MobileTestPro: A Real-World Evaluation

Michelle Paredes[1], Danilo Martinez[1,3], Mireya Zapata[2(✉)], and Marcelo Rea[1,3]

[1] Grupo de Investigación en Modelos de Producción de Software (GrIMPSoft), Universidad de Las Fuerzas Armadas ESPE, Av. Rumiñahui, Sangolquí, Ecuador
{meparedes14,mdmartinez,amerea1}@espe.edu.ec
[2] Centro de Investigación en Mecatrónica y Sistemas Interactivos (MIST), Facultad de Ingenierías, Universidad Indoamérica, Quito, Ecuador
mireya.zapata@uti.ec
[3] Instituto Superior Tecnológico Wissen, Av. 10 de Agosto s/n y José Ma. Sánchez, 010107 Cuenca, Ecuador

Abstract. This paper presents the validation of the *MobileTestPro* framework, a systematic, dimension-based approach to assessing the quality of mobile applications. The validation was conducted through an empirical case study involving three application contexts: video streaming, communication, and social media, using YouTube, WhatsApp, and Instagram as units of analysis. The framework assesses key quality attributes, including performance, usability, reliability, and resource consumption, under controlled testing conditions. Specific test cases were designed to simulate real-world scenarios, including variations in video resolution, message exchange processes, and user interaction patterns. Quantitative data were collected and analyzed to compare application behavior across the evaluated dimensions. The results demonstrate that *MobileTestPro* provides a consistent and replicable mechanism for evaluating mobile application quality. Furthermore, the framework facilitates the identification of performance issues and usability limitations. These findings confirm its effectiveness as a structured mobile testing tool.

Keywords: Mobile Testing · Software Quality · MobileTestPro · Case Study · Android · Security Testing · Accessibility · Stress Testing

1 Introduction

The mobile telephony market has experienced exponential growth over the past three decades. Concurrently, the development of mobile applications and platforms has also shown significant evolution, although at a relatively slower pace of expansion. Since the launch of the first iPhone lineup, app development has undergone a complete transformation, making apps more user-friendly and gradually integrating them into everyday life, encompassing all kinds of services: communication, entertainment, and personal and professional productivity. This has brought with it the need to improve app quality in terms of functionality, security, energy consumption, usability, and accessibility.

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026, LNCS 16769, pp. 38–52, 2027.
https://doi.org/10.1007/978-3-032-30494-0_3

Apps must function in complex environments with a variety of devices, different operating system versions, variations in communication networks, and changes in system policies—what Martinez et al. [1, 2] refer to as "Mobile Illities".

In this context, ensuring the quality of mobile applications is challenging. It is not enough to verify that the application performs the expected actions correctly; it is also necessary to evaluate its behavior under these conditions and test its performance. As an alternative to improve application quality, MobileTestPro [3] emerged. This application evaluation framework consists of three blocks and covers six evaluation dimensions: performance, power consumption, security, accessibility, compatibility, usability, and stress.

MobileTestPro is a framework proposed by Romero-Collaguazo et al. [3] that aims to improve app quality by evaluating six dimensions (energy, usability, security, compatibility, stress, and accessibility). The framework was evaluated using a "toy app", which allowed for verification of its overall functionality and sequence of activities. However, no tests were conducted in real-world environments with widely used applications, where usage conditions are more demanding.

In this article, we present the validation of MobileTestPro in a real-world environment through a case study, applying the guidelines of Runeson et al. [4]. A holistic multicase design was implemented, as proposed by Yin [5]. Three distinct contexts were identified, each with its own unit of analysis (UA). The UAs represent three widely used applications: YouTube (video streaming), WhatsApp (messaging), and Instagram (social media). For data collection, a test case was created for each dimension of MobileTestPro and applied to each UA. The aim was to evaluate whether MobileTestPro allows for the systematic identification of problems and the comparison of results across contexts.

The remainder of the document is organized as follows: Sect. 2 provides background, including the context of MobileTestPro and relevant approaches to mobile testing. While, Sect. 3 describes the case study methodology, its planning, and the research questions. In Sect. 4 we detail the execution of the test case and the results for each unit of analysis. Section 5 discusses the most important findings and their implications. Finally, in Sect. 6, we present conclusions and future work.

2 Background

2.1 MobileTestPro

MobileTestPro is a framework proposed by Romero-Collaguazo [3] that aims to systematize app testing across various quality dimensions, thereby reducing the need for ad-hoc testing and improving the traceability of results. The framework is designed to provide a structure for defining test objectives, activities and evidence to evaluate those attributes which, in the mobile environment, determine the user experience.

MobileTestPro defines a set of activities and tools for each dimension, as shown in Fig. 1, so that testing is carried out optimally in each case and the results can be analysed with greater consistency. This structure facilitates comparison between applications and usage contexts using common criteria, while integrating technical evidence into a framework that supports identifying opportunities for improvement and decision-making during the maintenance and evolution of mobile applications.

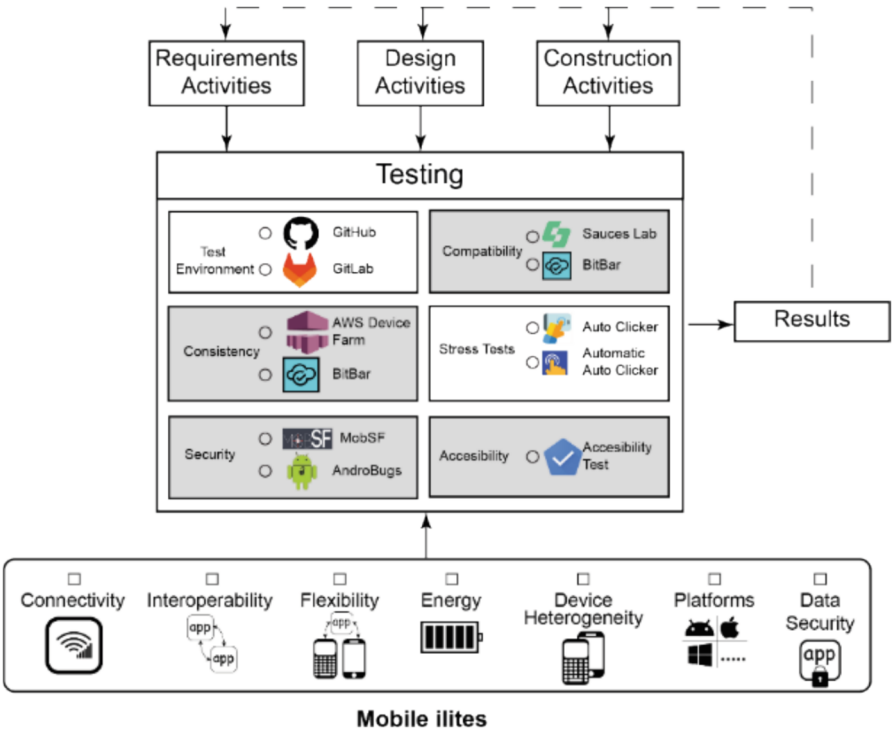

Fig. 1. MobileTestPro (from Romero-Collaguazo): Mobile testing framework to evaluate multiple quality dimensions and mobile-specific attributes.

To test the real effectiveness of MobileTestPro, it is relevant to observe its behavior with apps used daily and by a large number of users, since that is where they have the necessary conditions to obtain data that can confirm its effectiveness in identifying app errors and, therefore, contribute to the best quality of their products.

2.2 Software Testing

According to Myers et al. [6], software testing is a process, or a series of processes, designed to make sure a computer code. Software should be predictable and consistent, offering no surprises to users.

When you want to add some value to a program, you should test it. The reliability of the program means finding and removing errors. The testing process helps to identify errors.

According to Osterweil et al. [7], testing plays an important role in achieving and assessing the quality of a software product. Hence, when we repeat a test-find defects-fix cycle during development, we improve the quality of the product.

Friedman and Voas [8] describe software testing as a verification process for software assessment and improvement. The activities for software quality can be divided into two broad categories: static and dynamic analysis.

Static analysis suggests examining a number of documents, such as requirements documents, software models, design documents, and source code. Traditional static analysis includes code review, inspection, walkthrough, algorithm analysis and proof of correctness.

Dynamic analysis of a software application includes its execution to expose possible program failures. Furthermore, its behavioural and performance are also observed. Programs are executed with a set of input values and the outputs are observed. In testing process we observe some representative behaviors and reach a conclusion about the its quality.

A test case is a simply pair data, input data and outcome expected.

Testing Activities

In order to test a program, we need a set of sequence activities. Naik et al. [8] proposed the following steps.

- Identify an objective to be tested: Identify the objective to be tested. Define the purpose of the designed one or most test cases to ensure that the program supports the objective.
- Select inputs: Selection of test inputs based on the requirements or goal of the application.
- Compute the expected outcomes: Calculate or define the outcomes of the applications with the selected inputs.
- Set up the execution environment: Prepare the scenario to execute the program that includes setting up the run environment (hardware, software, and communication).
- Execute program: Run the programs with selected inputs and observe the outcomes of the programs.
- Analyze results: The final activity is to observe the outcomes and compare them with the expected results.

2.3 Mobile Testing

Mobile apps, like any other software system, require testing, but apps are different from traditional ones and require different and specialized testing techniques and processes [9]. The difference between traditional applications and mobile applications Martinez [1] et al. named Mobile Ilities.

Haller [10] presents device testing and real-world testing as a dual approach for testing applications. Device test consist in testing the same test case (app) in different mobile devices to ensure compatibility, performance, usability, etc. While testing in the wild approach is testing an app in its real usage context, on other hands, in a real environmental.

When we used Mobile Ilities and Haller's approach, we obtained the best conditions for applying MobileTestPro in our study, because we created a set of test cases based on Mobile Ilities to apply to MobileTestPro's dimensions. In the test, we apply two Haller approaches. Any case test was testing on some devices (physical or emulator) on a real-world application in its real environment through a case study.

3 Case Study

3.1 Case Study Design

To study the behavior of MobileTestPro in a real environment, a case study was proposed using the guidelines proposed by Runeson et al. [4] because it is the most appropriate method to study the behavior of a phenomenon in its real context, especially when the conditions of the environment directly influence the observed behavior, and it is not possible to control all variables.

The following research questions were posed for the study:

RQ1: How effectively can MobileTestPro identify performance, stability, security, usability, compatibility, and accessibility issues in applications?
RQ2: How do the dimensions of MobileTestPro relate to the identification of recurring failures and the comprehensive evaluation of mobile application quality?

For the study, the mobile applications that register the highest use according to the DataReportal [11] have been considered, being Instagram, YouTube, and WhatsApp, which in turn have determined that the study is carried out in 3 contexts (streaming, communication, and social networks).

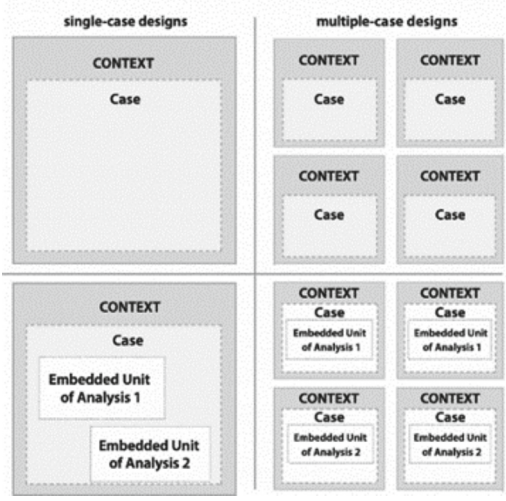

Fig. 2. Use Case Design (from Robert Yin): Types of case study designs proposed by Yim [5] according to the defined contexts and units of analysis.

The design adopted is a holistic, multi-case approach, as described by Yim et al. [5] as shown in Fig. 2. It is multi-case because three contexts are analyzed (video Streaming, Communication, and social media), and it is holistic because each context has its unit of analysis, see Fig. 3. This design allows for comparing the performance of MobileTestPro across different functional domains and observing how dimensions such as usability, compatibility, accessibility, security, performance, and stability interact in everyday scenarios.



Fig. 3. Holistic Multiple-case Design: Design composed of 3 contexts (video streaming, communication, social media), with a unit of analysis for each context (YouTube, WhatsApp, Instagram).

3.2 Data Collection

For data collection, a set of test cases was developed for each of the units of analysis. The test cases were planned based on the context and the MobileTestPro dimensions that best suited the nature of the application.

The planned test cases for the YouTube analytics unit are shown in Table 1.

Table 1. YouTube planned test cases

Dimension	Objective	Input	Outcome	Tool
Compatibility and Usability	Verify that YouTube is compatible and usable on different device models and Android versions	The YouTube APK runs on available devices and emulators	The app should be installed, open, and allow basic navigation without errors on the selected devices	BitBar
Performance and energy	Evaluate YouTube's behavior during continuous video playback in different qualities	5 and 10-minute video playback sessions in different qualities	The device must maintain stable operation, without unexpected shutdowns or appreciable performance degradation	AccuBattery
Stress	Simultaneous user loading on services exposed by the YouTube API	K6 scripts configured with different levels of concurrent users	The invoked services must maintain response times within acceptable parameters for the defined test scenario	k6

(*continued*)

Table 1. (*continued*)

Dimension	Objective	Input	Outcome	Tool
Security	Static analysis of the YouTube APK to identify potential security risks	YouTube APK loaded into the tool	Detect vulnerabilities classified as critical in the static analysis report	MobSF
Accessibility	Check compliance with basic accessibility criteria on the main YouTube screens	Home, search, and playback screens	Interactive elements and texts must comply with basic guidelines for contrast, size, and labeling	Accessibility Test/Accessibility Scanner

Meanwhile, for the WhatsApp analysis unit, the test cases shown in the Table 2.

Table 2. WhatsApp planned test cases

Dimension	Input	Outcome	Objective	Tool
Compatibility and Usability	Check WhatsApp compatibility and the correct display of the messaging interface on different Android devices	WhatsApp APK running on multiple devices	App must correctly display the main screens and allow access to chats without interface failures	BitBar
Performance and energy	Analyze WhatsApp's resource consumption during active messaging use	Sessions of 5 and 10 minutes, sending text messages, voice notes, and images	App should respond smoothly and the operating system should not exhibit any obvious crashes or delays	AccuBattery
Stress	Evaluate the API's behavior when the app is sending and querying messages under load	k6 scripts executing concurrent calls to selected endpoints	The system must support the defined load without exhibiting widespread failures in its responses	k6

(*continued*)

Table 2. (*continued*)

Dimension	Input	Outcome	Objective	Tool
Security	Review permissions, settings, and potential security weaknesses in the WhatsApp APK	WhatsApp APK analyzed in the tool	The report should not reveal clearly insecure configurations or practices at the level of permissions and data storage	MobSF
Accessibility	Evaluate the accessibility of WhatsApp's chat interface and main menus	Chat list screen and conversation view under analysis	Buttons, fields and interactive elements must be identifiable and accessible via the tool	Accessibility Test / Accessibility Scanner

Finally, for the Instagram analysis unit, the test cases shown in the Table 3.

Table 3. Instagram planned tests

Dimensión	Objective	Input	Outcome	Tool
Compatibility and Usability	Check Instagram's compatibility with dynamic layouts across different screen sizes and Android versions	The Instagram app runs on devices with different screen sizes and resolutions and versions of Android	The feed and main screens must load correctly and respond to basic user interactions	BitBar
Performance and energy	Evaluate Instagram's performance during continuous Reels viewing	5 and 10-minute Reels viewing sessions	Navigation should remain smooth, and the application should not crash or freeze during continuous use	AccuBattery
Stress	Validate the stability of the API associated with the Instagram feed and authentication under load scenarios	K6 scripts with concurrent users accessing content query endpoints	Services must maintain consistent availability and response times throughout the test execution	k6

(*continued*)

Table 3. (*continued*)

Dimensión	Objective	Input	Outcome	Tool
Security	Evaluate the Instagram APK to identify potential vulnerabilities in internal components	Instagram APK processed in the tool	High-impact vulnerabilities that compromise data integrity or confidentiality should not be registered	MobSF
Accessibility	Check accessibility in the feed and dynamic content sections, including Reels.	Feed and Reels screens analyzed with the accessibility tool	The interface components must be navigable, readable, and have appropriate labels for assistive technologies	Accessibility Test / Accessibility Scanner

3.3 Execution

For the tests, a series of free simulators was used, as well as a smartphone running Android 12, a Unisoc Tiger T606 processor, a 6.5-inch screen, 4 GB of RAM, and a 500 mAh battery. The phone was prepared exclusively for testing; therefore, factory settings were applied, and only the applications necessary for testing were installed.

The test cases were executed by applying the MobileTestPro dimensions for each usage context and selected unit of analysis. To ensure consistency and comparability between cases, each application was evaluated following the same execution logic: (i) definition of the use scenario corresponding to the context, (ii) execution of test cases by dimension using specific tools, and (iii) systematic recording of evidence for later consolidation.

The planned test cases were executed by dimensions, each associated with a mechanism for obtaining evidence: compatibility using BitBar; security using MobSF; performance under load using k6; energy consumption using AccuBattery; and accessibility using an accessibility scanner.

4 Results

The results obtained from running the planned test case are shown in this section.

4.1 Video Streaming

The planned test cases designed for this analysis unit was carried out in two parts. For the first part, a 5-minute video with a resolution of 1080 was used. For the second part,

another video with a higher resolution and a duration of 10 min was used. The results obtained for each dimension are shown in the Fig. 4.



Fig. 4. UA YouTube: Results obtained after performing the YouTube UA test cases in the planned dimensions.

Table 4 shows the details of the test cases execution for each dimension of MobileTestPro.

Table 4. Details of tests performed on YouTube

Dimension	Details
Compatibility (BitBar)	YouTube worked correctly on the Pixel 3a (Android 12). On the Pixel 9 Pro (Android 14), it was reported to be incompatible, and the error INSTALL_FAILED_MISSING_SPLIT was recorded
Security (MobSF):	The analysis reported a score of 52/100 (classification: medium risk-go) and 4 vulnerabilities: Cleartext Traffic, use of MD5, exported components and enableBackup = true
Stress (k6):	The test showed approximately 81% successful results and 19% failures, with connection termination being the most common failure
Energy (AccuBattery):	In the YT-02 scenario (720p, 10 min), an average power consumption of 592 mA was recorded
Accessibility	A total of 49 suggestions were identified, the most notable of which concerned opportunities for improvement in touchpoints and in the text or labels

4.2 Communications

The test cases carried out by WhatsApp's analytics team were conducted on two devices running different versions of the Android operating system. The test involved sending text messages, photos, videos, and voice notes in rapid succession over two time periods: the first lasting 5 min and the second 10 min. The results are shown in Fig. 5. Table 5 shows the details of the test results for this unit of analysis.



Fig. 5. UA: WhatsApp: Results obtained after performing the WhatsApp UA test cases in the planned dimensions.

Table 5. Details of tests performed on WhatsApp

Dimension	Details
Compatibility (BitBar)	WhatsApp was reported to be compatible on the Pixel 3a (Android 12) and the Nexus 5 (Android 7.1.1). In the latter case, CPU usage was recorded at approximately 80–150%
Security (MobSF):	The analysis yielded a score of 51/100 (classification: medium risk). The findings reported include: unencrypted traffic, user certificates, SHA1 + V1 and hard-coded secrets
Stress (k6):	The results showed approximately 37% correct answers and 62% errors, with connection errors being the most common
Energy (AccuBattery):	In scenario WA02 (10 min), an average current consumption of 691.2 mA was recorded
Accessibility	A total of 35 suggestions were recorded, with frequent improvements to touch targets and text/labels

4.3 Social Media

Finally, for Instagram, test cases were designed to assess compatibility across different devices and versions of Android. In addition, installation and execution were verified, and CPU and memory usage was monitored during startup on the BitBar platform. The results are shown in Fig. 6.

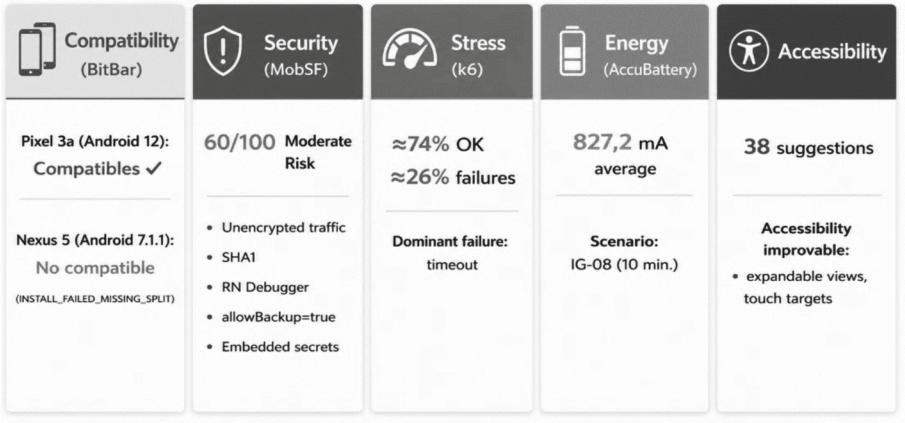

Fig. 6. UA: Instagram: Results obtained after performing the Instagram UA test cases in the planned dimensions.

Table 6 shows the details of the results obtained during the testing of this analysis unit.

Table 6. Details of tests performed on Instagram

Dimension	Details
Compatibility (BitBar)	It has been reported that Instagram is compatible with the Pixel 3a (Android 12). On the Nexus 5 (Android 7.1.1), it has been reported that it is not compatible, and the error INSTALL_FAILED_MISSING_SPLIT has been recorded.
Security (MobSF):	The analysis returned a score of 60/100 (classification: medium risk), with findings related to: unencrypted traffic, SHA1, the presence of RN Debugger, and the setting allowBackup = true
Stress (k6):	The results showed approximately 74% correct answers and 26% errors, with delay time being the most common error
Energy (AccuBattery):	In scenario IG-08 (10 min), an average current consumption of 827.2 mA was recorded
Accessibility	A total of 38 suggestions were received, with expandable views and touch targets standing out as areas for improvement

5 Discussion

The data obtained during the case study have enabled us to identify several faults in the performance of the evaluated apps, demonstrating the effectiveness of the framework. The findings suggest that the dimensions do not function as isolated assessments; rather, they tend to converge into consistent relationships that facilitate the interpretation of the

system's quality. Furthermore, a number of recurring faults were identified among the apps evaluated.

Security is a cross-cutting issue linked to the robustness and reliability of the system. In this area, it was found that the "allowBackup = true" vulnerability is present on both YouTube and Instagram. These apps require users to register an account in order to run. This vulnerability allows backup data to be stored on the device, which can then be stolen from it via some form of attack. Consequently, users are exposed to identity theft, data exposure, social engineering, scams and extortion, amongst other risks.

Furthermore, unencrypted traffic affects both WhatsApp and Instagram, as both apps use encryption protocols such as HTTPS/TLS to send messages, photos, videos, credentials and session tokens. The vulnerability can arise when apps fail to validate certificates correctly, use insecure TLS configurations, do not implement 'certificate pinning', or allow 'man-in-the-middle' attacks. The risks associated with incorrect encryption implementation include interception of traffic between the user and the server via an attack, which could lead to the disclosure of transmitted information, modification of requests, theft of authentication tokens, and session hijacking.

The results show a close relationship between compatibility and accessibility. Both aspects are reflected in user interaction with the interface and often manifest as similar issues when the environment changes (such as changes in resolution, pixel density or operating system version). In this regard, issues such as small touch targets or insufficient labels not only affect users with accessibility needs but also tend to become more critical when the application runs on different configurations. Therefore, the evidence supports the view that compatibility and accessibility share a pattern: when the interface is not optimised, the impact is amplified across multiple runtime environments.

In terms of efficiency and resource usage, a correlation was observed between stress and energy consumption, as both factors reflect the system's behaviour when resource demand increases. Under higher-load scenarios, increased activity (CPU and network) can result in a greater energy impact; therefore, a combined assessment makes it possible to determine not only whether the application responds, but also whether it does so whilst making efficient use of the device's resources.

6 Threats to Validity

The study operationalizes mobile application quality using the dimensions defined by MobileTestPro; however, the selected test cases and tools may not fully capture the complexity of real-world usage. In particular, stress testing was based on simulated API calls rather than actual user behavior, which may limit the realism of the observed system responses. The execution protocol was consistent, the use of predefined scenarios may bias defect detection compared to more flexible approaches such as ad-hoc or exploratory testing. The absence of a direct comparison with a baseline approach (e.g., measuring time to detect defects using MobileTestPro versus ad-hoc testing) also restricts the ability to attribute efficiency gains solely to the framework.

The generalizability of the findings is constrained by the limited number of devices (2–3 devices and emulators), which does not fully represent the diversity of the mobile ecosystem, including variations in screen sizes, aspect ratios, hardware configurations,

and emerging form factors such as foldable devices. Similarly, the evaluation is restricted to three applications and contexts, which may not capture the full spectrum of mobile application behaviors. In terms of robustness, the results are based on a limited set of executions without extensive variability in conditions such as network performance or repeated trials. Expanding device coverage, incorporating real user load scenarios, and including comparative and statistical analyses would improve the validity and reliability of the findings.

7 Conclusion and Future Works

MobileTestPro was validated through a holistic multi-case study applied to three usage contexts (video streaming, communications and social media), represented by the following analysis units YouTube, WhatsApp and Instagram. The results show that the dimensional approach enabled tests to be planned and executed systematically, evidence to be collected that was comparable across units of analysis, and the detection of recurring patterns relevant to the evaluation of mobile applications in real-world scenarios.

The findings show that these dimensions cannot be interpreted in isolation; compatibility and accessibility tend to reflect common interface and interaction issues. Furthermore, stress and energy consumption provide complementary evidence on resource usage under greater demand. Finally, the security dimension enables the identification of risks associated with configuration and best practices, even when these do not manifest as visible failures to the user.

Taken together, these results confirm that MobileTestPro is useful for organising and comparing evaluation by dimension in scenarios that closely resemble real-world use.

Future work includes expanding the validation coverage with more execution configurations (more models and system versions) and increasing the number of applications based on their context. Furthermore, it is recommended to strengthen the synthesis of results across dimensions using integration criteria that allow for more direct case comparisons and support improvement decisions.

References

1. Martinez, D., Ferre, X., Guerrero, G., Juristo, N.: An agile-based integrated framework for mobile application development considering Ilities. IEEE Access. **8**, 72461–72470 (2020). https://doi.org/10.1109/access.2020.2987882
2. Martínez, D., Ferré, X., Marcillo, D.: Proposal for an integrated framework for mobile applications development. In: 2018 International Conference on Information Technology & Systems, Santa Elena, pp. 813–819 (2018). https://doi.org/10.1007/978-3-319-73450-7_77
3. Romero-Collaguazo, J., Martinez, D., Zapata, M., Ferre, X.: MobileTestPro: testing framework for mobile application. In: Gervasi, O., et al. (eds.) ICCSA 2025. LNCS, vol. 15650, pp. 68–80. Springer, Cham (2025). https://doi.org/10.1007/978-3-031-96962-1_5
4. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empir. Softw. Eng. **14**, 131–164 (2009). https://doi.org/10.1007/s10664-008-9102-8
5. Yin, R.K.: Case Study Research. Design and Methods, vol. 4, pp. 264–267. Sage (2009). https://doi.org/10.1007/BF01103312

6. Myers, G.J., Badgett, Thomas, T.M., Sandler, C.: The Art of Software Testing. Wiley, Hoboken (2004)
7. Osterweil, L.: Strategic directions in software quality. ACM Comput. Surv. **28**, 738–750 (1996). https://doi.org/10.1145/242223.242288
8. Naik, K., Tripathy, P.: Software Testing and Quality Assurance: Theory and Practice
9. Muccini, H., Di Francesco, A., Esposito, P.: Software testing of mobile applications: challenges and future research directions. IEEE (2012)
10. Haller, K.: Mobile testing. ACM SIGSOFT Softw. Eng. Notes **38**, 1–8 (2013). https://doi.org/10.1145/2532780.2532813
11. Global Social Media Statistics — DataReportal – Global Digital Insights. https://datareportal.com/social-media-users. Accessed 12 Mar 2026