Mobile Testing: New Challenges and Perceived Difficulties from Developers of the Italian Industry

Mobile Testing: New Challenges and Perceived Difficulties from Developers of the Italian Industry

Riccardo Coppola, Luca Ardito, Maurizio Morisio, Marco Torchiano
Department of Computer and Automation Engineering
Politecnico di Torino
Turin, Italy
Email: first.last@polito.it

Abstract

Automated GUI testing is a fundamental part of the Verification and Validation process of every software, but it is often linked to notable maintenance costs, especially for mobile applications. The literature reports a general lack of automated testing adoption among mobile developers in the industry. In this paper we present the outcome of seven interviews centered on how companies automate the testing process of mobile applications. The interviews confirmed that automated testing is still not widely adopted and rarely formalized by industry, with manual testing being still the primary form of testing. Test fragility and evolution of the user interface are seen as a relevant issue by developers, with a cost of around 30% of the overall maintenance performed on test suites. Some clear shared needs emerged during our interviews, that can be considered hints for the added research effort from academia in meeting the needs of industry.

Index Terms

GUI-testing; Android Testing; Automated Testing

I. Background

Mobile devices have overtaken desktop computers in terms of shipped units, with the Android operating system becoming the favorite among mobile users. Current mobile Apps have reached a very high complexity and provide a wide range of features, from infotainment to security-critical

operations. Apps are relatively easy to deploy, thanks to the availability of marketplaces where they can be sold or released for free. In such a highly competitive scenario, a thorough Verification and Validation phase could be fundamental to ensure reliability and guarantee conformity to the users' needs. Testing the application GUIs (i.e., Graphical User Interfaces) is a key task, since they vehicle most of the interaction with the final user. The GUI is particularly involved when testing the flow of an App end-to-end (E2E).

Many approaches to testing mobile applications are available, ranging from the execution of manual test cases to the adoption of complex automated testing frameworks. Test automation is a preferable alternative, since it can create more dependable and adaptable test suites. Automated testing tools can be categorized according to the way the elements of the interface are recognized and interacted [1]. Android widget-based testing techniques, like Robotium, Appium, Espresso and UiAutomator, are based on the tester's knowledge of the structure of the app GUI. Fuzzy or Random testing tools, like Monkey, inspect interfaces through sequences of aleatory inputs; the definition of the inputs can also be refined using models of the user interface. ACRT by Chien et al. [11] and RERAN by Gomez et al. [7] are examples of mobile C&R testing tools, which entail manually executing test cases that are then translated into repeatable test scripts. Amalfitano et al. propose MobiGUITAR [2], a specialization for Android of the general purpose GUITAR [12], which extracts models of the GUI through a process called GUI Ripping and then systematically traverse them for testing purposes.

However, automated GUI test cases require significant maintenance and are brittle, i.e., they may fail due to minor GUI modifications, even without any functional change in the app. The impact of fragility is significant especially when Non-Regression Testing is performed, with test cases that ought be adapted to the changes in the GUI appearance and definition. A number of studies in the literature, like the one by Kropp et al. [9], highlight the unreliability of manual testing techniques for mobile applications. However, as underlined by Kaur et al. [3], several characteristics of mobile devices increase the costs to perform V&V activities. Kochar et al., in an empirical study about the test automation performed by mobile developers, also emphasize a lack in test automation culture of Android open source developers, and a common lack of proper documentation for automated testing tools [8]. Leotta et al. explored the problem of fragility in the field of web applications [10]. In our previous work we quantified the issue of fragility among open source Android applications [5], and provided a taxonomy of maintenance reasons for Android test suites [4].

This work aims to provide insights into the testing procedures operated by developers working in the Italian industry, and to understand the principal difficulties faced in mobile application testing, the differences perceived with respect to testing web or desktop applications, and the desired directions for academia to move towards. To do so, we conducted – in seven different companies – a set of semi-structured interviews covering the topics described above.

II. Research approach

A. Research Questions

Our goal is gathering insights into the kind of testing performed and the most used automated testing techniques, for understanding the practices adopted by mobile developers. Thus we ask:

How and to what extent are mobile apps tested?

We want to understand which aspects of mobile applications are considered of primary importance for testers, and – on the other hand – which ones are a deterrent for the adoption of testing techniques. Hence, our second research question is:

What are the peculiarities in mobile application testing? What aspects discourage testing mobile apps?

Finally, we wanted to characterize the interest of mobile developers from the industry about testing techniques that are currently emerging in academic literature. We also aimed at summarizing the most burdensome challenges in mobile application testing, and the amount of human labour needed for setting up and maintain mobile test suites. Therefore it would be useful to know:

What are the main challenges in mobile application testing and the directions research should take?

Each research question was developed in a separate part of the interview, that is detailed in the following subsection.

B. Interviews

We conducted a series of structured interviews in Italy with representatives from medium and large-sized enterprises. Information about the interviewees is reported in Table 1. The interviewer asked a set of questions arranged in three sections, each pertaining to one of the Research Questions. The interview sessions lasted around 30 minutes each. A transcript was obtained at the end of every session, and the findings were catalogued and organized to answer the individual Research Questions. All the questions were open ended, and are available online[1].

Each interview focused on an individual company, with one or more representatives from it. We selected enterprises based on our location. All the interviewed representatives were involved in development and testing of mobile applications that were part of the portfolio of their enterprises. In each interview we stated the motivation of our study, and provided the definition of testing fragility for mobile applications formulated in our previous works. To avoid biases, we did not state hypotheses during the interviews.

C. Threats to Validity

The results of this study are based on the interpretation of qualitative data gathered from the transcripts of the interviews. The researcher bias of the study is however limited, since at the time of this writing none of the authors was involved in the development of testing tools or has interests in defending any specific thesis. The reported results are also consistent with other experience reports available in the literature.

We also recognize, as a threat to the external validity, that our findings are based on a small sample of companies working on a limited set of application domains, and hence their generalizability to all mobile developers is not assured.

III. Findings

Table II reports a summary of the main findings that could be extracted from the interviews we conducted, regarding the tool usage in the companies, and the developers' perception about the value of E2E testing for mobile apps and its most critical aspects. A coding of the answers was performed by the authors of the present paper, after an inspection of the transcripts taken during the interview sessions.

[1] https://figshare.com/articles/Automated_mobile_testing_Interview_to_Developers/9821381

TABLE I

INTERVIEWED DEVELOPERS FROM THE INDUSTRY

Interview ID	No. of representatives	Company and project
A	1	Distributor of testing tools for various typologies of applications.
B	2	Test factory for third party applications and test consulting.
C	1	Insurance company: web and mobile apps for insurers and customers.
D	2	Insurance company: platform for insurance management.
E	1	Test factory for third party applications and test consulting.
F	1	Full-stack development of mobile applications for multiple platforms.
G	2	Test factory for consulting of test and test management for banking applications.

A. Adoption of mobile testing techniques

Adopted testing techniques. Our respondents were developing both web and mobile applications, and all of them highlighted a priority in testing web applications over mobile ones. Among our respondents, mobile testing is executed mostly at system and acceptance level. A certain amount of unit and integration testing is performed with automated techniques by all respondents, except companies B and E that, being test factories, leave low level testing to the developers.

For what concerns non-functional testing, the main focus of the respondents is on usability and performance.

In general, manual testing is always performed on mobile applications, with purposes that may vary: all of the respondents performed manually system and acceptance testing, in compliance with test cases written by the business department of the company. Manual testing was also used by respondent B to verify non-regression between consecutive releases.

Capture&Replay (C&R) and Scripted testing techniques were adopted by more than half of the respondents. Respondent C adopted C&R for data-driven test cases.

All except respondent F tested actual devices and not emulators. Respondent F was also the only one fully leveraging techniques of random/monkey testing.

Company B adopted techniques of Mobile APM (i.e., Application Performance Management), capable of evaluating the compliance to non-functional requirements on the application after its release, monitoring the usage of the application running on the users' devices; company G

TABLE II

SUMMARY OF THE FINDINGS OF THE INTERVIEWS.

	A	B	C	D	E	F	G
Uses random-monkey testing	–	–	–	–	–	✓	–
Uses manual testing	✓	✓	✓	✓	✓	–	✓
Uses Capture&Replay	✓	✓	✓	–	✓	–	✓
Uses scripted testing	–	✓	✓	✓	✓	✓	–
Uses model-based testing	–	✓	–	–	–	–	–
Mobile E2E testing is an activity with an high ROI	1	3	2	2	4	4	5
Automation provides significant benefits to mobile E2E testing	4	5	2	2	3	2	1
Device diversity is a critical aspect to test for mobile apps	1	5	1	5	5	1	3
Performance and UX is a critical aspect to test for mobile apps	1	1	5	4	5	4	5
Energy consumption is a critical aspect to test for mobile apps	1	4	4	1	1	1	4
Fragility is a significant issue for the maintenance of test cases	2	5	5	4	5	1	3
More advanced testing techniques w.r.t. current ones are necessary	2	1	1	2	5	4	2

In the coded answers the following legend is used:
1) strongly disagree; 2) disagree; 3) neutral; 4) agree; 5) strongly agree.

adopted techniques for static source code analysis and reporting instruments.

What emerged from our interviews is that the rapid development lifecycle of mobile applications, and the frequent addition of new features, is seen as a deterrent for the adoption of structured and well-documented test suites. This also suggests that mobile testing methodologies are not well integrated in the Continuous Integration workflow of mobile development. An easier interface of mobile testing techniques with existing CI techniques should provide significant benefits in having structured test suites at lower levels than Acceptance and System testing.

Adopted tools. Selenium is the most widely used tool for test scripts of web-based and hybrid mobile applications; Selenium IDE is also used for the creation of C&R scripts.

Some commercial tools were cited by the interviewed developers: four respondents cited HP UFT, used for web-based applications; two cited Silk Mobile, used for tests on native applications; three cited PerfectoMobile, used for cloud-based functional tests on real devices, using scripts created by C&R.

Other test frameworks, like those that are part of the Android Instrumentation Framework

(e.g., Espresso, UIAutomator, Monkey and MonkeyRunner), were cited by some respondents. Only one respondent used advanced AI-based and model-based testing techniques, adopting two tools named Qualitia and AppliTools.

B. Peculiarities of mobile application testing

Several differences have been highlighted by the interviewed developers when testing mobile applications, with respect to testing desktop and web applications.

Apps may be divided into three different categories: Native, if they are engineered for a particular OS/platform; Web-based, that are typical web applications loaded inside browsers; Hybrid, if they have a native part that loads dynamically web pages. As respondent B pointed out, the testing procedures for the three categories of apps vary significantly in terms of adopted instruments as well as test case definition.

For mobile applications, the complexity of the testing procedure has increased dramatically mainly because of device diversity. Mobile applications must ensure compatibility with a set of different devices, pixel densities, resolutions, screen orientations. If the applications are multi-platform, they must also be tested on the main OSs. Finally, apps must comply to the design guidelines and features of new releases of the OSs while guaranteeing backward compatibility with past versions. As respondent B told us, *"device diversity is a relevant enabler for test automation, because it is impossible to execute manual tests on many devices; we pick the devices that are sold the most in the market, and the ones that are used the most by the final users according to geographic statistics."*; respondent F also pointed out that *"device diversity and form factor are the fundamental variables to take into account, much more than for web application testing, for which it is sufficient to test the main browsers."* Albeit being an important push towards automation, the extreme device diversity of the Android world (counting more than 24 thousand different devices, built by more than 12 hundred vendors[2]) is an insurmountable impediment for testers to test all possible renderings of their apps.

Other non-functional properties are peculiar of mobile applications and require specialized testing procedures. The topic of Green Energy is another very perceived issue: ensuring a battery consumption that is adequate to the typology of the app is fundamental for the users' satisfaction.

Finally, being mobile applications strongly GUI-based, the rendition of the graphics on the

[2]https://www.hongkiat.com/blog/android-fragmentation/

device screens, and an evaluation of the provided user experience (UX), is a crucial element of acceptance testing. However, the usage of previously defined and tested mockups may relieve the developers from extensively testing the final appearance of the app once it is deployed.

C. Challenges and desires of mobile app testers

Factors limiting mobile application testing. Several problems hampering the practice of mobile application testing emerged from our interviews. For commercial applications, the companies often want a fixed time-to-market, and compromises are necessary to find an optimal balance between cost and quality. Respondent B considered that *"clients want the application to be published anyhow, and often the quality aspect is sacrificed, offering limited features. The quality of the app then grows with time, in parallel with the number of the users that use them, and thanks to their feedback."* Respondent E added that *"rarely projects have a test strategy which is carefully defined, validated and approved, with a reasonable time to perform it; testing typically suffers from delays in the previous phases of the development."*

Many of the interviewees underlined that the culture of testing mobile applications is still limited. Respondent A highlighted that, in large companies producing software, the testing department is often managed by members of the business department, who have a different perception of testing with respect to ICT people. Therefore, manual testing is often preferred and, in general, it is difficult to go particularly far beyond C&R techniques. Still according to respondent A, *"the mobile device, from the business point of view, is mainly seen as a proxy to access services that are located on the web."* Respondent D confirmed that the focus is often kept on the backend, without particular interest towards app testing. Respondent B pointed out that *"mobile application testing is still not treated with sufficient maturity, and clients are just beginning to see the return of investment that test automation can guarantee; only companies creating apps that manage sensitive and economically critical data tend to adopt automated testing"*.

We can conclude that a constant issue that emerged from the interviews is a scarce dissemination and documentation of automated testing tools. The problem appears to be amplified for open-source tools, that are henceforth rarely adopted by companies.

Maintenance and fragility. Test fragility was deemed as the main cause for maintenance of existing test suites.

Respondent A, who used C&R techniques, had to completely re-register test cases when the interface was modified between a version of the application and the subsequent one. Respondent C underlined a scarce adaptability of Selenium, estimating the modification of existing scripts as 30% of total testing effort.

Respondent B experienced fragilities in the regression testing of its applications, with an estimation of 20-30% of the total testing effort for maintaining test scripts. This respondent pointed out that *"changes in the user interface significantly limit the adoption of test automation, and the issue is amplified when it is not the same company performing developing and testing"*.

Respondent D defined fragility as a *"problem that is perceived and that has to be fought on a daily basis: test suites must be maintained daily"*. For projects, fragility is identified as a critical problem, especially for possible shortages of time: it is often not possible to do complete maintenance of test cases that fail even though they should not. This developer identified the effort for maintenance of test suites as 10% of total testing effort.

The estimated cost of fragility was even higher for respondent E, that identified the cost of maintenance of already present test scripts as 60% of total maintenance cost. The developer pointed out that *"the impact of fragility is higher for mobile applications, because mobile interfaces and features evolve more rapidly than traditional applications. The investment for test case maintenance is mandatory and grows with time, even though the changes in the user interfaces are limited."*

The information gathered from the interviews was in line – if not higher – with our previous measurements of maintenance on open-source Android code, where we quantified the amount of testing effort dedicated to maintenance as around 20% of testers' effort. Even if a certain amount of fragility is inevitable, since test code must always co-evolve with application code, we believe that the absence of reliable ways to automatically fix and refactor test cases is an hindrance for mobile testing as serious as the device diversity issue.

Requests to academia. All the respondents told us that a solution, possibly automated, to the problem of fragilities of automated tests, especially GUI-related ones, should come in handy to companies testing both web and mobile applications.

Respondent C expressed a desire for a more direct link between incidents in tests, or in running applications, to the defects in the source/system.

Respondent B highlighted the problem of test prioritization, to take into account changes to

production code and maximize the coverage of the modified source only.mSuch tools are seen as useful when resources and time for testing are limited, because it theoretically maximizes the portion of useful testing.

Respondent D identified the need for a finer way to calculate the code coverage for web/mobile test suites, with fine-grained metrics able to represent the actual economic value of the testing procedure.

Automation in test case development (and not only in the execution) was a need expressed by respondent F, who pointed out that *"an algorithmic creation of test cases during the definition of the application logic to an even limited coverage of features to be tested, would be very happily welcomed by developers, who still see writing test cases as an overhead."*

Model-based testing is not felt as a primary need, or something that can be useful at least in the near future. Only respondent A showed enthusiasm towards the possibility of adopting model-based testing, expressing the need for a *"trustable mobile ripper, capable of semantically interpret wgat happens during the exploration of an interface, and proposing test cases."* Respondent B highlighted that model-based testing is indeed an interesting topic for academia, but at a first approach the techniques that have emerged are too complex and require too much knowledge to be actually adopted by companies. Respondent E was the only one performing a sort of modeling of apps for the definition of test cases, and pointed out that an extended modularity of tests should be encouraged by research.

IV. Conclusion

We conducted seven interviews with relevant players of the Italian software industry, who develop mobile applications along web and desktop ones; all of them reported experience with both manual and automated testing.

Even though all of the respondents adopted some automated testing tools, to some extent they considered manual testing as the first option to test their apps. According to their responses, the need for automated mobile testing is still not perceived as it is for other kinds of applications; the practice is also hampered by the inexperience of the members of testing departments, by the difficulty of using automated testing tools (which often come with insufficient documentation, especially if they are open-source), and by the very rapid time-to-market of mobile apps. In general, a limited amount of C&R tests and an even smaller amount of scripted tests are developed. It is evident from the responses that the adaptability to different devices and performance

are important concerns for mobile apps testing, higher than it happens for desktop application testing.

For those developing scripted tests, the amount of maintenance estimated, due to fragility, is around 30%, with a peak estimation of 60% reported by one of the respondents. These estimates are actually even higher than the ones that we found in open-source projects within a previous study [6], using automated analysis of modifications performed on source code using diff files. This means that fragility is a very important issue even for industrial developers, and not only for open-source developers and researchers.

Almost all respondents expressed a desire for better ways to manage fragility and to reduce the amount of maintenance needed for test suite evolution. Little enthusiasm, on the other hand, has been shown towards new paradigms of testing, like model-based testing or visual recognition testing. Skepticism has been shown towards the adoption of techniques that are more complex than C&R or basic scripting with tools like Selenium, or similar.

We may put forward a few recommendations to practitioners:

- the more critical your app the more important automated tests are;
- be prepared to pay significant effort to maintain tests;
- to reduce such effort try to keep test and development teams as close as possible: make app changes as little disruptive to tests as possible.

What finally emerged by some of our interviews is that test automation is seen as an additional difficulty rather than an enabler for better testing procedures. This finding makes us suppose that the dissemination of automated testing is still scarce, and that developers of testing tools should focus on making them better documented, easier to use, and more dependable than those currently available, to foster adoption from the industry.

Riccardo Coppola is a Post-Doctoral Research Fellow at Dept. of Control and Computer Engineering at Politecnico di Torino, where he received his MSc and PhD degree in Computer Engineering. He is currently a member of the Software Engineering research group, and his research interests include automated GUI testing for web and mobile applications, and the evaluation of non-functional properties of testware.

Luca Ardito is an Assistant Professor at Dept. of Control and Computer Engineering at Politecnico di Torino where he works in the Software Engineering research group. He received

BSc, MSc, and PhD in Computer Engineering from Politecnico di Torino. His current research interests are: mobile development and testing, green software and empirical software engineering methodologies.

Maurizio Morisio received the Ph.D. degree in software engineering from Politecnico di Torino, Turin, Italy. He is currently a Professor of computer science with Politecnico di Torino, where he leads the software engineering research group (http://softeng.polito.it). His research interests include software engineering for safety-critical systems, software and system engineering for sociotechnical systems, and smart cities and green ecosystems.

Marco Torchiano is an associate professor at the Control and Computer Engineering Dept. of Politecnico di Torino, Italy; he has been post-doctoral research fellow at Norwegian University of Science and Technology (NTNU), Norway. He received an MSc and a PhD in Computer Engineering from Politecnico di Torino. He is Senior Member of the IEEE and member of the software engineering committee of UNINFO (part of ISO/IEC JTC 1). He is author or co-author of over 140 research papers published in international journals and conferences, of the book "Software Development—Case studies in Java" from Addison-Wesley, and co-editor of the book "Developing Services for the Wireless Internet" from Springer. He recently was a visiting professor at Polytechnique Montréal studying software energy consumption. His current research interests are: green software, UI testing methods, open-data quality, and software modeling notations. The methodological approach he adopts is that of empirical software engineering.

REFERENCES

[1] Alégroth, Emil, Robert Feldt, and Pirjo Kolström. "Maintenance of automated test suites in industry: An empirical study on Visual GUI Testing." Information and Software Technology 73 (2016): 66-80.

[2] Amalfitano, Domenico, et al. "MobiGUITAR: Automated model-based testing of mobile apps." IEEE Software 32.5 (2015): 53-59.

[3] Anureet Kaur. 2015. Review of Mobile Applications Testing with Automated Techniques. interface 4, 10 (2015).

[4] Coppola, Riccardo, Morisio, Maurizio, and Torchiano, Marco. "Maintenance of Android Widget-based GUI Testing: A Taxonomy of test case modification causes." 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2018.

[5] Coppola, Riccardo., Morisio, Maurizio, and Torchiano, Marco. (2018). Mobile GUI Testing Fragility: A Study on Open-Source Android Applications. IEEE Transactions on Reliability.

[6] Coppola, Riccardo, Morisio, Maurizio, and Torchiano, Marco. "Scripted UI Testing of Android Apps: A Study on Diffusion, Evolution and Fragility." Proceedings of the 13th International Conference on Predictive Models and Data Analytics in Software Engineering. ACM, 2017.

[7] Gomez, Lorenzo, et al. "Reran: Timing-and touch-sensitive record and replay for android." Software Engineering (ICSE), 2013 35th International Conference on. IEEE, 2013.

[8] Kochhar, Pavneet Singh, et al. "Understanding the test automation culture of app developers." Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on. IEEE, 2015.

[9] Kropp, M., & Morales, P. (2010). Automated GUI testing on the Android platform. Testing Software and Systems, 67.

[10] Leotta, Maurizio, et al. "Visual vs. DOM-based web locators: An empirical study." International Conference on Web Engineering. Springer, Cham, 2014.

[11] Liu, Chien-Hung, et al. "Capture-replay testing for android applications." Computer, Consumer and Control (IS3C), 2014 International Symposium on. IEEE, 2014.

[12] Nguyen, Bao N., et al. "GUITAR: an innovative tool for automated testing of GUI-driven software." Automated Software Engineering 21.1 (2014): 65-105. APA