Software Testing of Mobile Applications: Challenges and Future Research Directions

Henry Muccini
Dipartimento di Informatica
University of L'Aquila, Italy
henry.muccini@di.univaq.it

Antonio Di Francesco
Dipartimento di Informatica
University of L'Aquila, Italy
antonio.difrancesco84@gmail.com

Patrizio Esposito
Dipartimento di Informatica
University of L'Aquila, Italy
patexp80@gmail.com

***Abstract*—While mobile applications are becoming so extraordinarily adopted, it is still unclear if they deserve any specific testing approach for their verification and validation. This paper wants to investigate new research directions on mobile applications testing automation, by answering three research questions: (RQ1) *are mobile applications (so) different from traditional ones, so to require different and specialized new testing techniques?*, (RQ2) *what are the new challenges and research directions on testing mobile applications?*, and (RQ3) *which is the role automation may play in testing mobile applications?*. We answer those questions by analyzing the current state of the art in mobile applications development and testing, and by proposing our view on the topic.**

I. Introduction

Applications running on mobile devices (e.g., mobile applications running on smart phone or new generation tablets) are becoming so popular that they are representing a re-volution in the IT sector. In three years, over 300,000 mobile applications have been developed *(mobithinking.com)*, they have been downloaded 10.9 billion times in 2010 *(www.idc.com)* with a forecast of 29 billions applications downloaded in 2011 *(www.abiresearch.com)* and a prediction of 76.9 billion global downloads in 2014 that will be worth US$35 billion *(www.idc.com)*.

While mobile applications were initially developed mostly in the entertainment sector, they are now touching more critical domains: think about the NFC or SQUARE *(https://squareup.com/)* technologies intended to revolutionize the current payment system, to the m-government *(www.mobih.org/resources/)* and mobile-health initiatives[1], or to the huge amount of enterprise and industry-specific mobile applications expected to be produced in the next years [2].

The exponential growth of this market and the criticality of the developed (or, to be developed) systems impose an increased attention to dependability aspects of applications running on them. As demonstrated in some studies [7], [9], mobile applications are not bug free and new software engineering approaches are required to test those applications [17].

But, *are mobile applications (so) different from traditional ones, so to require different and specialized new testing techniques (RQ1)?* And if so, *what are the new challenges and research directions on testing mobile applications (RQ2)?*, and *which is the role automation may play in testing mobile applications (RQ3)?*

Goal of this work is to provide an answer to those research questions. For answering RQ1, we first analyze what a mobile application is or is going to be (Section II-A). We look at mobility and context-awareness as the most peculiar characteristics of mobile applications, and based on those, define two types of mobile applications. Then, we analyze how mobile applications peculiarities drive specialized research on mobile applications' testing (Section II-B). Section III helps to provide an answer to RQ2 and RQ3, by identifying challenges and future research directions on automated testing of mobile applications. We outline the most relevant aspects in systematic testing of mobile applications, followed by a brief analysis of state-of-the art solutions, and future research directions. Section IV concludes this paper.

II. Mobile Applications Implications on Testing

Are mobile applications (so) different from traditional ones, so to require different and specialized new testing techniques? While some studies have analyzed additional requirements of mobile applications not commonly found in traditional ones [17], this section wants to investigate how mobile applications testing differs from testing traditional applications.

A. What a Mobile Application Is

A mobile application is vaguely defined as an application running on mobile devices (see [7], [17]) and/or taking in input contextual information [5]. In order to improve our degree of knowledge on the topic, we decided to analyze the role of mobility and context-awareness on mobile applications.

[1]See for example: http://www.imedicalapps.com/, http:// iphonemedicalapps.com/, https://market.android.com/apps/ MEDICAL?hl=en

978-1-4673-1822-8/12/$31.00 © 2012 IEEE

AST 2012, Zurich, Switzerland

In *mobile computing* an application is considered to be mobile if it runs on an electronic device that may move (e.g., mp3 readers, digital camera, mobile phones). Satyanarayanan in [13] synthetically captures the uniqueness and conceptual difference of mobile computing through four constraints: limited resources, security and vulnerability, performance and reliability variability, and finite energy source.

In *context-aware computing*, instead, an application is aware of the computing environment in which it runs, and adapts/reacts according to its computing, user, physical, or time context [14], [5]. Schmidt in [15] categorizes contextual information into *human factors* (i.e., user, social environment, and task) and *physical environments* (i.e., location, infrastructure, and physical conditions). Challenges in context-aware computing are represented by contextual sensing (e.g., location, time, nearby objects, orientation, social contexts), contextual adaptation/reconfiguration, context-triggered actions, and contextual resource discovery [11], [14].

Figure 1 summarizes our view on mobile applications by classifying them in two different categories: traditional applications rewritten to run on mobile devices (e.g., web navigation and searching, social networking mobile applications, collaborative work applications) hereafter referred as *App4Mobile*, and mobile applications that make use of contextual information to generate context-based outputs (denoted as *MobileApp*s). As a direct impact on testing, App4Mobile applications inherit the peculiarities of mobile applications (e.g., mobility, autonomy, and connectivity) [13], while MobileApps inherits the challenges associated to context-aware applications as well.

From a testing perspective, we thus define an App4Mobile as an application that, driven by user inputs, runs on a mobile device with limited resources. A MobileApp is a particular App4Mobile that inputs data from the (potentially evolving) environment surrounding the mobile device and/or from user actions, producing context-based outputs.

Different testing techniques will be required for those two types of mobile applications.

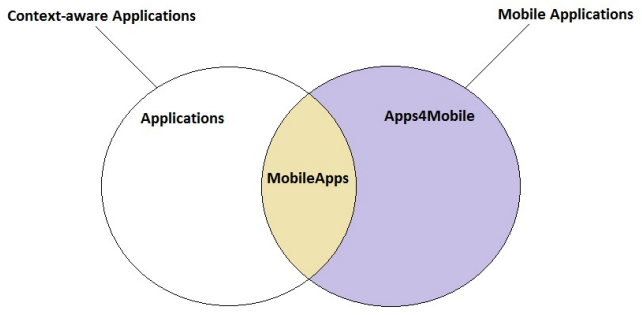

Figure 1. Types of Mobile Applications

B. Implications of Mobile Application Peculiarities to Software Testing

By keeping in mind the difference between *Apps4Mobile* and *MobileApps*, this section highlights how mobile applications differ from traditional desktop applications and the implications on software testing. This analysis, while building on mobile applications peculiarities defined in [17] and mobile context-aware applications peculiarities [14], [5], [13], re-shapes and extends available knowledge by looking at it from a testing perspective. Table 2 summarizes peculiarities and implications discussed in the sequel of this section.

Type of Mobile Application			Peculiarity	Implications on Testing
Mobile Applications	MobileApps	Apps4Mobile	Mobile Connectivity	Reliability, performance, security, and functional testing through different networks
			Limited Resources	Performance and functional monitoring
			Autonomy	Monitoring for energy consumption
			User Interface	GUI Testing
			Context Awareness	Context-dependent functional and extra functional testing
			Adaptation	Adaptation correctness
			New programming languages	New white box and black-box testing, byte-code analysis
			New O.S.s	Compatibility and O.S. testing
			Diversity of phones and O.S.s	Diversity coverage
			Touch Screens	Usability, and response to screen touch

Figure 2. Table

1) Peculiarities of Apps4Mobile Applications and Implications on Software Testing: The main peculiarities of Apps4Mobile are:

Mobile Connectivity:

- Peculiarities: mobile connectivity is one of the most peculiar, as well as critical, characteristics of a mobile application. Mobile applications typically connect to mobile networks, that may vary in speed, security, and reliability.
- Implications on Testing: the application reliability, performance, security, and correct functioning strongly relies on the available connectivity type. Functional and extra functional testing has to be performed in different connectivity scenarios and networks. Bugs related to unreliable Wi-Fi, 3G, or bluetooth connections have been reported [1].

Limited Resources:

- Peculiarities: while mobile devices are becoming more and more powerful, their resources are still far away from those available on a laptop or desktop computers (e.g., the most powerful iPad can account for a modest -if compared with laptop devices- 512 MB of RAM, 64 GB of disk space, and 1 Ghz dual core CPU).
- Implications on Testing: mobile applications resource usage has to be continuously monitored so to avoid per-

formance degradation and incorrect system functioning. Actions have to be taken in case of resource shortage.

`Autonomy:`

- Peculiarities: differently from desktop or laptop computers, mobile devices can never rely on an electrical outlet for functioning. Different mobile applications may require very different energy consumption. Considering that a 200 hours stand-by autonomy of an iPhone 4S drops to 9 hours when a Wi-Fi connection is enabled, and to 6 hours when a 3G connection is active, an application requiring full time 3G connectivity strongly impacts the device autonomy. Reduced autonomy can be an issue (e.g., patches have been produced to fix battery life issues[2]); autonomy improvement can instead be a business advantage[3].
- Implications on Testing: as for the *limited resources* issue above, the energy consumption of mobile applications can be evaluated through testing or continuous monitoring.

New `User Interfaces:`

- Peculiarities: when developing a mobile application, developers have to follow some strictly defined guidelines for producing mobile applications GUIs. Mobile applications may in fact look differently depending on the mobile device screen resolution and dimension.
- Implications on Testing: GUI testing is a first class testing need in mobile applications. Different mobile devices can react differently to the same application code and this needs to be tested, as claimed in [7].

2) Peculiarities of MobileApp Applications and Implications on Software Testing: MobileApp applications, while inheriting the App4Mobile peculiarities, also present their own distinctive characteristics that are listed below.

`Context Awareness:`

- Peculiarities: *MobileApp*s heavily rely on sensed data provided by context providers, that are sensing (like noise, light, motion, image sensors) and connectivity (like bluetooth, GPS, Wi-Fi, 3G) devices. All those devices may provide a huge amount of inputs (e.g., brightness, temperature, altitude, noise level, type of connectivity, bandwidth, neighboring devices) that vary (even unpredictably) depending on the environment and/or user actions.
- Implications on Testing: testing whether the application is going to correctly work on any environment and under any contextual input is a challenge and may lead to combinatorial explosion. Context-specific test selection techniques and coverage criteria have to be produced. Bugs associated to contextual inputs are quite frequent [1].

`Adaptation:`

- Peculiarities: MobileApps may adapt and evolve driven by contextual information. Such an evolution may happen during the application operation and without stopping the service. The application dependability may be impacted by unforeseen, run-time adaptations.
- Implications on Testing: approaches for testing the application correct adaptation/evolution have to be developed.

3) Other Peculiarities Related to Mobile Applications in General: This paragraph lists peculiarities that, while not directly ascribable to MobileApp or App4Mobile applications, may introduce new testing issues for mobile applications.

New Mobile Programming Languages:

- Peculiarities: programming languages of mobile applications have been designed for supporting mobility, managing resource consumption, and handling new GUIs. Objective-C, for example, uses the UIKit framework for implementing GUIs, and it is a dynamic language that can be extended at run-time. Android, instead, replaces the `java.awt` and `java.swing` APIs with new libraries, supports the bluetooth through third parties libraries, and uses the Dalvik opcode compiler optimized for mobile devices.
- Implications on Testing: traditional structural testing techniques (and associated coverage criteria) need to be revised in order to be applied to new mobile languages. Bytecode analysis tools and functional testing techniques may be required to deal with binary code.

New (and rapidly Evolving) Mobile Operating Systems:

- Peculiarities: mobile applications run on top of new operating systems that are still only partially reliable. Many are the examples of applications failing due to operating systems problems, like the iOS alarm clock failure[4]. Moreover, new versions of mobile operating systems are frequently released, not always backward compatible with previous versions.
- Implications on Testing: while testing mobile applications, failures not ascribable to mobile applications bugs can be generated due to OSs unreliability and variability.

Diversity of Phones and Phone Makers:

- Peculiarities: hundreds of different mobile devices are on the market, produced by different vendors, and with

[2]http://tuto4log.blogspot.com/2011/11/apple-releases-ios-501-for-fix-autonomy.html

[3]http://www.yousaytoo.com/samsung-s2-galaxy-android-2-dot-3-5-brings-a-considerable-improvement-o/1228272

[4]See, for example, http://www.macworld.com/article/156793/2011/01/ios_alarm.html

different software features and hardware components. Mobile applications, while running on different devices, may behave differently due to variations in the hardware or O.S. components (a study reported the existence of 1.800 hardware/O.S. different configurations[5]). For example, sensors are typically calibrated differently, so that two (different) phones running the same application, in the same environment, may compute different outputs.
- Implications on Testing: testing techniques for maximizing the diversity coverage while minimizing the devices under test have to be devised.

`Touch Screens:`

- Peculiarities: touch screens are the main means for inputting user data into a mobile application. However, the system response time to a touch strongly depends on the device resource utilization, and may become slow in certain circumstances (e.g., entry level hardware, busy processor). Touch screens lack of responsiveness has been reported[6].
- Implications on Testing: testing techniques have to be introduced to test the touch screen functioning under different circumstances (e.g., resources usages, processor load, memory load) and within different mobile devices.

III. Challenges and Future Research Directions on Testing Mobile Applications

A systematic software engineering approach to software testing is aimed at maximizing fault detection, making results reproducible, and reducing the influence of external factors, such as random selection of test cases or the testers mood. When dealing with systematic testing, different dimensions can be considered. For the purpose of this work, we focus on a subset of what we consider the most relevant aspects in systematic software testing of mobile applications. Specifically, we analyze challenges and potential future research directions on i) the testing process (test selection and execution), ii) the testing artifacts (structural and functional), iii) the testing levels (unit and integration), and iv) the types of tests to be run.

Each topic is structured so to discuss what we consider the most relevant *challenges* first, followed by some reasoning about *potential* research directions and support to test *automation*, and the most recent state-of-the art (*SOTA*).

[5]http://www.mobileappstesting.com/2012/01/10/seetest-the-android-multi-device-challenge/

[6]see, for example, http://code.google.com/p/android/issues/detail?id=23382

A. The Testing Process: Challenges and Research Directions in Test Selection and Test Execution of Mobile Applications

The testing process consists of different activities. Those on which we focus on this paper are *test selection* and *test execution*.

- `Test Selection.` *(Challenge):* while selecting test cases for App4Mobile seems not to introduce specific peculiarities, MobileApp are expected to receive inputs from different context providers (i.e., users, sensors and connectivity devices), inputs that vary from the different (and changing) contexts the mobile device can move towards. This may lead to the unpredictability and high variability of the inputs the application is potentially receiving. *(Potentials and Automation):* new testing criteria are required to provide the guidelines, rules, and strategies by which mobile test cases are selected so to maximize coverage in case of unpredictable and variable contextual inputs. Models of contextual scenarios may be used to drive the systematic selection and coverage of MobileApps. For example, if a tracking application making use of GPS data wants to be tested, the test suite could already include parametric inputs associated to the GPS input scenarios, e.g., the entrance/exit to/from a gallery. *(SOTA):* in [12] different contexts are created and used for testing purposes, but without a coverage criteria. In [8], contexts that have a higher probability to generate faults are selected. Overall, coverage criteria for context-aware inputs seem to be still missing.
- `Test Execution.` *(Challenge):* Apps4Mobile applications are, in terms of test execution, quite similar to traditional applications. In MobileApps, instead, contextual information leads to the challenge on how to execute test cases including rich contextual inputs. *(Potentials and Automation):* existing simulators are unable to simulate real-world phones with their sensors, GPS, connectivity. New capture-and-replay automated techniques can be realized for executing (as a replay) contextual inputs selected during the test selection phase. *(SOTA):* Android Robotium[7] is an open-source tool that enables the automated and black box test execution of third parties applications. It resembles Selenium but for Android applications.

B. The Testing Artefacts: Challenges and Research Directions in Structural and Functional Testing of Mobile Applications

Structural (code-based) or functional (model-based) testing techniques can be devised for testing mobile applications. More specifically:

[7]http://code.google.com/p/robotium/

- `Structural`. *(Challenge):* mobile appplication languages add some specific constructs for managing mobility, sensing, and energy consumption. The peculiarities of those new programming languages have to be taken into account when producing control or data flow graphs (and their respective coverage criteria) out of the mobile programming language. *(Potentials and Automation):* New coverage criteria (and if needed, new control and data flow graphs) shall be though as a way to consider at best the new mobility, sensing, and energy constructs. In case the source code is not available, new bytecode analysis tools can be realized. *(SOTA):* In [6] the authors propose a structural testing approach to execute test cases on real devices. The tool JaBUTi/ME is proposed for this purpose.
- `Functional`. *(Challenge):* mobile applications functional testing requires to specify both the application and the environment it may operate in (especially in MobileApps). *(Potentials and Automation):* state-based approaches can be particularly useful to specify the different states a mobile application may reach when different sensed data are received. We can expect to model the ranges of value an environmental variable can get, and model the impact of various environmental inputs into the application under test. We can also use state-based approaches to model different execution modes (e.g., low battery, meeting, and flying mode) the system can be. *(SOTA):* Android Robotium[8] is an open-source tool that enables the automated and black box test execution of third parties applications. The MonkeyRunner tool [9] can run an automated functional test for Android applications. MobileTest [4] adopts a sensitive-event based approach for the automatic black box testing of software running on mobile devices.

C. Unit and Integration Testing: Challenges and Research Directions in Unit and Integration Testing of Mobile Applications

While guidelines and tools have been proposed to unit testing mobile applications, integration testing seems to be still quite unexplored, as briefly shown below.

- `Unit testing`. *(Challenge):* no specific challenge seems to apply to unit testing of mobile applications. *(Potentials and Automation):* there is the need and potential to realize automated testing tools supporting and enabling the specification of contextual values and context-based coverage. *(SOTA):* iOS provides guidelines on how to perform unit testing on Apps4Mobile applications[10]. Android enables the use of JUnit for unit testing Android applications[11]. However, in both cases, contextual values are not contemplated.
- `Integration testing`. *(Challenge):* while most of current testing approaches consider mobile applications in isolation, inter-application communication via intents and content providers are also possible. When testing such cooperating applications, intra- and inter-application information flow must be followed (e.g., when testing the absence of security leaks). *(Potentials and automation):* existing integration testing approaches have the potentials to be adapted for mobile applications. *(SOTA):* at the best of our knowledge, mature research on integration testing of mobile applications is still missing today.

D. Type of Testing: Challenges and Research Directions in Testing Mobile Applications

Different types os testing strategies can be applied to mobile applications, and more specifically:

- `Performance and reliability testing`. *(Challenge):* performance and reliability of mobile applications strongly depends on the mobile device resources, on the device operational mode, on the connectivity quality and variability, and other contextual information. *(Potentials and Automation):* new techniques for performance and reliability analysis have to explicitly consider characteristics related to (changing) contexts and different devices. Run-time analysis techniques can also be adopted to monitor the resources and connectivity state and prevent performance degradation. *(SOTA):* Berardinelli et al. in [3] introduce a framework for analyzing the performance of context-aware mobile software systems.
- `Memory and Energy testing`. *(Challenge):* memory leaks may preempt the (limited) memory resource, and active processes (of killed applications) may reduce the device (battery) autonomy. *(Potentials and Automation):* metrics for estimating energy and memory consumption could be devised so to automatically predict memory leaks and energy outage. *(SOTA):* the iOS `Instruments` tool enables memory leaks analysis. However such analysis is not exhaustive. Some development guidelines[12] explain how to avoid null pointer exceptions due to silent processes, but approaches and tools for testing those type of failures are still missing. Thompson et al. [16] propose a model driven engineering tool for modeling mobile software architectures successively used to generate synthetic emulation code to estimate power

[8] http://code.google.com/p/robotium/

[9] http://developer.android.com/guide/developing/tools/monkeyrunner_concepts.html

[10] https://developer.apple.com/library/ios/♯documentation/DeveloperTools/Conceptual/ UnitTesting/00-About_Unit_Testing/about.html

[11] http://developer.android.com/reference/android/test/ AndroidTestCase.html

[12] http://www.bettersoftware.it/conference/talks/tutto-il-testing-possibile-su-android

consumption properties. Palit et al. [10] propose a methodology to select user level test cases for performing energy cost evaluation of smartphone applications.

- `Security testing.` *(Challenge):* security is particularly relevant due to the mobility of the device into networks with different security levels. A trojan might infact have access to personal date, private networks, and private contextual information (e.g., where the application user is located). Moreover, the rich contextual information presents real privacy concerns. The democratization of mobile application publishers also increases the number of apps store the final user does not know anything about. *(Potentials and Automation):* since mobility towards different networks represent a peculiarity of mobile applications, traditional security testing approaches shall be revised so to keep in consideration contextual factors, that shall be simulated so to check which data is transmitted from the mobile device. *(SOTA):* in [18] the authors analyze the threats Android applications pose to the security and privacy of enterprise and propose several approaches for defending enterprises against security risks.
- `GUI testing.` *(Challenge):* two are the main challenges we foresee in GUI testing of mobile applications: i) to test whether different devices provide an adequate rendering of data, and ii) to test whether native applications are correctly displayed on different devices. *(Potentials and Automation):* an idea can be to automatically execute scripts that are captured during the user interaction and replayed and modified even on different devices. It is important to make this task automatic, so to avoid to manually interact with the GUI that is a time consuming task. *(SOTA):* a few testing approaches have been proposed for capturing GUI failures. In [7] the authors propose a technique for detecting GUI bugs by automatically generating test cases, feeding the application with random events, instrumenting the VM, and producing log/trace files and analyzing them post-run. The Android Monkey tool[13] provides features for stress testing of mobile applications user interfaces.
- `Product line testing.` *(Challenge):* testing an application on a multitude of mobile devices is certainly an important challenge, especially in the Android O.S., where different mobile phones provide different features and hardware components, and phone manufacturers may deeply customize the O.S. Considering that as far as today there are around 130 different mobile phones running Android, seven versions of the Android OS, and assuming two firmwares per device, this will count to about 1800 different combinations[14]. *(Potentials and Automation):* the current *"test on many devices"* practice has to be replaced by (cost) effective automated techniques. Commonalities and variabilities (using a jargon coming from the software product line community) in hardware devices and operating systems may be used for a systematic testing of the cell phone product lines. A different approach could consider to release (free of cost) instrumented beta versions of the application to be tested, have them running on a multitude of devices, and collect, store, and analyze run-time data and failures. *(SOTA):* A service provided by LessPainful.com enables users to upload their applications on their web-site and test them on a number of devices. A short article (available at http://www.mobileappstesting.com/2012/01/10/seetest-the-android-multi-device-challenge/) presents two strategies to maximize coverage while minimizing the number of devices used for testing.

[13]http://developer.android.com/guide/developing/tools/monkey.html

E. Concluding Remarks

The *need of automation* in testing mobile applications is exacerbated by two orthogonal aspects: i) `Cost of testing`: the common perception on mobile applications is that they must be cheap, and cost much less than traditional applications. On the other side, they must be performant, reliable and correct[15]. Automation is certainly among the most important means for keeping the cost of testing low, while guaranteeing and adequate degree of dependability. ii)`Testing through the layers`: current bugs are due to interoperability problems that exist today among the application, application framework, operating system, and hardware (sensoring) layers (as also remarked in [9]). Bugs reports show how certain types of failures do not depend on buggy applications, but more on operating systems malfunctioning, or sensored data lack of precision. This remark highlights the need of testing automation towards all the different layers, and able to clearly separate application-level failures from application framework or OS failures.

Solutions that may enable *cost-effective* testing of mobile applications include outsourcing, cloud- and crow-based testing. We can expect that companies will start offering *as-a-service* software testing services, providing specialized testing skills and laboratories to make thorough testing of critical mobile applications affordable. lesspainful.com, for example, has already launched a service to verify mobile applications on a variety of devices. Cloud- and crow-based testing strategies may also enable a distributed, cost-effective way of testing mobile applications on a multitude of devices.

[14]Data coming from http://www.mobileappstesting.com/2012/01/10/seetest-the-android-multi-device-challenge/

[15]see for example: http://www.webperformancetoday.com/2011/07/20/new-findings-mobile-web-users-are-more-disappointed-than-ever/

IV. Conclusions

This short paper has tried to provide an overview on what testing mobile applications is and can be in the next few years. Given the first research question (RQ1) *are mobile applications (so) different from traditional ones, so to require different and specialized new testing techniques?* and based on what discussed in Section II and III the natural answer seems to be *yes, they are*. About (RQ2) *what are the new challenges and research directions on testing mobile applications?*, the challenges seems to be many, related to the contextual and mobility nature of mobile applications. Performance, security, reliability, and energy are strongly affected by the variability of the environment where the mobile device moves towards. As far as concern (RQ3) *which is the role automation may play in testing mobile applications?*, some potentials for automation have been outlined, being aware that a much deeper and mature study shall be conducted. We look at this paper as a working document that can be amended and extended with the help of practitioners and researchers. In this line, we are contacting colleagues with expertise on specific topics discussed in this paper, for making this paper a more through repository of knowledge and a reference for future research on mobile application testing.

Acknowledgments

The authors want to thank Alfredo Morresi, Iulian Neamtiu, and Tony Wasserman for sharing some of their results with us, and the anonymous reviewers for the useful insights.

References

[1] "GoogleCode Android open issues," http://code.google.com/p/android/issues/list.

[2] "The 2011 IBM Tech Trends Report," ibm.com/developerworks/techtrendsreport, November 2011.

[3] L. Berardinelli, V. Cortellessa, and A. Di Marco, "Performance modeling and analysis of context-aware mobile software systems," in *Fundamental Approaches to Software Engineering*, ser. LNCS, D. Rosenblum and G. Taentzer, Eds. Springer Berlin / Heidelberg.

[4] J. Bo, L. Xiang, and G. Xiaopeng, "MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices," in *Proc. of the Second Int. Workshop on Automation of Software Test*, ser. AST '07, 2007, pp. 8–.

[5] G. Chen and D. Kotz, "A Survey of Context-Aware Mobile Computing Research," Hanover, NH, USA, Tech. Rep., 2000.

[6] M. E. Delamaro, A. M. R. Vincenzi, and J. C. Maldonado, "A strategy to perform coverage testing of mobile applications," in *Proc. of the 2006 Int. Workshop on Automation of Software Test*, ser. AST '06, 2006, pp. 118–124.

[7] C. Hu and I. Neamtiu, "Automating GUI Testing for Android Applications," in *Proc. of the 6th Int. Workshop on Automation of Software Test*, ser. AST '11, 2011, pp. 77–83.

[8] B. Jiang, X. Long, X. Gao, Z. Liu, and C. W.K., "FLOMA: Statistical fault localization for mobile embedded system," in *3rd International Conference on Advanced Computer Control (ICACC)*, January 2011, pp. 396–400.

[9] A. K. Maji, K. Hao, S. Sultana, and S. Bagchi, "Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian," *Software Reliability Engineering, International Symposium on*, vol. 0, pp. 249–258, 2010.

[10] R. Palit, R. Arya, K. Naik, and A. Singh, "Selection and execution of user level test cases for energy cost evaluation of smartphones," in *Proc. of the 6th Int. Workshop on Automation of Software Test*, ser. AST '11, 2011, pp. 84–90.

[11] M. J. Pascoe, "Adding Generic Contextual Capabilities to Wearable Computers," in *Proc. of the 2nd IEEE Int. Symposium on Wearable Computers*, ser. ISWC '98, 1998, pp. 92–.

[12] M. Sama, "Context-Driven Testing and Model-Checking for Embedded Context-Aware and Adaptive Systems," in *ISSTA 2008 poster*, 2008.

[13] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing," in *Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, ser. PODC '96. New York, NY, USA: ACM, 1996, pp. 1–7. [Online]. Available: http://doi.acm.org/10.1145/248052.248053

[14] B. Schilit, N. Adams, and R. Want, "Context-Aware Computing Applications," in *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 1994, pp. 85–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=1439278.1440041

[15] A. Schmidt, "Implicit human computer interaction through context," *Personal and Ubiquitous Computing*, pp. 191–199, 2000.

[16] C. Thompson, J. White, B. Dougherty, and D. C. Schmidt, "Optimizing mobile application performance with model—driven engineering," in *Proceedings of the 7th IFIP WG 10.2 International Workshop on Software Technologies for Embedded and Ubiquitous Systems*, ser. SEUS '09. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 36–46. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-10265-3_4

[17] A. I. Wasserman, "Software Engineering Issues for Mobile Application Development," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*, ser. FoSER '10. New York, NY, USA: ACM, 2010, pp. 397–400. [Online]. Available: http://doi.acm.org/10.1145/1882362.1882443

[18] X. Wei, L. Gomez, I. Neamtiu, and F. Michalis, "Malicious android applications in the enterprise: What do they do and how do we fix it?" in *ICDE Workshop on Secure Data Management on Smartphones and Mobiles - SDMSM 2012*, 2012.