INTERNATIONAL JOURNAL OF MULTIDISCIPLINARY RESEARCH AND ANALYSIS

ISSN(print): 2643-9840, ISSN(online): 2643-9875

Volume 09 Issue 01 January 2026

DOI: 10.47191/ijmra/v9-i1-49, Impact Factor: 8.266

Page No. 374-382

Cost Estimation Based on Use Case Points in The Development of AIoT Monitoring Infusion Test Software

Zakiy Qiros Muhammad[1], Renny Sari Dewi[2]

[1,2]Department of Digital Business, Faculty of Economics and Business, Surabaya State University, Indonesia

ABSTRACT: Artificial Intelligence of Things (AIoT) based infusion drip monitoring software plays an important role in improving the effectiveness and safety of healthcare services. However, most existing studies on infusion monitoring systems still focus primarily on hardware aspects, sensor accuracy, and technical integration, while software development effort planning and cost estimation, particularly for web based monitoring dashboards, remain insufficiently discussed. Therefore, this study aims to estimate the development effort and cost of AIoT based infusion drip monitoring software when implemented at an industrial scale. The proposed approach employs the Use Case Point s (UCP) method, beginning with problem identification and a literature review related to software cost estimation techniques. Subsequently, a Use Case Diagram is developed based on observations conducted during the prototype development process to identify system actors and core software functionalities. The results indicate a UCP value of 52,08, corresponding to an estimated development effort of 1.041,6 working hours. Referring to standard working time regulations of 8 hours per day or 40 hours per week, this effort is equivalent to an estimated development duration of approximately 6,5 months. Furthermore, by allocating the estimated effort across development activities and involved roles, the total development cost is estimated at IDR 98.148.855. These findings provide a quantitative and systematic reference for time and cost planning in the development of AIoT based monitoring software within the healthcare domain.

KEYWORDS: Software Cost Estimation, Use Case Point, Software Effort Estimation, Healthcare Information System

I. INTRODUCTION

The development of IoT and AI based infusion drip monitoring systems has been widely explored in recent years. Through a Kumbarani et al. (2025) found that most studies on IoT and AI based infusion monitoring still focus primarily on hardware design, prototype accuracy, and communication modules, resulting in systems that are largely standalone and not integrated with broader clinical systems. This finding aligns with other research emphasizing the importance of continuous infusion monitoring to maintain therapy stability and prevent patient complications, which generally concentrates on the technical aspects of the system and sensors [2], [3], [4]. These studies highlight that monitoring software plays a crucial role in managing the collected data, performing advanced processing, and presenting information through applications or web dashboards to support healthcare professionals in decision making. Therefore, while much(Afiyat et al., 2023; Cahyanurani et al., 2019; Lee et al., 2021) of the development effort for IoT and AI based infusion monitoring systems has focused on hardware and communication modules, the operational success of such systems heavily depends on the quality and design of the software used.

Medical devices continue to evolve as software increasingly supports their functionality and applications. Software failures can have serious consequences, including patient safety risks, emphasizing that software development is a critical aspect and must be treated as a core component of medical device manufacturing (Carroll & Richardson, 2016). Challenges in software development remain high. Portman (2021) presents a visualization from the 2020 CHAOS Report, showing that only 31% of global IT projects are considered successful, while 50% are challenged and 19% fail. Furthermore, among the challenged projects, 52% exceeded their initial budget estimates by up to 189%. Overall, the economic loss due to software project failures is estimated at USD 260 billion (Krasner, 2020). Consequently, the development of reliable software requires the implementation of project management based on the Software Development Life Cycle (SDLC), with an emphasis on accurate effort and cost estimation to minimize the risk of failure (Hariyanto & Wahono, 2015; Prayitno, 2017).

Various cost estimation methods have been widely developed, such as the Function Point Approach (Albrecht, 1979), COCOMO II (Boehm et al., 2022), and Use Case Points (UCP) (Karner, 1993). The UCP method is an early estimation technique based on Use

Case Diagrams, used to identify potential software issues, estimate project size, and outline the general system architecture (Adhitama, 2018). Research by Prayitno (2017) applied UCP to estimate effort and development costs for object oriented information systems, showing an estimation deviation of approximately 17% compared to actual effort. Further, Yahya & Korespondensi (2023) emphasized that functional requirements based cost estimation provides a more realistic picture of resource and budget needs than intuitive approaches. Research by (Permana et al., 2023) specifically applied UCP to estimate the effort and cost of developing business application based inventory software, with a total UCP value of 94, resulting in an estimated effort of 256.93 hours versus an actual effort of 310.5 hours, yielding a deviation of about 17.2%. This result was more accurate than manual expert estimations, which averaged a 20% deviation (Anda, 2000). Meanwhile, Hayuhardhika et al. (2020) compared several UCP variants including UCP, Extended UCP (E-UCP), and Revised UCP (Re-UCP) across conventional information systems such as project management and administrative systems. The study found that UCP and its variants could produce estimations relatively close to actual project costs, although accuracy depends on technical and environmental factor adjustments.

Although previous studies indicate that UCP methods have relatively low deviation rates, no research has yet addressed the application of UCP for estimating the development cost of AIoT based infusion monitoring software prototypes, a medical domain software that requires careful development planning due to its impact on patient safety and healthcare service continuity. Considering the scope limitations, this study focuses on applying the UCP method to estimate software development costs for an AIoT based infusion drip monitoring prototype system. The estimation emphasizes the software component, particularly the web monitoring dashboard, after system requirements have been validated using a prototyping approach. It is expected that this study will provide a quantitative overview of software development cost needs, support more structured healthcare information system project planning, and serve as an initial reference for the development and potential commercialization of AIoT based infusion monitoring systems.

II. RESEARCH METHOD

The cost estimating of development was conducted using the Use Case Point (UCP) method, where the system's functional requirements, actors, and interactions were derived from direct observation of the AIoT-based infusion drip monitoring prototype. The Technical Complexity Factor (TCF) and Environmental Complexity Factor (ECF) were determined based on the researcher's experience during development. These factors, combined with the use case data, were used to calculate the Use Case Points (UCP), forming the basis for estimating the software development effort and cost.

A. Use Case Point Method

The Use Case Points (UCP) method was first introduced by Karner in 1993 as an approach for estimating development effort in software projects. UCP leverages the number and complexity of use cases within a system to produce a development effort estimate. The calculation involves two main components: the system-level Use Case Diagram and the use case scenarios. The system-level diagram presents all actors and use cases, while the scenarios describe the normal execution paths along with all alternative flows for each use case (Kusumoto et al., 2004).

The initial stage of cost estimation using UCP is actor weighting, which is performed by identifying and categorizing the actors in the Use Case Diagram. The details of actor weights according to Wasono et al. (2019) are presented in Table 1.

Table 1. Category Unadjusted Actor Weight (UAW)

Actor	Description	Weight
Simple	Actor representing another system interacting via API	1
Average	Actor representing another system interacting via a protocol, such as TCP/IP	2
Complex	Actor interacting with the system through a GUI	3

And the Unadjusted Actor Weight (UAW) is calculated using Equation:

$$UAW = \sum_{k=1}^{n} BobotAktor_k$$

Next, the Unadjusted Use Case Weight (UUCW) is determined by reviewing each use case and its interactions with the system. Actors are classified into three categories based on interaction complexity, as shown in Table 2.

Table 2. Category Unadjusted Use case Weight (UUCW)

Category	Description	Weight
Simple	Use case with 1 to 3 transactions	5
Average	Use case with 4 to 6 transactions	10
Complex	Use case with 7 or more transactions	15

UUCW represents the overall use case complexity and is obtained by summing the transaction weights for each use case, as expressed in Equation:

$$UUCW = \sum_{k=1}^{n} BobotUseCase_k$$

The Unadjusted Use Case Points (UUCP) are then calculated by summing UAW and UUCW (Equation 3):

UUCP = UAW + UUCW

The Technical Complexity Factor (TCF) reflects the technical aspects of the software, assessed across 13 factors (T1 to T13), including distributed systems, performance, and special security features. Each factor is scored from 0 (no influence) to 5 (strong influence).

Table 3. Technical Complexity Factor (TCF)

Ti	Technical Factor	Weight
T1	Distributed System	2
T2	Response Adjectives	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

The Technical Factor (TF) is first calculated by multiplying each factor score by its weight and summing the results using equation:

$$TF = \sum_{k=1}^{13} Skor \times Bobot_k$$

The TCF is then derived using equation:

TCF = 0.6 + (0.01 × TF)

Similarly, the Environmental Complexity Factor (ECF) captures non-technical or environmental influences, evaluated across eight factors (E1 to E8), such as familiarity with development methods, application experience, and motivation.

Table 4. Environmental Complexity Factor (ECF)

Ti	Environmental Factor	Weight
E1	Familiar with Rational Unified Process	1,5
E2	Application Experience	0,5
E3	Object-oriented Experience	1

E4	Lead Analyst Capability	0,5
E5	Motivation	1
E6	Stable Requirements	2
E7	Part-time Workers	-1
E8	Difficult Programming Language	1

Each factor is rated from 0 to 5. The Environmental Factor (EF) is first calculated *using* equation 6:

$$EF = \sum_{k=1}^{13} Skor \times Bobot_k$$

ECF is then obtained using Equation 7:

ECF = 1.4 + (−0.03 × EF)

The final Use Case Points (UCP) value is calculated by multiplying UUCP, TCF, and ECF (Equation 8):

UCP = UUCP × TCF × ECF

Once UCP is obtained, it is converted into estimated effort in hours (Hours of Effort) by multiplying with the man-hour factor of 20 hours as suggested by Anda (2000):

Effort = UCP × man-hour

The estimated effort is then allocated across three main software development activities: Software Development (approximately 42%), Ongoing Activity (approximately 21%), and Quality and Testing (approximately 37%) (Saleh, 2011). For labor cost estimation, IT salaries were based on the Kelly Services survey (2024), using the lower end of the salary range to reflect small and medium-sized IT companies in Indonesia. The development team consisted of three members with defined roles: project manager, software engineer, and technical lead. Finally, the development cost for each phase was calculated by multiplying the standard hourly wage with the hours of effort for that phase. The total system development cost was obtained by summing the costs across all phases, resulting in the estimated overall development cost.

III. RESULT

A. Use Case Diagram

From the development of the AIoT-based infusion drip monitoring prototype, the final Use Case Diagram was obtained, representing the software's core functionalities, as shown in Figure 2. The diagram includes 12 use cases that illustrate the system's main functions. A total of five actors were identified in the diagram, including two primary actors, Admin and Nurse, who interact directly with the system, and three supporting actors, which are external systems involved in data exchange and provision. This Use Case Diagram serves to define the system boundaries and general interactions, and it forms the basis for estimating the software development cost using the Use Case Point (UCP) method.

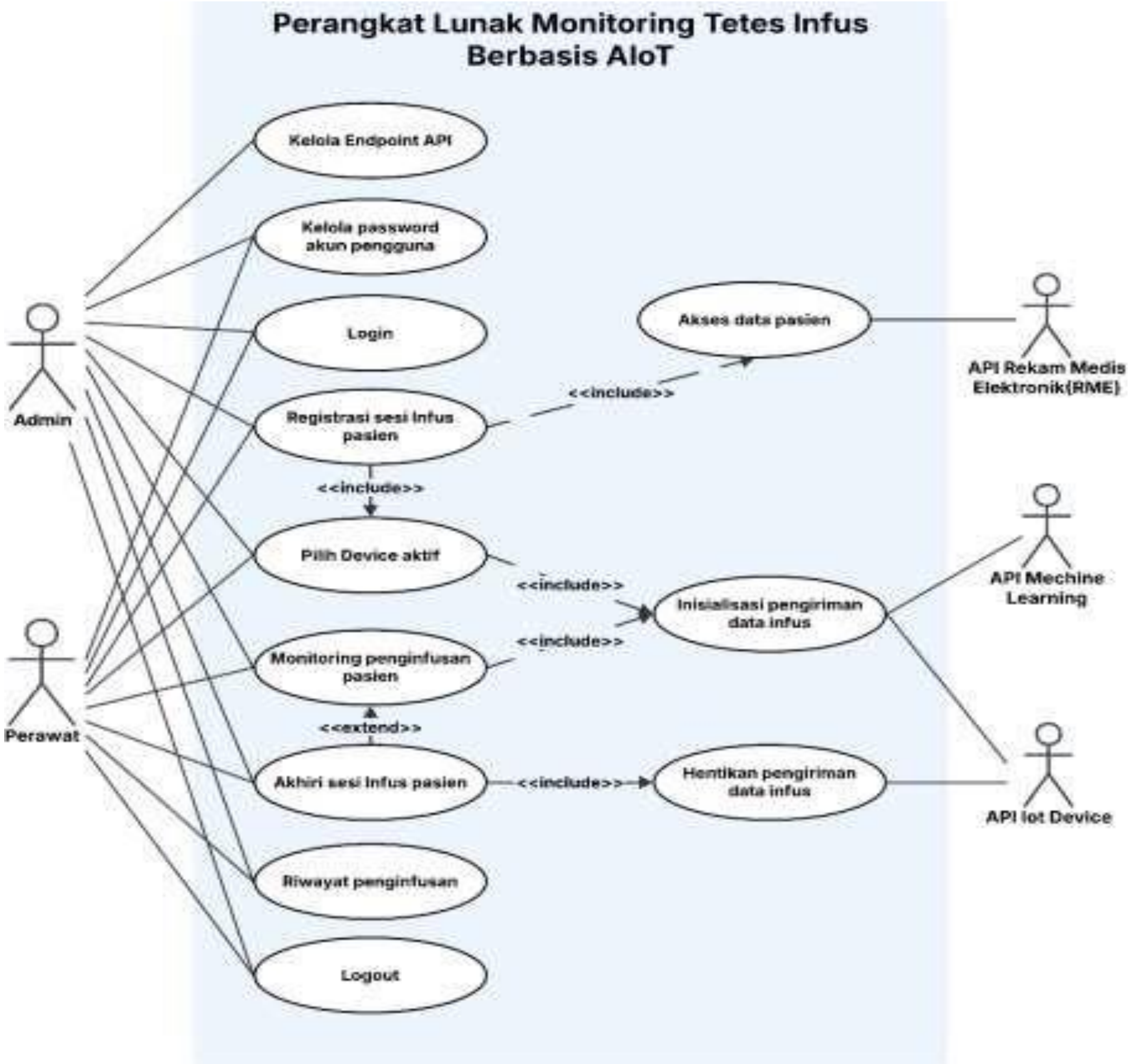

Figure 1. The Use Case Diagram of the AIoT-based infusion drip monitoring software prototype

B. Counting Unadjusted Use Case Point s (UUCP)

To calculate the Unadjusted Use Case Points (UUCP), the first step is to determine the Unadjusted Actor Weight (UAW) for all actors involved in the system. Based on the actor identification derived from the Use Case Diagram of the AIoT-based infusion drip monitoring system, five actors were identified and classified according to the complexity of their interactions with the system. The total UAW value for these five actors is presented in Table 5.

Table 5. Result Unadjusted Actor Weight (UAW)

Category	Weight	Number of Actors	Weight*Number of Actors
Simple	1	3	3
Average	2	0	0
Complex	3	2	6
	Total UAW		**9**

The next step is to calculate the Unadjusted Use Case Weight (UUCW) by grouping the use cases according to their interaction complexity. In the simple category, the use cases include Login with two transactions, Password Management with two transactions, Delete Infusion Session Registration with two transactions, Infusion History with three transactions, Logout with one transaction, and End Infusion Session with four transactions after considering include relationships. Meanwhile, the average category consists of API Endpoint Management with four transactions, Infusion Session Registration with six transactions including include relationships, and Infusion Monitoring with four transactions. This classification reflects the weight of each use case based on the number of transactions and interaction complexity, as structured in the monitoring software Use Case Diagram and detailed in Table 6.

Table 6. Result Unadjusted Use Case Weight (UUCW)

Category	Weight	Number of Use case	Weight*Number of Use case
Simple	5	6	30
Average	10	3	30
Complex	15	0	0
Total UUCW			**60**

Finally, the total UAW and UUCW values are summed, resulting in a value of **69**. This value represents the total unadjusted use case and actor points prior to applying technical and environmental complexity adjustments.

C. Calculating Technical Complexity Factor (TCF)

The Technical Complexity Factor (TCF) was determined by evaluating the system's technical complexity based on a set of relevant criteria. The impact factor for each criterion was derived from the researcher's observations during the development of the AIoT-based infusion drip monitoring software prototype, reflecting the actual conditions and technical challenges encountered throughout system implementation. The assessment results for each criterion, along with their corresponding weights, are presented in Table 7.

Table 7. Result Technical Complexity Factor (TCF)

Ti	Technical Factor	Weight	Impact (0-5)	Weight* Impact
T1	Distributed System	2	5	10
T2	Response Adjectives	1	4	4
T3	End-user Efficiency	1	4	4
T4	Complex Processing	1	5	5
T5	Reusable Code	1	1	1
T6	Easy to Install	0,5	3	1.5
T7	Easy to Use	0,5	4	2
T8	Portable	2	3	6
T9	Easy to Change	1	3	3
T10	Concurrent	1	4	4
T11	Security Features	1	4	4
T12	Access for Third Parties	1	5	5
T13	Special Training Required	1	4	4
Total Factor				**53,5**

Based on the values in Table 7, the resulting TCF is calculated as follows:

*TCF=0.6+0,535=**1,135***

D. Calculating Environmental Complexity Factor (ECF)

The Environmental Complexity Factor (ECF) was determined to account for the influence of environmental conditions on the Use Case Points calculation. The ECF value was obtained by evaluating several relevant environmental criteria encountered during the development of the AIoT-based infusion drip monitoring software prototype. The assessment results for each criterion, along with their respective weights, are presented in Table 8.

Table 8. Environmental Complexity Factor (ECF)

Ti	Environmental Factor	Weight	Impact (0-5)	Weight* Impact
E1	Familiar with Rational Unified Process	1,5	4	6
E2	Application Experience	0,5	1	0.5
E3	Object-oriented Experience	1	2	2
E4	Lead Analyst Capability	0,5	4	2
E5	Motivation	1	5	5
E6	Stable Requirements	2	4	8
E7	Part-time Workers	-1	1	-1
E8	Difficult Programming Language	1	2	2
	Total Factor			**24.5**

Based on the values in Table 8, the ECF is calculated as follows:

ECF=1.4+(−0.03×24.5) = ***0,665***

E. Calculating Use Case Point (UCP)

After obtaining the values of **UUCP**, **TCF**, and **ECF**, the next step was to calculate the **Use Case Points (UCP)**. The UCP value was determined by multiplying these three components, resulting in the following calculation: *UCP = 69×1,135×0,665 =* ***52,08***

F. Calculating Effort

Based on the Use Case Points (UCP) calculation, the system complexity was estimated at 52.08 UCP. To determine the development effort, this value was multiplied by a man-hour factor of 20 hours. The results indicate that the total estimated development effort is **1.041,6** hours.

G. Cost Estimation

After the development effort was obtained and allocated to each development activity, as shown in Table 9, the cost estimation was carried out based on the distribution of effort and the roles involved in each phase. The table presents the activity breakdown, the assigned roles, and the corresponding working time in hours of effort, which serve as the basis for estimating the system development cost.

Table 9. Cost Estimation

Activity Category	Position (Based on Salary Guide)	%Effort	Hours of Effort	Standard Hourly Rate	Total Cost
Software Development		**42,5%**	**442,68**		
Requirement	Project Manager	78,12	32,31	IDR 57.803	IDR 4.515.570
Specification	Project Manager	78,12	32,31	IDR 57.803	IDR 4.515.570
Design	Technical Lead	104,16	43,08	IDR 161.850	IDR 16.858.296
Implementation	Software Engineer	104,16	43,08	IDR 46.243	IDR 4.816.671
Acceptance & Deployment	Software Engineer	78,12	32,31	IDR 46.243	IDR3.612.503
Ongoing Activity		**20,82%**	**216,86**		
Project Management	Project Manager	86,87	35,94	IDR 57.803	IDR5.021.314
Configuration Management	Software Engineer	43,33	17,93	IDR 46.243	IDR2.003.735
Documentation	Software Engineer	43,33	17,93	IDR 46.243	IDR2.003.735
Training & Support	Software Engineer	43,33	17,93	IDR 46.243	IDR2.003.735
Quality & Testing		**36,68%**	**382,06**		
Integration Testing	Technical Lead	78,12	32,31	IDR 161.850	IDR12.643.722
Quality Assurance	Project Manager	86,87	35,94	IDR 57.803	IDR5.021.314
Evaluation & Testing	Technical Lead	217,07	89,76	IDR 161.850	IDR35.132.689
Total		**100%**	**1.041,6 hours**		**IDR98.148.855**

Based on these calculations, the total estimated cost of system development is IDR 98.148.855, representing the projected cost of developing the AIoT-based infusion drip monitoring software when implemented at an industrial scale.

IV. CONCLUSIONS

This study provides an estimation of the development effort and cost of an Artificial Intelligence of Things (AIoT)–based infusion drip monitoring software using the Use Case Points (UCP) method when applied at an industrial scale. Based on the UCP calculation, the total development effort was estimated at 1.041,6 man-hours. Based on Law of the Republic of Indonesia No. 11 of (2020) on Job Creation, the standard working time is defined as 8 hours per day or 40 hours per week, this effort corresponds to approximately 26.04 weeks, or about 6.5 months of development time. Furthermore, by distributing the estimated effort across development activities and the roles involved, the total development cost was estimated at IDR 98.148.855, representing the projected budget required for developing the proposed AIoT-based monitoring system.

Despite these results, this study has several limitations. The use cases used as the basis for the UCP calculation were derived from a software prototype, which may not fully represent a system that has been fully implemented and deployed in a production environment. In addition, the estimated effort and cost were not validated against actual development effort from a real-world implementation, and therefore the accuracy of the estimation cannot yet be empirically confirmed. Moreover, this study employed only the conventional Use Case Points method. Future research is recommended to compare the estimation results with other software cost estimation approaches, such as Function Point Analysis, COCOMO II, and alternative UCP variants including Fuzzy Use Case Points and Activity-Based Costing (ABC), to obtain more comprehensive and accurate estimations. Nevertheless, the findings of this study can serve as an initial quantitative reference for planning development time and costs for AIoT-based software systems in the healthcare domain.

ACKNOWLEDGMENT

The authors would like to express their sincere appreciation to the supervisor for valuable guidance, constructive feedback, and academic support provided throughout the research process.

REFERENCES

1) Adhitama, R. (2018). Effort Estimation Menggunakan Metode Use Case Point untuk Pengembangan Perangkat Lunak. *Journal of Informatics, Information System, Software Engineering and Applications (INISTA)*, *1*(1). https://doi.org/10.20895/inista.v1i1.14
2) Afiyat, N., Navilla, R. H., & Hariyadi, M. (2023). Sistem Monitoring Cairan Infus Berbasis IoT Menggunakan Protokol MQTT. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, *12*(1), 50–55. https://doi.org/10.22146/jnteti.v12i1.5862
3) Albrecht, A. J. (1979). *Measuring Application Development Productivity*. 83–92.
4) Anda, B. (2000). *Comparing Effort Estimates Based on Use Case Points with Expert Estimates*. https://www.researchgate.net/publication/228597738
5) Boehm, B. W., Abts, C., Brown, A. W., Chulani, S., Clark, B. K., Horowitz, E., Madachy, R., Reifer, D. J., & Steece, B. (2022). *Software Cost Estimation with COCOMO II*. Prentice Hall.
6) Cahyanurani, A., Hadiyoso, S., Aulia, S., & Faqih, M. (2019). Design and development of a monitoring and controlling system for multi-intravenous infusion. *Journal of Physics: Conference Series*, *1367*(1), 012075. https://doi.org/10.1088/17426596/1367/1/012075
7) Carroll, N., & Richardson, I. (2016). Software-as-a-Medical Device: demystifying Connected Health regulations. *Journal of Systems and Information Technology*, *18*(2), 186–215. https://doi.org/10.1108/JSIT-07-2015-0061
8) Hariyanto, M., & Wahono, R. S. (2015). Estimasi Proyek Pengembangan Perangkat Lunak dengan Fuzzy Use Case Points. *IlmuKomputer.com Journal of Software Engineering*, *1*(1), 54–56.
9) Hayuhardhika, W., Putra, N., & Perdanakusuma, A. R. (2020). Estimasi Biaya Proyek Perangkat Lunak Menggunakan Use Case-Based Effort Estimation. *TECNOSCIENZA*, *4*(2), 283–300. https://doi.org/https://doi.org/10.51158/8jh3ws13
10) Karner, G. (1993). *Resource Estimation for Objectory Projects*. Objective Systems SF AB.
11) Krasner, H. (2020). *The Cost of Poor Software Quality in the US: A 2020 Report*.
12) Kumbarani, K. T., Arifin, Z. A., Nurhasanah, Y., Agus, N., & Raya, J. (2025). Internet of Things-Based Infusion Monitoring System in Optimizing Nurse Performance: A Literature Review. *JENDELA NURSING JOURNAL*, *9*(1), 29–44. https://doi.org/10.31983/jnj.v9i1.12487

13) Kusumoto, S., Matukawa, F., Inoue, K., Hanabusa, S., & Maegawa, Y. (2004). Estimating effort by use case points: method, tool and case study. *10th International Symposium on Software Metrics, 2004. Proceedings.*, 292–299. https://doi.org/10.1109/METRIC.2004.1357913
14) Lee, J. K., Yoon, K. C., & Kim, K. G. (2021). Design of a remote monitoring system based on optical sensors to prevent medical accidents during fluid treatment. *Applied Sciences (Switzerland)*, *11*(21). https://doi.org/10.3390/app112110124
15) Permana, R., Abdilah, A., Hasan, F. N., & Syarif, M. (2023). Estimation Effort Pengembangan Software Inventory PT. Infinity Global Mandiri Menggunakan Metode Use Case Point. *Jurnal RESTIKOM : Riset Teknik Informatika dan Komputer*, *5*(2), 73–84. https://doi.org/10.52005/restikom.v5i2.144
16) PERSOLKELLY Indonesia. (2024).*Indonesia Salary Guide 2024*. https://player.flipsnack.com/?hash=NTdFRjk1NzdDNkYraHAzZXVoNzJneA==
17) Portman, H. (2021). *Project Success Quick Reference Card*. https://hennyportman.wordpress.com/2021/01/25/projectsuccess-quick-reference-card/
18) Prayitno, E. (2017). Penggunaan Metode Estimasi Use Case Points (UCP) Dalam Proyek Software Domain Bisnis. *JURNAL INFORMATIKA*, *4*(2).
19) Saleh, K. (2011). Effort and Cost Allocation in Medium to Large Software Development Projects. *INTERNATIONAL JOURNAL OF COMPUTERS*, *5*(1). files/202/Saleh - 2011 - Effort and Cost Allocation in Medium to Large Soft.pdf
20) Undang-Undang Republik Indonesia Nomor 11 Tahun 2020 Tentang Cipta Kerja, Pub. L. No. 11 (2020).
21) Wasono, B., Putra, W. H. N., & Rachmadi, A. (2019). Estimasi Biaya Pembangunan Perangkat Lunak Menggunakan Metode Use Case Point (Studi Kasus CV. Utomo Sakti (Cloud Astro Technology)). *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *3*(8), 7673–7680. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/6042
22) Yahya, M. T., & Korespondensi, P. (2023). *ESTIMASI USAHA DAN BIAYA PROYEK PENGEMBANGAN APLIKASI WEB DI CV. TECHNOBIT INDONESIA MENGGUNAKAN METODE USE CASE POINT* (Vol. 4, Issue 2).