Ingénierie des Systèmes d'Information
Vol. 29, No. 4, August, 2024, pp. 1453-1460
Journal homepage: http://iieta.org/journals/isi

Software Quality Measurement Analysis on Academic Information Systems

Dedi Setiadi[1*], Tata Sumitra[1], Ahmad Karim[2], Ritzkal[3]

[1] Informatics Management, Universitas Dirgantara Marsekal Suryadarma, Jakarta 13610, Indonesia
[2] Management, Universitas Pembinaan Masyarakat Indonesia, Medan 20214, Indonesia
[3] Informatics Engineering, Universitas Ibn Khaldun, Bogor 16161, Indonesia

Corresponding Author Email: dedi@unsurya.ac.id

Copyright: ©2024 The authors. This article is published by IIETA and is licensed under the CC BY 4.0 license (http://creativecommons.org/licenses/by/4.0/).

https://doi.org/10.18280/isi.290418

Received: 13 July 2024
Revised: 26 July 2024
Accepted: 2 August 2024
Available online: 21 August 2024

Keywords:
software measurement, software quality, software metrics, software indicators, academic information system, quality management

ABSTRACT

Measurement, which can be done directly or indirectly, is the process of providing a quantitative indication of the scope, quantity, dimension, capacity, or characteristics of a process or product. Software may be measured directly in a number of ways, such as cost, effort, amount of code lines, functionality, speed of execution, memory size, documentation, number of inputs and outputs, and flaws in individual units. In contrast, features like functionality, efficiency, dependability, and maintainability are measured by indirect metrics like software quality. A metric is a quantitative measure of the degree of an attribute of an object, system, or process that is produced by the gathering of data for measurement. These metrics need to be gathered and converted into indicators in order to assess the quality of the program. Metrics, or sets of metrics, known as indicators, give management thorough information about a product and aid in process and product control. A system is made up of several interrelated parts that work together to accomplish a certain objective. These systems are broken down into more manageable subsystems that assist the main system. This research seeks to examine how software quality is measured on Marshal Suryadarma Aerospace University's Academic Information System. In addition to helping university administration regulate and enhance the information systems in use, this research is anticipated to offer a thorough grasp of the application of metrics and indicators in assessing and enhancing the caliber of academic software.

1. INTRODUCTION

Measurement quantifies a process or product's extent, amount, size, capacity, or other characteristics. There are two types of measurement: direct and indirect [1, 2]. Direct measurement in software refers to tangible aspects such as expenses, labor, number of functions, memory size, documentation, inputs, outputs, lines of code, execution speed, and defects. Indirect measurement, such as assessing software quality, includes attributes like functionality, efficiency, reliability, and maintainability [3-5]. The measurement process begins with data collection, resulting in metrics-quantitative measures of an attribute's degree for an item, system, or process [6-8]. To evaluate software quality, these metrics must be transformed into indicators. An indicator is a statistic or a combination of metrics that provides comprehensive information about a product, assisting management in controlling the process and the resulting goods [9-11].

System is a collection of two or more components that constantly communicate and work together to achieve a specific goal [12]. Systems typically function as smaller subsystems that support larger, more complex systems [13, 14]. Effective systems require organization, structure, integration, and clear objectives [15, 16]. Thus, the author feels obligated to prepare a study under the heading "Analysis of Quality Improvement of Software Systems in Academic Information at Marsekal Suryadarma University."

While the methods for measuring software quality are extensively described [17-19], there is a need for detailed case studies to illustrate the practical application of these techniques. This research aims to bridge this gap by demonstrating the effectiveness of these measurement methods through case studies, comparing their application across different types of information systems within the academic context. By doing so, it will provide a deeper understanding of how these metrics and indicators can be used to improve software quality and assist university management in refining their information systems.

2. RESEARCH METHODS

The research method consists of several stages: data collection, numeric computation (metrics), and numeric computation (indicators) [20-22]. The results include data collection on software, numerical computation calculations on metrics, and numerical computation calculations with ISO 9126 quality indicators, which cover functional quality indicators, reliability quality indicators, usability quality indicators, efficiency quality indicators, maintainability quality indicators, and portability quality indicators. Data

collection involves gathering information from various sources, including academic databases, university records, and user feedback. Ensuring data quality is paramount; thus, we implement data validation techniques and cross-referencing with multiple sources to confirm accuracy.

For numeric computation of metrics, we use specific software tools such as MATLAB and Python for statistical analysis and computation [23, 24]. Metrics are derived from raw data and processed to quantify various attributes of the software. For instance, the number of lines of code is measured using code analysis tools like SonarQube, while execution speed and memory usage are assessed using performance profiling tools such as JProfiler and VisualVM [25, 26]. To compute the quality indicators according to the ISO 9126 standard, we use specialized software tools like Quality Modeler (QMOOD) and software quality assessment frameworks. These tools facilitate the evaluation of functional, reliability, usability, efficiency, maintainability, and portability quality indicators [27, 28]. Each indicator is calculated based on predefined metrics, and the results are analyzed to provide comprehensive insights into the software's quality [29, 30]. By elaborating on the software tools used, sources of data, and methods for ensuring data quality, this research aims to enhance the transparency and reproducibility of the study. This detailed approach allows for a more precise analysis of the software quality in the Academic Information System at Marsekal Suryadarma University (Figure 1).

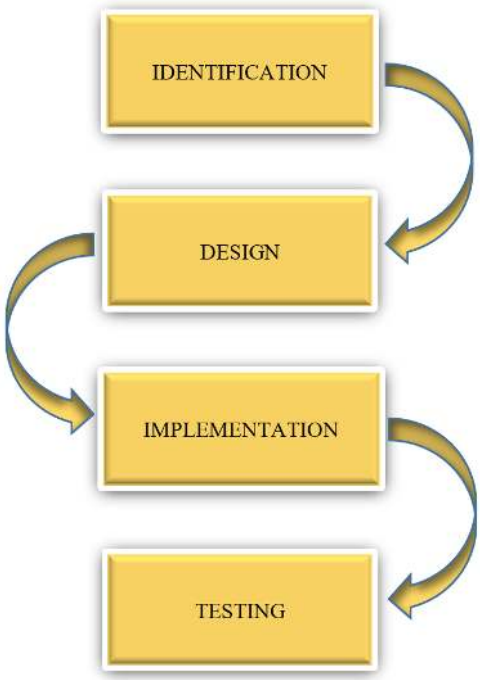

Figure 1. Research methods

2.1 Identification

Descriptive research follows much the same methods as other types of research procedures. Variations may arise, particularly when employing distinct analytical tools, such as distinguishing between qualitative and quantitative research in the given study. Generally speaking, though, descriptive research follows these steps:

a) Choose a noteworthy issue that will be resolved using descriptive techniques.

b) Define the issue precisely and set a limit.

c) Establish the goals and advantages of the study.

d) Look into relevant literature about the issue.

g) Establish the framework, as well as the research questions and/or hypotheses.

f) Creating the study methodology, which entails selecting the population, sample, sampling strategies, data gathering tools, and data analysis.

g) Gather information, arrange it, and apply pertinent statistical methods to evaluate it.

2.2 Design

The subjects in this study were a sample of Suryadarma University Faculty of Industrial Technology (FTI) students who had progressed through the even semester of the 2020-2021 academic year. Specifically, the sample included students from semesters 2, 4, 6, and 8. These students had experienced a shift to distance learning due to the COVID-19 pandemic, which began about a year prior to the study. This transition significantly altered the educational environment, as students and faculty adapted to new methods of interaction and instruction. The focus of this research is on the management of Distance Learning (PJJ) between students and lecturers within the FTI, with particular attention to the changes and challenges encountered post-pandemic. The study aims to assess both the direct and indirect impacts of this shift, including how well the new online systems facilitated communication, engagement, and overall educational effectiveness. By examining these elements, the research seeks to provide a comprehensive understanding of the effectiveness and efficiency of distance learning practices and their implications for future educational strategies in the context of ongoing and post-pandemic conditions.

2.3 Implementation

The techniques used to collect data in this study are as follows:

a. Prior observation

Here the researcher makes indirect observations through an online questionnaire (google form) regarding the impact of covid 19 on PJJ management in order to find out whether there is an impact directly or indirectly experienced by students.

b. Interview

Conducting interviews with the academic department to find out which students are active during PJJ. In this unstructured interview, the author can have an initial picture of PJJ management so far in his element.

c. Questionnaire

Questionnaires are often also known as questionnaires, basically a questionnaire is a list of questions that must be filled in by the person to be measured (respondent). With this questionnaire, people can be known about their circumstances / personal data, experiences, knowledge attitudes or opinions, and others. In terms of who answers, there are direct and indirect (closed) questionnaires. In this study, researchers will use a direct closed questionnaire, which consists of questions with a number of answers as options, in other words, the person subjected to the questionnaire must have the answers provided in the questionnaire. Regarding the form of questionnaire used is a multiple choice system. The reasons the author uses a questionnaire are: a) Save energy. b) Save costs because it does not require much equipment. c) Save time, meaning that with a short time you can get a lot of data.

2.4 Testing

Testing involves conducting observations, numerical computations, and calculating ISO 9126 quality indicators to determine the quality of software products. The ISO 9126 standard provides a comprehensive framework for evaluating software quality, encompassing six key quality characteristics:

1. Functionality: Measures how well the software performs its intended functions.
2. Reliability: Assesses the software's ability to maintain performance under specified conditions.
3. Usability: Evaluates how easily users can learn and use the software.
4. Efficiency: Examines the software's performance in terms of resource usage.
5. Maintainability: Determines how easily the software can be modified for corrections, improvements, or adaptation to changes.
6. Portability: Assesses the software's ability to be transferred from one environment to another.

Each of these characteristics is evaluated using specific metrics. For example, functionality is assessed through correctness, reliability through fault tolerance, usability through learnability, efficiency through time behavior, maintainability through analyzability, and portability through adaptability. These metrics are processed into indicators that provide comprehensive insights into the software's quality, helping management in the control and improvement of the system.

3. RESULT

A globally accepted benchmark for software quality is ISO 9126. Software product quality is defined by ISO 9126, along with models, quality attributes, and related metrics that are used to assess and determine software product quality. Furthermore, management must also adhere to ISO standards. An ISO standard certificate cannot be issued for the job if the management does not adhere to the ISO standards. Six quality criteria are included in the ISO 9126 list of quality factors.

Software testing is one way to evaluate the quality of software, along with other metrics and techniques. ISO 9126 is a software quality benchmark developed by the International Electrotechnical Commission (IEC) and the International Organization for Standardization (ISO).

A globally accepted benchmark for software quality is ISO 9126. Software product quality is defined by ISO 9126, along with related metrics, models, and quality attributes that are used to assess and determine a software product's quality. Furthermore, management must also adhere to ISO standards. An ISO standard certificate cannot be issued for the job if the management does not adhere to the ISO standards. Six quality characteristics are included in the ISO 9126 list of quality criteria. They are as follows:

Usability. The capacity of software to fulfil user requirements and provide user satisfaction.

Dependability. The software's capacity to sustain a particular degree of functionality (e.g., precision, consistency, ease of use, and fault tolerance).

Practicality. Software should be easy to use, learn, understand, and appealing to users.

Effectiveness. The software's capacity to provide acceptable performance in relation to the quantity of resources consumed in the given situation (e.g., storage efficiency).

Reliability. software capacity for modification. Corrections, enhancements, or adjustments made in response to alterations in the environment, specifications, and functional needs are examples of modifications (ex: consistency).

Mobility. The software's capacity to change and adapt to different environments (e.g., self-documentation, organization) or specific uses.

The ISO 9126 model divides each software quality characteristic into many qualities sub-characteristics, which include:

(1) The functionality of ISO 9126.
(2) Dependability (ISO 9126).
(3) Usability (ISO 9126).
(4) The portability of ISO 9126.

3.1 Observation results

Based on the observations, the data as shown in Table 1 is obtained.

3.2 Numeric computation calculation

The numeric computation of the metric is that the function-oriented software metric is derived based on a functionality measurement delivered by the application as a normalized value. Since functionality cannot be measured directly, it must be derived indirectly from other direct measurements. The function-oriented metric created by Alan J. Albrecht (1979) is called a function point. Currently, there are many variations on how to calculate function points after this method was developed and revised by the International Function Point User Group (IFPUG) since 1986. However, in this research, the author will focus on using the function point created by Albrecth. Function points are derived using an empirical relationship based on direct measurement of the software's computable information domain and estimated software complexity. Function points are calculated using a rating scale as shown in Table 2.

3.3 Calculation of ISO 9126 quality indicators

After the data is collected, the next step is to look for ISO 9126 quality indicators, namely functionality, reliability, usability, efficiency, maintainability, and portability.

3.3.1 Functionality

Functionality indicators can be derived from function points. The function point calculation requires data in the form of user input, user output, user requests, files, and external interfaces. Each of these data must be assessed for complexity in general, namely simple, medium or complex. With this data and assessment, the function point calculation is as shown in Table 3.

3.3.2 Complexity modifier

The complexity modifier factor is obtained from the assessment of the fourteen attributes contained in the software. The fourteen attributes are used as factors to normalize the function point calculation. The calculation of the function point complexity variable factor is as shown in Table 4.

Table 1. Observation results

Data Name	Item	Total
User Input	Student data entry KRS entry Lecturer entry Class entry Lecture schedule entry Grade entry Financial data entry Class schedule	8
User Output	Semester grades Grade transcript EDOM assessment Test schedule Final exam schedule Lecture teaching material Student finance Consultation with PA soden	8
User Request	Consultation with KaProdi Online survey Check bank bills Change password Tool menu Information board menu Log out button Profile button Help desk menu	10
File	Course material data Lecture schedule User data Lecturer data Student data Student financial data	6
External Interface	TCP/IP	1
Current Program Module	SKS program SKS plus Active bank billing	3
Replaced Module	Application migration	1
Deleted Module		0
Programming Language	PHP Delphi	2
Failure Data Name User Input	Patch update Student data search Print learning results Lecturer data search Course change	5

Table 2. Numeric computation and functional indicators

Param. Measurement	Total		Simple	Average	Complex	Total
J. User Input	8	X	3			24
J. User Output	8	X	4			32
J. User Inquiry	10	X		5		50
J. Files	6	X		10		60
J. External Interface	1	X	6			6
			Total			172

Unknown: Total=172

$$\sum Fi = 30 \quad (1)$$

The formula for finding function points is as follows:

$$\text{FP=Total Number} \times (0{,}65 + 0{,}01 \sum Fi) \quad (2)$$

$$FP = 172 \times (0.65 + (0.01 \times 30))$$
$$FP = 163.4$$
$$FPMax = 1.35 \times Total$$
$$FPMax = 1.35 \times 163.4$$
$$FPMax = 220.59$$

where, FP = Function Point; FPMax = Function Point Maximum; Total Sum = Total value of the information domain; ∑Fi = Sum of complexity adjustment prices.

Table 3. ISO 9126- Functionality

SUB – Characteristics Steak	Description
Suitability	The ability of the software to provide a set of functions that appropriate for specific tasks and user goals.
Accuracy	Software capabilities in providing precise results and correct according to the needs.
Security	Software capabilities to prevent unwanted access, facing intruders (hacker) as well as authorization in the modification of data.
Interopabillity	The ability of the software to interact with one or more specific system.
Compilance	Software capabilities in meeting standards and needs in accordance with applicable regulations.

Table 4. Complexity modifier

No.	Variable Factors	Nilai
1	Backup and recovery	0 1 2 3 4 5
2	Data communication	0 1 2 3 4 5
3	Data processing distribution	0 1 2 3 4 5
4	Performance	0 1 2 3 4 5
5	Operational configuration	0 1 2 3 4 5
6	On-line entry	0 1 2 3 4 5
7	On-line data change (on-line update)	0 1 2 3 4 5
8	Data transaction rate	0 1 2 3 4 5
9	User efficiency	0 1 2 3 4 5
10	Processing complexity	0 1 2 3 4 5
11	Processing complexity	0 1 2 3 4 5
12	Conversion and installation	0 1 2 3 4 5
13	Doubling of installation	0 1 2 3 4 5
14	Facilitate change	0 1 2 3 4 5
		30

Table 5. ISO 9126- Reliability

SUB-Characteristics Steak	Description
Maturity	The ability of software to avoid failure as a result of from errors in the software.
Fault Tolerance	The ability of the software to maintain its performance if a software error occurred.
Recoverability	Software ability to rebuild performance levels when a system failure occurs, including data and network connections.

Based on function points and maximum function points, the level of achievement of SIAKAD software functionality is as

follows:

$$\begin{gathered}\text{Unknown: } FP = 163.4 \\ FPMax = 220.59 \\ \text{Functionality} = \frac{\text{FP}}{\text{FPMax}} \\ \text{Functionality} = \frac{\text{FP}}{\text{FPMax}} = 0.7410\end{gathered} \quad (3)$$

So, the SIAKAD software functionality value is 0.7410.

3.3.3 Reliability

Software reliability indicators are obtained from the rate of failure occurrence (ROCOF) metric. To calculate ROCOF, the variables function point (FP), and number of failures (Failure) are needed (Table 5).

Diketahui: FP=163.4

$$\begin{gathered}Failure = 5 \\ \text{ROCOF} = \frac{Failure}{Function\ Point} \\ \text{ROCOF} = \frac{5}{163,4} = 0.0305\end{gathered} \quad (4)$$

Reliability can be derived using the following formula:

$$\begin{gathered}\text{Reliability} = 1 - ROCOF \\ Reliability = 1 - 0.0305 \\ Reliability = 0.969\end{gathered} \quad (5)$$

Thus, the reliability of SIAKAD is 0.969. A value of 0.969 indicates that in 1,000 operations, it is estimated that the software of the SIAKAT application is able to work correctly for 969 times. In other words, in 1000 operations the application has been estimated to fail 31 times.

3.3.4 Usability

The software usability indicator is obtained from the speed of operation metric. The more the speed of operation approaches a value of 0, indicating that the usability is increasing. Conversely, the more the speed of operation approaches a value of 1, the more the usability decreases (Table 6).

Table 6. ISO 9126- Usability

SUB – Characteristics	Description
Understandability	Software capabilities in ease to understand.
Learnability	Software capabilities in ease to learn.
Operability	Software capabilities in ease to operated.
Attractiveness	The ability of software to attracts users.

Unknown: Function Point(FP)=163.4

$$\begin{gathered}User\ Input = 8 \\ User\ Request = 10 \\ Speed\ of\ Operation = \frac{User\ Input + User\ Request}{Function\ Point} \\ Speed\ of\ Operation = \frac{8+10}{163,4} = 0.4895 \\ Usability = 1 - Speed\ of\ Operation \\ = 1 - 0,4895 = 0.5105\end{gathered} \quad (6)$$

3.3.5 Efficiency

Efficiency is related to performance, resources required, and savings gained from product use. To obtain efficiency indicators, several variables are needed to calculate them. These variables are total benefits, total costs, and effort required to build the software (Table 7).

Table 7. ISO 9126- Efficiency

SUB – Characteristics TIK	Description
Time Behavior	Software capabilities in providing response and time appropriate processing when performing its functions.
Resource Behavior	Software capabilities in using resources it has when performing the specified function.

Metric Approximation

$$\begin{gathered}\text{Known: } FP = 163.4 \\ E = -13.39 + 0.0545\ FP \\ E = -13.39 + (0.0545 \times 163.4) \\ E = 4.4847\ person - month\end{gathered} \quad (7)$$

3.3.6 Maintenance

In software, the more changes that occur in the software indicate that maintenance will be more difficult to do. To find the value of maintainability, the metric used is the Software Maturity Index (SMI) metric. The more SMI approaches the value of 1, the more stable the product will be. Conversely, the more SMI moves away from the value of 1, the more unstable the product will be. The variables needed to find the SMI value are the current number of modules (MT), the number of added modules (Fa), the number of modules that have changed (Fc), and the number of modules that have been deleted since the initial design (Fd) (Table 8).

Table 8. ISO 9126- Maintainability

SUB – Characteristics K	Description
Analyzability	Software capabillities in diagnosing deficiencies or causes or failure.
Changeability	The ability of the software to be modified certain.
Stability	The ability of the software to minimize the unpredictable effects of software modification.
Testability	Software capability to be modified and validated other software.

Unknown: MT=3

$$\begin{gathered}Fa = 0 \\ Fc = 0 \\ Fd = 0 \\ SMI = \frac{MT - (Fa + Fc + Fd)}{MT} = \frac{3 - (0+0+0)}{3} = \frac{3}{3} = 1\end{gathered} \quad (8)$$

A maintenance indicator of 1 indicates that out of every 3 program modules, there is 1 module that is expected to be stable so that it does not require significant changes at

maintenance time and 2 other modules that are expected to undergo changes at the maintenance stage.

3.3.7 Portability

After obtaining the six ISO 9126 quality indicators, namely functionality, reliability, usability, efficiency, maintainability, and portability, the last step is to generalize. The generalization in question is an overall assessment of the achievement of the quality of the SIAKAD system software the number of added modules (Fa), the number of modules that have changed (Fc), and the number of modules that have been deleted since the initial design (Fd) shown in Table 9.

The Table 10 lists different characteristics related to the quality of a software application, in this case, an SIAKAD application. Each characteristic has a value assigned to it. Here's a detailed explanation of each term: Functionality (Value: 0.74): This measures how well the software performs its intended functions. A value of 0.74 indicates a certain level of functionality provided by the software. Reliability (Value: 0.96): This measures the software's ability to consistently perform its functions without failure. A value of 0.96 suggests high reliability. Usability (Value: 0.51): This measures how easy and user-friendly the software is. A value of 0.51 implies that usability could be improved. Efficiency (Value: 0.4): This measures how well the software utilizes resources (like memory and processing power) while performing its tasks. A value of 0.4 indicates that efficiency might need enhancement. Maintainability (Value: 1): This measures how easy it is to maintain and update the software. A value of 1 is the highest among the listed characteristics, indicating that the software is quite maintainable. Portability (Value: 2): This measures how easily the software can be transferred from one environment to another. A value of 2 suggests that portability is a strong point of the software.

Table 9. ISO 9126- Portability

SUB – Characteristics Steak	Description
Adaptability	The ability of software to be adapted to the environment different.
Instalability	The ability of the software to be installed in an environment different.
Coexistence	The ability of the software to the coexist with the device other software in one environment by sharing resources.
Replacementy	The ability of the software to be used as a substitute other software.

Table 10. Portability

No.	Characteristics of	Value
1	Functionality	0.74
2	Reliability	0.96
3	Usability	0.51
4	Efficiency	0.4
5	Maintainability	1
6	Portability	2
	Total quality	5.61

Total Quality (5.61): This is the sum of all the values assigned to the characteristics listed above.

$$\text{Unknown: Maximum Quality} = 6$$
$$Quality\ Achievement = 5.61 \quad (9)$$

$$Percentage\ of\ Quality\ Achievement = \frac{Quality\ Achievement}{Maximum\ Quality} \times 100\%$$
$$= \frac{5.61}{6} \times 100\% = 93.5$$

4. CONCLUSIONS

A measurement is necessary to ascertain the quality of a software application. Standards, tools, and variables are used in quality measurement. The SIAKAD application software attained a quality of 93.5% through measurements utilizing metrics and ISO 9126 criteria. Dependability, with a value of 0.9888, is the greatest quality attribute of the SIAKAD application; efficiency, with a value of 0.4, is the lowest. The study reveals that the measurements of some metrics are directly correlated with software quality, whereas others are negatively correlated. Metrics such as the Software Maturity Index (SMI), function points, and portability show a strong correlation with software quality. To normalize data from negatively correlated metrics, the number 1 is subtracted from these values.

The research findings indicate that, with a quality attainment of 93.5%, the SIAKAD application is viable for implementation. However, the administration retains the authority to decide whether to utilize this program or seek alternatives. This study successfully achieved its aim of determining the degree of software quality attainment using the function-oriented metric approach and ISO 9126 quality criteria.

Summary of Research Findings: 1) Overall Quality Attainment: The SIAKAD application achieved an overall quality score of 93.5%, indicating high software quality. 2) Dependability: With a score of 0.9888, dependability emerged as the highest quality attribute, highlighting the software's robustness and reliability. 3) Efficiency: Efficiency scored the lowest at 0.4, suggesting areas for improvement in resource usage and performance optimization. 4) Correlation of Metrics: The study found a direct correlation between certain metrics (e.g., SMI, function points, portability) and software quality, while others showed a negative correlation.

Prospects for Future Work: 1) Detailed Analysis of Efficiency: Future research should focus on in-depth analysis and improvement strategies for the efficiency attribute, as it scored the lowest. 2) Broader Metric Evaluation: Expanding the range of metrics evaluated to include more diverse and comprehensive indicators of software quality. 3) Comparative Studies: Conducting comparative studies with other academic information systems to benchmark SIAKAD's performance and identify best practices. 4) Longitudinal Studies: Implementing longitudinal studies to monitor the SIAKAD application's quality over time and assess the impact of continuous improvement efforts.

By addressing these areas, future research can build on the findings of this study and contribute to the ongoing enhancement of academic information systems' quality.

REFERENCES

[1] Arfan, A. (2010). Implementation of quality measurement in software using lines of code (LOC) and function point (FP) methods. Department of Computer Science. Thesis. University of North Sumatra.

https://repositori.usu.ac.id/handle/123456789/76200, accessed on Jun. 5, 2023

[2] Hariyanto, B. (2004). Object oriented system engineering. Informatics, Bandung. ISBN: 978-623-8385-50-8. https://repository.dinus.ac.id/docs/ajar/REKAYASA_SISTEM_BERORIENTASI_OBJEK-_RINGKASAN.pdf, accessed on May. 15, 2012.

[3] ISO/IEC 9126. (1991). Information technology-software product evaluation-quality characteristics and guidelines for their use. International Organization for Standardization. https://cdn.standards.iteh.ai/samples/22749/d293dbe1fea54b3e853dfc5a07549390/ISO-IEC-9126-1-2001.pdf.

[4] Longstreet, D. (2002). Function Point Analysis Training Course. https://sun.aei.polsl.pl/~jfrancik/lectures/download/ozpi/fpt.pdf, accessed on Jul. 5, 2011.

[5] Nazir, M. (2005). Metode Penelitian. Ghalia Indonesia, Bogor. ISBN: 979-450-173-5. https://elibrary.bsi.ac.id/readbook/207604/metode-penelitian, accessed on Jul. 5, 2011.

[6] Batra, R.C. (1984). Recent advances in non-linear computational mechanics. Engineering Analysis, 1(1): 64. https://doi.org/10.1016/0264-682x(84)90024-8

[7] Ritzkal, S., Syaiful, S. (2020). The application of academic information system measurement software with iso standardization. In Proceedings of the International Conference on Industrial Engineering and Operations Management.

[8] Simamarta, J. (2010). Software Engineering. Andi, Yogyakarta. ISBN: 9792913475 & 9789792913477. https://books.google.co.id/books/about/Rekayasa_Perangkat_Lunak.html?id=QP1BjG_VIsoC&redir_esc=y.

[9] Sommerville, I. (2007). Software Engineering. 8th ed. Pearson Education Limited, Harlow. https://fita.vnua.edu.vn/wp-content/uploads/2013/06/Software-Engineering-By-Ian-Sommerville-8th-Edition.pdf, accessed on May. 15, 2021.

[10] Meilich, A., Rickels, M. (1999). 1 an application of Object Oriented Systems Engineering (OOSE) to an army command and control system: A new approach to integration of system and software requirements and design. In INCOSE International Symposium, 9(1): 1005-1016. https://doi.org/10.1002/j.2334-5837.1999.tb00268.x

[11] Clausen, H.D., Hofmann, A., Pawlitzek, G. H. (1992). Integration of different system engineering methods using an object-oriented model. Acm Sigplan Oops Messenger, 4(2): 179-181. https://doi.org/10.1145/157710.157748

[12] Amalia, R., Wijaya, A. (2018). Evaluasi kepuasan penggunasistem informasi akademik perguruan tinggi menggunakan standar ISO 9126. Jurnal Ilmiah Matrik, 20(2): 147-156. https://doi.org/10.33557/jurnalmatrik.v20i2.117

[13] Dwi, L., Siti, R., Umi, L.Y. (2017). Compilation and testing of operability metrics for academic information systems based on ISO 25010. Lesmideyarti. https://doi.org/10.35585/inspir.v7i2.2442

[14] Aswati, S., Ramadhan, M.S., Firmansyah, A.U., Anwar, K. (2017). Analysis study of rapid application development model in information system development. Matrix Journal. https://journal.universitasbumigora.ac.id/index.php/matrik/article/view/10, accessed on May. 20, 2019.

[15] Putra, I.H., Ismanto, S.U., Runiawati, N. (2022). Efektivitas sistem informasi manajemen daerah barang milik daerah (Simda Bmd) pada pemerintah daerah kota bandung. JANE-Jurnal Administrasi Negara, 14(1): 238-255.

[16] Hermanto, A., Supangat, S., Mandita, F. (2017). Evaluasi usabilitas layanan sistem informasi akademik berdasarkan kombinasi servqual dan webqual studi kasus: SIAKAD politeknik XYZ. Journal of Information Systems Engineering and Business Intelligence, 3(1): 33. https://doi.org/10.20473/jisebi.3.1.33-39

[17] Anwar, M.A.H., Kurniawan, Y. (2019). Dokumentasi software testing berstandar IEEE 829-2008 untuk sistem informasi terintegrasi universitas. Kurawal, 2(2): 118-125. https://doi.org/10.33479/kurawal.v2i2.261

[18] Saputera, S.A., Sunardi, D., Syafrizal, A., Samsidi, P. (2020). Evaluasi sistem informasi akademik menggunakan metode mccall. Journal of Technopreneurship and Information System, 3(2): 38-45. https://doi.org/10.36085/jtis.v3i2.878

[19] Aditya, A., Purwiantono, F.E. (2020). Penyusunan kerangka konseptual pengukuran kualitas sistem informasi akademik di kampus STIKI malang berdasarkan standart ISO 9126. Ilmu Komputer, 7(5): 979-984. https://doi.org/10.25126/jtiik.2020711513

[20] Buchholz, G., Engel, J., Märtin, C., Propp, S. (2007). Model-based usability evaluation-evaluation of tool support. Lecture Notes in Computer Science, 4550: 1043-1052. https://doi.org/10.1007/978-3-540-73105-4_114.

[21] Alturki, R., Gay, V. (2017). Usability testing of fitness mobile application: Case study aded surat app. International Journal of Computer Science and Information Technology, 9(5): 105-125. https://doi.org/10.5121/ijcsit.2017.9509

[22] Hayuni, S., Hidayah, M., Rahman, M.A., Welnof, S. (2021). Sistem informasi penerimaan mahasiswa baru di universitas dharmawangsa. Journal of Software Engineering, Computer Science & Information Technology, 2(1): 135-138. https://doi.org/10.46576/syntax.v2i1.1328

[23] Jordan, G.A., Ballance, R.H. (1983). A microcomputer-based annual ring measurement system. The Forestry Chronicle, 59(1): 21-25. https://doi.org/10.5558/tfc59021-1

[24] Marcus, A., Poshyvanyk, D. (2005). The conceptual cohesion of classes. In 21st IEEE International Conference on Software Maintenance (ICSM'05), Budapest, Hungary, pp. 133-142. https://doi.org/10.1109/ICSM.2005.89

[25] Prasad, L., Nagar, A. (2009). Experimental analysis of different metrics (object-oriented and structural) of software. In 2009 First International Conference on Computational Intelligence, Communication Systems and Networks, Indore, India, pp. 235-240. https://doi.org/10.1109/CICSYN.2009.22

[26] Srinivasan, K.P., Devi, T. (2014). Software metrics validation methodologies in software engineering. International Journal of Software Engineering & Applications, 5(6): 87. https://doi.org/10.5121/ijsea.2014.5606

[27] Fountas, S., Carli, G., Sørensen, C.G., Tsiropoulos, Z., Cavalaris, C., Vatsanidou, A., Liakos, B., Canavari, M., Wiebensohn, J., Tisserye, B.A. (2015). Farm management information systems: Current situation and future perspectives. Computers and electronics in Agriculture, 115: 40-50. https://doi.org/10.1016/j.compag.2015.05.011

[28] Masrek, M.N., Gaskin, J.E. (2016). Assessing users satisfaction with web digital library: The case of Universiti Teknologi MARA. The International Journal of Information and Learning Technology, 33(1): 36-56. https://doi.org/10.1108/IJILT-06-2015-0019

[29] Rohayani, H., Gaol, F.L., Soewito, B., Warnars, H.L.H.S. (2017). Estimated measurement quality software on structural model academic system with function point analysis. In 2017 International Conference on Applied Computer and Communication Technologies (ComCom), Jakarta, Indonesia, pp. 1-5. https://doi.org/10.1109/COMCOM.2017.8167085

[30] Saputri, N.A.O., Alvin, A. (2020). Measurement of user satisfaction level in the bina darma information systems study program portal using end user computing satisfaction method. Journal of Information Systems and Informatics, 2(1): 154-162. https://doi.org/10.33557/journalisi.v2i1.43