Journal of Applied Informatics for Sustainability and Artificial Intelligence
AISAI Journal
https://jurnal.polbeng.ac.id/index.php/AISAI

Volume 1 Issue 1 Year 2026 | Page 48-54 | e-ISSN: xxxx-xxxx | ISSN: xxxx-xxxx
Received: 20-01-2026 | Revised: 30-01-2026 | Accepted: 10-02-2026

Software Project Cost Estimation System Based on the COCOMO II Method

Recki Walestra [1], Fajar Ratnawati [2]
[1,2] Politeknik Negeri Bengkalis, Bengkalis, Riau, Indonesia, 28711
E-mail: *reckiwalestra@gmail.com*[1] , *fajar@polbeng.ac.id*[2]
*Correspondence: *reckiwalestra@gmail.com*

Abstrak: *A project is considered successful if the system can be delivered on time, according to the desired cost and quality. Even if the target set by the project manager is reasonable but does not take into account the team's productivity record, there is a possibility that the deadline will not be met due to incorrect initial estimates. Therefore, cost estimation in software development is considered very important for a project manager. Therefore, this study was conducted with the aim of mitigating the problems that arise by creating a software project cost estimation application to help project managers calculate the estimated costs to be used using the COCOMO II method in developing an application.*

Keywords: *COCOMO II, Cost Estimation, Software Project.*

1. Introduction

The rapid advancement of information technology requires organizations to continuously adapt to increasingly complex software requirements. However, real-world evidence indicates that not all software projects proceed as planned. According to the Standish Group CHAOS Report 2020, only approximately 31% of software projects are successfully completed on time, within budget, and in accordance with specifications. Meanwhile, about 50% experience schedule delays or budget overruns, and the remaining 19% are cancelled [1]. This situation highlights that one of the main challenges in software development lies in the inaccuracy of cost and time estimation. Recent studies also indicate that the accuracy of classical models such as COCOMO remains challenging when applied to modern software projects [8].

Many organizations still rely on manual methods, such as spreadsheets, to calculate cost estimates. While this approach is relatively easy to implement, it has several drawbacks, including a high risk of calculation errors, limited flexibility in accommodating changing requirements, and an inability to provide comprehensive predictions [2]. Consequently, project managers often face significant risks of cost overruns and schedule delays.

Several previous studies have attempted to address these issues. Maqdum *et al.* [3] applied the COCOMO II method to a software project at CV Profile Image Studio and demonstrated improved estimation accuracy. Harsono *et al.* [5] developed a COCOMO II–based estimation system to support government agencies in determining the Owner's Estimate (HPS) for software projects. Mariana and Adiyanto [4] designed a web-based application for coachwork cost estimation, replacing the previously manual Excel-based approach. Tursono [6] combined the Function Point method with COCOMO II to enhance estimation accuracy. Furthermore, Ariyanto *et al.* [7], through a Systematic Literature Review, found that integrating COCOMO II with artificial intelligence techniques, such as Genetic Algorithms, can produce more accurate

estimations. Other studies also report that integrating COCOMO II with Artificial Neural Networks (ANN) and other intelligent estimation approaches yields more adaptive and reliable results [9], [10].

Based on the reviewed literature, it can be concluded that the COCOMO II method has proven effective; however, its application remains limited to specific case studies and is not yet widely available in the form of a web-based application that can be directly utilized by project managers [11]. Therefore, this study proposes the development of a web-based software cost estimation application using the COCOMO II method. The objective is to provide an accurate, efficient, and standardized solution for estimating project cost, development time, and resource requirements.

2. Research Methods

2.1 COCOMO II

COCOMO (Constructive Cost Model) II is an algorithmic cost estimation model that supports individuals or teams in estimating project cost, effort, and schedule during software development planning activities. COCOMO II is an evolution of COCOMO 81, originally introduced by Boehm in 1981 and later refined and published in 1997 [6]

2.2 COCOMO II Implementation Stages

In the COCOMO II model, effort is expressed in person-months. The following steps describe the procedure for estimating effort using the COCOMO II method:

Determine the Total Unadjusted Function Points (TUFP) Function Points are used to address limitations associated with using lines of code (LOC) as a software size measure and to support the development of effort prediction mechanisms can be seen in the table 1.

Table 1. TUFP Calculation

Description	*Total Number*	*Low*	*Medium*	*High*	*Total*
Inputs	_	_x3	_x4	_x6	_
Outputs	_	_x4	_x5	_x7	_
Queries	_	_x3	_x4	_x6	_
Files	_	_x7	_x10	_x15	_
Program Interface	_	_x5	_x7	_x10	_
Total Unadjusted Funcion Point					_

Each Function Point component is classified according to its level of complexity. The assigned complexity level determines the corresponding quantity of Unadjusted Function Points (UFP).

2.3 Adjusted Processing Complexity

After obtaining the total Unadjusted Function Points (UFP), the next step is to determine the system's complexity level. This is done by selecting the Project Complexity Adjustment (PCA) factor from the following standard range:

- 0.65 for Simple Systems
- 1.0 for Normal Systems
- 1.35 for Complex Systems

2.4 Processing Complexity (PC)

Once the Total Unadjusted Function Points (TUFP) have been obtained, the Processing Complexity (PC) value is calculated. By combining the TUFP and PC values, the Total Adjusted Function Points (TAFP) are derived. After determining the TAFP, the value is then converted into

Lines of Code (LOC). Table X presents the LOC conversion based on the Quantitative Software Management (QSM) standard can be seen in the table 2.

Table 2. Processing Complexity Values

Description	Scale (0–3)
Data communication within the system process	_
System configuration	_
Presence of transactions in the system	_
User efficiency in interacting with the system	_
Existence of complex processing in the system	_
Ease of system installation	_
System connectivity with external sites	_
System performance	_
Online data entry capability	_
Online system updates	_
System reusability	_
Operational complexity	_
System extensibility potential	_
Total Processing Complexity (PC)	_

After converting the Total Unadjusted Function Points (TUFP) and Total Processing Complexity (PC) into Lines of Code (LOC)**,** the KLOC value is obtained.

Effort Estimation

$$Effort = 1.4 \times KLOC \quad (1)$$

Time Estimation

$$\text{Time} = 3.0 \times (\text{Person-Months})^{1/3} \quad (2)$$

4. Cost Estimation

To estimate the project cost, the calculation considers the project duration and the required number of personnel, multiplied by the software labor rate. For example, the labor rate can be based on the average wage in Bengkalis City.

$$Cost = Person\text{-}Months \times Labor \quad (3)$$

where Labor Rate refers to the software workforce tariff (e.g., average local wage), and Cost represents the total estimated development cost can be seen in the table 3 and 4.

Table 3. Function Point Components

Component	**Description**
External Input (EI)	A function that transfers data from outside the application into the system without presenting data manipulation.
External Output (EO)	A function that transfers data from within the system to the user and presents output where some data have been processed or manipulated.
External Inquiry (EQ)	Provides information to users through data retrieval/processing or control information derived from ILF/EIF.
Internal Logical File (ILF)	A logically related group of persistent data maintained by the application, managed through external inputs.
External Interface File (EIF)	A logically related group of persistent data used by the application but maintained outside the application boundary.

Table 4. Quantitative Software Management QMS Conversion Standards

Language	QSM SLOC/FP Data			
	Avg	*Medium*	*Low*	*Hight*
ABAP (SAP) *	28	18	16	60
ASP*	51	54	15	69
Assembler *	119	98	25	320
Brio +	14	14	13	16
C *	97	99	39	333
C++ *	50	53	25	80
C# *	54	59	29	70
COBOL *	61	55	23	297
Cognos Impromptu Scripts +	47	42	30	100
Cross System Products (CSP) +	20	18	10	38
Cool:Gen/IEF *	32	24	10	82
Datastage	71	65	31	157
Excel *	209	191	131	315
Focus *	43	45	45	45
FoxPro	36	35	34	38
HTML *	34	40	14	48
J2EE *	46	49	15	67
Java *	53	53	14	134
JavaScript *	47	53	31	63
JCL *	62	48	25	221
LINC II	29	30	22	38
Lotus Notes *	23	21	19	40
Natural *	40	34	34	53
.NET *	57	60	53	60
Oracle *	37	40	17	60
PACBASE *	35	32	22	60
Perl *	24	15	15	60
PL/I *	64	80	16	80
PL/SQL *	37	35	13	60
Powerbuilder *	26	28	7	40
REXX *	77	80	50	80
Sabretalk *	70	66	45	109
SAS *	38	37	22	55
Siebel *	59	60	51	60
SLOGAN *	75	75	74	75
SQL *	21	21	13	37
VB.NET *	52	60	26	60
Visual Basic *	42	44	20	60

3. Results and Discussion

The website used as a case study for cost estimation is Seeday, a JavaScript-based web application obtained from GitHub can be seen in the figure 1.

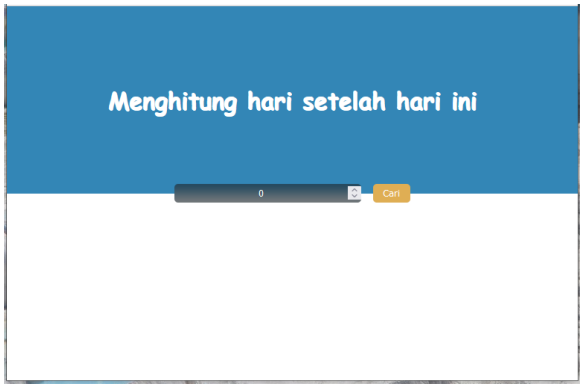

Figure 1. Seeday Website

(Source: https://github.com/devnazir/seeday)

3.1 Determining the Total Unadjusted Function Points (TUFP)

The analysis of the Seeday application, used as an example system, is conducted by identifying the number of functions and the level of complexity implemented by the developer. The functions can be categorized into five components, as Function Point counting is based on functional elements can be seen in the table 5.

Table 5. TUFP Values

Complexity Description	Total Number	Low	Medium	High	Total
Inputs	1	1×3	0×4	0×6	3
Outputs	1	0×4	1×5	0×7	5
Queries	1	0×3	1×4	0×6	4
Files	1	0×7	1×10	0×15	10
Program Interface	0	0×5	0×7	0×10	0
Total Unadjusted Function Points (Tufp)					**22**

3.2 System Processing Complexity (PC)

In this stage, the researcher conducts an interview with the project stakeholder. The questions may be left unanswered if the stakeholder does not wish to make assumptions can be seen in the table 6.

Table 6. Total Processing Complexity (PC) Table

Description	Scale (0–3)
Data communication within the system process	_
System configuration	_
Presence of transactions in the system	_
User efficiency in interacting with the system	1
Existence of complex processing in the system	_
Ease of system installation	1
System connectivity with external sites	_
System performance	1
Online data entry capability	1
Online system updates	_
System reusability	1
Operational complexity	_
System extensibility potential	1
Total Processing Complexity (PC)	**6**

After obtaining the total Processing Complexity (PC) value, the following calculations are performed:

1. Processing Complexity (PC)

$$PC = 6$$

2. Adjusted Processing Complexity (PCA)

   The Processing Complexity result is normalized as follows:

$$PCA = 0.65 + (0.01 \times PC)$$

$$PCA = 0.65 + (0.01 \times 6) = 0.71$$

3. Total Adjusted Function Points (TAFP)

$$TAFP = TUFP \times PCAT$$

$$TAFP = 22 \times 0.71 = 15.62$$

4. Converting TAFP to Lines of Code (LOC/SLOC) After obtaining the Total Adjusted Function Points (TAFP), the value is converted to Source Lines of Code (SLOC) as follows:

$$LOC = TAFP \times \left(\frac{LOC}{FP}\right)$$

$$SLOC = 15.62 \times 53 = 827.86 \text{ LOC}$$

In this calculation, the factor 53 is taken from the QSM conversion standard because the application is developed using JavaScript.

5. EffortEstimation

   Effort is estimated using the COCOMO-based formulation in person-months:

$$Effort = 1.4 \times \frac{LOC}{1000}$$

$$Effort = 1.4 \times \frac{827.86}{1000} = 1.16 \text{ person-months}$$

6. Time Estimation

   The project duration is estimated (in months) as:

$$Time = 3.0 \times (Person\text{-}Months)^{1/3}$$

$$Time = 3.0 \times (1.16)^{1/3} = 3.15 \text{ months}$$

7. Cost Estimation

   Cost estimation is calculated based on the required effort and a standard labor rate for software development. As an example, the labor rate can be set using the average programmer salary in Indonesia.

$$Costmonthly = Person - Months \times Labor\ RateCost_{monthly}$$

$$Cost_{monthly} = 1.16 \times 5{,}200{,}000 = 6{,}032{,}000 \text{ IDR/month} Costmonthly$$

$$Cost_{total} = Cost_{monthly} \times Time$$

$$Cost_{total} = 6{,}032{,}000 \times 3.15 = 19{,}000{,}800 \text{ IDR}$$

Based on the above procedure, the estimated cost required to design and develop the Seeday website is IDR 19,000,800, assuming an average programmer salary in Indonesia of IDR 5,200,000 and the complexity values described in the computation steps.

4. Conclusions

This study developed a COCOMO II–based software project cost estimation application that supports project managers in computing cost requirements more accurately. The application was evaluated using a case study of the Seeday website, implemented in JavaScript. The results indicate that the project has a Total Adjusted Function Points (TAFP) value of 15.62, approximately 827.86 lines of source code, an estimated effort of 1.16 person-months, an estimated development time of 3.15 months, and a total estimated cost of IDR 19,000,800, assuming an average programmer salary in Indonesia of IDR 5,200,000 per month. Overall, the proposed application demonstrates that the COCOMO II method can provide a realistic cost estimation based on the project's complexity level and productivity characteristics.

References

[1] B. B. Tursono and A. S. M. Lumenta, "Analisis Estimasi Proyek Perangkat Lunak," J. Teknik Informatika, pp. 1–8, 2013. [Online]. Available: https://repo.unsrat.ac.id/3596/.

[2] S. G. Powell, B. Lawson, and K. R. Baker, "Impact of Errors in Operational Spreadsheets," arXiv preprint arXiv:0801.0715, 2008. [Online]. Available: https://arxiv.org/abs/0801.0715

[3] A. N. Maqdum, A. R. Perdanakusuma, W. Hayuhardhika, and N. Putra, "Implementasi Metode COCOMO II untuk Estimasi Biaya Pengembangan Perangkat Lunak di CV. Profile Image Studio," J. Pengemb. Teknol. Inf. Ilmu Komputer, vol. 3, no. 6, pp. 6238–6247, 2019. [Online]. Available: http://j-ptiik.ub.ac.id

[4] A. R. Mariana, E. Adiyanto, and S. A. Sakti, "Aplikasi Estimasi Biaya pada Karoseri Berbasis Web di PT Mitra Toyotaka Indonesia," J. Tren Bisnis Global, vol. 1, no. 1, 2021, doi:10.38101/jtbg.v1i1.354.

[5] E. F. Harsono, A. Sukmaaji, A. P. Widodo, and S. Sholiq, "Sistem Aplikasi Penentu Estimasi Biaya Proyek Pengembangan Perangkat Lunak Menggunakan Metode COCOMO II," J. Sistem Inform. dan Komputer (JSIKA), vol. 9, no. 1, pp. 1–8, 2020. [Online]. Available: https://media.neliti.com/media/publications/439427-none-a63cc746.pdf

[6] N. Merlo, "COCOMO (Constructive Cost Model)," *Seminar on Cost Estimation WS 02/03*, Zurich: Requirement Engineering Research Group, University of Zurich, 2002, p. 8.

[7] D. Pitaloka, A. R. Irawati, and Y. T. Utami, "Perkiraan Sumber Daya Pengembangan Sistem Informasi Menggunakan COCOMO II," *Jurnal KLIK*, vol. 8, no. 2, pp. 139–151, 2021. [Online]. Available: http://klik.ulm.ac.id/index.php/klik/article/view/378

[8] U. Jeklin, M. I. Saad, and H. Ekawati, "Evaluation Of COCOMO Model Accuracy In Software Effort Estimation," *BIT*, vol. 6, no. 2, 2025, doi:10.47065/bit.v6i2.2027.

[9] K. K. T. M. and Y. K. K. P., "Software Effort Estimation for COCOMO-II Projects Using Artificial Neural Network," *Int. J. Res. Sci. Innov.*, vol. 7, no. 6, pp. 129–132, Jun. 2020. [Online]. Available: https://www.rsisinternational.org/journals/ijrsi/digital-library/volume-7-issue-6/129-132.pdf

[10] M. S. Rekha and B. M. Malakreddy, "Need for Intelligent Software Cost Estimation and its Methods: A Comprehensive Overview," *Int. J. Intell. Syst. Appl. Eng.*, vol. 12, no. 19s, pp. 43–59, 2024. [Online]. Available: https://ijisae.org/index.php/IJISAE/article/view/5043

[11] Tedyyana, A., Ahmad, A. A., Idrus, M. R., Mohd Shabli, A. H., Abu Seman, M. A., Ghazali, O., & Abd Razak, A. H. (2024). Enhance Telecommunication Security Through the Integration of Support Vector Machines. *International Journal of Advanced Computer Science & Applications*, *15*(3).