**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


# Application Security Analysis Mobile Commerce With Mobile Security Framework(MOBSF) and OWASP Mobile Application Security Testing Guide(MASTG)

**Eri Rustamaji [1*], Shaqila Erbeliza[2], Elsy Rahajeng[3], Nuryasin Nuryasin[4], Oshid Akbar Pratama[5]**

[1, 2, 3, 4] Department of Information System, UIN Syarif Hidayatullah Jakarta, Indonesia
[5]Department Master of Information Technology, UIN Syarif Hidayatullah Jakarta, Indonesia

[1] eri.rustamaji@uinjkt.ac.id
[2]shaqilaerbeliza@gmail.com
[3]elsy.rahajeng@uinjkt.ac.id
[4]nuryasin@uinjkt.ac.id
[5]oshidakbarpratama24@mhs.uinjkt.ac.id

Received: 29 January 2026, Revised: 28 February 2026, Accepted: 4 March 2026, Published: 30 April 2026

## Abstract

The development of Information Technology is currently growing very rapidly, one of which is the use of mobile devices today. In recent years, the use of mobile applications has increased in various areas of Indonesian society's life. However, cyber crimes such as data leaks are also increasing in Indonesia. One of them is the case of data theft in mobile commerce applications in Indonesia, where as many as 91 million user data were traded by hackers illegally on Dark Web sites. Jakmall's mobile commerce application also stores sensitive user data for use in its business processes such as email, passwords, addresses, telephone numbers and account numbers. The purpose of this study is to analyze and identify security vulnerabilities or loopholes that could harm providers and users of the Android-based Jakmall mobile commerce application with the Mobile Security Framework (MOBSF) and the OWASP Mobile Application Security Testing Guide (MASTG). This research was carried out in 5 (five) stages, namely Preparation, Intelligence Gathering (data collection), Mapping the Application (mapping vulnerabilities), Exploitation (exploitation), and Reporting (reports). The results of the study found that the Jakmall mobile commerce application has a security gap issue in the Data Storage range in parameter (MSTG-STORAGE-5) and in the Authentication Architectures range in parameter (MSTG-AUTH-5 and MSTG-AUTH-6).

**Keywords:** Android Apps, Mobile Commerce, Security, MOBSF, OWASP.

## I. Introduction

In recent years, the use of mobile applications has increased in various areas of Indonesian society. Based on Reportal Data in the Digital 2022: Indonesia report, the number of internet users in Indonesia has reached 204.7 million with a penetration rate of 73.7 percent. This figure increased compared to last year by 1 percent [1]. However, the increase in the number of internet users in Indonesia is not directly proportional to a good level of cybersecurity. Cybersecurity in Indonesia is a serious problem with frequent data leaks.

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


According to a report from the National Cyber Security Index (NCSI) in 2022, Indonesia has a cyber security index score of 38.96 points out of 100. This assessment aims to measure the country's readiness to prevent cyber threats and manage cyber security incidents. This places Indonesia at 83rd rank out of 160 countries in the world and 6th in ASEAN and is ranked 3rd lowest compared to the G20 countries [2].

The increase in cybercrimes such as data leaks has been a succession in Indonesia. One of them is the case of data theft in mobile commerce applications in Indonesia, in research [3] As many as 91 million Tokopedia application user data are illegally traded by hackers on the Dark Web site. User data that may be exposed is personal data such as email, name, address, date of birth and telephone number. The perpetrators sold the data for $ 5,000 or around Rp. 74 million [4]. Application providers or in that case mobile commerce applications in general often ask for permission regarding personal data from their users. These data are used to support the application's business processes. Primary user identities such as email, address and telephone number are unique and vulnerable to abuse [5].

According to [6], Mobile commerce is an online channel that can be reached by someone through a mobile device that is used by sellers and customers to make transactions online. The increase in the number of electronic commerce users in Indonesia is predicted to reach 189.6 million users in 2024 [7]. Therefore, it is important for all parties involved in online transactions to consider threats to information security and must be equipped with relevant and up-to-date knowledge to minimize the risks [8]. At the end of 2018, Jakmall is one of the largest E-Commerce in Indonesia [9], seeks to develop its company by releasing an Android-based mobile commerce application. Until now, users of the Android-based Jakmall application have reached more than 100,000 users. Jakmall is a mobile commerce application that offers different things in the midst of intense e-commerce competition. Jakmall provides a platform that opens opportunities for resellers to pick up goods from stalls on this platform. Therefore, Jakmall conducts strict selection for users to provide products at low prices. Based on observations in this study, Jakmall collects User information in processing and expediting the process of using the application [10].

There are several frameworks or frameworks that can help us in testing the security of Android applications. Among them are the Mobile Security Framework (MOBSF) and the Open Web Application Security Project Mobile Application Security Testing Guide (OWASP MASTG) which can provide clear security analysis results to serve as security improvement materials for Android application developers [11]. Mobile Security Framework (MobSF) is a framework used as an automated test of open-source mobile applications (Android, iOS, Windows) that performs penetration testing, malware analysis and is able to detect vulnerabilities that can be exploited by attackers in mobile applications [12]. In addition, OWASP MASTG is a security standard used in mobile applications accompanied by a comprehensive testing guide that includes processes, techniques, tools, and case studies related to carrying out mobile application security tests. According to research [13] mentions that OWASP MASTG can find more android application vulnerabilities from the analysis results on the AndroBugs framework.

Identification of the problem based on the background that has been described is:

1. There are still many cases of cybercrime that are increasing in Indonesia, one of which is the theft of sensitive mobile application user data being traded by hackers illegally.
2. Jakmall is one of the largest e-commerce sites in Indonesia, with a total of 100,000 (one hundred thousand) Android-based application users, so it is very important to pay attention to the security factor in this application.

Based on this identification, the formulation of the problem will be discussed in this studyHow to analyze the security of Jakmall's mobile commerce application to identify security holes and provide recommendations to reduce the impact of the vulnerabilities found?

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


The author limits the scope so that research can be carried out with more focus and depth. Therefore, the author provides the following limitations:

1. This study uses the Android-based Jakmall mobile commerce application as the research object.
2. Using MOBSF version 3.5.2 as a security detection framework for Android applications that are made into objects using static analysis methods.
3. This study uses a testing guide with the Open Web Application Security Project (OWASP) Mobile Application Security Testing Guide (MASTG) to conduct penetration testing as well as suggestions for fixing existing loopholes.
4. This study only tested 2 (two) scopes on OWASP MASTG, namely Data Storage and Authentication Architectures.
5. This research only tests at the mobile application level, not at the mobile device operating system level.
6. This research uses Android application files with public access that can be downloaded from Playstore.
7. The device used to carry out security testing has an Intel Core i5 8th Gen process specification, 4GB RAM, 256GB SSD, and the Windows 11 operating system.
8. This research uses tools Jadx 1.4.4, ADB (Android Debug Bridge) and tools Burp Suite Community Edition v2022.12.5 connected to Nox Emulator v7.0.5.0 to perform penetration testing.

Based on the background and formulation of the problem, this study has the following objectives:

1. Analyze and identify security vulnerabilities or gaps that could harm providers and users of the Android-based Jakmall mobile commerce application.
2. Generate a security evaluation report from the analysis results on the Android-based Jakmall mobile commerce application using the OWASP MASTG guide.
3. Provide suggestions or recommendations for improvements to security issues found from the results of research on the Jakmall mobile commerce application..

## II. Related Work

Research on software maintainability initially centered on structural code metrics as objective indicators of internal quality. Classical object-oriented metrics such as the Chidamber & Kemerer suite and Lines of Code (LOC) remain widely validated predictors of complexity, coupling, and structural stability. Although these metrics were introduced decades ago, empirical studies consistently confirm their relevance, demonstrating that fundamental structural characteristics continue to shape maintenance effort and system evolution. Subsequent research expanded beyond static measurement toward qualitative degradation factors, particularly code smells and technical debt. Empirical investigations show that unmanaged code smells significantly increase maintenance complexity, prompting the development of prioritization models and automated refactoring techniques, including optimization-based and fuzzy genetic approaches. In parallel, machine learning–driven predictive models have become dominant in estimating maintainability levels, leveraging ensemble algorithms such as Random Forest variants. However, class imbalance within software datasets remains a persistent limitation, requiring resampling strategies to enhance predictive robustness and reduce bias toward majority classes.

More recent studies introduce a dynamic and sustainability-oriented perspective by incorporating historical software evolution data and exploring maintainability's relationship with energy consumption. Evolution-aware models demonstrate that historical metric changes across releases provide stronger predictive power than snapshot-based approaches, recognizing maintainability as a temporal property shaped by cumulative modifications. Additionally, exploratory empirical evidence suggests that structural characteristics

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


such as LOC may influence energy usage, extending maintainability implications beyond technical quality toward green software engineering. Despite significant advances across metrics, predictive analytics, refactoring automation, and sustainability research, existing studies remain fragmented. An integrative framework that unifies structural metrics, historical evolution, imbalance-aware prediction, and environmental considerations therefore represents a critical research opportunity

## III. Research Methodology

### A. Method of collecting data

1. Literature review

The author reads and learns from books, journals and articles from the internet that have something to do with the research being taken. The author conducted a literature study to provide an overview of the stages of analyzing mobile application security based on the MOBSF and OWASP MASTG guidelines.

2. Field Study/Observation

The author made direct observations interacting with the Jakmall e-commerce Android mobile application or in other words using the features contained in the application, because by making direct observations researchers can easily understand and collect information sent by the application.

### B. Data analysis method

In testing mobile application security [14] there are several stages, which consist of:

1. preparation
2. Intelligence Gathering
3. Mapping the Analysis
4. Exploitation
5. reporting

### C. Characteristics and Reference Selection

The study identification and selection process was conducted through several systematic stages. First, an initial search was conducted in major academic databases such as IEEE Xplore, ACM Digital Library, Scopus, and SpringerLink using a combination of keywords relevant to software maintainability and maintainability prediction. The number of references screened in this initial stage was substantial, reflecting the breadth of research in the field of software maintainability. The initial search resulted in 118 articles published in international journals from 1981 to 2025. However, the SLR process typically involves thousands of initial results that are then filtered through inclusion and exclusion criteria.[19], [20], [21]

Inclusion criteria focused on studies published between 2014 and 2025 that: (1) explicitly discussed software maintainability or its influencing factors (e.g., technical debt, code smells), (2) proposed or evaluated models/metrics/approaches for predicting or improving maintainability, (3) used empirical research methodologies (case studies, experiments, surveys) or systematic reviews, and (4) were published in peer-reviewed scientific journals. Exclusion criteria included studies that: (1) only discussed software quality in general without focusing on maintainability, (2) were in gray literature (e.g., theses, dissertations, blog posts) without peer review, (3) were non-systematic review articles or opinion pieces, and (4) were not relevant to

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


the context of predicting or improving software maintainability, (5) or were reputable conference proceedings in English. The study selection method was conducted in two stages: first, screening based on titles and abstracts to identify potentially relevant studies; second, reading the full text of the studies that passed the first stage to thoroughly verify their relevance. This process ensures that only the most relevant and high-quality studies are included in this review, reflecting current and relevant research trends on software maintainability.

### D. Validity and Reliability

The validity and reliability of an SLR depend heavily on thoroughness at every stage of the process. To ensure internal validity, the research protocol was explicitly designed and documented, including a comprehensive search strategy, objective selection criteria, and standardized data extraction procedures. This helps minimize study selection and data extraction bias. External validity was achieved by ensuring broad coverage of relevant databases and publications, so that the synthesized findings can be generalized to various research contexts and practices in the field of software maintainability.

The reliability of this study was enhanced through the PRISMA guidelines, which mandate a systematic and transparent approach. The interpretation and data extraction of selected studies will be achieved by involving at least two researchers independently in the screening and data extraction process, then resolving discrepancies through discussion and consensus. The use of a reference manager also ensures consistency and accuracy of citations and bibliographies. Although some studies may have limitations in their own internal validity or reliability (e.g., imbalanced data in predictive models that could impact accuracy), this SLR will critically evaluate and address these limitations in the synthesis of findings. Focusing on empirical studies that have gone through a rigorous peer-review process also inherently supports the reliability of the data collected.

The data collection procedures in this study were systematically designed to ensure transparency and replicability. The preparation phase involved designing a detailed SLR protocol, including a clear definition of the research question, a comprehensive keyword search strategy, and specific inclusion and exclusion criteria, similar to the practice in other SLR studies. [22], [23]. The data collection period will cover the period from the protocol drafting date to the completion of the study selection process, with an estimated duration of several weeks to allow for careful screening of the significant volume of literature. The data collection will be conducted online through access to the sci-hub database, which is standard practice in systematic literature reviews. The specific technique applied for data collection is data extraction from each article that has passed the inclusion criteria, where relevant information such as the year of publication, the metrics used (e.g., Chidamber & Kemerer, lines of code), the methodology (e.g., machine learning, refactoring), the main results, and the limitations of the study are recorded in a structured manner. For example, data regarding the impact of code smells and the prioritization of bad smells on maintainability will be extracted from related studies. [24]. In addition, information regarding the handling of imbalanced data in maintainability prediction is also a focus of data extraction, considering its significance in improving model accuracy.

The data analysis methods used in this study are narrative and thematic synthesis, chosen due to their relevance in processing diverse qualitative and quantitative data from the included studies. The thematic approach allows for the identification of recurring patterns, trends, and research gaps across the collected literature. Quantitative data, such as maintainability metrics or predictive model accuracy from the surveyed studies, will be analyzed descriptively to identify the most frequently used metrics and the effectiveness of various machine learning algorithms. Qualitative data, such as findings on maintainability challenges or proposed solutions to address code smells, will be synthesized to build a comprehensive understanding of the

ITDM: Information Technology Development and Management
Volume 2, (1) 2026, p. 15-25
E-ISSN: XXXX-XXXX


current research landscape. This approach will also make it possible to identify how previous studies addressed issues such as imbalanced data in maintainability predictions or the relationship between maintainability and energy consumption, as well as the limitations they faced [25]. By comparing findings from various sources, this thematic analysis aims to understand and formulate recommendations for robust and relevant adaptive predictive models.

## IV. Result

### 4.1. General description

Jakmall's mobile commerce is a handheld device application released at the end of 2018 to make purchases and sales online. This application can be used free of charge by downloading the application in the Google Play Store for Android-based applications and the App Store for iOS-based applications. The Android-based Jakmall application has main features including Home, Categories, Baskets, Transactions, Jakwallet and Accounts. This application requires data that is included in PII (Personal Identifiable Information), namely a person's identity such as name, email, address and telephone number, date of birth to financial account numbers.

### 4.2. preparation

This stage aims to determine the scope or coverage used in security testing of applications. The scope used in this test is Data Storage and Authentication Architectures in the OWASP MASTG framework. DataStorageis the scope that includes data storage media. Data storage is very important in mobile applications, poor implementation of this scope can have fatal consequences such as leaked data so that it is misused by irresponsible parties. Meanwhile, Authentication Architecures is a scope that includes authentication and authorization. Authentication and authorization issues are common security vulnerabilities. In fact, this scope consistently ranks second highest in OWASP because most applications implement user authentication [26].

### 4.3. Intelligence Gathering

Based on the results of automatic scanning with MOBSF, it provides information that the APK file that is the target for analysis is version 1.3.83 with version code 35. Android-based Jakmall can run with a minimum of API (Application Programming Interface) level 21 or Android 5.0 and with a target SDK (Software Development Kit) on API level 32 or Android 12. The authenticity of the Jakmall APK file can be checked using the SHA256 cryptographic algorithm (Secure Hasing Algorithm) with MD5 (Message Digest) and SHA1 hashes.

### 4.4. Mapping the Application

Based on the results of the static analysis from MOBSF, there are several security issues found from the code analysis. This stage performs mapping or categorization of vulnerabilities found from the results of static analysis based on the OWASP MASVS (Mobile Application Security Verification Standard) within the scope of Data Storage and Authentication Architectures. The following is a vulnerability categorization obtained from static analysis by the MOBSF tool. The Jakmall mobile commerce application found a vulnerability in the MSTG-STORAGE-2 parameter which is included in the OWASP MASTG Data Storage scope in the

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


Testing Local Storage for Sensitive Data section. MOBSF did not find any security issues in other parameters, so the authors conducted further tests to validate the findings from MOBSF.

### 4.5. Exploitation

The purpose of this stage is to verify the test cases from the OWASP MASTG guide related to the assessment of the OWASP Mobile Application Security Verification Standard (MASVS) parameters generated from the automated testing of the MOBSF code analysis tools.

1. Testing Local Storage for SensitiveData (MSTG-STORAGE-1 and MSTG-STORAGE-2). These test cases focus on identifying potentially sensitive data stored by applications and verifying that it is stored securely. This is intended so that sensitive data stored is not leaked due to malicious application attacks [27]. Android devices have several information storage media, such as internal storage, external storage, and cloud storage. In this test case, it is carried out to identify the source of the type of storage in the application.However, after further testing the shared preferences (/data/data/com.jakmall/shared_prefs) to look for XML files that hold sensitive data. The result is that there is no sensitive user data stored in the application's shared preferences.
2. Testing Local Storage for Input Validation(MSTG-PLATFORM-2). At any publicly accessible data store such as login forms, or any process can modify that data. This can be used by irresponsible parties. Then it is necessary to apply input validation when the data is read back. In this test, an input validation analysis of the login mechanism was carried out based on the source code in the AuthHolder file of the mobile commerce application obtained from the reverse engineering results.

Based on result, an input validation analysis was carried out on the login mechanism on the Jakmall mobile commerce application source code. Where in the AuthHolder file used in the LoginActivity mechanism there is a getSharedPreference, it can be said that the input validation mechanism has been carried out in shared preferences. So that it can be stated that the input validation carried out on shared preference does not result in sensitive data or information. Logging is useful for application developers in tracking crashes, application usage errors, and viewing application usage statistics. Testing of these cases focuses on identifying any sensitive application data in the application logs. Testing is carried out by static analysis of the source code which records related code such as Log.d, Log.e, Log.i, Log.v, Log.w, and Log.wtf.

Based on the test, this case test was carried out using the adb tool by giving the "adb logcat" command through the command prompt. The process is carried out in the form of login features, changing user data, and checking out. Based on this test, no sensitive data was found in the Jakmall mobile commerce application log activity recording. The test in this case is an attempt to intercept HTTP (Hypertext Transfer Protocol) traffic between the client and server when logging in to the Jakmall mobile commerce application. Testing was carried out using the Burpsuite tool. The purpose of this test is to find out what information is obtained during the communication process between the client and server.

In this test the username and password used are the author's. The system sends data to the server in the form of input from the user without going through the process of encrypting data against usernames and passwords, thus allowing intercept. HTTP requests that easily change data to learn how applications behave when performing different actions. So that with these conditions it can be categorized as HTTP traffic for the Jakmall mobile commerce application in an unsafe condition. This test case identifies whether there is sensitive data exposed through the keyboard cache. When the user types in the input field, the application can suggest data, this means that the application allows storing data on the keyboard cache. However, cached keyboards can reveal sensitive data when the user selects an input field that uses this type of data. Based on

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


test, the application does not store keyboard cache regarding sensitive data such as email and passwords. Entering sensitive data, such as when registering an account or logging in, is an important part of using many applications. This data can be in the form of information such as user passwords. Data can be exposed if the app doesn't hide it properly while it's being typed. To prevent disclosure and mitigate risks such as shoulder surfing (direct observation) cybercrime, it is necessary to verify that no sensitive data is exposed through the user interface. Sensitive data in this case user passwords must be properly disguised, usually by showing asterisks or dots instead of clear text.

Based on the test in this case, it is carried out by checking the account registration form. It can be seen, that the password column has been masked or disguised with a dot symbol so that the text entered is not visible on the application interface. Password strength is a major concern when data is used for authentication. The password policy defines the requirements that must be complied with by the user. The password policy usually determines the length of the password, the complexity of the password, and the topology of the password. A "strong" password policy makes cracking manual or automated passwords difficult or impossible. In this test case, it is done by testing the password policy when registering an account. Testing was carried out using Burpsuite tools by looking at requests and responses during the process of registering a new account. The data used in this test is the new user data that the author created.

In testing is carried out by creating a new account. The author creates a new account by entering the password “panthers”. This password is a weak password because all characters only use lowercase letters [22]. The application gives a response "200 OK" which means that using a weak password in the application is allowed. This proves that the Jakmall mobile commerce application does not have a password policy for new users. Weak passwords are easier to attack and can be abused. The author will test this mechanism by carrying out a brute force attack technique. This technique is done by trial and error in breaking the password. It can be seen that the Jakmall mobile commerce application can be carried out with a brute force attack on the login feature. The author performs a brute force attack technique on the user's password by trying to log in 1000 (one thousand) times with the user's username and the result is that 1 (one) password matches that username. This is also tied to the previous parameters in the Data Storage scope, namely Determining Whether Sensitive Data is Sent to Third Parties (MSTG-STORAGE-4) so that the login feature is easily exposed to brute force attack techniques.

### 4.6. Reporting

Based on the exploitation stage that has been carried out from 7 (seven) parameters in the scope of Data Storage and Authentication Architectures, it was found that 2 (two) test parameters that were verified were security issue vulnerabilities, 2 (two) parameters namely Determining Whether Sensitive Data is Sent to Third Parties and Testing Best Practices for Passwords. In testing the Determining Whether Sensitive Data is Sent to Third Parties parameter, a security issue is found where the application sends data to the server in the form of input from the user without encrypting the data on the username and password. So that sensitive data is exposed to HTTP traffic. When the hacker can intercept or intercept traffic from the client, the hacker can explore and understand what he is doing or rewrite it to get another response from the server.

Then in testing the Testing Best Practices for Password parameter, a security issue was found where the application did not apply a password policy to the mechanism for creating a new account or registering an account. The use of weak passwords is a serious vulnerability problem because it is easier for hackers to take control of user accounts. By not applying the blocking mechanism or account locking against incorrect login requests many times, it gives an attacker the opportunity to freely guess an account's password with a brute force attack technique.

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


After that, recommendations for improvements were made based on the vulnerabilities found to improve the security of the Jakmall mobile commerce application. The security recommendations provided are based on the OWASP MASTG guidelines, namely:

- Determining Whether Sensitive Data is Sent to Third Parties(MSTG-STORAGE-4). Applying data encryption techniques when communicating between clients and servers, especially in the Login mechanism.

- Testing Best Practices for Passwords(MSTG-AUTH-5 and MSTG-AUTH-6). Enforce password policies when users register accounts, so users are required to use strong passwords. Limiting the number of login attempts to no more than 5 times so brute force techniques will be difficult. Implement an account lock system (temporary or permanent) if a user logs in more than 15 times. Added the Captcha feature to the login form to distinguish between human and robot logins. Maximizing the login mechanism by implementing two factor authentication with validation on other devices so as to prevent brute force attacks.

## V. Conclusion

This study systematically synthesized contemporary research on software maintainability to map the transition from traditional metric-based evaluation toward predictive intelligence–driven approaches. The findings confirm that classical structural indicators such as the Chidamber & Kemerer (C&K) metrics and Lines of Code (LOC) remain foundational in assessing internal software quality. However, maintainability research has significantly evolved beyond static measurement, incorporating machine learning–based prediction models, automated refactoring strategies, and historical evolution analysis. The dominance of predictive analytics demonstrates a paradigm shift in which maintainability is no longer assessed solely as a descriptive property but increasingly treated as a forecastable and manageable attribute of large-scale software systems.

The review also highlights critical challenges that shape the current research landscape. Class imbalance in maintainability datasets continues to undermine prediction reliability, necessitating resampling strategies and enhanced ensemble algorithms to improve model robustness. Additionally, the persistent impact of code smells and technical debt underscores the importance of prioritization frameworks and automated mitigation techniques to allocate maintenance resources effectively. The integration of historical metric changes further reveals that maintainability should be conceptualized as a dynamic and temporal characteristic rather than a static condition. These findings collectively reinforce the need for adaptive predictive models capable of capturing structural complexity, evolutionary trends, and data distribution constraints simultaneously.

In a broader perspective, the emerging linkage between maintainability and energy consumption expands the conceptual boundaries of the field, positioning maintainability within the broader discourse of sustainable and green software engineering. Although empirical validation remains limited, preliminary evidence suggests that improving structural quality may contribute not only to reduced maintenance costs but also to enhanced energy efficiency. Future research should therefore focus on developing integrated, evolution-aware, and imbalance-sensitive predictive frameworks, while also exploring interdisciplinary connections with sustainability and organizational factors. By consolidating fragmented streams of research, this study contributes a structured understanding of the maintainability paradigm and offers strategic direction for both researchers and practitioners seeking long-term software sustainability.

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


## REFERENCES

[1] I. Sommerville, Software engineering, 10. ed. Boston, Munich: Pearson, 2016.
[2] R. Saheb Nasagh, M. Shahidi, and M. Ashtiani, "A fuzzy genetic automatic refactoring approach to improve software maintainability and flexibility," Soft Comput., vol. 25, no. 6, pp. 4295–4325, Mar. 2021, doi: 10.1007/s00500-020-05443-0.
[3] M. Fowler and K. Beck, Refactoring: improving the design of existing code, 28. printing. in The Addison-Wesley object technology series. Boston: Addison-Wesley, 2013.
[4] T. Alshammari and M. Alshayeb, "Toward a Software Bad Smell Prioritization Model for Software Maintainability," Arab. J. Sci. Eng., vol. 46, no. 9, pp. 9157–9177, Sep. 2021, doi: 10.1007/s13369-021-05766-6.
[5] M. Gradišnik, T. Beranič, and S. Karakatič, "Impact of Historical Software Metric Changes in Predicting Future Maintainability Trends in Open-Source Software Development," Appl. Sci., vol. 10, no. 13, p. 4624, Jul. 2020, doi: 10.3390/app10134624.
[6] P. Avgeriou, P. Kruchten, I. Ozkaya, and C. Seaman, "Managing Technical Debt in Software Engineering (Dagstuhl Seminar 16162)," Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2016. doi: 10.4230/DAGREP.6.4.110.
[7]T. Alshammari and M. Alshayeb, "Toward a Software Bad Smell Prioritization Model for Software Maintainability," Arab. J. Sci. Eng., vol. 46, no. No. 9, pp. 9157–9177, Sep. 2021, doi: https://doi.org/10.1007/s13369-021-05766-6.
[8] S. Gupta and A. Chug, "Software Maintainability Prediction Using an Enhanced Random Forest Algorithm," J. Discrete Math. Sci. Cryptogr., vol. 23, no. No. 2, pp. 441–449, Feb. 2020, doi: https://doi.org/10.1080/09720529.2020.1728898.
[9] T. Besker, A. Martini, and J. Bosch, "Impact of Architectural Technical Debt on Daily Software Development Work — A Survey of Software Practitioners," in 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Vienna, Austria: IEEE, Aug. 2017, pp. 278–287. doi: 10.1109/SEAA.2017.16.
[10]R. Malhotra and K. Lata, "Using Ensembles for Class-Imbalance Problem to Predict Maintainability of Open-Source Software," Int. J. Reliab. Qual. Saf. Eng., vol. 27, no. No. 05, p. 2040011, Oct. 2020, doi: https://doi.org/10.1142/S0218539320400112.
[11] B. C. Mourão, L. Karita, and I. Do Carmo Machado, "Green and Sustainable Software Engineering - a Systematic Mapping Study," in Proceedings of the XVII Brazilian Symposium on Software Quality, Curitiba Brazil: ACM, Oct. 2018, pp. 121–130. doi: 10.1145/3275245.3275258.
[12] J. Mancebo, C. Calero, and F. García, "Does maintainability relate to the energy consumption of software? A case study," Softw. Qual. J., vol. 29, no. 1, pp. 101–127, Mar. 2021, doi: 10.1007/s11219-020-09536-9.
[13] M. Gradišnik, T. Beranič, and S. Karakatič, "Impact of Historical Software Metric Changes in Predicting Future Maintainability Trends in Open-Source Software Development," Appl. Sci., vol. 10, no. 13, p. 4624, Jul. 2020, doi: 10.3390/app10134624.
[14] R. Saheb Nasagh, M. Shahidi, and M. Ashtiani, "A fuzzy genetic automatic refactoring approach to improve software maintainability and flexibility," Soft Comput., vol. 25, no. 6, pp. 4295–4325, Mar. 2021, doi: 10.1007/s00500-020-05443-0.
[15] M. J. Page et al., "The PRISMA 2020 statement: an updated guideline for reporting systematic reviews," BMJ, p. n71, Mar. 2021, doi: 10.1136/bmj.n71.
[16] A. F. Herrera Ortiz et al., "A Practical Guide to Perform a Systematic Literature Review and Meta-analysis," Princ. Pract. Clin. Res. J., vol. 7, no. 4, pp. 47–57, Dec. 2021, doi: 10.21801/ppcrj.2021.74.6.
[17] N. Shaheen et al., "Appraising systematic reviews: a comprehensive guide to ensuring validity and reliability," Front. Res. Metr. Anal., vol. 8, Dec. 2023, doi: 10.3389/frma.2023.1268045.
[18] Y. Riyadi, M. Wahidin, and A. Elanda, "Systematic Literature Review Implementasi Service Operation Dalam Kerangka Kerja Information Technology Infrastructure Library (ITIL) di Indonesia: Tren Penelitian, Manfaat dan Tantangan," J. Interkom J. Publ. Ilm. Bid. Teknol. Inf. Dan Komun., vol. 17, no. 2, pp. 81–97, Jul. 2022, doi: 10.35969/interkom.v17i2.232.
[19] P. Candra Susanto, D. Ulfah Arini, L. Yuntina, J. Panatap Soehaditama, and N. Nuraeni, "Konsep Penelitian Kuantitatif: Populasi, Sampel, dan Analisis Data (Sebuah Tinjauan Pustaka)," J. Ilmu Multidisplin, vol. 3, no. 1, pp. 1–12, Apr. 2024, doi: 10.38035/jim.v3i1.504.
[20] N. Lestari, P. Paidi, and S. Suyanto, "A systematic literature review about local wisdom and sustainability: Contribution and recommendation to science education," Eurasia J. Math. Sci. Technol. Educ., vol. 20, no. 2, p. em2394, Feb. 2024, doi: 10.29333/ejmste/14152.
[21] P. M. Khristodas, K. Palanivelu, A. Ramachandran, J. Anushiya, B. A. K. Prusty, and S. Guganesh, "ASSESSMENT OF CLIMATE-INDUCED SEA-LEVEL RISE SCENARIOS AND ITS INUNDATION IN COASTAL ODISHA, INDIA," Appl. Ecol. Environ. Res., vol. 20, no. 4, pp. 3393–3409, 2022, doi: 10.15666/aeer/2004_33933409.
[22] R. I. Williams, L. A. Clark, W. R. Clark, and D. M. Raffo, "Re-examining systematic literature review in management research: Additional benefits and execution protocols," Eur. Manag. J., vol. 39, no. 4, pp. 521–533, Aug. 2021, doi: 10.1016/j.emj.2020.09.007.

**ITDM: Information Technology Development and Management**
**Volume 2, (1) 2026, p. 15-25**
**E-ISSN: XXXX-XXXX**


[23] S. Young et al., "PROTOCOL: Searching and reporting in Campbell Collaboration systematic reviews: An assessment of current methods," Campbell Syst. Rev., vol. 17, no. 4, Dec. 2021, doi: 10.1002/cl2.1208.
[24] T. Alshammari and M. Alshayeb, "Toward a Software Bad Smell Prioritization Model for Software Maintainability," Arab. J. Sci. Eng., vol. 46, no. 9, pp. 9157–9177, Sep. 2021, doi: 10.1007/s13369-021-05766-6.
[25] R. Malhotra and K. Lata, "An Empirical Study on Predictability of Software Maintainability Using Imbalanced Data," Softw. Qual. J., vol. 28, no. No. 4, pp. 1581–1614, Dec. 2020, doi: https://doi.org/10.1007/s11219-020-09525-y.