A Glance at Automated Mobile Application Testing: Overview, Challenges and the Future

Cheng Bao
Wenzhou-Kean University
1335784
1335784@wku.edu.cn

Xinxin Qiu
Wenzhou-Kean University
1335784
1335783@wku.edu.cn

Abstract—**Automated mobile application testing is a crucial aspect of software development lifecycle (SDLC), ensuring the quality and reliability of mobile applications. This paper provides an overview of the current state of automated mobile application testing, highlighting its significance in the software development lifecycle, as long as the challenges faced in this domain, including device fragmentation, varying operating systems, and the need for efficient test automation frameworks. Furthermore, we explore the future trends and advancements in automated mobile application testing, such as AI-driven testing solutions and cloud-based testing platforms. By addressing these challenges and embracing emerging technologies, we can enhance the efficiency and effectiveness of automated mobile application testing.**

Index Terms—**Automated testing, Mobile application, Cases study, automated testing tools**

I. Introduction

Over the past decades, the development of Information & Communication Technologies (ICT)—including 5G and fiber to the home (FTTH)—has driven rapid growth in both mobile devices and the mobile applications market. According to International Data Corporation (IDC), more than 1.2 billion mobile devices were in circulation in 2014 [1]. The Grand View Research's mobile market report indicated that the mobile application market reached USD 252.89 billion in 2023 and is projected to grow at a compound annual growth rate (CAGR) of 14.3% from 2024 to 2030, reaching USD 626.39 billion [2]. As Fig. 1 shows, mobile device operating systems have consolidated from multiple options like Symbian OS [3] and Windows Phone [4] to two main systems: Google Android [5] and Apple iOS [6]. Yet this reduction in operating system choices has not simplified mobile application development.

With the rapid growth of mobile devices and applications, the demand for high-quality mobile applications has increased significantly [7]. This surge has created a pressing need for effective mobile software development across all lifecycle phases to ensure application timeliness, reliability, and performance. Automated testing, as an integral part of the software development lifecycle, has emerged as a key solution to these challenges, offering faster and more efficient testing processes than manual approaches [8]. This paper examines the comprehensive landscape of automated mobile application testing, addressing both its significance and challenges. Drawing from previous research, we explore future trends in automated mobile testing, particularly the

integration of cutting-edge technologies such as AI and cloud computing, to outline the field's trajectory. Through this analysis, we aim to identify ways to enhance the efficiency and effectiveness of automated mobile application testing.
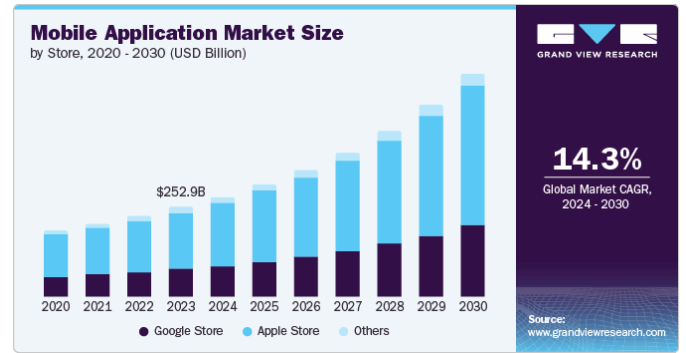

Fig. 1: The Mobile Application Market Size Insight (by store, 2020 - 2030), Grand View Research report, 2025

II. Background Knowledge

A. The definition of Mobile Testing Automation

To clarify the topic we discuss, the definition of mobile application testing must be specified. According to Wikipedia, mobile application testing is a process by which application software developed for handheld mobile devices is tested for its functionality, usability and consistency. [9] The types of mobile application testing divide into two kinds: manual testing and automated testing. Automated mobile application testing, also called mobile testing automation in some fields, involves using specialized tools and scripts to execute tests, validate an application's behavior, and ensure it meets the desired quality standards without human intervention. Because of the exclusion of the human factor, it accelerates the testing process and allows for continuous integration and continuous delivery (CI/CD) in the development life cycle. [10]

B. The Types of Mobile Application Testing

As Table I shows, The types of mobile application testing can be divided into two categories: functional testing and non-functional testing. Functional testing focuses on verifying the application's functionality, ensuring that it behaves as expected and meets the specified requirements. This includes unit testing, integration testing, system testing, and acceptance testing. Non-functional testing, on the other hand,

TABLE I: TYPES AND WAYS OF MOBILE APPLICATION TESTING

Categories	Sub-categories	Way of testing
Functional Testing	Unit Testing	tools & frameworks
	Intergration Testing	
	System Testing	
	Acceptance Testing	
Non-Functional Testing	Performance Testing	automated & manual testing
	Security Testing	
	Usability Testing	
	Compatibility Testing	

evaluates the application's performance, security, usability, and compatibility across different devices and platforms. This includes performance testing, security testing, usability testing, and compatibility testing. [10] When in the context of mobile application testing, functional testing is often performed using automated testing tools and frameworks, while non-functional testing may require a combination of automated and manual testing approaches. The choice of testing methods depends on the specific requirements of the application and the goals of the testing process.

a) *Functional Testing:* Functional testing ensures that an app's features and functionalities work as intended. It involves testing user interactions, APIs, databases, security, and client/server applications. In addition, automated functional testing verifies the end-to-end workflow of an app to confirm it meets the specified requirements. [11]

b) *Performance Testing:* Performance testing evaluates how well a mobile application performs under various conditions, including different network speeds, high user loads, and varied device configurations. It identifies performance bottlenecks and ensures the app can handle peak usage times without degrading user experience.

Performance testing is critical for mobile applications due to the vast array of devices and network conditions they must support, emphasizes Michael Facemire, Vice President and Principal Analyst at Forrester. His extensive research underlines the complexities of mobile performance testing. [12], [13], [14], [15]

c) *Usability Testing:* Usability testing identifies vulnerabilities within the mobile application to protect it from potential threats. Automated security tests check for weaknesses such as unauthorized access, data leaks, and other security flaws. [16]

d) *Security Testing:* Security testing identifies vulnerabilities within the mobile application to protect it from potential threats. Automated security tests check for weaknesses such as unauthorized access, data leaks, and other security flaws. [16]

e) *Compatibility Testing:* Compatibility testing ensures the app works across different devices, operating systems, and screen sizes. Automated compatibility tests verify that the application delivers a consistent user experience regardless of the device or platform. [17]

f) *Regression Testing:* Regression testing ensures that new code changes do not adversely affect the existing functionality of the application. Automated regression tests are crucial for maintaining the integrity of the app after updates or enhancements. [18]

III. AUTOMATED TESTING FRAMEWORKS & TOOLS

With the development of mobile testing automation going on, there are several frameworks and tools used widely in this field. The "framework" here refers to a combination of tools or software that provide the foundation for establishing automation testing by the users' own desire. Using the framework can accelerate the process of performing automation testing and prevent "re-inventing the wheel", which is considered a modern way of mobile testing automation in real-world practice. The following shows some of the most popular frameworks or tools:

A. Appium

Appium is a popular open-source framework used for automated mobile app testing. It allows developers to automate the testing of native or hybrid iOS and Android applications. Appium doesn't work alone. It runs the test cases using the WebDriver interface.

a) *Features:* Similar to Selenium, Appium is a versatile testing framework that supports multiple programming languages including Java, JavaScript, PHP, Ruby, Python, and C#. As a flexible cross-platform solution, it enables testers to create scripts that work across Windows, iOS, and Android platforms using a unified API. One of its key advantages is the ability to reuse source code between Android and iOS, significantly reducing development time and effort. Notably, Appium doesn't require modifications to the app's source code for automation purposes. The framework supports testing of native, hybrid, and mobile web applications, while also integrating seamlessly with popular cross-platform development frameworks like React Native, Xamarin, and Flutter. To enhance the testing experience, Appium provides useful built-in tools such as Appium Desktop and Appium Inspector, which help in inspecting app elements and generating test scripts.

b) *Pros & Cons:* The following are the pros of using Appium:
- Supports multiple programming languages (Java, JavaScript, Python, C#, etc.).
- Enables cross-platform testing for iOS and Android with a single API.
- Doesn't require modification of the app's source code.
- Supports native, hybrid, and mobile web applications.
- Works with frameworks like React Native, Xamarin, and Flutter.
- Offers built-in tools (Appium Desktop, Appium Inspector).

Along with the pros, there are also some cons of using Appium:
- Slower test execution compared to native frameworks.
- Requires a learning curve for beginners.
- Limited support for advanced gestures and animations.

B. NightwatchJS

NightwatchJS is a Node.js based framework that is developed and maintained by BrowserStack. Nightwatch uses Appium under the hood to achieve mobile application automation on virtual simulators & real devices. Nightwatch also takes care of the entire installation with just a single command.

a) *Features:* NightwatchJS offers several key features that make it a powerful tool for mobile testing automation. It provides a streamlined setup process with one-command installation of all necessary SDKs and libraries. The framework comes with native Nightwatch commands and integrates seamlessly with Appium Inspector. One of its standout features is support for parallel test execution, enabling tests to run simultaneously across multiple browsers and devices to significantly reduce testing time. Additionally, Nightwatch.js is versatile in its testing capabilities, supporting both real devices and emulators/simulators for web-based mobile application testing.

b) *Pros & Cons:* The following are the pros of using NightwatchJS:

- Easy setup with one command installation.
- Supports parallel test execution for faster test runs.
- Works with real devices and emulators/simulators.
- Integrates seamlessly with Appium and Appium Inspector.
- Uses JavaScript, making it developer-friendly.

Along with the pros, there are also some cons of using NightwatchJS:

- Limited community support compared to Appium and Espresso.
- Requires Node.js, which may not suit all development teams.
- Best suited for web-based mobile apps rather than native apps.

C. Calabash

Calabash is a mobile test automation framework that works with multiple languages. It supports Ruby, Java, Flex, and NET. Testers can use APIs to enable native applications that run on touchscreen devices. This framework has libraries that allow test scripts to interact programmatically with native and hybrid apps.

a) *Features:* Calabash enables writing and executing automated tests for mobile applications across both iOS and Android platforms. It allows test scripts to be reused across different platforms and devices, which significantly reduces maintenance effort. The framework supports both native and hybrid mobile applications, and offers compatibility with multiple programming languages including Ruby and .NET languages like C#. Additionally, its parallel testing capability helps decrease overall test execution time substantially.

b) *Pros & Cons:* The following are the pros of using Calabash:

- Supports multiple languages (Ruby, Java, .NET).
- Works for both iOS and Android platforms.
- Allows interaction with native and hybrid apps.
- Enables parallel test execution to speed up testing.

Along with the pros, there are also some cons of using Calabash:

- Requires additional setup for integration with CI/CD pipelines.
- Official support and development have slowed down.
- Not as widely adopted as Appium or Espresso.

D. XCUItest

XCUITest is Apple's native automation framework for testing iOS applications. Among mobile testing tools, this one is best known for testing iOS apps. Launched by Apple in 2015, the XCUITest framework is meant to create and run UI tests on iOS apps using Swift / Objective-C.

a) *Features:* XCUITest, built and maintained by Apple, ensures seamless compatibility with the latest iOS versions and features. Known for its fast execution, intuitive operation, and low flakiness, it allows developers to write test scripts in Swift or Objective-C while leveraging their existing language skills. The framework supports comprehensive testing on both real iOS devices and simulators, while providing APIs to interact with various device features like camera, GPS, and accelerometer. XCUITest also includes accessibility testing features to ensure app usability for individuals with disabilities. With its robust automation features and tight integration with XCode and XCTest, it stands as a reliable choice for iOS app automation testing.

b) *Pros & Cons:* The following are the pros of using XCUITest:

- Native framework for iOS, ensuring high performance.
- Faster execution compared to cross-platform tools.
- Deep integration with Xcode and Swift.
- Supports accessibility testing.

Along with the pros, there are also some cons of using XCUITest:

- Limited to iOS and cannot be used for Android testing.
- Requires knowledge of Swift/Objective-C.
- No built-in cross-platform testing support.

IV. CHALLENGE OF MOBILE TESTING AUTOMATION

As the techniques are developing rapidly in the field of mobile testing automation, the situation of testing remains challenging. Several significant issues, due to the fragmented nature of devices, operating systems, and testing environments, needed to be solved desperately. One major issue is the diversity of devices and platforms, such as iOS, Android, and others, each with varying hardware specifications, screen sizes, and OS versions. This fragmentation necessitates testing across numerous combinations to ensure compatibility, which is both time-consuming and resource-intensive. Additionally, automation scripts must be maintained and updated frequently to accommodate app updates and new features, requiring robust frameworks that can handle regression testing without introducing errors.

Another critical challenge is the reliance on both emulators and real devices. While emulators are cost-effective, they often fail to replicate real-world conditions, such as processor limitations, network variability, and user interactions, leading to gaps in test coverage. Furthermore, distributed

testing across different geographical locations introduces complexities, including time zone differences, network latency, and local carrier-specific issues, which can affect app performance and user experience. The tools mentioned above help mitigate some of these issues, but challenges like script maintenance, cross-platform compatibility, and accurate crash reporting persist, requiring continuous innovation in testing methodologies.

Despite the challenges listed above, some key problems will also be faced when implementing automated mobile testing, they are:

- Application downloadable: The application must be available for the specific platform, typically through an app store.
- Diversity in mobile platforms/OSes: The market includes several mobile operating systems, primarily Android, iOS, and Windows Phone. Each operating system has distinct limitations.
- Device availability: With an ever-growing variety of devices and operating system versions, accessing the right test devices is a persistent mobile application testing challenge. This becomes more complex when testing teams are geographically distributed.
- Mobile network operators: Over 400 mobile network operators worldwide use different standards, including CDMA, GSM, FOMA, and TD-SCDMA.
- Scripting: Device diversity complicates test script execution. Since devices vary in keystrokes, input methods, menu structures, and display properties, a single script rarely works across all devices.
- Test method: Mobile applications can be tested on either real devices or emulators. While emulators are convenient, they may miss issues that only real devices can detect. However, testing on real devices is costly and time-consuming due to the wide variety of devices available.
- Compatibility: Applications must maintain compatibility across various device properties, including different screen sizes and resolutions.
- Phone functionality: Applications must handle incoming calls appropriately during execution.
- Device diversity: Mobile devices feature various input methods (QWERTY, touch, standard) and different hardware capabilities.
- Load testing limitations: Engineers often lack sufficient hardware resources to create comprehensive user scenarios for performance testing of mobile applications.

V. About the Future

Starting from 2023, the advancements in AI and machine learning are enhancing almost every aspect of computer science. The shift towards AI-powered automated testing is not just a trend, but a fact. With the help of AI, the pioneers in mobile testing automation are trying a new efficient way to change the very cornerstone of automated testing, which is element identification. By introducing hybrid strategies that include context-aware locators, heuristic-based identi-

fication, and early implementations of adaptive and self-healing locators, to reduce the usage of traditional element identification methods, such as XPath, CSS selectors, and static attributes, to solve the key challenges above. [19]

Another breakthrough of the mobile testing automation will be the change of the testing device or environment. The definition of Test-as-a-Service (TaaS) was introduced back in 2015 [20], but because of the limitations of cloud resources and other technical barriers, there have been no significant breakthroughs recently. Nowadays, with the concept of cloud-native becoming more and more acceptable for developers, many applications, as well as mobile applications, are "born" with cloud-native. With cloud-based platforms like AWS Device Farm and Firebase Test Lab, tester can quickly and efficiently test their mobile application across multiple devices and operating systems without any local preparation. Using cloud-based platforms also makes it easier for testers to keep track of their results and analyze their performance.

Finally, in the foreseeable future, mobile app testing will continue to become more accessible and affordable. The cost of testing mobile applications has been steadily decreasing over the years, and this trend is likely to continue in the coming years. This will make it easier for developers and testers to access the tools and resources they need to ensure that their apps are of the highest quality.

VI. Conclusion

In summary, automated mobile application testing has become an indispensable part of the software development lifecycle, with various tools and frameworks available to address different testing needs. While established tools like Appium, Calabash, and XCUITest provide robust solutions for current testing requirements, significant challenges persist in the form of device fragmentation, platform diversity, and resource constraints. However, the future of mobile testing automation appears promising with the emergence of AI-powered testing solutions, cloud-based testing platforms, and more accessible testing tools.

The integration of artificial intelligence in element identification and the growing adoption of cloud-native testing approaches are set to revolutionize how mobile applications are tested. The shift towards AI-powered automated testing is particularly significant, as it addresses one of the fundamental challenges in mobile testing - element identification. With advanced techniques like context-aware locators and self-healing mechanisms, testing frameworks are becoming more resilient and adaptable.

Furthermore, the evolution of cloud-based testing platforms has opened new possibilities for scalable and efficient testing environments. The concept of Test-as-a-Service (TaaS) is gaining momentum, allowing organizations to leverage cloud resources for comprehensive testing across multiple devices and operating systems without significant infrastructure investments. This democratization of testing resources, combined with decreasing costs, is making quality assurance more accessible to developers and organizations of all sizes.

As the mobile application landscape continues to evolve, the testing methodologies and tools will need to adapt accordingly. The challenges of today are driving innovation in testing approaches, and the solutions being developed are not just addressing current limitations but are also paving the way for more sophisticated, efficient, and accessible testing practices in the future. This ongoing evolution in mobile application testing will ultimately contribute to higher-quality applications and better user experiences across the mobile ecosystem.

References

[1] D. Bernardo Silva, A. T. Endo, M. M. Eler, and V. H. S. Durelli, "An analysis of automated tests for mobile Android applications," in *2016 XLII Latin American Computing Conference (CLEI)*, Valparaíso, Chile: IEEE, Oct. 2016, pp. 1–9. doi: 10.1109/CLEI.2016.7833334.

[2] Grand View Research, "Mobile Application Market Size, Share & Growth Report 2030." [Online]. Available: https://www.grandviewresearch.com/industry-analysis/mobile-application-market/segmentation

[3] "Symbian OS," *Wikipedia*, May 2025, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Symbian&oldid=1288336212

[4] "Windows Phone," *Wikipedia*, Apr. 2025, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Windows_Phone&oldid=1286569954

[5] "Android (Operating System)," *Wikipedia*, May 2025, [Online]. Available: https://en.wikipedia.org/w/index.php?title=Android_(operating_system)&oldid=1288461632

[6] "iOS," *Wikipedia*, Apr. 2025, [Online]. Available: https://en.wikipedia.org/w/index.php?title=iOS&oldid=1286015929

[7] M. E. Joorabchi, A. Mesbah, and P. Kruchten, "Real Challenges in Mobile App Development," in *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement*, Baltimore, Maryland: IEEE, Oct. 2013, pp. 15–24. doi: 10.1109/ESEM.2013.9.

[8] M. Linares-Vasquez, K. Moran, and D. Poshyvanyk, "Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing," in *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Shanghai: IEEE, Sep. 2017, pp. 399–410. doi: 10.1109/ICSME.2017.27.

[9] "Mobile Application Testing," *Wikipedia*, Jan. 2025, Accessed: May 11, 2025. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Mobile_application_testing&oldid=1272006010

[10] "Mobile Test Automation: Everything You Need to Know." Accessed: May 11, 2025. [Online]. Available: https://www.tricentis.com/learn/mobile-test-automation-a-practical-introduction/

[11] "Functional Testing." Accessed: May 11, 2025. [Online]. Available: https://www.tricentis.com/learn/functional-testing/

[12] M. Facemire, "Mobile's Next Era: Not Apps, Not Websites — Experiences!." Accessed: May 11, 2025. [Online]. Available: https://www.forrester.com/blogs/mobiles-next-era-not-apps-not-websites-experiences/

[13] M. Facemire, "Development In The Enterprise: The Mobile Path Is Clear And Getting Easier!." Accessed: May 11, 2025. [Online]. Available: https://www.forrester.com/blogs/13-04-22-development_in_the_enterprise_the_mobile_path_is_clear_and_getting_easier/

[14] M. Facemire, "The Breathtaking Future of Software Development – It's Already Here!." Accessed: May 11, 2025. [Online]. Available: https://www.forrester.com/blogs/the-breathtaking-future-of-software-development-its-already-here/

[15] "Michael Facemire." Accessed: May 11, 2025. [Online]. Available: https://www.forrester.com/blogs/author/michael_facemire/

[16] "Software Testing: What It Is and How to Do It Right - Tricentis." Accessed: May 11, 2025. [Online]. Available: https://www.tricentis.com/learn/software-testing

[17] "Learn with Tricentis: What Is Software Testing?." Accessed: May 11, 2025. [Online]. Available: https://www.tricentis.com/learn/software-testing/

[18] "Regression Testing." Accessed: May 11, 2025. [Online]. Available: https://www.tricentis.com/learn/regression-testing/

[19] Vinaysimha Varma Yadavali, "Advancing Element Identification for Web and Mobile Automation," *International Journal of Advanced Research in Science, Communication and Technology*, pp. 45–56, Aug. 2024, doi: 10.48175/IJARSCT-6232.

[20] C. Tao, J. Gao, and B. Li, "Cloud-Based Infrastructure for Mobile Testing as a Service," in *2015 Third International Conference on Advanced Cloud and Big Data*, Yangzhou, Jiangsu, China: IEEE, Oct. 2015, pp. 133–140. doi: 10.1109/CBD.2015.30.