Level of end-user efficiency
8	Level of master file update
9	Level of online real-time update
10	Level of reusability
11	Level of installation ease
12	Level of operational ease
13	Level of customer organization variation
14	Level of change possibility
15	Level of security
16	Level of user training

3. RESULTS AND DISCUSSION

The estimates presented in this study are based on the activities performed during the project. Therefore, the project work activities listed in Table 8 serve as supporting data for this research and were obtained from project management records at one of our software house sources. Costs are calculated using the Indonesian National Association of Consultants (Inkindo) rate, converted into units of days (8 working hours per day) as shown in Table 8. The cost rate per activity in software development is determined by multiplying the rate by the effort allocated to each activity. This project's six activities involved in software development are: Requirement Gathering, Development & Implementation, Installation on Testing Environment, System Integration Test, User Acceptance Test, and Implementation.

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

Table 8. *Resources and payrate*

Class	Experts	Person per Month	Person per Day	Roundoff	Person per Hour
Young Expert	Project Manager	Rp23,750,000	Rp1,079,545	Rp1,050,000	Rp131,250
Young Expert	Business Analyst	Rp20,750,000	Rp943,182	Rp943,000	Rp117,875
Young Expert	System Analyst	Rp20,750,000	Rp943,182	Rp943,000	Rp117,875
Personnel	Programmer	Rp11,700,000	Rp531,818	Rp531,000	Rp66,375
Personnel	Quality Assurance	Rp10,000,000	Rp454,545	Rp454,000	Rp56,750
Personnel	Staff Admin	Rp6,400,000	Rp290,909	Rp290,000	Rp36,250
Personnel	Documenter	Rp6,400,000	Rp290,909	Rp290,000	Rp36,250

Displayed in Table 9 are the actual costs incurred for the researched work, which will be compared with the Adjusted Function Point method and the Use Case Point Activity Based Costing (UCPabc) method. This comparison will enable us to evaluate the effectiveness of the UCPabc and Adjusted Function Point methods in estimating software development costs. By comparing the actual costs with the estimates generated by these methods, we can identify any discrepancies and assess the accuracy of each method. This information will be valuable for software development organizations, as it can inform their decision-making processes and help them to choose the most suitable cost estimation method for their projects. Additionally, this study will contribute to the broader field of software engineering by providing insights into the strengths and weaknesses of different cost estimation methods and advancing our understanding of how to improve the accuracy of software cost estimation.

Table 9. *Actual cost*

Activities	Actual Cost
Requirement Gathering	Rp 20,025,000
Development & Implementation	Rp 59,958,500
Installation on Testing Environment	Rp 1,056,000
System Integration Test	Rp 16,572,000
User Acceptance Test	Rp 15,572,000
Implementation	Rp 11,891,000
Total Cost	Rp 125,074,500

Formulas (8) and (9) were used to calculate the percentage of resources for each activity presented in Table 10, based on the duration of the resources used. The percentage of resources allocated to each activity is an important factor in software cost estimation, influencing the project's overall cost. Therefore, accurately measuring resource allocation is crucial for estimating software development costs. The formulas used in this study consider the duration of the resources used for each activity, providing a more detailed and accurate assessment of resource allocation than methods that rely solely on functional requirements or lines of code. Using this approach, we can better understand how resources are allocated throughout the project and make more informed decisions about resource allocation in the future. This information is particularly important for organizations working on large-scale software development projects, where resource allocation can significantly impact the project's success.

$$Resource\ Percentage = \frac{Total\ mandays\ per\ activity}{Total\ day\ per\ activity} \times 100\% \quad (8)$$

$$Cost\ Rate = \sum(resource\ percentage \times resource\ payrate\ per\ hour) \quad (9)$$

After collecting several variables at the initial stage, the researchers performed estimation calculations using The estimation calculations were conducted to generate cost estimates for the software development project based on the functional requirements identified in the initial stage. The UCPabc and Adjusted Function Point methods were chosen for comparison because they are widely used and well-established methods for estimating software development costs. By applying both methods, the researchers were able to compare the results and assess each method's accuracy in estimating the project's costs.

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

Table 10. *Activity cost rate*

Activities	Resources	Cost rate
Requirement Gathering	Project Manager (37%), System Analyst	Rp 166,438
Development & Implementation	Project Manager (41%), System Analyst (40%), Programmer (59%), Quality Assurance (7%), Business Analyst (7%), Staff Admin (11%), Documenter (22%)	Rp 164,310
Installation on Testing Environment	Project Manager (50%), Programmer	Rp 132,000
System Integration Test	Project Manager (40%), System Analyst (80%), Programmer (80%), Staff Admin (20%)	Rp 207,150
User Acceptance Test	Project Manager (40%), System Analyst (89%), Programmer (89%)	Rp 216,483
Implementation	Project Manager, System Analyst (29%), Programmer (71%)	Rp 212,560

3.1. Calculation with Use Case Point Activity Based Costing (UCPabc)

To begin, it is necessary to examine the project's functionality. The functionality is extracted by calculating the use case type, which is summarized in Table 11 and Table 12.

Table 11. *Calculation of unadjusted use case weight (UUCW)*

Use case Type	Description	Weight	Num of Use Case	Weight x Num of Use Case
Simple	Using at most 3 transactions	5	3	15
Medium	Using 4 to 7 transactions	10	6	60
Complex	Using more than 7 transactions	15	3	45
			Total UUCW	120

Table 12. *Calculation of unadjusted actor weight*

Use case Type	Description	Weight	Num of Use Case	Weight x Num of Use Case
Simple	Actors in use cases interact via the API or command prompt	1	0	0
Medium	Actors interact via protocols, such as TCP/IP	2	0	0
Complex	Actors interact through a GUI or Web Page	3	4	12
			Total UAW	12

UUCP = $UUCW + UAW$
$= 120 + 12 = 132$

The Unadjusted Use Case Point value obtained is 132. Next, the Environment Factor calculation is carried out as displayed in Table 13, which has its basis presented in Table 6. The results obtained in this calculation will then be used to calculate the Environment Complexity Factor. The value of the Environment Complexity Factor obtained is 0.755.

Table 13. *Calculation of environment factor*

No	Environment Factor	Weight	Score (0-5)	EF
E1	Familiarity with development process used	1.5	3	4.5
E2	Application experience	0.5	4	2
E3	OO Programming Experience	1	4	4
E4	Lead Analyst Capability	0.5	4	2
E5	Motivation	1	4	4

Table 13 continued…

No	Environment Factor	Weight	Score (0-5)	EF
E6	Stable Requirements	2	5	10
E7	Part-Time Staff	-1	3	-3
E8	Difficult Programming Language	-1	2	-2
			Total EF	21.5

$$ECF = C_1 + C_2 \times \sum_{i=1}^{8} Weight\ Factor_i \times Influence_i$$
$$= 1.4 + (-0.03) \times EF$$
$$= 1.4 + (-0.03) \times 21.5$$
$$= 0.755$$

Then calculate the Technical Factor whose calculation basis is presented in Table 5. The results of this calculation will be used to calculate the Technical Complexity Factor (TCF) as displayed in Table 14. The Technical Complexity Factor value obtained is 1.06.

Jurnal Teknik Informatika Vol. 18 No. 2, October 2025 (316-326)
ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

Table 14. *Calculation of technical factor*

No	Technical Factor	Weight	Score (0-5)	TF
TCF01	Distributed System Required	2	0	0
TCF02	Response time is important	1	4	4
TCF03	End-user efficiency	1	4	4
TCF04	Complex internal processing is required	1	2	2
TCF05	Reusable Code Must be a focus	1	1	1
TCF06	Installation Ease	0.5	5	2.5
TCF07	Operation Ease (usability)	0.5	5	2.5
TCF08	Cross-platform support (portability)	2	5	10
TCF09	Easy to change (changeability)	1	5	5
TCF10	Highly Concurrent	1	4	4
TCF11	Custom Security	1	4	4
TCF12	Provide direct access for third parties	1	3	3
TCF13	User Training	1	4	4
			Total TF	**46**

$$TCF = C_1 + C_2 \times \sum_{i=1}^{13} weight\ factor_i \times influence_i$$

$$= 0.6 + 0.1 \times TF$$

$$= 0.6 + 0.1 \times 46$$

$$= 1.06$$

After getting the UUCP, TCF and ECF values, then we get the UCP value. The Use Case Point (UCP) value obtained is 105.6396, calculated using formula (7). If the UCP value has been obtained, then it is multiplied by the Effort Rate of 8.2 to calculate the Total Effort [15], so the Total Effort in this project is 866.24. Table 15 displays the Effort Distribution in this project.

Table 15. *Effort distribution of the UCPabc method*

Activities	%	Effort
Requirement Gathering	17.05%	147.69
Development & Implementation	52.27%	452.78
Installation on Testing Environment	1.14%	9.88
System Integration Test	11.36%	98.40
User Acceptance Test	10.23%	88.62
Implementation	7.95%	68.87
Total Effort	**100%**	**866.24**

So that the estimated costs based on the UCPabc method approach are obtained as shown in Table 16.

Table 16 .*Cost estimation UCPabc method*

Activities	%	Effort	Cost
Requirement Gathering	17.05%	147.69	Rp 24,581,807
Development & Implementation	52.27%	452.78	Rp 74,396,881
Installation on Testing Environment	1.14%	9.88	Rp 1,303,518
System Integration Test	11.36%	98.40	Rp 20,384,568
User Acceptance Test	10.23%	88.62	Rp 19,183,889
Implementation	7.95%	68.87	Rp 14,638,174
Total Effort	**100%**	**866.24**	
Total Cost			**Rp 154,488,837**

3.2. Calculations with Adjusted Function Points

In calculating estimates using AFP, the first thing to do is to determine the Crude Function Point (CFP) first. In the project that is used as a reference for this research, the CFP is shown in Table 17. After that, the MCAF is calculated, this process is present in Table 18.

Table 17. *CFP calculation*

Type	Complexity Level									Total CFP
	Simple			Average			Complex			
	J	B	P	J	B	P	J	B	P	
External Input (EI)	3	3	9	2	4	8	0	6	0	17
External Output (EO)	0	4	0	0	5	0	0	7	0	0
Internal Logical File (ILF)	2	3	6	1	4	4	0	6	0	10
External Interface File (EIF)	0	7	0	3	30	30	0	15	0	30
External Inquiry (EQ)	0	6	0	0	0	0	0	10	0	0
								Total		**57**

Table 18. *MCAF Calculation*

No.	Modification Complexity Adjustment Factor	Score (0-5)
1	Level of reliability for recovery	0
2	Level of data communications	1
3	Level of distributed data processing	1
4	Level of performance needs	3
5	Level of environment configuration	3
6	Level of transaction rate	2
7	Level of end-user efficiency	4
8	Level of master file update	2
9	Level of online real-time update	3
10	Level of reusability	2
11	Level of installation ease	1
12	Level of operational ease	2
13	Level of customer organization variation	2
14	Level of change possibility	2
15	Level of security	4
16	Level of user training	3
	Total	35

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

$$\begin{aligned} AFP &= CFP \times (0.65 + 0.01 \times MCAF) \\ &= 57 \times (0.65 + 0.01 \times 35) \\ &= 57 \times (1) \\ &= 57 \end{aligned}$$

The Adjusted Function Point value obtained is 57. After that, a value of 467.4 is obtained as the Total Effort value. This value is then converted to the activities in the project to get the Effort distribution shown in Table 19.

Based on these results, an estimated cost can be calculated using AFP, the value is shown in Table 20, and the results obtained from calculating the two estimates on the implemented project can be formulated as in Table 21. It can be observed that the UCPabc approach has a different value with the Actual Cost that is less than AFP. In the context of cost management for this project, UCPabc is also considered safer because the estimated value exceeds the actual cost. The results of calculations using AFP on this project indicate a higher probability of overbudgeting than the UCPabc method.

Table 19. *AFP method effort distribution*

Activities	%	Effort
Requirement Gathering	17.05%	79.69
Development & Implementation	52.27%	244.31
Installation on Testing Environment	1.14%	5.33
System Integration Test	11.36%	53.10
User Acceptance Test	10.23%	47.82
Implementation	7.95%	37.16
Total Effort	**100%**	**467.40**

Table 20. *AFP cost estimation*

Activities	%	Effort	Cost	
Requirement Gathering	17.05%	79.69	Rp	13,263,687
Development & Implementation	52.27%	244.31	Rp	40,142,573
Installation on Testing Environment	1.14%	5.33	Rp	703,344
System Integration Test	11.36%	53.10	Rp	10,998,969
User Acceptance Test	10.23%	47.82	Rp	10,351,115
Implementation	7.95%	37.16	Rp	7,898,368
Total Effort	**100%**	**467.40**		
Total Cost			**Rp**	**83,358,056**

Table 21. *Difference in cost between the actual cost of the project and the estimation using UCPabc and AFP*

Activities	Actual Cost		UCPabc		Adjusted Function Point	
Requirement Gathering	Rp	20,025,000	Rp	24,581,807	Rp	13,263,687
Development & Implementation	Rp	59,958,500	Rp	74,396,881	Rp	40,142,573
Installation on Testing Environment	Rp	1,056,000	Rp	1,303,518	Rp	703,344
System Integration Test	Rp	16,572,000	Rp	20,384,568	Rp	10,998,969
User Acceptance Test	Rp	15,572,000	Rp	19,183,889	Rp	10,351,115
Implementation	Rp	11,891,000	Rp	14,638,174	Rp	7,898,368
Total Cost	**Rp**	**125,074,500**	**Rp**	**154,488,837**	**Rp**	**83,358,056**
Deviation Cost				**23.52%**		**33.35%**

CONCLUSION

In conclusion, the integration of Use Case Point and Activity-Based Costing in the UCPabc model has proven to be an effective and reliable approach for estimating software development project costs. This model's precision of cost estimates enhances decision-making processes within software development organizations, enabling better resource allocation and project planning. This contributes to project success and mitigates the risks associated with inaccurate cost estimations.

The applicability of the UCPabc model extends beyond cost estimation, fostering a culture of data-driven decision-making within software development teams. It sets a precedent for organizations to optimize resource allocation, enhance project management practices, and achieve more cost-effective outcomes. The integration of these methodologies aligns perfectly with the dynamic nature of the software development landscape, offering a systematic approach to address the complexities of estimating and managing costs.

Furthermore, the positive outcomes of this research prompt a reevaluation of prevailing industry practices and methodologies. This study encourages a shift towards more holistic and comprehensive cost estimation approaches by showcasing integrated models' tangible benefits and enhanced accuracy. It also underscores the importance of accurately assessing functional requirements and project activities in cost estimation, highlighting the potential financial consequences of underestimating or overestimating project costs.

For software development organizations, this study leads to several recommendations. First, adopting the UCPabc model is recommended due to its superior accuracy in cost estimation. Second, maintaining thorough data collection and analysis throughout the project lifecycle is crucial to achieve precise estimates. Third, organizations should carefully assess their functional requirements and project activities to choose the most suitable estimation method. Lastly, future research should explore alternative cost estimation approaches and investigate how project size and complexity affect estimation accuracy. Additionally, future studies are encouraged to examine the integration of machine learning or AI-based techniques with traditional estimation models, which could further enhance accuracy and adaptability to various project environments. By applying these recommendations, software development organizations can improve the precision of their cost estimates, leading to greater project success and cost efficiency.

REFERENCES

[1] K. Zhang, X. Wang, J. Ren, and C. Liu, "Efficiency Improvement of Function Point-Based Software Size Estimation with Deep Learning Model," *IEEE Access*, 2021, doi: 10.1109/ACCESS.2020.2998581.

[2] D. Khairani, "Studi Kasus Pengukuran Sistem Informasi Menggunakan Function Point (Fp)," *Jurnal Teknik Informatika*, vol. 8, no. 2, pp. 1–7, 2015, doi: 10.15408/jti.v8i2.2442.

[3] H. LEUNG and Z. FAN, "SOFTWARE COST ESTIMATION," 2002.

[4] R. S. Dewi, A. P. Subriadi, and Sholiq, "A Modification Complexity Factor in Function Points Method for Software Cost Estimation Towards Public Service Application," *Procedia Computer Science*, vol. 124, pp. 415–422, 2017, doi: 10.1016/j.procs.2017.12.172.

[5] F. Fachruddin and Y. Pratama, "Eksperimen Seleksi Fitur Pada Parameter Proyek Untuk Software Effort Estimation dengan K-Nearest Neighbor," *JURNAL INFORMATIKA : Jurnal Pengembangan IT*, vol. 2, no. 2, pp. 53–62, 2017.

[6] R. Saptono and G. D. Hutama, "Peningkatan Akurasi Estimasi Ukuran Perangkat Lunak dengan Menerapkan Logika Samar Metode Mamdani," *Scientific Journal of Informatics*, 2016, doi: 10.15294/sji.v2i1.4527.

[7] A. Syaputra, "Aplikasi E-Kelurahan Untuk Peningkatan Pelayanan Administrasi dalam Mendukung Penerapan E-Government," *MATRIK : Jurnal Manajemen, Teknik Informatika dan Rekayasa Komputer*, vol. 20, no. 2, pp. 379–388, 2021, doi: 10.30812/matrik.v20i2.1180.

[8] B. N. Fadillah, D. Khairani, and N. F. Rozi, "MOBILE BANKING SYSTEM DEVELOPMENT USING NEAR FIELD COMMUNICATION TECHNOLOGY WITH ANDROID-BASED OPERATING SYSTEM," *JURNAL TEKNIK INFORMATIKA*, vol. 13, no. 1, pp. 43–50, 2020, doi: 10.15408/jti.v13i1.15731.

[9] A. Naderpour, J. M. Sardroud, M. Mofid, Y. Xenidis, and T. P. Rostam, "Uncertainty management in time estimation of construction projects: A systematic literature review and new model development," *Scientia Iranica*, 2019, doi: 10.24200/sci.2017.4605.

[10] R. Adhitama, "Effort Estimation Menggunakan Metode Use Case Point untuk Pengembangan Perangkat Lunak," *Journal of Informatics, Information System, Software Engineering and Applications (INISTA)*, 2018, doi: 10.20895/inista.v1i1.14.

[11] Sholiq, R. S. Dewi, and A. P. Subriadi, "A Comparative Study of Software Development Size Estimation Method: UCPabc vs Function Points," 2017, doi: 10.1016/j.procs.2017.12.179.

[12] S. Kumari and S. Pushkar, "Comparison and Analysis of Different Software Cost Estimation Methods," *International Journal of Advanced Computer Science and Applications*, 2013, doi: 10.14569/ijacsa.2013.040124.

[13] V. Tunalı, "Software Size Estimation Using Function Point Analysis – A Case Study for a Mobile Application," 2014.

[14] W. A. Chapetta, J. S. Das Neves, and R. C. S. Machado, "Quantitative metrics for

ISSN: p-ISSN 1979-9160 (Print)| e-ISSN 2549-7901 (Online)
DOI: https://doi.org/10.15408/jti.v18i2.46669

performance monitoring of software code analysis accredited testing laboratories," *Sensors*, 2021, doi: 10.3390/s21113660.

[15] R. S. Dewi, G. F. Prassida, Sholiq, and A. P. Subriadi, "UCPabc as an integration model for software cost estimation," *Proceeding - 2016 2nd International Conference on Science in Information Technology, ICSITech 2016: Information Science for Green Society and Environment*, pp. 187–192, 2017, doi: 10.1109/ICSITech.2016.7852631.

[16] L. Lavazza, G. Liu, and R. Meli, "Using extremely simplified functional size measures for effort estimation: An empirical study," 2020, doi: 10.1145/3382494.3410691.

[17] S. K. Khatri, S. Malhotra, and P. Johri, "Use case point estimation technique in software development," 2016, doi: 10.1109/ICRITO.2016.7784938.

[18] K.-T. Kwon, "Use Case Points Estimation for the Software Cost Appraisal," *Journal of Software Assessment and Valuation*, 2020, doi: 10.29056/jsav.2020.06.04.

[19] I. M. Keshta, "Software Cost Estimation Approaches: A Survey," *Journal of Software Engineering and Applications*, 2017, doi: 10.4236/jsea.2017.1010046.

[20] K. Schwalbe, "Information Technology Project Management, Seventh Edition," *Nursing management*, 2012.

[21] C. Jones, *Applied Software Measurement: Global Analysis of Productivity and Quality*. 2008.

[22] G. Karner, "Resource estimation for objectory projects," *Objective Systems SF AB*, pp. 1–9, 1993.