Malaysian Journal of Computing, 10 (1): 2084-2098, 2025
Copyright © UiTM Press
eISSN: 2600-8238

APPLYING FUNCTION POINT METHOD TO MEASURE ENTREPRENEURSHIP SYSTEM SIZE AT UiTM

Gazairi Ghazali[1]*, Mohd Hairy Mohamaddiah[2], Ahmad Sobri Abdul Haris[3], and Syarifah Nurul Afzan Syed Mohammed[4]

[1, 2,3,4]Infostructure Department, Office of Infrastructure & Infostructure, Universiti Teknologi MARA, 40450 Shah Alam, Selangor, Malaysia*

[1*]gazairi@uitm.edu.my, [2]hairy@uitm.edu.my,
[3]ahmadsobri@uitm.edu.my, [4]sy.nurul.afzan@uitm.edu.my

ABSTRACT

Traditional estimation methods for assessing the size and scope of software systems often result in inaccuracies, leading to budget overruns and timeline delays. This is particularly problematic in the development of complex systems like the Entrepreneurship Information Management System (MASMED2U) at UiTM. There is a need for a more precise and user-centric approach to measure system size, costs, and development timelines effectively. This paper investigates the use of Function Point Analysis (FPA) to assess the size of the Entrepreneurship Information Management System (MASMED2U) at UiTM. Adhering to the IFPUG standard ISO/IEC 20926:2009, FPA measures system size using function points, effectively addressing the inaccuracies of traditional estimation methods. By focusing on user-centric functional features, including data and transactions, FPA offers a more precise solution for estimating costs and timelines. FPA also enables system size measurement throughout the development lifecycle, with Early and Quick FPA facilitating initial phase measurements for budget proposals.

Keywords: *Costing, Software Development, Software Engineering, Software Estimation, User Requirement.*

Received for review: 03-01-2025; Accepted: 20-03-2025; Published: 01-04-2025
DOI: 10.24191/mjoc.v10i1.4535

1. Introduction

Accurate estimation of software size and scope is crucial for effective project management, budgeting, and scheduling. Traditional estimation methods often fall short in providing the necessary precision, leading to common issues such as budget overruns and timeline delays. These issues are especially critical in the development of complex information systems like the Entrepreneurship Information Management System (MASMED2U) at Universiti Teknologi MARA (UiTM). To address these challenges, this paper explores the application of Function Point Analysis (FPA) as a more precise, user-centric approach for estimating system size, costs, and development timelines. Traditional estimation methods have the limitations which can lead to underestimation of effort and missed deadlines.

FPA, standardized by the International Function Point Users Group (IFPUG) under ISO/IEC 20926:2009, offers a structured methodology to measure system size in terms of function points. This approach focuses on user-centric functional features, including data and transactions, providing a more accurate estimation framework compared to traditional methods. Additionally, FPA considers non-functional requirements and enables system size measurement throughout the development lifecycle. This study evaluates the implementation of FPA in the development of MASMED2U, aiming to demonstrate its effectiveness in improving estimation accuracy and achieving cost savings since the development is executed in house.

This system consists of three main modules. The MyENT module manages the registration of entrepreneurial students, while the Tabung Keusahawanan Siswa (TKS) module handles student loan applications, including interviews, loan disbursements, and repayment management. The Program module oversees various entrepreneurial initiatives, such as the Teaching Innovation Exploration (TIE), which manages entrepreneurship courses for postgraduate students, and the Unicorn Scholar Program (USP), which facilitates entrepreneurial apprenticeship programs. This system plays a crucial role in supporting student entrepreneurship programs, aligning with the Ministry of Higher Education's vision of equipping students with entrepreneurial skills and knowledge. In the rapidly evolving digital ecosystem, understanding and anticipating customer demands, preferences, and buying habits can be effectively achieved through data analytics.

The rest of the paper is structured as follows: Section 2 (Related Work) presents a review of several studies on estimation methods and a comparison between Function Point Analysis and other traditional methods. This section also provides a brief overview of the methodological techniques used in the study. Section 4 and (Results and Discussion) presents the experimental findings, followed by Section 6 (Conclusion), which summarizes the research results.

2. Related Work

Software development projects rely heavily on accurate estimation to ensure successful delivery within budget and timeframe. Traditional estimation methods, while facing challenges, remain prevalent and offer valuable tools for project planning.

2.1 Traditional Estimation Methods

Traditional software estimation methods, such as expert judgment, analogy-based estimation, and parametric models, often lack the precision required for complex systems (Bohem, 1981, Wideman, 2002, Alshammari et. al, 2022). These methods rely heavily on historical data and subjective assessments, which can lead to significant variances in estimates (Jørgensen, 2004). For instance, COCOMO (Constructive Cost Model) and SLIM (Software Life Cycle Management) are widely used parametric models that, while useful, may not adequately capture the intricacies of user requirements and functional complexities (Boehm, 1981; Putnam & Myers, 1997). COCOMO (Constructive Cost Model) is a parametric model that estimates software development effort based on the size of the project (measured in LOC) and other cost drivers, such as complexity, reliability, and experience of the development team (Boehm, 1981). Lines of Code (LOC) is one of the oldest and most straightforward methods for measuring software size (McConnell, 1993). This metric counts the number of lines in a program's source code, including both executable statements and comments (Bohem, 1981).

SLIM (Software Life Cycle Management) is another parametric model that estimates software size, effort, and duration based on historical project data and the Rayleigh curve (Putnam, 1978, Sharma et. al, 2021). Man-days estimation approach estimates the total effort required to complete a software project by calculating the number of days a person (or a team) would need to deliver the project (Jørgensen & Sjøberg, 2004). Expert judgment is one of the most used traditional estimation methods, relying on the experience and intuition of seasoned developers and project managers to estimate software size and effort (Jørgensen, 2004). Analogy-based estimation involves comparing a new software project with similar past projects to estimate size, effort, and cost (Shepperd et. al, 1997, Kumar, et. al, 2023). Both methods heavily rely on the experience and knowledge of individuals, typically seasoned developers, project managers, or subject matter experts, to make accurate estimations. However, both methods may struggle to scale effectively for large or highly complex projects, where more systematic and data-driven approaches might be necessary for accurate estimation.

Function Point Analysis (FPA) is a structured technique used to measure the functional size of a software system based on the functions it provides to the user. Unlike traditional methods like Lines of Code (LOC) or man-days estimation, which focus on the volume of code, or the time required to complete a project, FPA assesses the software's functionality from the user's perspective, offering a more objective and consistent measure of software size (Albrecht, 1979, Van Hai et. al, 2022). Function Point Analysis (FPA) was developed by Allan Albrecht at IBM in the late 1970s to measure the functionality delivered by software. Unlike traditional methods that focus on the volume of code or effort required, FPA quantifies software based on its functional requirements from the user's perspective. This approach considers various elements, including inputs, outputs, user interactions, internal files, and external interfaces. The International Function Point Users Group (IFPUG) standard, ISO/IEC 20926:2009, provides a comprehensive framework for applying FPA, ensuring consistency and reliability in measurements (IFPUG, 2009). Numerous studies have shown that FPA can enhance estimation accuracy and provide a more reliable basis for project planning and resource allocation. For instance, Albrecht's foundational work demonstrated that FPA could effectively measure software size across various types of projects, making it a versatile tool for software estimation (Albrecht, 1979). Additionally, FPA's focus on user requirements allows for a better alignment between software functionality and project goals, reducing the risk of scope creep and ensuring that development efforts are directed towards meeting user needs (Jørgensen, 2004).

2.2 Comparison Between Function Point Analysis and Other Traditional Methods

Table 1 below derives the comparison between Function Point Analysis method and other Traditional Methods in Software Estimation:

Table 1(a). Comparison Matrix between Function Point Analysis and Other Traditional Methods in Software Estimation

Method	Strength	Weakness
FPA	• Independent of programming languages, development methodologies, or technologies. This makes it a versatile tool for various projects (Abran & Nguyenkim, 1991). Standardized by the International Function Point Users Group (IFPUG), ensuring consistency and reliability in measurement across different projects and organizations (IFPUG, 2010).	• Can be complex to learn and apply, requiring specialized training and expertise. (Gencel & Demirörs, 2008). The process of counting function points can be time-consuming, especially for large or complex systems. (Abran & Nguyenkim, 1991).
LOC	LOC has been widely used because it is simple to understand and apply across various programming languages (McConnell, 1993).	LOC does not consider non-coding tasks, such as design and testing, which are crucial in modern software development (Fenton & Pfleeger, 2014).
Man-days	Commonly used in project management to allocate resources, schedule timelines, and estimate costs (Boehm, 1981).	Highly dependent on the accuracy of the initial assumptions regarding team productivity, project complexity, and unforeseen challenges. (Jørgensen, 2004).
COCOMO (Boehm, 1981).	Effective for high-level cost estimation	Reliance on LOC as a primary input can be a limitation

Table 1(b). Comparison Matrix between Function Point Analysis and Other Traditional Methods in Software Estimation

Method	Strength	Weakness
SLIM (Putnam, 1978).	Ability to model the entire software development lifecycle, from initial design to maintenance	Relies heavily on historical data and statistical models, which may not always capture the specific functional requirements of a new project
Expert judgement (Jørgensen, 2004).	Can be quick and cost-effective	Inherently subjective and prone to biases, such as overconfidence or reliance on recent experiences.
Analogy-based estimation (Shepperd & Schofield, 1997).	Can be effective when there is a strong similarity between projects	Can also lead to inaccurate estimates if the differences between projects are not adequately accounted for

Based on the above comparison, FPA able to address and complement other traditional methods in software estimation. For COCOMO, FPA, in contrast, focuses on the functionality delivered to the user rather than the amount of code written (Albrecht, 1979). For SLIM, FPA complements SLIM by providing a functional perspective on software size (Putnam, 1978; Symons, 1991). For expert judgment, FPA offers a more objective alternative to expert judgment by providing a structured and standardized approach to software estimation (Albrecht, 1979; Gencel & Demirörs, 2008). For analogy-based estimation, FPA improves upon analogy-based estimation by providing a detailed breakdown of the software's functional components, making it easier to identify and adjust for differences between projects (Albrecht, 1979; Shepperd & Schofield, 1997). As a result, these traditional methods are often supplemented or replaced by more sophisticated approaches, such as Function Point Analysis (FPA), which aim to provide a more comprehensive and accurate measure of software size (Albrecht, 1979; Symons, 1991). The error associated with function point measurement by analysts with varying levels of experience was found to be relatively low, with a mean difference of approximately 10%, as reported in a case study by Kemerer (1990).

3. Methods

We used Function Point Analysis to assess the size of the Entrepreneurship Information Management System (MASMED2U) at UiTM. FPA calculation breaks down the software into five main components: External Inputs (EI), External Outputs (EO), External Inquiries (EQ), Internal Logical Files (ILF), and External Interface Files (EIF). Each component is assigned a weight based on its complexity, and the total function points are calculated by summing the weighted components. This method allows for a more detailed understanding of the software's scope and provides a common metric that can be used across different projects and technologies (Symons, 1991).

Early and Quick FPA methods aim to deliver timely and reasonably accurate estimates, aiding in budget proposals and project planning during the project's inception phase. These variants are valuable tools for project managers seeking to make informed decisions with limited information (Jørgensen, 2004; Symons, 1991). The calculation of Function Points for the MASMED2U system follows a systematic approach to quantify the system’s functionality. We follow the standards or guidelines published by International Function Point Users Group (IFPUG) in 2009. It will involve several steps.

The first step is the Functional Components Calculation. The calculation begins with identifying and quantifying the system's functional components, which are categorized into two main groups: transaction functions and data functions. Transaction functions include inputs such as user input and data submissions while outputs include reports or data retrievals. Data functions encompass internal files such as databases or data stores while external interfaces cover interactions with other systems.

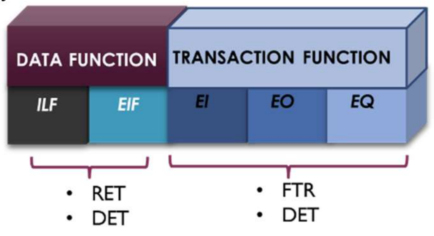

Figure 1. Functional Components in FPA.

Referring to Figure 1, We define data function complexity in table 2 below:

Table 2. Data Function Complexity.

Entity Name	Attribute	Component Type	Ret	Det	Complexity
Student	atrribute 1 attribute n +1	ILF	3	21	S
Course	atrribute 1 attribute n +1	ILF	1	14	R
Scholarship	attribute n attribute n +1	EIF	1	10	L

Next, we calculate the transaction function complexity. The complexity is defined based on table 3 below:

Table 3. Transaction Function Complexity.

Function	Component Type	Ftr	Det	Complexity
View list	EQ	1	1	L
View report summary	EO	2	5	L
Add student	EI	3	21	H
Edit student	EQ	3	29	H
	EI	3	29	H
Delete Student	EI	1	3	H

Each complexity is defined based on Complexity Metric referring to the guide in IFPUG. The second step is to calculate Unadjusted Function Points (uFP). The total function points are derived by summing up the values assigned to the transactions and data functions. Each function is assigned a specific weight based on its complexity and contribution to the system's functionality. Table 4 below defines the uFP. The value in the table is as an example, to derive the uFP. The size of FP in table 4, is defined based on Complexity Metric referring to the guide by IFPUG.

Table 4. Unadjusted Function Point.

Component Type	Complexity Level			
	Low (L)	Medium (M)	High (H)	Total
ILF	1 X 7	1 X 10	0 X 15	17
EIF	1 X 5	0 X 7	0 X 10	5
EI	1 X 3	0 X 4	2 X 6	15
EO	1 X 4	0 X 5	0 X 7	4
EQ	1 X 3	0 X 4	1 X 6	9
Total Unadjusted FP				50

The third step is calculating Value Adjustment Factor (VAF). This factor accounts for non-functional requirements such as performance, security, and user experience. The VAF adjusts the uFP to reflect the impact of these non-functional aspects on the system. Table 5 below define the VAF, with a value as an example to derive the VAF value.

Table 5. Value Adjustment Factor (VAF).

General System Characteristic (GSC)	**(0-5)**	**General System Characteristic (GSC)**	**(0-5)**
1. Data Communications	5	8. On-Line Update	5
2. Distributed Data Processing	3	9. Complex Processing	2
3. Performance	5	10. Reusability	0
4. Heavily Used Configuration	0	11. Installation Ease	0
5. Transaction Rate	5	12. Operational Ease	0
6. On-Line Data Entry	5	13. Multiple Sites	0
7. End-User Efficiency	5	14. Facilitate Change	0
Total Degree of Influence (TDI)		Sum (1-14)	35
Value Adjustment Factor (VAF)		(TDI * 0.01)+0.65	1.00

The fourth step is calculating Adjusted Function Points (AFP). The final measurement of software size is calculated by multiplying the Unadjusted Function Points (uFP) by the Value Adjustment Factor (VAF). The formula is based on equation 1 below:

$$AFP = UFP\ x\ VAF \tag{1}$$

This adjusted measure provides a comprehensive estimate of the software's functionality, considering both its functional and non-functional aspects.

The fifth step is calculating effort of system development in man-days. The formula is based on equation 2 below:

$$Effort(mandays) = AFPx10/8\ hour \tag{2}$$

For equation 2, AFP value is derived from the FPA calculation while the effort conversion factor is based on the average development effort in Malaysia, which is 10 man-hours per 8-hour workday (Czarnacka-Chrobot, B., 2012).

The sixth step is calculating system development cost**.** The formula is as follows:

$$Development\ Cost = AFP\ x\ Development\ Cost\ per\ FP\ (1200) \tag{3}$$

Based on above formula, AFP is derived from the FPA calculation while RM1,200.00 is the cost per Function Point, reflecting the average development cost in Malaysia (Czarnacka-Chrobot, B., 2012). All of the steps above will be applied to estimate the effort and cost associated with MASMED2U system at the following section. We will also compare the software costing from FPA calculation with analogy-based estimation to show the different.

4. Results

4.1.1 FPA Calculations

The calculation of Function Points for the MASMED2U system follows a systematic approach as defined in Method topic as above involves following steps. For the first step, we performed Functional Components Calculation, Data Function Complexity for MASMED2U. The result is defined in table 6:

Table 6: Complexity Defined for Data Function in MASMED2U.

ENTITY NAME	COMPONENT TYPE	RET	DET	COMPLEXITY
tbl_bil_create_fais	EIF	1	10	L
tbl_bil_request_fais	EIF	1	3	L
tbl_bill	ILF	1	10	M
tbl_category_parameter	ILF	1	3	L
tbl_disbursement	ILF	1	6	L
tbl_disbursement_loan	EIF	1	2	L
tbl_didbursement_loan_details	EIF	1	4	L
tbl_events	ILF	1	3	L
tbl_menu_item	ILF	1	5	L
tbl_parameter	ILF	1	4	L
tbl_permissions	ILF	1	2	L
tbl_perniagaan	ILF	2	17	L
tbl_perniagaan_pinjaman	ILF	3	24	M
tbl_program	ILF	7	26	H
tbl_program_category	ILF	2	11	L
tbl_receipt_payment	ILF	2	26	M
tbl_request_payment_fais	EIF	1	15	L
tbl_roles	ILF	1	2	L
tbl_users	ILF	1	9	L
tbl_users	EIF	1	9	L
tbl_user_information	ILF	1	2	L

Transaction Function Complexity for MASMED2U is defined based on following table 7:

Table 7(a). Complexity Defined for Transaction Function in MASMED2U.

Function	Component Type	FTR	DET	Complexity
Senarai Menu	EQ	1	8	L
Menu Baru	EI	1	9	L
Edit Menu	EI,EQ	1	9	L
Senarai Peranan	EQ	1	4	L
Daftar Baru Peranan	EI	1	4	L
Senarai Pengguna dan Peranan	EI,EQ	1	7	L,L
Daftar Baru tetapan Pengguna	EI	1	12	L
Edit Tetapan Pengguna	EI,EQ	1	13	L,L
Senarai Kategori Parameter	EQ	1	5	L
Daftar Baru Kategori Parameter	EI	1	5	L
Edit Kategori Parameter	EI,EQ	1	5	L,L
Senarai Parameter	EQ	1	6	L
Daftar Baru Parameter	EI	1	6	L
Edit Parameter	EI,EQ	1	6	L,L
Senarai MyEnt	EQ	1	7	L
Daftar Baru MyEnt	EI	1	22	M
Senarai Status dan Jumlah	EQ	1	2	L
Kriteria Carian	EQ	1	6	L
Senarai TKS	EQ	1	10	L
Daftar Baru TKS	EI	2	31	H
Edit TKS	EI,EQ	2	31	M,M
Senarai Bil	EQ	1	8	L
Daftar Baru Billing	EI	1	4	L
Billing Information	EI,EQ	1	9	L,L
Senarai Interview TKS	EQ	1	6	L
Tambah Temuduga	EI	1	6	L
Edit TKS	EI,EQ	2	12	L,L
Edit Status	EI,EQ	1	11	L,L
Senarai Pengeluaran Pinjaman	EQ	1	7	L
Daftar Baru Pengeluaran Pinjaman	EI	1	3	L
Edit Pengeluaran Pinjaman	EI,EQ	2	12	M,M
Senarai Kategori Program	EQ	1	7	L
Daftar Baru Kategori Program	EI	1	8	L

Table 7(b). Complexity Defined for Transaction Function in MASMED2U.

Function	**Component Type**	**FTR**	**DET**	**Complexity**
Edit Kategori Program	EI,EQ	2	10	L,L
Daftar Baru Aktiviti	EI	1	6	L
Edit Aktiviti	EI,EQ	1	6	L,L
Senarai Program	EQ	1	7	L
Daftar baru Program	EI	1	9	L
Edit program	EI,EQ	3	19	H,M
Tambah Pelajar	EI	1	7	L
Update Overall Result	EI,EQ	1	7	L,L
Senarai Mengikut Status	EQ	1	6	L
Details Aktiviti	EQ	2	8	M
Edit Aktiviti	EI,EQ	1	6	L,L
Senarai Program	EQ	1	7	L
Daftar baru Program	EI	1	9	L
Edit program	EI,EQ	3	19	H,M
Tambah Pelajar	EI	1	7	L
Update Overall Result	EI,EQ	1	7	L,L
Senarai Mengikut Status	EQ	1	6	L
Details Aktiviti	EQ	2	8	M

The second step, is to calculate Unadjusted Function Points (uFP), table 8 calculates the UFP in MASMED2U:

Table 8. Calculation for Unadjusted FP in MASMED2U.

Component Type	**Complexity Level**			
	Low (L)	**Medium (M)**	**High (H)**	**Total**
ILF	10 X 7	3 X 10	1 X 15	115
EIF	7 X 5	0 X 7	0 X 10	35
EI	23 X 3	3 X 4	2 X 6	93
EO	0 X 4	0 X 5	0 X 7	0
Eq	25 X 3	4 X 4	0 X 6	91
Total Unadjusted FP				334

For the third step, we calculate *Value Adjustment Factor (VAF)*. Table 9 derived the *VAF* results in MASMED2U:

Table 9. Calculation for Value Adjustment Factor in MASMED2U.

General System Characteristic (GSC)	**(0-5)**	**General System Characteristic (GSC)**	**(0-5)**
1. Data Communications	**5**	8. On-Line Update	**5**
2. Distributed Data Processing	**3**	9. Complex Processing	**2**
3. Performance	**5**	10. Reusability	**0**
4. Heavily Used Configuration	**0**	11. Installation Ease	**0**
5. Transaction Rate	**5**	12. Operational Ease	**0**
6. On-Line Data Entry	**5**	13. Multiple Sites	**0**
7. End-User Efficiency	**5**	14. Facilitate Change	**0**
TOTAL DEGREE OF INFLUENCE (TDI)		**SUM (1-14)**	**35**
VALUE ADJUSTMENT FACTOR (VAF)		**(TDI * 0.01)+0.65**	**1.00**

For the fourth step, based on equation 1, we calculate *Adjusted Function Points (AFP*),:

$$AFP = 334\,x\,1 = 334 \tag{4}$$

This adjusted measure provides a comprehensive estimate of the software's functionality, taking into account both its functional and non-functional aspects. For the fifth step, the effort for system development in man-days based:

$$\begin{aligned} Effort(mandays) &= AFPx10/8\,hour \\ &= 334\,x\,10/\,8 \\ &= 417.5 \text{ mandays} \end{aligned} \tag{5}$$

Next, the final step, the system development cost will be:

$$Development\ Cost: 334\,x\,RM1200\ =\ RM400{,}800.00 \tag{6}$$

4.2 FPA Comparison with Analogy-based Estimation

During the initial stages of the MASMED2U system development project, cost estimation was conducted using the analogy-based estimation method. This approach involved estimating the cost by multiplying the monthly salary corresponding to each job grade by the project's duration, which was set at 18 months. The project duration was determined based on previous experience with similar system development projects, providing a reliable benchmark for the estimation process. Table 10 defines the costing for MASMED2U at the initial stage:

Table 10. Cost estimation for MASMED2U using Analogy-Based Estimation.

Project Item	**One Time Cost (Rm)**				**Total Cost (RM)**
	Project Component (RM)	**Human Resource (Existing)**	**Human Resource (Contract)**	**Humean Resource (JV/ Outsource)**	
System Development Technical Team: • 1 F48 (RM11761 x • 18 months) • 1 F44 (RM10504 x • 18 months) • 1 FA29 (RM5684 x • 18 months)	0	512,946.00	0	0	512,946.00

For comparison, as the project neared completion, we performed a software cost estimation using FPA for MASMED2U based on 5.1.

Table 11 depicts the comparison of cost estimation value, using both methods, function point analysis and analogy-based estimation:

Table 11. Comparison of Cost Estimation between FPA and Analogy based Estimation.

Estimation Method	**Function Point Analysis**	**Analogy-based Estimation**
Cost (RM)	400,800	512,946

5. Discussion

The relationship between development cost and effort in FPA is intrinsically linked to the accuracy of function point measurement. By providing a standardized way to assess software size based on user requirements, FPA helps in predicting the required effort and associated costs more accurately. This allows organizations to plan and allocate resources more effectively, ensuring that projects are completed within budget while meeting functional requirements (Albrecht, 1979; Jones, 1996). The result in table 10, depicts the calculated value for analogy-based estimation. In common practices for analogy-based estimation, cost estimation process is crucial at the early phases of software development. However, changes may also occur in other phases, such as the development and deployment processes (Seetharaman et al.,2005). This is due to the changing needs and requirements over time. It will influence the software cost and the development effort.

The results in table 11, indicate that the calculation using FPA is more accurate. This is because FPA includes both transaction functions and data functions, which contribute to the overall system. On the other hand, analogy-based estimation method estimates the timeline and cost based on the number of modules involved and the experience with similar previous systems. In addition, this can be problematic as the previous systems might differ in domain and business processes. The effectiveness of analogy-based estimation relies heavily on the similarity between the new project and the past projects. If the new project differs significantly in scope, technology, or requirements, the estimates may be inaccurate (Shepperd & Schofield, 1997).

There are several benefits and insights by using FPA. Calculation of effort and cost provides a clear and quantifiable measure of the project's requirements, leading to more accurate budget and scheduling forecasts. By applying the cost per Function Point, the MASMED2U project was able to identify the total development cost effectively, facilitating the cost saving calculation. FPA also enables the estimation accuracy and cost savings**.** It provides more precise measurements of system size, leading to more accurate cost and timeline estimates. The in-house development of MASMED2U, guided by FPA estimation, resulted in significant cost savings compared to outsourcing or using less precise estimation methods.

6. Conclusion

Implementing Function Point Analysis (FPA) in the MASMED2U system development has provided valuable insights into the effectiveness of this methodology for software size estimation. FPA's structured approach, which involves quantifying functional components based on user requirements, proved instrumental in achieving precise estimations for effort and cost.One of the primary strengths of FPA is its ability to provide an objective measure of software size by focusing on functionalities rather than code metrics or subjective assessments. This approach helped in accurately estimating the total effort required for system development. By applying the formula *Effort in man-days*, we could translate Function Points into practical effort estimates. This method ensures that the estimated effort aligns with local industry standards, enhancing the reliability of the project plan.

Furthermore, the cost estimation using FPA, calculated with *Development Cost*, enabled precise financial forecasting. This calculation reflected the average development cost in Malaysia, allowing for effective budget management and resource allocation. Reflecting on the use of FPA, its application contributed to improved project planning and cost control. The method provided a more comprehensive understanding of the system's requirements and complexities, leading to more accurate and actionable estimates. However, it is also important to recognize that while FPA offers significant advantages, its effectiveness depends on the accuracy of the initial functional requirements and the consistent application of its principles throughout the project lifecycle.

The Function Point Analysis (FPA) method provides UiTM with a reliable, user-centric approach for estimating the size, costs, and development timelines of software systems at various project stages—whether in the initial phases, midway, or at completion. By adhering to the IFPUG standard ISO/IEC 20926:2009, FPA addresses the limitations of traditional estimation methods, ensuring more accurate and reliable project planning and budgeting. Its application in projects like MASMED2U has demonstrated significant benefits, including improved estimation precision and notable cost savings. For in-house projects, FPA aids in calculating cost efficiencies, while for outsourced initiatives, it helps determine optimized costs. This aligns with UiTM's objectives and supports the Ministry of Higher Education's mission to cultivate entrepreneurial skills among students. Future research should explore the broader applicability of FPA across diverse educational and entrepreneurial settings to further validate its effectiveness and impact.

Acknowledgements

Myzatul Akmam Sapaat from Jabatan Digital Negara contributed valuable insights for the reference on *What Is the Cost of One IFPUG Method Function Point? – A Case Study* by Beata Czarnacka-Chrobot.

Funding

The author(s) received no specific funding for this work.

Author Contribution

Author1 wrote the literature review, method, result, reflection and discussion. Author2 wrote the method and oversaw the article writing. Author3 and Author4 conducted the result calculations and analysis.

Conflict of Interest

The authors have no conflicts of interest to declare.

References

Abdul Aziz, M., Mustakim, N. A., & Abdul Rahman, S. (2024). Decision tree and rule-based classification for predicting online purchase behavior in Malaysia. *Malaysian Journal of Computing (MJoC)*, *9*(2), 1905-1915.

Abran, A., & Nguyenkim, H. (1991). "Measurement of the Functional Size of Software: New Variations on Old Themes." *IEEE Transactions on Software Engineering*, 17(7), 635-643.

Albrecht, A. J. (1979). *Measuring Application Development Productivity*. Proceedings of the IBM Application Development Symposium, 83-92.

Alshammari, F. H. (2022). Cost estimate in scrum project with the decision-based effort estimation technique. *Soft Computing*, *26*(20), 10993-11005.

Boehm, B. W. (1981). Software Engineering Economics. Prentice-Hall.

Czarnacka-Chrobot, B. (2012). What Is the Cost of One IFPUG Method Function Point?-Case Study. In *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)* (p. 1). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Davide, F., Giovanni, C. and Roberto, M. (2023) Estimating Software Functional Size via Machine Learning.

Fenton, N. E., & Pfleeger, S. L. (2014). *Software Metrics: A Rigorous and Practical Approach*. CRC Press.

Gencel, C., & Demirörs, O. (2008). "Functional Size Measurement Revisited." *IEEE Transactions on Software Engineering*, 34(6), 796-810.

IFPUG (International Function Point Users Group). (2009). *ISO/IEC 20926:2009 – Software and Systems Engineering – Function Point Analysis – Counting Practices Manual*. IFPUG.

IFPUG (International Function Point Users Group). (2010). *Function Point Counting Practices Manual, Release 4.3.1*.

Jones, C. (1996). *Applied Software Measurement: Assuring Productivity and Quality*. McGraw-Hill.

Jørgensen, M. (2004). "A Review of Studies on Expert Estimation of Software Development Effort." Journal of Systems and Software, 70(1-2), 37-60.

Jørgensen, M., & Sjøberg, D. I. K. (2004). "A Survey of Expert Estimation of Software Development Effort." *Empirical Software Engineering*, 9(3), 223-249.

Kammy, M. and Ryan, H. (2020). But Wait, There's More! Using Simple Function Point Analysis for Your Cost, Schedule & Performance Needs

Kumar, K. H., & Srinivas, K. (2023). Preliminary performance study of a brief review on machine learning techniques for analogy based software effort estimation. *Journal of Ambient Intelligence and Humanized Computing*, *14*(3), 2141-2165.

Kemerer, C. F. (1990). *Reliability of function points measurement: A field experiment*. Communications of the ACM, 33(2), 85-97.

McConnell, S. (1993). Code Complete: A Practical Handbook of Software Construction. Microsoft Press.

Putnam, L. H. (1978). *A General Empirical Solution to the Macro Software Sizing and Estimating Problem*. IEEE Transactions on Software Engineering, SE-4(4), 345-359.

Putnam, L. H. (1992). Long-Term High-Quality Software: A New Way of Measuring and Estimating. Prentice-Hall.

Putnam, L. H., & Myers, W. (1997). Measures for Excellence: Reliable Software On Time, Within Budget. Prentice-Hall.

Shepperd, M., & Schofield, C. (1997). "Estimating Software Project Effort Using Analogy." *IEEE Transactions on Software Engineering*, 23(12), 736-743.

Seetharaman,N., Senthilvelmurugam,M. and Subramanian,T. Budgeting & Accounting of Software Cost. Journal of Digital AssetManagement. Vol. 1 No.5.pp 347-359.(2005).

Sharma, S., & Vijayvargiya, S. (2021). Applying soft computing techniques for software project effort estimation modelling. In *Nanoelectronics, Circuits and Communication Systems: Proceeding of NCCS 2019* (pp. 211-227). Springer Singapore.

Symons, C. J. (1991). *Function Point Analysis: Measurement Practices for Successful Software Projects*. John Wiley & Sons.

Van Hai, V., Nhung, H. L. T. K., Prokopova, Z., Silhavy, R., & Silhavy, P. (2022). Toward improving the efficiency of software development effort estimation via clustering analysis. *IEEE Access*, *10*, 83249-83264.

Wei, X., Danny, H., Luiz, F. C. and Faheem, A. (2020). Updating Weight Values for Function Point Counting.

Wideman, R. M. (2002). "Project Management and the Triple Constraint." *Project Management Knowledge Area*, 1(1), 1-6