Frontiers in Business, Economics and Management
ISSN: 2766-824X | Vol. 14, No. 2, 2024

Design and Analysis of a Mobile Automation Testing Framework: Evidence and AI Enhancement from Chinese Internet Technological Companies

-- A Case Study

Jun Cui[1, *], Wangmei Chen[2], Qiang Wan[3], Zhongxin Gan[4], Zihao Ning[4]

[1] Solbridge International School of Business, Woosong University, Senior Digital Expert and Ph.D. Student, Korea
[2] Universiti Kebangsaan Malaysia (UKM), Computer Science (software technology), Master Student, Malaysia
[3] Foshan E-government Technology Co.,Ltd, Senior Engineer, China
[4] Solbridge International School of Business, Ph.D. Student, Korea

[*] Corresponding author: Jun Cui (Email: jcui228@student.solbridge.ac.kr)

Abstract: This present study mainly describes the implementation and ideas of the mobile automation framework, which supports iOS and Android mobile automation testing technology. This study mainly uses a combination of qualitative and documentary analysis methods, designs some technical architecture diagrams, and writes some open-source code to implement respectively. Meanwhile, The automation code is not made public due to mobile automation security and privacy concerns. The paper is mainly driven by the so file inside Android and the ADB command, while iOS uses some class libraries based on the iOS system. The research framework also uses the Appium framework for encapsulation and research and carries out secondary encapsulation and call based on the internal Appium framework. It is convenient for the internal quality team's automated testing staff to use and execute and improve the efficiency and speed of automated software testing, which is the contribution of this research. Beside, The limitation of this research is that the framework is based on the secondary development and encapsulation of Appium framework, so this study are inevitably some imperfections and bugs that need to be continuously improved and used. Moreover, these contribution results demonstrate that it can combine the businesses of different china internet enterprises to complete. Next, the paper discusses specific cases of mobile automation testing framework and artificial intelligence in mobile framework design and evaluates their effects and impacts. Finally, the paper summarizes the roles and challenges of mobile automation testing framework design and AI enhancement elements and looks ahead to the future.

Keywords: The qualitative and quantitative mixed analysis, Appium framework, IOS and Android framework, mobile automation framework and AI enhancement technologies.

1. Introduction

In recent years, some mobile automation testing and quality engineering has become very important for Internet enterprises, and the investment of enterprises in automated testing represents the final quality delivery of products. moreover, previous research papers and articles research Gap mainly introduced some TAM, Documentary analysis theories and did not introduce how to implement and design a mobile test framework, specially design ideas. The main contribution of this paper is to introduce and inspire the packaging and design of an automated test framework, as well as the design ideas for a mobile terminal automated test framework. For automated testing, we often talk about the automation pyramid, which is familiar in the field of testing, where UI stands for page-level system testing. service stands for service business test (interface test), and unit stands for unit test. The higher the pyramid, the more effort and work are required. The APPIUM framework is very popular for mobile automated testing. Thus, this paper mainly introduces the packaging and design of a mobile automated testing framework to solve the mobile automation capability of software enterprises and accelerate the product delivery and product quality of enterprises. Perhaps many people's impression of the automated testing framework is that it is a program that can automate testing. In fact, this is not comprehensive; the real automated testing framework cannot be a program; it is just a collection of ideas and methods. To put it simply, it is an architecture (Pyshkin, E., & Mozgovoy, M. (2018)). On the other hands, Everyone should know that the operating system is actually an architecture. You can understand it as a basic automated testing framework for a simple operating system. It defines several layers of architecture and defines how the layers communicate with each other. Through this architecture, we can expand our test objects (core functions and AI enhancement libraries), test libraries (link libraries), test cases (individual Windows processes), test cases (threads), and communicate between them by passing parameters (that is, equivalent to messaging in the mobile system). This paper also mainly refers to TAM and Documentary analysis theoretical model, combined with the theory of computer science and the theory of mobile testing technology to analyze and describe. For the contribution of this research, this study also put forward research questions about the design and implementation of mobile automation framework ideas and how to apply them to their own enterprise mobile automation design framework (Coppola, R., Ardito, L., & Torchiano, M. (2019, August)). From this research paper, the author put forward the following research questions.

RQ1. How do you combine the Android and iOS platforms

to design a universal mobile testing framework? How to use the open-source APPIUM framework to design the mobile testing framework?

RQ2. What is the design idea of the mobile testing framework? How do you design common components, Enhanced using AI technology and what do you need to know about designing and testing frameworks for iOS and Android?

Furthermore, this study designs our research automation framework using the mobile automated test framework model, which includes Android automation and iOS automation, common libraries and AI enhancement libraries, and third-party libraries. In the meanwhile, this study come up with the research model formula (1) to calculate mobile automation framework. Where TC_{MM}^{2} is testcase, the testcase include different modules and products. the F is framework design. and λ is firm size and automation team size. Where AF indicates the number of android test framework, that i is android automation functions, and the IF indicates the IOS Automation testing functions, C is Common functions of mobile automation framework. At first, this study will design the android and IOS automation testing framework, and then, this study will study design the final mobile automation framework version that consider the influence China technical firm's mobile automation testing. in addition, some parameters of the formula variables are explained as follows: Where $\sum_{k=1}^{n} C_k$ represents the framework's generic library, which can encapsulate multiple different third-party libraries and AI enhancement libraries. E_n is error of mobile automation framework testing respectively.

$$F = TC_{MM}^{2} \cdot \left(\lambda \sum_{i=1}^{n} AF_i + \lambda \cdot \sum_{j=1}^{n} IF_j + \sum_{k=1}^{n} C_k \right) + E_n (1)$$

Additionally, according to the research model described in Formula 1 above, we designed the following structure diagram for an automated mobile test framework, which contains many levels, each of which is realized by decoupling and layering. The user layer is mainly responsible for the realization and implementation of user test cases, and the development layer includes the integration of mainstream unit test frameworks (Gu, R., & Rojas, J. M. (2023, September)). The following is the business layer, which uses the idea of design pattern and the page model to encapsulate and implement the different elements. Click the Android or iOS app to obtain the page elements of the app for event testing. Then there is the service layer, which encapsulates the form and data. The driver of various mobile terminals and page browsers is encapsulated and invoked(Pyshkin, E., & Mozgovoy, M. (2018)). In this way, the architectural design of our mobile test framework is realized. And then the mobile framework automation architecture and descriptions as shown in Figure 1 and Table 1.

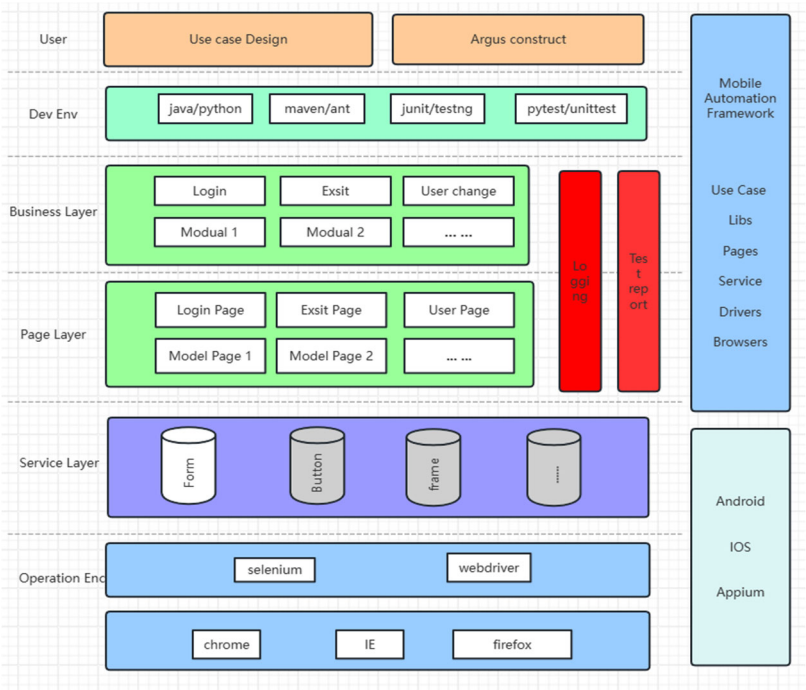

Figure 1. Mobile automation test framework universal design architecture framework.

Table 1. The Comparison of the number of different mobile and web-driven test version support summaries.

Function	Android	IOS	Web driver
Cross-Platform Compatibility	Supported	Not Supported	Supported
Popular Programming Languages Support	Supported	Supported	Supported
Free and Open Source	Supported	Not Supported	Supported
Compatibility with Testing Frameworks	Supported	Supported	Supported
Robust and Extensible Ecosystem	Supported%	Supported	Supported
Robust and Extensible Ecosystem	Supported%	Supported	Supported

From table 1. Based on the support features in the table above, enterprises can choose their own open-source

framework that needs to be encapsulated and integrated to implement a mobile automated testing framework.

2. Literature Review

2.1. Mobile automation framework

2.1.1. Mobile automation testing

Additionally, some of Mobile apps are an essential part of modern life, and more and more companies are beginning to invest significant time and resources in developing and maintaining mobile apps. As applications grow and become more complex, manual testing is no longer sufficient for quality assurance (Gu, R., & Rojas, J. M. (2023, September)). As a result, mobile automated testing is becoming an essential part of the test team. In this article, this study will discuss how to plan and design mobile automated testing. To design a mobile test framework, first of all, we need to determine the test scope and target. The goals should be specific and clear so that the test team can design the appropriate test cases. The scope of the test should cover all major functions and scenarios of the application. At the same time, different versions and platforms of the application need to be considered to ensure that the test covers all relevant devices and operating systems. Second, you need to choose the right automated testing tool for your company and project. Common mobile test automation tools include Appium, Calabash, and Selendroid. Choosing the right tool takes into account several aspects, such as the skill level of the test team, the characteristics of the application, the device, and the operating system. Next, you need to design automated test cases and AI testing functions. Test cases should be able to cover all functions and scenarios of the application, and test time should be minimized. When designing test cases, different versions and platforms of the application need to be considered to ensure that the test covers all relevant devices and operating systems. The test cases should be repeatable and maintainable so that the test team can re-run the tests after the application is updated (Gu, R., & Rojas, J. M. (2023, September)). Moreover, There is programming platform language compatibility, platform compatibility, mobile system version compatibility, and so on. After the final discussion and analysis, the APPIUM open-source framework is more suitable for enterprise packaging and secondary development. as previously described, There is little introduction to the design of the mobile terminal automated test framework, and there are no real design or implementation cases for reference (Berihun, N. G., Dongmo, C., & Van der Poll, J. A. (2023).). This study fills the gap in this research and analyzes and discusses different dimensions of design architecture, which will be helpful to the quality research and development of enterprises.

Indeed, This paper uses qualitative and documentary description theory analysis methods to study. Besides, This paper is arranged as follows; Section 2 presents the literature review ,research questions, Section3 presents the theoretical basis and research model , research description, variable statements. Section 4 presents the method and content analysis. Finally, Section 5 presents the discussion, limitation, and conclusions.

2.2. Android automation Implement

Indeed, For Google's Android automation framework testing, we all understand that the android automated testing framework is mainly based on the Google Android ecosystem. To achieve this requirements, the APPIUM open-source framework is also supported, so we have open source Appium packaging to complete. Android's UI Automator is an interface testing framework for cross-application functional interface testing across systems and installed applications. With the UI Automator API, you can interact with visible elements on The firm's mobile device (regardless of the activity in focus), enabling you to do things like open the Settings menu or app launcher on your test device. Tests can find interface components using convenient descriptors (for example, text displayed in the corresponding component or a description of its contents). The following is an mobile automation implementation architecture diagram for android mobile testing (Cruz, L., & Abreu, R. (2018, May)).

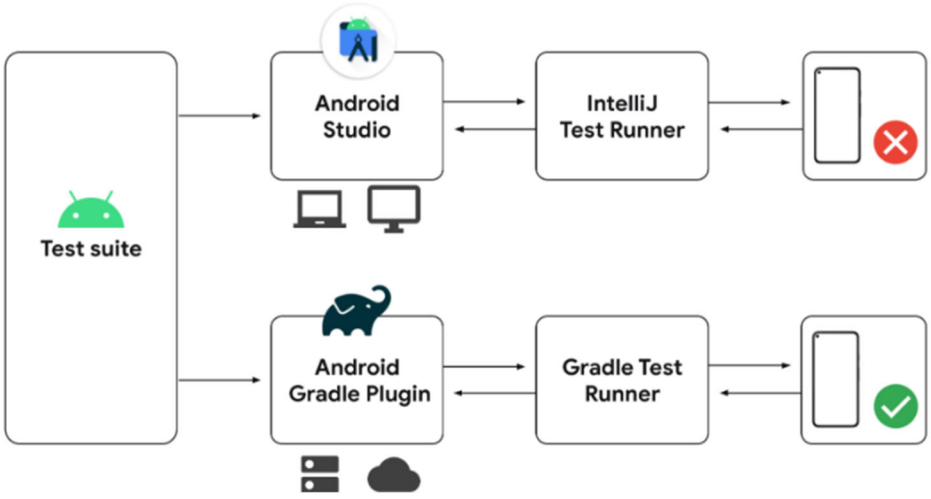

Figure 2. Google Android automation testing framework Implementation and Design.

Furthermore, from the above architecture figure 2 diagram, we can see that Android automated testing is generally completed based on the Android Studio and Grade Compilation Framework. In fact, the APPIUM open-source framework is also supported, and it also locates the elements of an Android app. Then find the element to automate the operation and complete the automated test.

2.3. IOS automation impalement

Furthermore, Mobile IOS test framework design is more

complex; consider some compatibility and how to install an automated test driver app on different iPhone devices. After the test is completed, the test results should be entered into the test report and log. For iOS automated testing, the APPIUM framework is very well supported. For white box testing, testers can directly contact the source code of the app to be tested (Cruz, L., & Abreu, R. (2018, May)). White-box testing is more of a unit test, where the tester analyzes various possible inputs for each unit and then tests its output. Alternatively,The test code for white box testing is usually written by iOS developers. Black box testing. During black box testing, the tester does not need to touch the source code. The correctness of its behavior and UI is verified at the app level, and the black box test is completed by the iOS test. iOS testing usually only has the following two levels: Unit testing to ensure that each class can work properly. UI testing, also known as integration testing, ensures that each business can work properly from the perspective of the business layer.

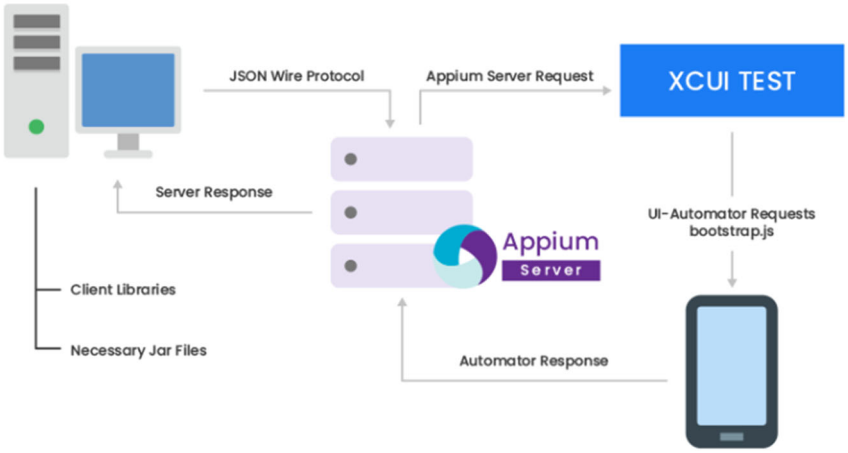

Figure 3. Mobile IOS automation test implementation framework.

Consequently, Through the above principle of the IOS automation testing framework, we see that IOS automation is actually required to have the support of APPIUM SERVICE, and after starting the service, to request the IOS app application, obtain permissions, and operate the simulated mouse click event to achieve this. On the other hand, For the element location of iOS automation, enterprise engineers can use the inspector tool of iOS Appium to locate app elements on mobile iOS. Inspector of http://localhost:8100/inspector, the inspector is used to check the UI layer, making it convenient to write the test script. From figure 4 how that this is a elements picture demonstration of positioning.

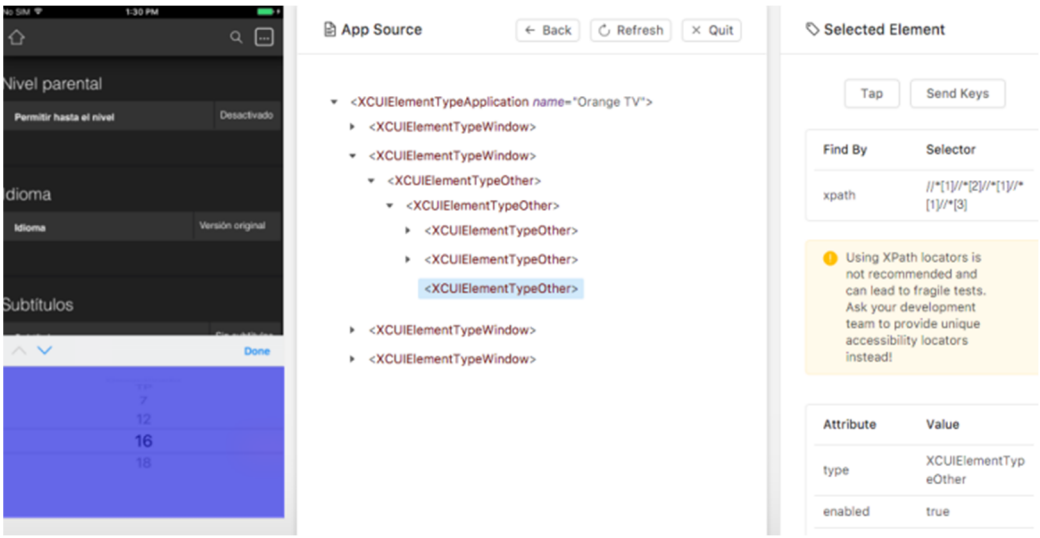

Figure 4. Some Mobile App automation elements design.

2.4. Design of common class library

Furthermore, for mobile testing general-class libraries and AI enhancement libraries, we generally use some third-party libraries for packaging and research and development. For the Google Android ecosystem, Android's own test framework is the basis for many test frameworks, and you can load the components under test in the same process. With many rich high-level packages, users can use other frameworks based on instrumentation to avoid too much secondary development. However, instrumentation does not support cross-application, causing frameworks based on instrumentation to inherit this shortcoming. For iOS automation framework's UIautomators libraries, it is generally used in the design of frameworks (Thant, K., Khaung, T., & Ind, H. H. I. (2023)). Now, the Android automation testing framework 's Instrumentation Test Framework is an automated testing tool for Android application interfaces launched by Android, which is an ideal tool for automated functional regression testing for APK packages. And then Click, find, slide, long press, and other

operations event can be performed according to text, coordinates, and control ID to achieve manual operation logic consistent with humans and operate according to specified commands after Python coding according to test cases, check the expected results, and test. Finally, the use-case scripts are executed through unit tests to generate corresponding reports. Consequently, The IOS automation part of this paper mainly uses Appium, an open-source cross-platform automated testing framework that supports IOS, Android, and FirefoxOS platforms and is very popular in the testing field. Mobile applications can be tested without recompiling or adjusting apps. The companies technical team can give test team's test code access to back-end APIs and databases. For the general part, such as file processing, database processing, and caching technology, OCR technology can be designed into the general framework to achieve.

3. Results and Framework Design

Additionally, Through the above analysis and description, for the technical characteristics of different platforms (IOS and Android) and the characteristics of the APPIUM open-source framework, this study also use qualitative and quantitative framework design analysis methods. After all, this study need to design our own mobile terminal test framework according to the enterprise's own business, and this framework should be designed as broadly as possible.From previously published studies, the architecture diagrams in this study are all the framework architecture design models actually used by enterprises. Moreover, this paper can design the following mobile universal test framework. The internal components of the framework are as follows:

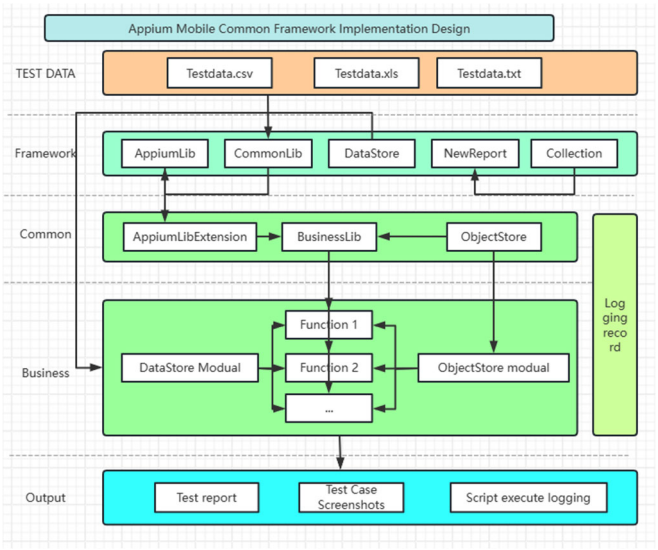

Figure 5. The final mobile automation design framework structure diagram.

Furthermore, from the figure 5, according to the above analysis and description, our final mobile automation design framework structure diagram, as can be seen from this android testing architecture diagram, is divided into a business layer, a functional layer, a common library, and the bottom part, as well as the output part. The output includes test reports and logs. The top layer is the part of test cases. Automation test engineers can use the framework to complete their own business automation test code and scripts. On the other hand, Previous research on mobile automation test framework design emphasized how to implement the framework and how to use it, but there was no detailed internal framework design how to plan the components, how to design the generated report and how to design the mobile test automatic framework at different levels. The above architecture diagram detailed the internal structure of the framework, which enterprises can refer to and use.

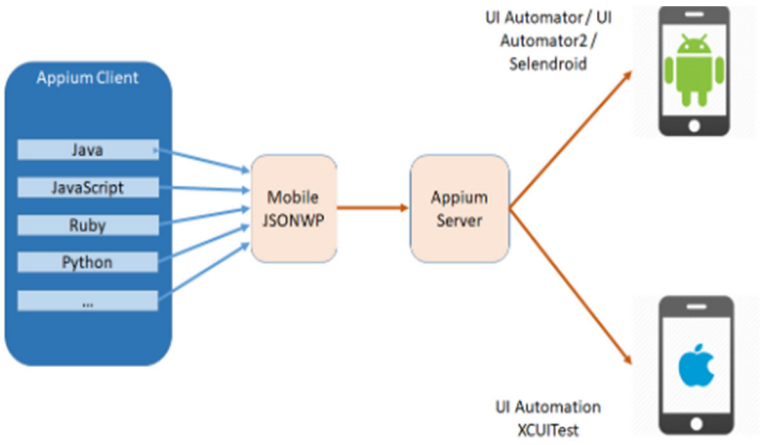

Figure 6. Appium Open-source framework design.

From the figure 6, this study can see that the Appium open-source framework structure diagram above, it shows that Appium is driven by Appium Server to test the automation of iOS and Android platforms. This study also designed the automation framework to be compatible with both platforms. Likewise, it also supports many programming languages and scripts, such as Java, Python, JavaScript, Ruby, etc.

4. Discussion and Suggestions

Indeed, compared to prior research methods, This research is more about the design of mobile automation frameworks.For the design of a mobile terminal test framework, first start with the product design. The first step in product design is "selection." Design from scratch and start from the end; from requirements analysis to framework selection, we need to think about how to meet the subsequent scalability. How to make our architecture more logical and become a standardized container interface is the first priority for a product manager to consider (Septian, I., & Alianto, R. S. (2018)). Students who have studied products should know the five-layer structure of user experience elements, from the structure layer to the presentation layer, which describes the process of demand to product design, from abstract to concrete, layer by layer. At the structural level, we need to figure out the "connections" that connect individual pages together. Where do users come from and where do they go? This requires us to consider the framework of a product, that is, navigation. Navigation is often a tool that leads users to where they want to go. The bottom navigation should be used to get to several top pages of similar importance. These pages require direct access from anywhere in the app. In the framework layer, we have to do more detailed thinking and discussion for the specific layout and interaction of individual pages. Finally, at the presentation level, UI designers can enjoy packaging products and let users see a beautiful visual experience. This paper takes China's WeChat software app as an example to analyze Thus, here are some UI automation test different styles for reference only. The mobile testing framework should be designed with touch and click event functions in mind.

Furthermore, Mobile automation framework terminal list navigation is the entry or content in accordance with the style of the list displayed on the page. In turn, list navigation is more suitable for content products as the main navigation or general products as auxiliary navigation. The navigation structure is simple, clear, easy to understand, calm, and efficient, and it can help users quickly locate the corresponding content (Vadan, A. M., & Miclea, L. C. (2023)). There are two types of tabular navigation: directly used as primary navigation or as secondary navigation to display content at a secondary or deeper level. In addition, the drawer navigation of the mobile app means that the menu is hidden on the current page, and the menu is pulled out like a drawer after clicking the entrance. The drawer navigation is more suitable for products with outstanding core functions and relatively single products. When your product information level has a lot of pages and content, it is difficult to display all the content on a screen. You must first think of designing a bottom or top tab navigation, but too much navigation is undoubtedly bloated, and it is difficult for users to click.

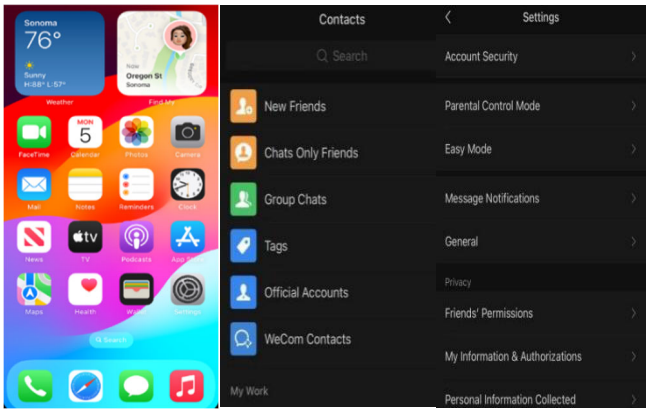

Figure 7. Some Mobile App automation design styles.

Additionally, According to figure 7, the mobile design result show that Mobile automatic test framework design needs mobile product UI to complete. Meanwhile, Mobile product navigation design is not the best; only the most suitable is needed. Enterprises need to choose the most suitable navigation design according to their product. All navigation design needs to highlight the core point of the product as far as possible to achieve a flat task path. Of course, in addition to the existing design patterns, there will be more updated navigation design and interactive experiences in the future, or there will be more comprehensive navigation models, so that product managers can find more free and reasonable framework patterns at the beginning of product design (Septian, I., & Alianto, R. S. (2018)).

Additionally, there are several discussions and research suggestions for mobile automation frameworks of China technical firms. The enterprise "automated test framework" is for testers. If the firms really want to scale automated testing, then you need to treat test engineers as your users. You can't expect them to have the patience to write test scripts or to have a good grasp of these ideas like you do. You want to treat them as if they don't know anything, so your framework must be the embodiment of "simplicity of “everything"—simplicity of operation, simplicity of maintenance, simplicity of extension. Secondly, the design of the mobile automation framework that needs to do an automated test framework is mainly based on hierarchical consideration rather than simply the application of an idea; it is a collection of various ideas (Singh, S., Gadgil, R., & Chudgor, A. (2014)). For example, to do GUI automated testing, it is simply generally divided into three layers, which can run through the various ideas of automated testing, such as the idea of keywords in the object layer, the object library that can be marked in Excel tables for management, or the application of dynamic search to pass object recognition parameters (Vadan, A. M., & Miclea, L. C. (2023)). The tasks layer can encapsulate various methods to form a large library of methods, and data-driven ideas can be applied to each method. Again, the real automated testing framework is combined with the process, rather than simply relying on technology to achieve it. Technology is not very complex; the key is to have a deep grasp of its architecture and process, and this takes a long time, so do not expect to eat fat in one breath, only step by step according to the demand and the application of the demand-guiding ideology. Finally, it is the framework used for automated testing. By definition, an automated test framework is either a basic or reusable automated test module.

5. Conclusion

In summary, at present, many China technical enterprises have mobile automation test framework, for different Chinese enterprises, the UI design idea is not the same, need to be combined with their own business to complete the implementation, the future can be combined with BDD test to further improve the implementation, an open-source cross-platform mobile application test framework, AI technology can be used to test screen shots, gestures and actual functional code (Lovreto, G., Endo, A. T., Nardi, P., & Durelli, V. H. (2018, October)). It is free and supports the Cucumber language, all statements are defined in Java, Python, Ruby programming languages, and there is a large community of open-source support. all in all, A mobile automated test framework needs to be implemented according to the business of your own enterprise. The technical companies can refer to the introduction and design of this article to complete the implementation of your own enterprise automatic test framework. as well as the limitation is that code of the automated test framework cannot be shared due to the security and privacy of the enterprise. This paper uses the qualitative analysis method, and adds some quantitative software architecture design and formula to explain the design and implementation of mobile automation test framework (Thant, K., Khaung, T., & Ind, H. H. I. (2023)).On the other hand, For the contribution of this study paper, the author has the following thoughts on the design of a mobile terminal automated test framework for the reference of enterprises, and now this study will introduce these "automated test framework ideas" one by one according to author's personal understanding:

1. The mobile automation so-called molecularity idea is to split several different test points in a test case and encapsulate the test steps of a single point to form a module in mobile framework. For example: a test case to test a login program, including user name input, password input, and confirmation of the user login; Then end user can input the user's name, password, confirm login, and cancel login. These four operations are packaged in four different modules. When testing, you simply call its module. This way, when a module changes, firm only need to maintain that module individually, or user can combine different test cases according to different modules (Berihun, N. G., Dongmo, C., & Van der Poll, J. A. (2023)).

2. The mobile automation so-called test library idea is the sublimation of the modular idea, which creates library files for the test of the application (which can be APIs, DLLs layer, AI enhancement layer etc.). These library files are a collection of functions. Different from the modular idea, it expands the interface idea, that is, parameters can be passed through the interface, rather than a blocked module, which can be said to be an interactive module with a "door.".For example, in the above test case, just the user's name input, password input, confirm Login, and cancel Login packages are imported into a library. This library contains a function login, which receives two parameters, "user name and password." The input of different user names and passwords can be different test cases. The firm can also have another function called Cancle.

3. The mobile automation so-called data-driven ideas and opinions vary; many Chinese technologies firms feel that only relying on the use of the EXCLE table to read different data is a high-level parameterize; in fact, how to understand is not important; the key is that its ideas can be applied to your framework. My understanding is that variables remain unchanged, data drives results, and different data leads to different results. For the import of data, you can use many methods, such as the EXCLE table, XML (used in the Web), Cache (Redis), database (DB), CSV file, TXT file respectively, etc.

4.The mobile automation so-called keyword thought, this design idea, we have been thinking about, its relationship with object orientation, and the difference between interactive modular thought. Later personal understanding, in fact, keyword-driven is an object-oriented idea, such as QTP or RFT automation software; the object can be data or a keyword; the capture of the object; the firm can encapsulate its test object into a keyword (that is, you can encapsulate GUI elements into a keyword); so, China technical Firms can carry out a variety of operations on its key objects; different objects can drive different test flow and results. The mobile automation test framework theory in this paper can also be used as a model supplement to documentary analysis and TAM theory, as a part of academic research for reference. In additional, the future work on automated test frameworks and mobile test framework design may add some AI technology, ChatGPT technology, etc. These results suggest that the maturity and popularity of AI technology(Thant, K., Khaung, T., & Ind, H. H. I. (2023)), automated test and mobile automatic test frameworks should also add some AI functions and technologies to increase the automated test technology well.

In the future, with the continuous development and advancements in technology and application of mobile automated testing frameworks, they will become wider and deeper. Businesses need to respond. Actively leverage the advantages of big data and artificial intelligence, integrate innovation, and improve The level and effectiveness of human resource management and the realization of sustainable development in enterprises.

Acknowledgment

We thank all friends, managers, classmates, technical architect and professors. This work was supported in part by a grant from Solbridge International School of Business. Moreover, I hope that my technical paper journals can help more and more people understand computer software mobile automation testing, mobile automation framework design and mobile automation testing technologies, automation testing framework design, and android automation, IOS automation framework testing thinking.

ORCID

Cui Jun https://orcid.org/0009-0002-9693-9145

References

[1] Septian, I., & Alianto, R. S. (2018). Comparison analysis of android gui testing frameworks by using an experimental study. Procedia Computer Science, 135, 736-748.

[2] Singh, S., Gadgil, R., & Chudgor, A. (2014). Automated testing of mobile applications using scripting technique: A study on appium. International Journal of Current Engineering and Technology (IJCET), 4(5), 3627-3630.

[3] Hussain, A., Razak, H. A., & Mkpojiogu, E. O. (2017). The perceived usability of automated testing tools for mobile

applications. Journal of Engineering, Science and Technology (JESTEC), 12(4), 89-97.

[4] Lovreto, G., Endo, A. T., Nardi, P., & Durelli, V. H. (2018, October). Automated tests for mobile games: An experience report. In 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames) (pp. 48-488). IEEE.

[5] Coppola, R., Ardito, L., & Torchiano, M. (2019, August). Fragility of layout-based and visual GUI test scripts: an assessment study on a hybrid mobile application. In Proceedings of the 10th acm sigsoft international workshop on automating test case design, selection, and evaluation (pp. 28-34).

[6] Pyshkin, E., & Mozgovoy, M. (2018). So You Want to Build a Farm: An Approach to Resource and Time Consuming Testing of Mobile Applications. ICSEA 2018, 101.

[7] Anjum, H., Babar, M. I., Jehanzeb, M., Khan, M., Chaudhry, S., Sultana, S., ... & Bhatti, S. N. (2017). A comparative analysis of quality assurance of mobile applications using automated testing tools. International Journal of Advanced Computer Science and Applications, 8(7).

[8] E. E. Reber, R. L. Michell, and C. J. Carter, "Oxygen absorption in the Earth's atmosphere," Aerospace Corp., Los Angeles, CA, Tech. Rep. TR-0200 (420-46)-3, Nov. 1988.

[9] Cruz, L., & Abreu, R. (2018, May). Measuring the energy footprint of mobile testing frameworks. In Proceedings of the 40th International Conference on Software Engineering: Companion Proceeedings (pp. 400-401).

[10] Vadan, A. M., & Miclea, L. C. (2023). Software testing techniques for improving the quality of smart-home iot systems. Electronics, 12(6), 1337.

[11] Thant, K., Khaung, T., & Ind, H. H. I. (2023). The impact of manual and automatic testing on software testing efficiency and effectiveness. Indian journal of science and research, 3(3), 88-93.

[12] Berihun, N. G., Dongmo, C., & Van der Poll, J. A. (2023). The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review. Computers, 12(5), 97.

[13] Godboley, S., Dalei, D., Sadam, R., & Mohapatra, D. P. (2023, March). Agile GUI Testing by computing novel Mobile App Coverage Using Appium Tool. In Proceedings of the 38th ACM/SIGAPP Symposium on Applied Computing (pp. 1026-1029).

[14] Heid, K., Tefke, T., Heider, J., & Staudemeyer, R. C. (2022). Android Data Storage Locations and What App Developers Do with It from a Security and Privacy Perspective. In ICISSP (pp. 378-387).

[15] Gu, R., & Rojas, J. M. (2023, September). An Empirical Study on the Adoption of Scripted GUI Testing for Android Apps. In 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW) (pp. 179-182). IEEE