Cost Estimation for Software Development Using Function Point Analysis Method

Ester Lumba[1], Destriana Widyaningrum[2], Alexander Waworuntu[3]
[1,2] Department of Informatics, Bunda Mulia University, Jakarta, Indonesia
[3] Department of Informatics, Universitas Multimedia Nusantara, Tangerang, Indonesia
[1]l0178@lecturer.ubm.ac.id, [2]l0178@lecturer.ubm.ac.id
[3]alex.wawo@umn.ac.id

Accepted 7 January 2025
Approved 17 January 2025

***Abstract*— Software development requires substantial financial resources. This study aims to examine the cost estimation for software development. The complexity of the software, its intangibility as a non-physical product, the technology utilized, and human resources can all influence the determination of software development costs. The method used for cost estimation is Function Point Analysis with a case study approach. The researchers conducted a case study on the software development for an employee savings and loan cooperative at XYZ Company. The formulation of the problem in this study is how to apply the Function Point Analysis method in estimating software development costs at the XYZ employee savings and loan cooperative. The result of this study is a cost recommendation that can serve as a reference for selecting software development vendors by the cooperative's management.**

***Index Terms*— Cost Estimation; Function Point Analysis; Savings and Loan Cooperative.**

I. INTRODUCTION

Information technology is rapidly advancing in the digital era [1], prompting various small, medium, and large organizations to develop software to support their business operations. Desktop, web, and mobile applications are necessary to enhance efficiency, productivity, and service quality.

Software development incurs costs [2]. Organizations, whether large or small, often struggle to determine the costs associated with software development. Several factors contribute to the difficulty of determining these costs, including the intangibility of software as a non-physical product, project complexity, the technology employed, and human resources. Estimation techniques can be classified into three categories: expert judgment based on historical data and similar software projects, algorithmic models, and machine learning [2][3][4].

Software comprises computer programs associated with software documentation [5]. This documentation is a collection of information, guidelines, and descriptions related to the development, operation, and maintenance of the software. Software does not become obsolete because software defects can be repaired. The purpose of documentation is to assist users in understanding how to use the software. For developers, it provides essential information for updating or repairing software and serves as a reference for teams involved in project development.

Software is built through an engineering process [5]. Generally, software development follows principles known as the software or system development life cycle (SDLC) [6][7]. The SDLC includes phases aimed at producing quality software that meets customer desires or the objectives for which the system was created. The software or system development life cycle can be seen in Figure 1.

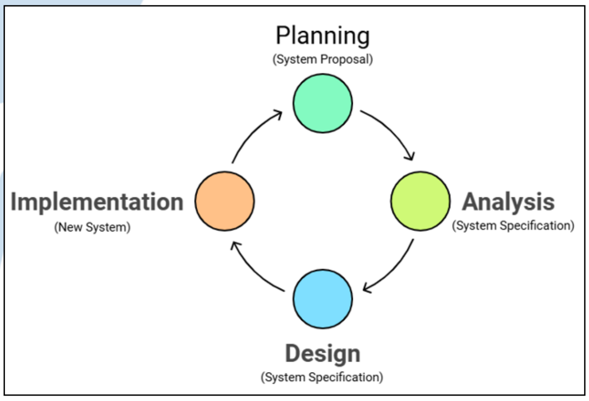

Fig. 1. Software Development Life Cycle

In software engineering, the concept of SDLC underlies various software development methodologies. These methodologies provide a framework for planning and controlling the creation of an information system, namely the software development process. Software development is divided into two approaches: structured and Object-Oriented Design (OOD). The structured approach represents the system based on data and the processes applied to this data, using modeling tools like Data Flow Diagrams (DFD). The object-oriented design approach views the system as a collection of objects consisting of data and processes, using modeling tools like Unified Modeling Language (UML) [5][6][7].

This research uses a case study approach to develop software for the XYZ Company's employee savings and loan cooperative. This cooperative is a type of microfinance organization that plays a role in enhancing the welfare of each member[8]. XYZ Company has both permanent and contract employees. Like other companies, XYZ Company also formed a savings and loan cooperative. This cooperative was established as a platform for employees to assist each other financially during times of need. All permanent employees are encouraged to become cooperative members [8]. Besides serving as a venue for saving and borrowing money, the cooperative also fosters solidarity and mutual aid among employees at the company [9].

To manage employee data, the cooperative's management still uses manual bookkeeping and records data using tools like Microsoft Excel. Amid the demands of their primary job roles within the company, employees who also serve as cooperative managers often face challenges in reporting to members. Therefore, during a member meeting, the cooperative's management at the company proposed developing software to aid in managing member data and recording transactions and borrowings conducted by employees.

Building software requires funding. The cooperative's management has contacted various software vendors. The development costs offered by these vendors vary significantly, ranging from very low to very high prices.

Addressing the challenges outlined above, the research team embarked on a study aimed at refining the cost estimation process for software development through the Function Point Analysis method. This study focuses on applying this method specifically to estimate the costs associated with developing software for the XYZ employee savings and loan cooperative. The core research question investigates the practical application of Function Point Analysis in this setting: "How is the Function Point Analysis method applied in estimating the costs of developing software for the XYZ employee savings and loan cooperative?"

The objectives are twofold: firstly, to implement the Function Point method to accurately estimate the development costs for software at the XYZ Employee Savings and Loan Cooperative, and secondly, to equip the cooperative's management with reliable recommendations for choosing a software development vendor who offers a reasonable cost.

The anticipated benefits of this study are manifold. For software developers, it promises to provide a clear and quantifiable guideline for estimating costs, based on the functional size of the software. For the cooperative's management, it aims to present an overview of what constitutes reasonable software development costs. Lastly, for the academic and research community, this study is intended to serve as a valuable reference in the field of software cost estimation, with a particular focus on the application of the Function Point Analysis method.

II. Research Methodology

This study employs a qualitative descriptive approach using a case study methodology. The software development project for the Employee Savings and Loan Cooperative at XYZ Company will be analyzed using Function Point Analysis to estimate costs [10]. The research process is divided into three stages, as illustrated in Figure 2.

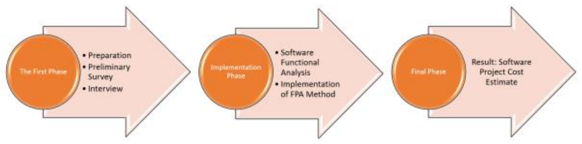
The First Phase
• Preparation
• Preliminary Survey
• Interview
Implementation Phase
• Software Functional Analysis
• Implementation of FPA Method
Final Phase
Result: Software Project Cost Estimate

Fig. 2. Research Step

The initial stage of the research is divided into three parts: preparation, preliminary survey, and interviews with the cooperative's management. In the preparation phase, the research team conducts studies related to topics that are relevant to the field of expertise. Subsequently, in the preliminary survey phase, the team visits XYZ Company to gather information about the software needs. The preliminary survey revealed that the management of the XYZ Company’s employee savings and loan cooperative needs software to manage financial and member data. Following this, the research team conducts further interviews to capture the user requirements.

The implementation stage is split into two parts: analyzing the software functions based on the interviews conducted in the first stage and applying the Function Point Analysis (FPA) method. In the user requirements determination phase, researchers document the interview results with the cooperative's management. This documentation includes a list of functional and non-functional requirements needed by the users. The next part involves analyzing the software functions, which results in seven modules. Each module defines a list of functional requirements. The final stage of this research produces a recommended cost estimate for the software project. This recommendation will serve as a reference for the cooperative's management in selecting a vendor to develop the software for the employee savings and loan cooperative at XYZ Company.

III. Function Point Analysis

Function Point Analysis (FPA) is a software measurement method introduced by Allan Albrecht in 1979[10] and is widely used globally. It has been updated by the International Function Point Users Group (IFPUG) [11]; a nonprofit organization managed by members worldwide. This organization helps improve software development processes according to software measurement standards. IFPUG has approximately 1,200 members from 30 countries who are experts in Function Point Analysis [11][12].

A. *User Function Identification and Complexity*

From a user’s perspective, software functionality is measured using five elements: External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), and Program Interfaces (PI). Each function has its own complexity level [13]. Therefore, each function can be classified based on its complexity as low for simple functions, medium for moderately complex functions, and high for complex or highly complex functions [13][14][15]. The functional complexity weight based on function type are shown in table 1 below:

TABLE I. FUNCTIONAL POINT COMPLEXITY

User Function Types	**Complexity Weight**		
	Low	***Medium***	***High***
External Inputs (EI)	3	4	6
External Outputs (EO)	4	5	7
External Inquiries (EQ)	3	4	6
Internal Logical Files (ILF)	7	10	15
Program Interfaces (PI)	5	7	10

External Input pertains to user inputs into the system. Based on the analysis, the function and complexity levels for the external input elements of the employee savings and loan cooperative are shown in Table 2.

TABLE II. EXTERNAL INPUTS

No.	**Description**	**Complexity**
1	Member Login	Low
2	Manager and Admin Login	Low
3	Member Registration	Low
4	Manager Registration	Low
5	Update Manager Data	Low
6	Update Member Data	Low
7	Loan Application	Medium
8	Deposits (Principal, Mandatory, Voluntary)	Low
9	Installment Deposits	Low

External Output relates to outputs produced by the system. Based on the analysis, the function and complexity levels for the external output elements of the cooperative are shown in Table 3

TABLE III. EXTERNAL OUTPUTS

No.	**Description**	**Complexity**
1	Member Data Search	Low
2	Manager Data Search	Low
3	Loan Application Status Search	Low
4	Membership Status Search	Low
5	Loan Payment Status Search	Low

External Inquiries involve user searches within the system [14][15]. The system will display search results if data is found and a message if no data is available. The functions and their complexities are shown in Table 4

TABLE IV. EXTERNAL INQUIRIES

No.	**Description**	**Complexity**
1	Member Card	Low
2	Voluntary Deposit Receipt	Low
3	Mandatory Deposit Receipt	Low
4	Loan Repayment Receipt	Low
5	Loan History per Member	Low
6	Savings History per Member	Low
7	Loan List for All Members	Low
8	Savings List for All Members	Low
9	Monthly Report	Medium
10	Annual Report	Medium
11	Profit and Loss Report	High
12	Cash Flow Chart	High
13	Email Notifications	Low
14	Smartphone Notifications	Log

Internal Logical Files consist of files that form the system, such as tables, images, text files, or other file formats [14][15]. The complexities involved in the cooperative's employee savings and loan system are shown in Table 5

TABLE V. INTERNAL LOGICAL FILES

No.	**Description**	**Complexity**
1	Member Table	Low
2	User Table	Low
3	Savings Table	Medium
4	Loan Table	Medium
5	Manager Table	Low
6	Installment Table	Low
7	Interest Table	Low
8	Image Files	Low

Program Interface involves interfaces such as the use of Application Programming Interfaces or other systems related to the developed software. Since the cooperative already has a system for processing employee data, employee and division data are taken from the existing system at the company. The functions and complexity levels are shown in Table 6.

TABLE VI. PROGRAM INTERFACE

No.	Description	Complexity
1	Company Employee Data	Low
2	Company Division Data	Low

B. Unadjusted Function Point

After identifying user functions, the next step is to calculate the Unadjusted Function Point (UFP). The formula for calculating UFP is (1).

$$UFP = \sum(\text{ Number of Functions } \times \text{ Complexity Weight }) \quad (1)$$

The UFP is derived from the calculation of the number of functions multiplied by their complexity weights. Table 7 shows the calculations for each function and their complexities.

TABLE VII. UFP VALUE CALCULATION

User Function Types	Total Number	Complexity Weight			Total
		Low	*Medium*	*High*	
External Inputs (EI)	**9**	8*3	1*4	0*6	28
External Outputs (EO)	14	10*4	2*5	2*7	64
External Inquiries (EQ)	5	5*3	0*4	0*6	15
Internal Logical Files (ILF)	8	6*7	2*10	0*15	62
Program Interfaces (PI)	2	2*5	0*7	0*10	10
Unadjusted Function Points					179

From Table 7 above, the total Unadjusted Function Points is 179, calculated from the total of EI + EO + EQ + ILF + PI, thus total UFP = 28 + 64 + 15 + 62 + 10 = 179.

C. Calculate Value Adjustment Factor

The software developed requires an operational environment. Therefore, in FPA, it's necessary to calculate factors that affect the operational complexity of the software. There are 14 General System Characteristics (GSC) that can influence the complexity of the software. Each characteristic or factor is rated between 0 to 5, with 0 meaning no effect and 5 meaning a significant effect. Table 7 shows the factors and the researcher's ratings for the cooperative's software.

Based on the Table 8, calculate the Total Degree of Influence (TDI) using the formula (2).

$$TDI = \sum(\text{ GSC Value }) \quad (2)$$

Thus, TDI = (3+0+0+0+0+0+0+1+1+0+0+1+0+0) = 6 After calculating the TDI, the next step is to compute the Value Adjustment Factor using the formula = 0.65 + (0.01 x TDI). Thus, VAF = 0.65 + (0.01 x 6) = 0.71 After calculating the VAF, calculate the Adjusted Function Points with the formula TUFP x VAF Thus, AFP = 179 x 0.71 = 127.09 rounded to 127.

TABLE VIII. VALUE ADJUSTMENT FACTORS

Factor	Value
Data Communication	3
Distributed Function	0
Performance Objectives	0
Heavily Used Configuration	0
Transaction Rate	0
Multiple Sites	0
Reusability	0
On-line Data Entry	1
On-line Update	1
Complex Processing	0
Installation ease	0
Operational ease	1
Extensibility	0
End-user Efficiency	0

D. Cost Estimation

To calculate the cost of software development using the FPA method, use the formula: Total cost = Adjusted Function Point * Hour/AFP * rates. The hourly rate is set at Rp. 100,000.00. The determination of the rate is based on consideration of several factors such as level of experience, specific expertise, type of project, location, and type of company. Thus, the estimated cost for developing the software for the XYZ company's employee savings and loan cooperative is 127 * 15 * 100,000 = Rp. 190,500,000.00

IV. RESULTS AND DISCUSSION

This study aimed to estimate the cost of developing software using the Function Point Analysis (FPA) method. The application of FPA was carried out through a case study approach at the XYZ company's employee savings and loan cooperative. The result of this study is an estimated software development cost for the cooperative amounting to Rp. 190,500,000.00. This cost estimation will serve as a reference for the cooperative's management to select a software development vendor for this company.

Based on the functional requirements obtained from the users, the software for the savings and loan cooperative produced seven modules. These modules are illustrated in Figure 3.

Using the FPA method, an Unadjusted Function Point of 179 and a Total Degree of Influence of 6 were obtained. Given that the project size of the savings and loan cooperative falls into the category of small-scale software, the calculation of the Value Adjustment Factor (VAF) used a constant of 0.65 and 0.01 to determine the influence of each TDI point on the final VAF result, resulting in a VAF of 0.71.

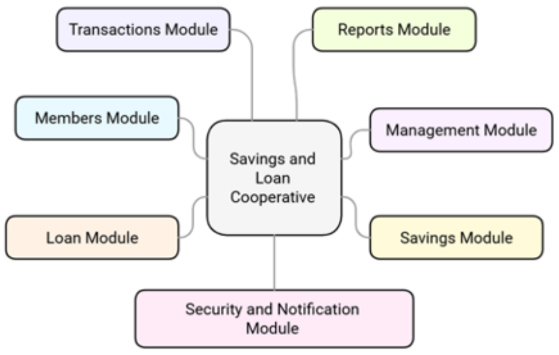

Fig. 3. Savings and Loan Cooperative Module

The variables needed to calculate the cost of the software include the adjusted function point, the average time taken for each function, and the hourly rate of programmers to perform the function. According to IFPUG standards, each function is typically completed in 15 hours. The hourly rate depends on the programmer's expertise level. For this calculation, the research team used an average hourly rate of Rp. 100,000. Therefore, the estimated cost for developing the software for the savings and loan cooperative is 127 * 15 * 100,000 = Rp. 190,500,000.00.

This study highlights the practicality of using FPA for software cost estimation, especially in settings where project scopes are relatively small but the functionalities involved are critical to the organization's operations. It provides a quantifiable and methodical approach to estimating costs that can significantly aid cooperative management in budget planning and vendor selection. Furthermore, these findings contribute to the broader understanding of applying FPA in different organizational settings, offering insights that may be valuable for other researchers and software development professionals.

V. Conclusion

The implementation of the Function Point Analysis (FPA) method requires a team that possesses programming skills to accurately determine the complexity level of each function. FPA is versatile and can be employed to estimate costs for small, medium, and large-scale software projects. The rates for software development are influenced by two factors: the skill or experience of the programmer and the policies of the software company.

In this study, the application of the FPA method for estimating the cost of developing software for the savings and loan cooperative resulted in the creation of seven modules with a total of 127 functions. The estimated cost for the development is Rp. 190,500,000.00. This study demonstrates that FPA is a practical and effective tool for financial planning in software development, providing a structured approach that can guide cooperatives and similar organizations in their vendor selection and budgetary processes. The adaptability of FPA to different project sizes and complexities also highlights its utility across various software development scenarios.

References

[1] S. P. D. Ardi, I. Bernarto, N. Sudibjo, A. Yulianeu, H. A. Nanda, and K. A. Nanda, "The Secret to Enhancing Innovativeness in the Digital Industry," *International Journal of Innovation, Creativity and Change*, vol. 12, no. 12, pp. 225-243, 2020.

[2] A. Latif, L. A. Fitriana, and M. R. Firdaus, "Comparative Analysis Of Software Effort Estimation Using Data Mining Technique And Feature Selection," *JITK (Jurnal Ilmu Pengetahuan dan Teknologi Komputer)*, vol. 6, no. 2, February 2020. doi: 10.33480/jitk.v6i2.1968, pp. 167-174.

[3] R. Henglie, Y. Purnomo, and J. A. Ginting, "Predicting Increased H-Index For Research Journals Using The Cost-Sensitive Selective Naive Bayes Classifiers Algorithm," *Jurnal Algoritma, Logika dan Komputasi*, vol. VII, no. 01, pp. 643-651, 2024. doi: http://dx.doi.org/10.30813/j-alu.v2i2.6028.

[4] E. W. dos Santos, I. Nunes, and D. Jannach, "Developer perceptions of modern code review processes in practice: Insights from a case study in a mid-sized company," *Journal of Systems and Software*, vol. 222, Apr. 2025. https://doi.org/10.1016/j.jss.2024.112288.

[5] E. Lumba and L. Hakim, "Digital Library Application Design," *International Journal of Multidisciplinary Research and Publications (IJMRAP)*, vol. 6, no. 2, pp. 193-197, 2023.

[6] A. Dennis, B. Wixom, and D. Tegarden, *Systems Analysis and Design: An Object-Oriented Approach with UML*. John Wiley & Sons, 2020.

[7] I. G. N. Suryantara, Michael, J. F. Andry, and J. A. Ginting, "Pengembangan Aplikasi Operasional Restoran Dengan Framework Scrum (Studi Kasus: Restoran PT. XYZ)," *INFOTECH: Journal of Technology Information*, vol. 9, no. 2, Nov. 2023. doi: https://doi.org/10.37365/jti.v9i2.168.

[8] D. Fahriani and T. R. Zubaidah, "Financial Performance Analysis of The Saving and Loan Cooperative," *JKIE (Journal Knowledge Industrial Engineering)*, vol. 10, no. 1, Apr. 2023, pp. 38-49. https://doi.org/10.35891/jkie.v10i1.4125.

[9] B. Kiiza and G. Omiat, "The Impact of Savings and Credit Cooperatives on Household Welfare: Evidence from Uganda," *Journal of Economics and Public Finance*, vol. 7, no. 3, 2021. doi: 10.22158/jepf.v7n3p33.

[10] International Function Point Users Group. Accessed on 10 October 2024. Available online: https://ifpug.org/.

[11] M. F. Hillman and A. P. Subriadi, "40 Years Journey of Function Point Analysis: Against Real-time and Multimedia Applications," *Procedia Computer Science*, vol. 161, pp. 266-274, 2019. https://doi.org/10.1016/j.procs.2019.11.123.

[12] "Function Point Analysis - Introduction and Fundamentals," Accessed on 12 October 2024. Available online: https://www.fingent.com/blog/function-point-analysis-introduction-and-fundamentals/.

[13] D. N. Malleswari, D. Rakesh, K. Subrahmanyam, and D. Vadlamudi, "An Efficient Model for Software Quality Analysis Based on User and Developer Interaction," *International Journal of Innovative Technology and Exploring Engineering (IJITEE)*, vol. 9, issue 2, Dec. 2019. DOI: 10.35940/ijitee.B6822.129219.

[14] A. W. Wambua and B. M. Maake, "Characterizing Software Quality Assurance Practices in Kenya," *INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING & COMPUTER SYSTEMS (IJSECS)*, vol. 8, issue 1, pp. 22-28, 2022. https://doi.org/10.15282/ijsecs.8.1.2022.3.0093.

[15] N. Rachmat and S. Saparudin, "Estimasi Ukuran Perangkat Lunak Menggunakan Function Point Analysis - Studi Kasus

Aplikasi Pengujian dan Pembelajaran Berbasis Web," *Prosiding Annual Research Seminar 2017*, vol. 3, no. 1.

[16] D. Ramos-Vidal, W. K. G. Assunção, A. Cortiñas, M. R. Luaces, O. Pedreira, and Á. S. Places, "SPL-DB-Sync: Seamless database transformation during feature-driven changes," *Journal of Systems and Software*, vol. 222, Apr. 2025. https://doi.org/10.1016/j.jss.2024.112285.