JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

ANALYSIS OF THE IMPLEMENTATION OF MVVM ARCHITECTURE PATTERN ON PERFORMANCE OF IOS MOBILE-BASED APPLICATIONS

Deri Indrawan*[1)], Dana Sulistyo Kusumo[2)], Shinta Yulia Puspitasari[3)]
1. Telkom University, Bandung, Indonesia
2. Telkom University, Bandung, Indonesia
3. Telkom University, Bandung, Indonesia

Article Info
Keywords: Architecture Pattern, Mobile Application, MVVM, Performance efficiency, RxSwift.

Article history:
Received 25 October 2022
Revised 7 November 2022
Accepted 13 November 2022
Available online 1 March 2023

DOI :
https://doi.org/10.29100/jipi.v8i1.3293

* Corresponding author.
Corresponding Author
E-mail address:
deriindrawan@student.telkomuniversity.ac.id

ABSTRACT
Performance efficiency is important in mobile application development because mobile devices have limitations in terms of power and resources. Performance efficiency can be improved by applying architecture patterns. In this paper, we use the Model View ViewModel (MVVM) architecture. The application of the architecture is carried out to analyze how practical the application of the MVVM architecture pattern is in increasing performance efficiency in the mobile application. Performance efficiency is measured based on CPU usage, memory usage, and execution Time. The case study shows that the CPU usage and execution Time on MVVM are smaller than Base architecture pattern from the AR Ruler. This is due to the third-party library RxSwift in the MVVM architecture that increases the application's response so that CPU usage and execution time is better than Base architecture pattern. However, the existence of the third-party library RxSwift has a negative impact on memory usage, resulting in higher memory usage than the Base Architecture Pattern. The MVVM pattern is highly recommended for mobile application development to improve performance efficiency.

I. INTRODUCTION

MOBILE application technology is developing very rapidly. Based on statistical data, the number of application downloads in 2021 will reach 150 billion [1]. Which shows that the need for mobile applications is so high. Android and iOS are the most widely used mobile operating systems (OS) and even control 99 percent of the market share [2]. However, the increased demand for mobile applications is accompanied by data that users' satisfaction responses to poor performance applications tend to delete applications [3]. Therefore, mobile applications must have good performance to increase user satisfaction in using mobile applications [4].

Devices for running mobile applications have limitations in power and resources, so there needs to be performance efficiency in mobile applications. Previous research reinforced this with user frustration with mobile applications stemming from performance problems [3]. ISO/IEC 25010 states that for mobile applications to produce efficient performance, it is necessary to have efficiency in CPU usage, memory, and how fast the application can execute programs [5]. One of them is by applying architecture patterns [6].

Several architectural patterns are used to build mobile applications, namely: Model View Controller (MVC), Model View Presenter (MVP), and Model View ViewModel (MVVM). In this paper, we test the MVVM architecture to see how much it improves in terms of performance efficiency. Architecture pattern is an architectural element that provides a packaged strategy to solve some of the problems faced by the system [7]. Architecture describes the elements, forms of interaction, and rules used to solve problems. The architecture pattern also has a framework. This framework aims to build the system incrementally to avoid the simultaneous integration of elements. This makes it easier to find problems at the beginning of development.

The MVVM architecture pattern was chosen because, in previous studies [6], it was stated that MVVM had better performance efficiency on CPU usage and execution time. However, it is necessary to conduct further research on performance issues because in previous research [6], the application used was only a sample application to-do list. Each application has different characteristics, so it is better to use an application with characteristics relevant to the problem to be tested. An architecture pattern is one alternative to improve performance on mobile applications because it can affect resource usage and execution time on mobile applications [6]. Among them is the high use of resources.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

Many studies have been conducted related to this research problem. A related study by Lou, T [6] A comparison of Android Native App Architecture MVC, MVP, and MVVM. This study conducts a thorough analysis of the MVC, MVP, and MVVM architecture patterns from the aspects of testability, modifiability, and performance. The analysis results show that MVP and MVVM are better than MVC. MVP and MVVM require less time to run tests, lower coupling levels, and use less memory. Another study was conducted by B. Wisnuadhi [8] Performance Comparison of Native Android Application on MVP and MVVM. This study analyzes the MVP and MVVM architecture patterns in the Point of Sales application regarding performance efficiency. The results show that the application of the MVVM architecture pattern on the android application produces better performance than MVP. The analysis results show that MVVM is better than MVP in CPU usage and execution time. However, MVVM in android applications uses more memory due to the additional databinding library. Based on related studies that have been done. The MVVM architecture pattern is better than MVP in terms of CPU usage and execution time [8]. This is due to the third-party library databinding that improves performance on the MVVM architecture pattern. However, this results in the use of more significant memory.

This study aims to apply the MVVM Architecture pattern to the AR Ruler application to improve performance efficiency. For the purpose research, we used the AR Ruler application. AR Ruler is an iOS-based application used to measure the length of an object by utilizing Augmented Reality technology. AR Ruler was chosen as a case study because it has many users and has been downloaded millions of times [9]. This application also applies Augmented Reality technology, which burdens CPU and memory performance [10].

II. Research Methodology

A. Research Stages

The research stages describe research groundwork concerning which research elements are required and what particular term are applied to complete the research study. The research methodology consists of an analysis and evaluation phase. The research stages shown in Fig. 1.

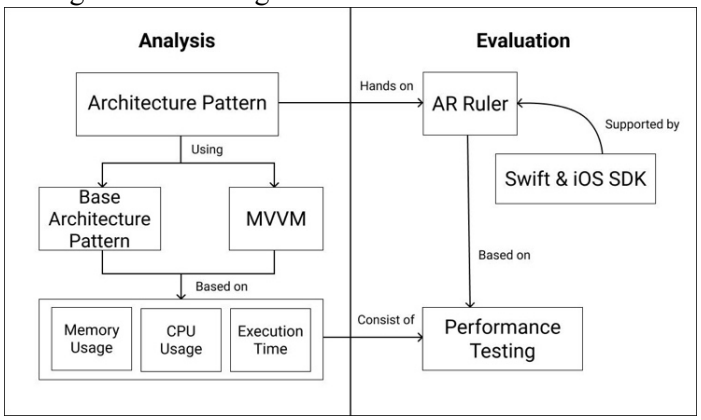

Fig. 1. Research Stages.

1) Analysis stage

The analysis stage aims to analyze how much the performance efficiency improvement from the MVVM architecture pattern. The analysis used data measurement (commonly called profiling) from the Base architecture pattern and the MVVM architecture pattern. Measurement of data architecture pattern based on: (1) CPU usage in percent, (2) memory usage (MB), and (3) execution time (ms). This measurement was obtained using tools Xcode Debugger and Instruments.

CPU is the brain of a device that processes all data and code to produce an application. The number of processors working on a device is called CPU usage [3]. memory is a place to store a series of data to be processed. memory usage displays the amount of memory used in the application [11]. The executed data and instructions will be calculated based on a specific time unit known as execution time [12].

The analysis was carried out by taking notes of any changes and additions made to the application. Then the comparison of the results of metric measurements between MVVM and Base Architecture is carried out. Architectures that have better performance efficiency are:

a) Less CPU allocation on applications per millisecond.

b) Less memory allocation in millisecond applications.

c) Less average execution time on millisecond applications.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

2) Evaluation stage

The evaluation is the stage to identify the information that has been collected at the analysis stage. The results obtained are compared with similar studies to ensure that the MVVM architecture pattern can be used to improve performance on iOS-based mobile applications. The result of evaluation stage can be used as material for consideration in choosing the right architecture pattern in the development of mobile applications.

B. Performance Testing

Performance testing is carried out by going through several iterations of testing. To avoid inconsistent results from application measurements, the MVVM pattern and Base architecture pattern was tested for performance with the following test scenarios:

a) Clear cache every time testing.

b) The camera is pointing at the same object.

c) Light testing: Idle application.

d) Moderate testing: An interaction occurred in the application

e) Heavy testing: There is interaction in the application; other applications are running in the background; low light on the camera object.

C. Application Selection

The case study used in this research is a mobile application based on augmented reality AR Ruler. This case study proves the usefulness of research outcomes and analysis of the impact of architecture patterns on the quality of iOS-based software. AR Ruler is an iOS-based mobile application used to measure the length of an object by utilizing Augmented Reality technology. Augmented reality is a technology that combines digital objects with the real world through a device, including a smartphone [13]. This technology will project virtual objects in real-time. Smartphones will display digital information according to what the user sees. Smartphones have a Global Positioning System service that locates the device in a specific position. It then compares data from the camera to other image-based data to see what the camera sees. Then the system adds data and overlays the data. The data is displayed by paying attention to the angle, height etc., to suit the user's wishes, so the application of augmented reality will provide more information than the actual conditions. AR Ruler is built on the ARKit framework. ARKit is an Augmented Reality framework created by Apple to integrate iOS device cameras and motion gestures. This framework requires setting the privacy controls related to the camera access rights of the application. This framework is available for iOS 11.0+ devices with a minimum System on Chip A9 Bionic [11].

D. Device and Tools

Xcode is an Integrated Development Environment for macOS, containing software development kit for native iOS development. Xcode supports C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, and Swift languages. Swift is a common language used for iOS, Mac, and Apple TV development. Xcode features a debug navigator to display CPU, memory, disk, and network usage. This feature covers the application whereas running on a simulator or actual device. Fig. 2. show graphs of the percentages used and the time the application ran tests.

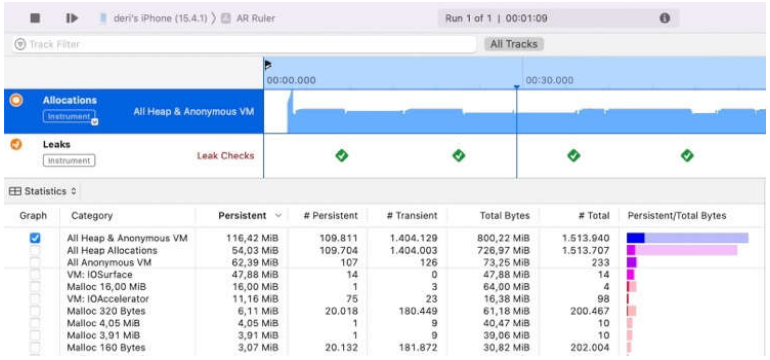

Fig. 2. Instruments for application analysis.

Feature debug navigation has a specific app to record when application testing. The instrument has many features related to application analysis. Fig. 2. shows a menu of the instrument, namely the activity monitor, leaks, time profiler, and others. The device test used in this research is the iPhone 8. with specifications OS version iOS 15,4,1 display phone 4,7 inches (diagonal), SoC Apple 11 Bionic 2.39 GHz, and 2 Gb RAM.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

III. RESULT AND DISCUSSION

A. Implementation Results

At this stage, it explains the MVVM implementation result. Implementing the MVVM architecture pattern in iOS App required a data binding mechanism, which aims to convey changes in the model into the View through the ViewModel. Realize that this mechanism can be implemented by adding a third-party library like Combine for applications built using SwiftUI and RxSwift for applications built using UIKit. SwiftUI or UIKit provides the required infrastructure for building an iOS app. It provides the window and views architecture for implementing an interface, event-handling, and other things to manage interactions in the system app. AR Ruler was built using UIKit so implementing RxSwift will result in more relevant measurement data because both apps are built with the same infrastructure.

The MVVM architecture pattern works very well with reactive programming because the reactive programming concept is oriented to the flow and changes of data, so changes in the model will affect the View through the ViewModel which can be seen in Fig. 3. When the data in the model changes, Model performs change callbacks to ViewModel, then ViewModel sends and receives data from Model, and ViewModel performs change callback to View. Because View performs observe data changes from ViewModel, View will automatically receive the latest data from ViewModel. Apart from reactive programming, this mechanism can be implemented with Key-Value Observing [14][15]. Key-Value observing it's useful for communicating changes between Logically, Models, and Views.

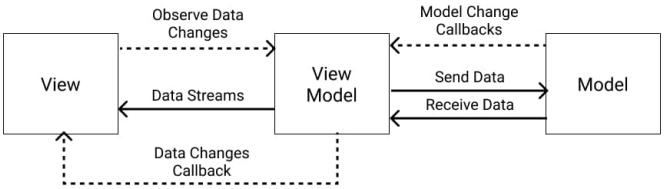

Fig. 3. MVVM Implementation Diagram.

B. Performance Results

At this stage, it explains the performance results of each performance metric such as CPU usage, Memory usage, and execution time before and after implementing the architecture pattern MVVM. Each performance metric was calculated with different units. CPU usage using percent, Memory usage using MB, and Execution time using ms. Architecture pattern with better performance is shown by reduced CPU usage, memory usage, and execution time required to execute programs.

1) CPU Usage

The measurement of CPU usage in the AR Ruler application was aimed in Fig. 4. In Test scenario 1 MVVM pattern reduces CPU usage from 71% to 69%, Test scenario 2 reduces CPU usage from 71.2% to 69.8%, and Test scenario 3 reduces CPU usage from 76% to 74.4%. The MVVM is always lower than the Base architecture pattern in each test scenario, with an average difference of 1.67%. This is because MVVM implements the RxSwift library, which allows each component to run separately. The data binding mechanism in MVVM also allows for automatic View updates when data changes. So implementing MVVM has a positive impact on performance because it reduces CPU usage.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

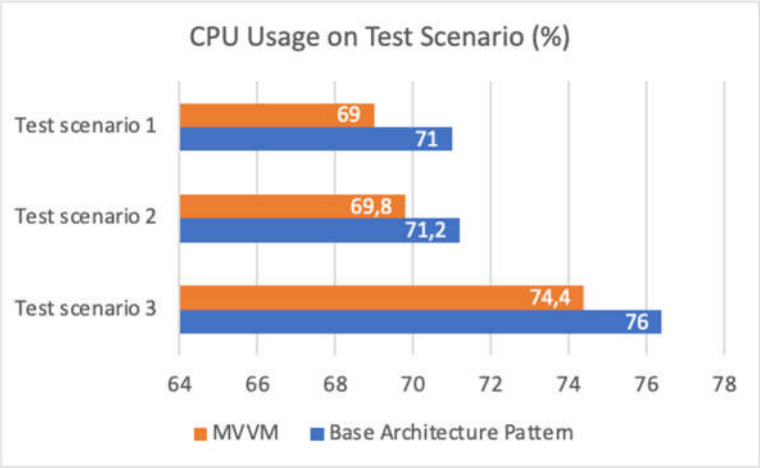

Fig. 4. CPU usage based on Test scenario.

2) Memory Usage

Fig. 5 shows the memory usage in the AR Ruler application. In Test scenario 1 MVVM pattern increases Memory usage from 126.82 MB to 129.9 MB, Test scenario 2 increases from 128.17 MB to 130.17 MB, and Test scenario 3 increases from 129.9 MB to 132.63 MB. Based on these results, The MVVM is higher than the Base architecture pattern in each test scenario, with an average difference of 2.73 MB. This is because additional libraries cause the code size to be larger and the data processed to be more. So, implementing MVVM has a negative on performance because it increases the memory usage.

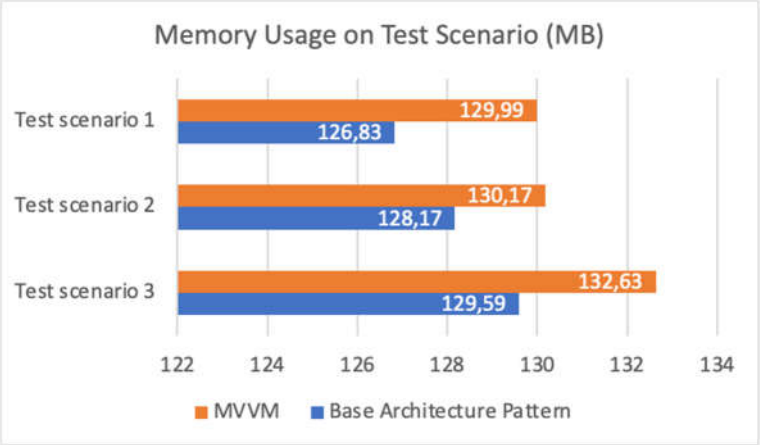

Fig. 5. Memory usage based on Test scenario.

3) Execution Time

The execution time measurements on iOS devices are shown in the Fig. 6. In Test scenario 1 the MVVM pattern decreases execution time from 142.31 to 115.23, Test scenario 2 decreases execution time from 147.38 ms to 115.41 ms, and Test scenario 3 decreases from 15.08 ms to 126.81 ms. The MVVM is faster than the Base architecture pattern with an average difference of 28.11 ms. This is because the declarations on the data used are in different components. So implementing MVVM has a positive on performance because it decreases the execution time.

JIPI (Jurnal Ilmiah Penelitian dan Pembelajaran Informatika)
Journal homepage: https://jurnal.stkippgritulungagung.ac.id/index.php/jipi
ISSN: 2540-8984
Vol. 8, No. 1, Maret 2023, Pp. 059-065

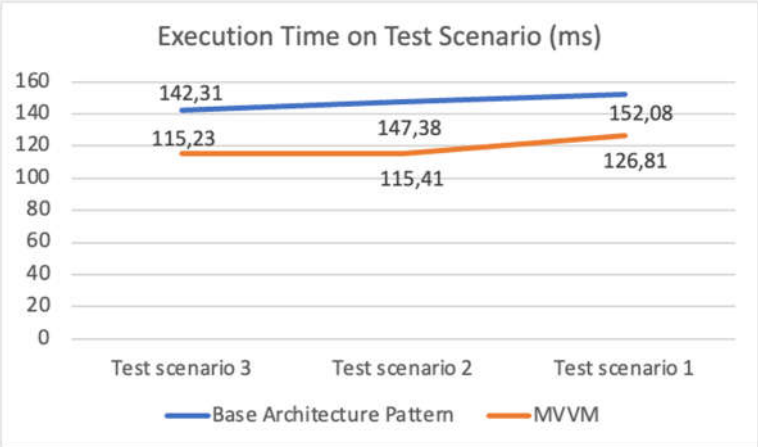

Fig. 6. Execution time based on Test scenario.

C. *Discussion*

Using different architecture patterns on iOS apps produces different outputs. In this case, it affects the efficiency performance aspects: (1) on CPU usage, MVVM is lower than the Base architecture pattern, with an average difference of 1.67%. (2) On Memory usage, MVVM is higher than Base architecture pattern, with an average difference of 2.73 MB. (3) in terms of execution time, the results show MVVM is faster than Base architecture pattern, with an average difference of 28.11 ms.

Based on the test scenario results, the MVVM architecture produces better performance values than the Base architecture pattern in terms of CPU usage and execution time. The MVVM architecture pattern is oriented to the flow and change of data, the mechanism works well by implementing reactive programming. The MVVM architecture pattern also implemented an additional RxSwift library that could improve application responsiveness by allowing each component to run separately, resulting in better CPU usage and execution time values. However, the use of third-party libraries resulted in higher memory usage. This is because a larger code size results in more data being processed.

The results are in accordance with previous research regarding the application of the MVVM architecture pattern on the android operating system [6][8]. The study stated the MVVM architecture pattern resulted in better performance efficiency, it was based on lower CPU usage and faster execution time. It can be stated that implementing the MVVM architecture pattern has a positive impact on improving the performance of mobile applications. Based on the results of the evaluation of the MVVM architecture, it is highly recommended to build iOS applications if CPU usage and execution time are the main concerns.

IV. CONCLUSION

Performance efficiency is an essential factor in improving the quality of mobile applications. Performance efficiency was analyzed based on CPU usage, memory usage, and execution time as a standard for evaluating performance. Performance efficiency can be achieved by implementing the MVVM architecture, which uses a data binding mechanism to link data or objects into the View. data binding on iOS can be implemented by implementing third-party libraries such as RxSwift. RxSwift library could improve application responsiveness by allowing each component to run separately, so resulting in better CPU usage and execution time values. The application of the MVVM pattern in iOS mobile applications is increasing the aspect of performance efficiency. In the case of AR Ruler, the MVVM architecture pattern provided increased CPU usage and execution Time. This increase was also due to the addition of third-party libraries. However, the presence of a third-party library increases the Memory Usage aspect. Based on the results, this can be used as a consideration for iOS developers in choosing an MVVM architecture pattern for better performance. Further research can be continued by analyzing the MVVM architecture pattern using other strategies or approaches, and more experiments should be carried out to study in depth.

REFERENCES

[1] Statista, “Mobile app downloads worldwide from 2018 to 2025, by store.” https://www.statista.com/statistics/1010716/apple-app-store-google-play-app-downloads-forecast (accessed Oct. 26, 2021).
[2] Statista, “Mobile operating systems’ market share worldwide from January 2012 to June 2021.” https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009 (accessed Oct. 26, 2021).
[3] Doug Sillars, High performance android apps: Improve ratings with speed, optimizations, and testing. Sebastopol, CA: O'Reilly, 2015.

[4] Rösler, F., Nitze, A., & Schmietendorf, "A. Towards a mobile application performance benchmark" In International Conference on Internet and Web Applications and Services. (2014). (Vol. 9, pp. 55-59).
[5] ISO/IEC 25010 (2012) 'System and Software Quality Requirements and Evaluation (SQuaRE) – System and Software Quality Models', Canadian Standards Association.
[6] T. Lou, "A comparison of Android Native App Architecture MVC, MVP and MVVM," M.S. thesis, Service Design and Engineering, Aalto University, Findland, 2016.
[7] L. Bass, P. Clements, and R. Kazman, Software Architecture in Practice Third Edition. 2015.
[8] B. Wisnuadhi, G. Munawar, and U. Wahyu, "Performance Comparison of Native Android Application on MVP and MVVM," In Proceedings of the International Seminar of Science and Applied Technology (ISSAT 2020), 2020.
[9] Appstore, "AR Ruler App data appstore." https://www.apple.com/id/search/ar-ruler?src=globalnav (accessed Nov. 16, 2021).
[10] A.-R. Baek, K. Lee, and H. Choi, "CPU and GPU parallel processing for mobile augmented reality," 2013 6th International Congress on Image and Signal Processing (CISP), 2013.
[11] Developer Apple, "Framework - ARKit." https://developer.apple.com/documentation/arkit/verifying_device_support_and_user_permission (accessed Feb. 21, 2022).
[12] Stone, H. S. A logic-in-memory computer. IEEE Transactions on Computers, C-19(1), 73–78. https://doi.org/10.1109/tc.1970.5008902, 1970.
[13] Berryman, D. R. (2012). Augmented reality: A Review. Medical Reference Services Quarterly, 31(2), 212–218. https://doi.org/10.1080/02763869.2012.670604.
[14] Ohrimenko, O., Costa, M., Fournet, C., Gkantsidis, C., Kohlweiss, M., & Sharma, D. "Observing and preventing leakage in mapreduce" In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, 2015.
[15] Garofalo, R, Building Enterprise Applications with Windows Presentation Foundation and the MVVM Model View ViewModel Pattern. 2011.