Problems and Solutions in Mobile Application Testing

Triin Samuel and Dietmar Pfahl(✉)

Institute of Computer Science, University of Tartu, Tartu, Estonia
{triin.samuel,dietmar.pfahl}@ut.ee

Abstract. In recent years the amount of literature published about mobile application testing has significantly grown. However, it is unclear to what degree stated problems and proposed solutions are relevant to industry. To shed light on this issue, we conducted a literature survey to provide an overview of what current scientific literature considers problems and potential solutions in mobile application testing, and how often proposed solutions were reportedly evaluated in industry. Then we conducted a case study involving six software companies in Estonia to find out which of the problems are considered relevant by professionals, and which of the proposed solutions are considered novel and applicable. In total, we identified 49 potential problems or challenges in the mobile application testing domain and 39 potential solutions, some of which were implemented software tools while others were just theoretical concepts. Although some of the solutions were reportedly applied in practice, in most cases the literature did not give much information on the actual usage in industry of the proposed solutions. The case study revealed that while the relevance of each identified problem was highly variable from one company to another, there are some key problems that are generally considered vital both by research and industry. Regarding solution proposals, it turned out they are often described too much on the conceptual level or are too unrelated to the most urgent test-related problems of our case companies to be of interest to them.

1 Introduction

In the recent years, mobile devices have grown from futile entertainment gadgets to popular and ever-present media with a wide range of uses from social applications to business, medicine and others. This has brought the importance of testing mobile applications into highlight. As mentioned by various researchers [1–4] mobile applications have some unique qualities that demand new or modified testing approaches to ensure effectiveness and efficiency. Accordingly, the number of scientific papers written about mobile application testing is steadily increasing. However, it is uncertain whether the proposed solutions are usable in industry and whether the problems mentioned in industry are actually relevant in real mobile application development and testing. In order to find answers to these questions, we decided to carry out a literature survey and followed by a case study to assess the practical relevance of the information collected from literature. The exact research questions are as follows:

P. Abrahamsson et al. (Eds.): PROFES 2016, LNCS 10027, pp. 249–267, 2016.
DOI: 10.1007/978-3-319-49094-6_16

- RQ1: What are the problems specific to testing of mobile applications as opposed to conventional applications, according to scientific literature?
- RQ2: What are the solutions (methods, tools) proposed by literature, if any?
- RQ3: According to literature, to what extent are these solutions used in industry?
- RQ4: Does industry consider the solutions proposed in the literature as relevant?
- RQ5: Does industry consider the solutions proposed in the literature as useful?

The rest of this paper is structured as follows. After a brief overview of the topic given in Sect. 2, the methodology is introduced in Sect. 3. Section 4 covers the results of the literature survey presenting answers to research questions RQ1 to RQ3. More specifically, it presents what scientific literature considers being problems in mobile application testing, which solutions are proposed, and how much the solutions are used in industry. Section 5 discusses the results of interviews conducted within six companies in order to evaluate how relevant industry considers the problems and solutions mentioned in literature. This addresses research questions RQ4 and RQ5. Section 6 provides a brief discussion of the limitations of the literature survey and case study. The paper ends with a summary of the results and conclusions.

2 Background

The first device that could be considered a smartphone was IBM Simon [5] released in 1994. Smartphones as we know them today started gaining mainstream popularity only in year 2007 when the first iPhone was released [6]. Since then, smartphone sales have skyrocketed [7]. What initially were thought to be just enhanced phones and entertainment devices have now developed into a wide range of different devices capable of performing business tasks, simplifying everyday life and enabling users to be continually connected to their work, social circles and service providers [1, 8]. Mobile devices are challenging conventional computers [9]. Consequently, the criticality of mobile applications has significantly increased [1, 3]. This has forced developers to focus more on the quality of their applications.

Testing of mobile applications incorporates many of the problems inherent to software testing in general. However, mobile devices also have qualities that differentiate them from conventional computers and therefore create testing challenges that are either unique to or more relevant in the case of mobile applications.

The three dominating mobile operating systems (OS) are Android, iOS and Windows Phone. According to net market share [10], Android was the most popular OS in the first quarter of 2016 with a 60 % market share. iOS followed with 32 %. Windows Phone was third with 3 %, followed by Java ME having 2 %.

Android is a free open-source operating system based on the Linux kernel and was released by Google in 2008. Android applications are normally developed in Java, compiled to Java bytecode and then to Dalvik bytecode to be run on Dalvik virtual ma-chine (DVM), with most of the code interpreted during runtime. From version 5.0, DVM has been replaced by Android Runtime (ART) that compiles the application to machine code during installation. Therefore, even though Android applications are commonly developed in Java, they cannot be run on Java Virtual Machine.

The second most popular operating system is iOS, a proprietary, closed source operating system released by Apple in 2007. The iOS operating system can be used only on Apple devices. Applications for iOS are normally developed either in Swift or Objective-C. The core of iOS is based on Darwin, a Unix operating system also used for Apple OS X, and Cocoa Touch is used for the user interface.

Windows Phone (previously Windows Mobile, now Windows 10 Mobile) is a proprietary closed-source operating system developed by Microsoft and released in 2010. Applications for Windows mobile devices can be developed in various languages like C#, .NET, C++ and HTML5. The latest mobile operating system from Microsoft was released as Windows 10 Mobile, reflecting Microsoft's intention to essentially merge the desktop and mobile versions of Windows [11] so that same apps could be run on both of them.

3 Methodology

In this section, we describe our research methods, i.e., how we conducted our literature survey and case study. We used the guidelines provided in [12, 13] as an orientation but didn't follow all of them to the letter.

3.1 Literature Survey

In order to get familiar with the literature, we first conducted an informal search in the ACM Digital Library. We searched for articles related to problems in mobile application testing published in 2007 or later because that was the year when the first iPhone, as well as the first alpha version of Android was released. Since we did not use any additional filtering, we got an excessive amount of results which we ordered based on relevance. We skimmed through the most relevant search results and manually chose 26 articles that seemed relevant to the question in hand by title.

Through reading the initial papers, we learned some additional keywords and search criteria that could be used. We also noticed that most of the results were conference papers and papers that mentioned problems usually also discussed solutions to them. We then conducted a second, more formal and structured search for journal articles. Since some relevant papers might not be indexed by the ACM Digital Library, we conducted the second search in the following four databases:

- ACM Digital Library
- SpringerLink (Computer science)
- Scopus (Computer science)
- ScienceDirect (Computer science)

We applied the following inclusion criteria:

- Only journal articles
- Published 2007 or later
- Full-text is available in the database

We applied the following exclusion criteria:

- Papers that were mainly about hardware-related, low-level communication or network issues, as opposed to end-user mobile applications.
- Papers to which we do not have full text access rights.
- Articles that do not analyse or make new contributions to the testing process itself; for example, if the paper was about developing a non-testing-related mobile application and at the end it was tested just to prove that the application works, then the article is not really about testing, even though it features it.
- Papers that are about mobile web application testing. Since web applications run in a browser or in a browser-like program, they don't inherit many of the challenges that native mobile applications have and are often more similar to web applications meant for desktop devices than to native mobile applications [14].
- Testing techniques that are not meant for consumer-oriented mobile applications.

The exact search queries can be found in Appendix I of [15]. The second, formal, search yielded 374 results, 355 of which were unique. Out of these, 84 were left after manual filtering based on the title. Therefore, the total set of abstracts to read was 26 + 84 = 110. Based on the abstract, 57 papers were discarded, which results in a set of 53 papers to read. While reading we discarded two papers because one had low relevance and one was superseded by another more recent publication. This left us with 50 publications.

Extracting Problems and Solutions. For each of the selected publications, we highlighted problems and solutions mentioned. If a solution was proposed in the paper, we assigned an approximate category to it and wrote the most important keywords concerning the solution to the front page. After reading all publications, we went through all the highlighted parts concerning RQ1 and wrote out all the found problems. Researchers rarely used the word 'problem', but often highlighted 'challenges' to justify the necessity of the solution they were going to propose. Offering a solution clearly shows that they considered the 'challenge' something that needed to be solved, so we counted these as problems. Some problems were also collected from general discussion parts of the papers.

After extracting a list of problems, we went through the papers again to write summaries of the proposed solutions (RQ2). The solutions were based mostly on the highlighted parts and the keywords we had written on the papers while reading, but details often needed to be clarified from other parts of the paper.

Assessing Whether Proposed Solutions were Evaluated in Industry. After we had extracted the list of solutions from the selected papers, we analysed the papers once again with the goal to find evidence that proposed solutions had actually been evaluated or applied in industry (RQ3).

3.2 Case Study

Selection of Industry Professionals. We compiled a set of 23 potentially relevant companies based on a Google search and our existing knowledge about software companies in Estonia. Then we explored web sites of the companies and selected those which fulfilled the following criteria:

- Operate in Estonia
- Deal with testing of native mobile applications. If a company develops native mobile applications, then we assumed that testing is done unless the home page hints that it is outsourced
- Are not a one-person company
- Seem professional enough to pay attention to the testing process

This restricted the list to seven companies, which we contacted. Five of the contacted companies replied and were willing to participate. In addition to these, one of the chosen companies put us in contact with a very suitable, but less known company that we were not aware of, which also agreed to participate. This resulted in a total of 6 companies to interview. We then asked our contacts to suggest interviewees involved in testing native mobile applications. In two cases we used pre-existing in-company contacts to find a suitable person in the company to interview.

Participating Companies. The following companies were selected:

- Fob Solutions: a mobile-oriented quality assurance company that on the side also provides development of web and native mobile applications. Fob Solutions has about 20 testers and some developers who work with Android, iOS and Windows Phone. We talked to the head of quality assurance.
- Testlio: a company that provides a community-based testing service. Testlio manages the testing process and prepares everything necessary, but actual testing is performed by a network of approximately 200 freelance testers who are not employees of Testlio. Testlio works with Android, iOS, Windows Phone and to a lesser degree BlackBerry. In Testlio testing is performed manually and the company doesn't diagnose the found problems. The company does have its own platform to facilitate testing but it mostly has management functionalities, not test running or generation. We interviewed a QA manager that we knew prior to the interview.
- TestDevLab: a quality assurance company that in addition to the more common testing services also provides battery, penetration and data usage testing. About 50 people are involved in Android, iOS and Windows Phone applications testing in TestDevLab. TestDevLab QA engineers are not oriented to a certain platform, therefore my interviewee had worked with different platforms (web, iOS, Android) in different projects. TestDevLab owns a test automation tool called Apimation[1]. The company has its headquarter in Latvia but it is common for their employees to

[1] https://apimation.com.

temporarily move to where the client is located. Therefore, we got a chance to talk to one of their QA engineers who lives in Estonia.
- Wazombi: a company focused on providing end-to-end solutions where everything from electrical engineering to UI design is done in one house. Wazombi works with Android and iOS, but as learned from the interview, most of their Android applications are not Java-based. Instead, Xamarin and C# are used. Xamarin also constitutes the only test generation tool mentioned by case study participants. Since the company is more oriented on development, it has only one person specifically assigned to mobile application testing, whom we interviewed.
- MoonCascade: a company that mainly provides mobile, responsive web and back-end development. From mobile platforms, Android, iOS and Windows Phone are used. There are four people working at mobile application testing. Some testing frameworks like Appium and Selendroid are used for test running. We interviewed the lead of the quality assurance team.
- Mobi Lab: a mobile application design and development company, formerly a part of current parent company Mobi Solutions. They work with Android, iOS and Windows Phone. We interviewed the only dedicated tester, but developers are also responsible for testing the applications that they are making.

Producing a Problem-Solution Matrix. Explaining every solution proposed in the literature to our interviewees would have resulted in unrealistically long and inefficient interviews. Therefore, we planned to only present solutions to problems that interviewees previously identified as being highly relevant to them. Since we did not have any information about the perceived relevance of each problem prior to the interview, it was not possible to choose the set of solutions to explain beforehand. We needed a mapping of problems and solutions that we could use during the interview to choose which solutions to explain.

In order to find out which problems a given solution solves, we used the knowledge of the solutions that we had gained from reading the papers, as well as the problems and challenges that researchers presented as justifications for their solution. For each problem-solution combination there were 4 options:

- 'Y' - a proposed solution significantly contributes to solving the given problem
- 'Partly' - partly solves the problem
- 'Maybe' - might be useful, but more information is needed to know
- Blank - a proposed solution does not address this problem

This resulted in the problem-solution matrix described in Sect. 5. Having this matrix handy allowed us to present exactly those solutions that were related to the most urgent problems highlighted by the interviewee, no matter which problem that was.

The Interview Process. The interview structure was as follows:

1. We introduced the research problem and collected some general information about the company. This information included the number of employees involved in testing mobile applications, whether the company is oriented at testing or development, mobile platforms the company works with, and experience with using or

developing automated solutions for mobile application testing. In addition to this, before showing the list of problems acquired from literature, we asked whether the interviewee sees any notable challenges in mobile application testing.
2. We presented the list of testing problems found from literature and asked the interviewee to rate the relevance of each problem for their actual mobile application testing. The answers were given on a multiple choice scale that also included options for "N/A" and "Already solved". The questionnaire can be found in Appendix II of [15].
3. We looked at those problems the interviewee considered important (marked as "Definitely") and used the Problem-Solution mapping presented in Sect. 5 to extract the set of corresponding solutions proposed in literature. Thereafter, we introduced these solution ideas to the professional and asked which of them seem potentially useful in practice. Since the respondents were only interested in practically applicable solutions and time was scarce, we omitted articles that were very general or too theoretical.

The time planned for each interview was 1.5 h. The first interview part took about 10 min while the duration of the second part was dependent on how fast the interviewee filled out the questionnaire, averaging at about 30 min. Duration of the third part was affected by how many problems the interviewee considered relevant in the questionnaire. Two respondents filled out the questionnaire very fast, which resulted in these interviews taking only 1 h. One interview was extended to 2 h because there were many potentially relevant solutions to present and discuss.

4 Results from the Literature Survey

The results from the literature survey were used to answer research questions RQ1-3:

- RQ1: What are the problems specific to testing of mobile applications as opposed to conventional applications, according to scientific literature?
- RQ2: What are the solutions (methods, tools) proposed by literature, if any?
- RQ3: According to literature, to what extent are these solutions used in industry?

4.1 Findings Related to RQ1

In this sub-section, we give an overview of problems and challenges (in the following subsumed under the term 'problems') that are specific to or especially relevant in the testing of mobile applications (cf. Table 1). We grouped problems according to their core causes. In reality, however, each problem can have more than one cause and, thus, the grouping shown in Table 1 should be taken as an approximation made in an effort to simplify reading. A detailed description of each problem can be found in [15]. In total 49 problems were identified.

Table 1. List of identified problems grouped by core cause

Core cause	Problems with references
Fragmentation (large variety of platforms with different operating systems, hardware, storage, and screen sizes)	P1 [3, 18, 20, 22–25] - P2 [1] - P3 [23, 26] - P4 [27, 28] - P5 [23] - P6 [22, 26] - P7 [22]
External software dependencies	P8 [1, 29] - P9 [30] - P10 [31]
Frequent external communication	P11 [1, 9, 16, 19] - P12 [19, 24, 29, 31–35] - P13 [1, 23, 24, 36] - P14 [1, 37] - P15 [38] - P16 [9, 21, 28, 32] - P17 [9, 39] - P18 [39] - P19 [39]
Variable users and usage contexts	P20 [40, 41] - P21 [21, 42] - P22 [21, 28, 43–45] - P23 [26] - P24 [35, 43] - P25 [21, 43, 46] - P26 [33, 45] - P27 [43]
Fast evolution	P28 [16, 32] - P29 [18, 21, 22, 47] - P30 [9, 46]
Limited resources	P31 [24, 35] - P32 [35] - P33 [1] - P34 [46]
Novelty	P35 [32, 41, 48–50] - P36 [21, 32, 46, 51] - P37 [36] - P38 [41, 51] - P39 [9, 21, 32, 43, 52]
Limitations related to platform implementation.	P40 [23, 29, 39, 48] - P41 [48] - P42 [19, 53] - P43 [54] - P44 [1, 24] - P45 [55] - P46 [32, 56] - P47 [35]
Other problems	P48 [9, 16] - P49 [9]

4.2 Findings Related to RQ2

In this sub-section, we present the tools and methods proposed in the literature for solving the problems described in the previous sub-section (cf. Table 2). The solutions

Table 2. List of identified solutions grouped by type

Type	Solutions with references
Theoretical	S1 [4] - S2 [21] - S3 [27] - S4 [47] -S5 [33] -S6 [57] - S7 [23]
General tools & methods	S8 [35] - S9 [17] - S10 [29] - S11 [31, 34, 48] - S12 [50] - S13 [26]
GUI-based testing	S14 [52, 58] - S15 [16, 59]
Record-and-replay	S16 [56]
Model-based	S17 [40] - S18 [30]
Model-learning	S19 [55, 60] - S20 [61] - S21 [24]
Search-based	S22 [61–64]
Performance testing	S23 [36]
Reliability testing	S24 [38] - S25 [19] - S26 [39]
Compatibility	S27 [22] - S28 [20] - S29 [18]
Usability and user testing	S30 [65] - S31 [32] - S32 [54] - S33 [28] - S34 [46] - S35 [45]
Security testing	S36 [37] - S37 [66] - S38 [25] - S39 [49]

are grouped by type. A detailed description of each solution can be found in [15]. In total, 39 solutions were identified.

4.3 Findings Related to RQ3

In this sub-section, we discuss to what extent proposed solutions (methods and tools) have reportedly been used in industry.

Most of the solutions listed in Sect. 4.2 were evaluated either on one or a few applications familiar to the researchers or on a more representative set of applications acquired from app stores. However, in both cases the evaluation was performed by the researchers themselves, usually in a controlled environment. Only one paper explicitly mentioned that their proposed solution was used in a company, i.e. Swisscom (S13). Also some proposed tools were evaluated on published apps (S16, S19, S21, S25, S37, S39) and one was partly tested on apps currently under development (S39).

However, publications were (co-)authored by individuals with a company affiliation. This applies to S1 (both authors affiliated with Microsoft), S5 (both authors affiliated with Nokia Research Center), S6 (one of the authors affiliated with Fujitsu Laboratories), S8 (all authors affiliated with Microsoft Research), S10 & S11 (one of the authors affiliated with NASA Ames Research Center), S25 (three out of four authors affiliated with Microsoft Research), S29 (first author is one of the founders of the TestDroid testing platform), S30 (one of the two authors affiliated with Ericsson Research), S35 (one of the authors affiliated with Telecom Italia). Even though in these cases it was not mentioned in the paper whether the proposed solution was evaluated in industry, it is highly probable that some of them are used in the companies to which (some of) the authors are affiliated. It is worth noting, however, that most of the affiliations are with research units within large corporations. Therefore, results reported in the related papers might not have been applied in the business units of these companies, and they might not be suitable for problems in the in the rest of the industry, i.e. in small and mid-sized companies.

5 Results from the Case Study

The results from the case study were used to answer research questions RQ4-5:

- RQ4: Does industry consider the solutions proposed in the literature as relevant?
- RQ5: Does industry consider the solutions proposed in the literature as useful?

In order to make the interviews conducted during the case study more efficient, we developed a problem-solution mapping based on the results from the literature survey. For easier viewing, we made the original file available on Dropbox[2]. Note that the columns in grey indicate solutions that were not presented in detail to the interviewees because they related to problems that were not of high priority for the interviewees.

[2] https://www.dropbox.com/s/ia8vgjr7a8ppxkr/Problem-solution%20matrix.ods?dl=0.

5.1 Findings Related to RQ4

Five of the six interviewed companies said that fragmentation was a significant problem in mobile application testing even before seeing the list of problems proposed in the literature. Testlio was the only company that didn't consider fragmentation as a significant problem because their community-based approach already ensures a high number of different platform, OS version, device, and screen size combinations. The interviewee from Testlio mentioned two challenges in mobile application testing. Firstly, there is a lack of fine-grained tools that testers could use to record GUI interactions leading to a fault. The ideal approach would be able to capture videos, screenshots with click positions and have better logs than the current approaches. The second problem was applications that need to be tested in very specific geographical locations, especially on iOS where location information is more difficult to mock than on Android.

The questionnaire answers provided by all of the participating companies are listed in Table 3. Questions corresponding to each question number can be found in Appendix II of [15]. Basically, interviewees were asked to tell for each problem P1 to P49 mentioned in the literature whether it is also a problem for them. The answer choices were: 'Definitely' – 'Maybe' – 'Probably not' – 'Definitely not', plus the answer options 'n/a' (not applicable) and 'Solved' (in case the problem existed but has been solved in the meanwhile). The first column shows the problem ID together with an indicator representing the relevance of the problem for the case companies. The symbol '++' indicates that at least four companies found the problem relevant (at least four times 'Definitely'), '+' indicates that two or three companies found the problem relevant (two or three times 'Definitely'), and '−' indicates that no company felt strongly that the problem is relevant (none of the companies stated 'Definitely').

The responses from the three companies that mainly focus on testing are displayed on the left while companies whose main area of business is mobile application development are displayed in the right half of Table 3.

The surveyed companies didn't agree on the relevance of any of the listed problems. However, some patterns can be pointed out. 18 of 49 listed problems were not considered relevant by all case companies. For example, testing inter-application communications (P14) and more sophisticated testing techniques like simulating external dependencies (P10), automatic page-load detection (P40) and ensuring completely clean application restart (P43) were never mentioned to be definitely relevant. None of the companies considered the lack of testing methods, tools or theory a significant problem (P33–P35, P37). Modelling applications before testing was not popular and some companies mentioned that testing on all devices is not needed because a set of supported devices is chosen before development. It can be argued that this practice of choosing a set of supported devices itself shows that testing an application on all potentially suitable devices is too difficult, expensive or time-consuming.

Some problems not considered highly relevant by companies mainly focusing on development were still considered potentially problematic by testing companies. These included acquiring a mental model of a complex application (P9), the unpredictability of external dependencies during testing (P12), ignoring unexpected user behavior

Table 3. Relevance of problems to our case companies

Problem	Testing companies			Development companies		
	Fob Solutions	Testlio	TestDevLab	Wazombi	MoonCascade	Mobi Lab
P1 ++	Definitely	Solved	Definitely	Maybe	Definitely	Definitely
P2 +	Solved	N/A	Probably not	Definitely	Definitely	Maybe
P3 +	Definitely	N/A	Definitely not	Maybe	N/A	Definitely
P4	Maybe	Solved	Definitely not	Definitely	Probably not	Definitely not
P5 +	Maybe	Solved	Definitely	Definitely not	Maybe	Definitely
P6 ++	Definitely	Solved	Definitely	Definitely not	Definitely	Definitely
P7 +	Definitely not	N/A	Definitely not	Solved	Definitely	Definitely
P8 +	Definitely not	Definitely	Definitely	Probably not	Definitely	Probably not
P9	Probably not	Maybe	Definitely	Definitely not	Definitely not	Definitely not
P10 −	Maybe	Maybe	Probably not	Maybe	Probably not	Definitely not
P11	Probably not	N/A	Definitely	Solved	Maybe	Probably not
P12 +	Definitely	Maybe	Definitely	Probably not	Probably not	Definitely not
P13 +	Definitely not	N/A	Definitely	Solved	Definitely	Definitely
P14 −	Probably not	Maybe	Probably not	Probably not	Maybe	Definitely not
P15 −	Maybe	Solved	Definitely not	Definitely not	Probably not	Definitely not
P16 +	N/A	Definitely	Probably not	Definitely not	Maybe	Definitely
P17 −	Maybe	Maybe	Maybe	Definitely not	Maybe	Definitely not
P18	Probably not	N/A	Definitely	Maybe	Maybe	Definitely not
P19 +	Probably not	N/A	Definitely not	Probably not	Definitely	Definitely
P20 +	Definitely	Definitely	Probably not	Probably not	Definitely	Maybe
P21 +	Maybe	Definitely	Definitely	Solved	Probably not	Maybe
P22	Maybe	Maybe	Definitely not	Solved	Definitely	Probably not
P23 +	Definitely	N/A	Definitely not	Definitely	Definitely	Maybe
P24	Maybe	Solved	Probably not	Probably not	Definitely	Probably not
P25	Probably not	Definitely	Definitely not	Probably not	Maybe	Probably not
P26	Probably not	Definitely	Probably not	Definitely not	Probably not	Definitely not
P27 +	Definitely	Definitely	Maybe	Probably not	Definitely	Definitely not
P28 +	Definitely	Maybe	Definitely	Probably not	Probably not	Definitely not
P29 −	Probably not	N/A	Maybe	Maybe	Maybe	Maybe
P30 −	Probably not	N/A	Maybe	Definitely not	Definitely not	Definitely not
P31	Maybe	Maybe	Maybe	Maybe	Definitely	Definitely not
P32	Definitely not	Maybe	Definitely	Definitely not	Probably not	Definitely not
P33 −	Probably not	N/A	Definitely not	Solved	Probably not	Definitely not
P34 −	Probably not	Maybe	Definitely not	Solved	Maybe	Definitely not
P35 −	Maybe	N/A	Definitely not	Definitely not	Probably not	Definitely not
P36 −	Solved	Maybe	Probably not	Solved	Solved	Definitely not
P37 −	Solved	Maybe	Probably not	Probably not	Solved	Definitely not
P38 +	N/A	N/A	Definitely	Definitely	Definitely not	N/A
P39 −	N/A	N/A	N/A	N/A	N/A	Probably not
P40 −	Probably not	N/A	Definitely not	Definitely not	Probably not	Probably not
P41 −	Solved	Maybe	Definitely not	Solved	Solved	Solved
P42 +	N/A	N/A	Definitely	Definitely not	Definitely	Probably not
P43 −	Probably not	Probably not	Definitely not	Definitely not	Solved	Solved
P44	Solved	Definitely	Definitely not	Solved	Solved	Solved
P45 −	N/A	N/A	N/A	Solved	Solved	Probably not
P46 −	Definitely not	Probably not	Definitely not	Probably not	Maybe	Definitely not
P47 −	Definitely not	Maybe	Probably not	Solved	Solved	Probably not
P48	N/A	Definitely	Probably not	Maybe	Probably not	Definitely not
P49 +	N/A	Maybe	Definitely not	Definitely	Definitely	Definitely not

(P15), users' variable mobile device usage experience (P21), insufficient OS failure logging (P30) and the usability and accessibility aspects of complex input mechanisms (P32). We suppose that testing companies do testing more thoroughly or are just more aware of their testing processes. In addition to this, if testing is performed by developers or at least in the same company, then the people doing the testing probably have a better overview of how the application is intended to function.

Since Testlio was the only company that actively uses a community-based testing approach as opposed to just testing in-house, different problems are sometimes considered relevant by them. Notably, fragmentation (P1) and the large number of test devices to buy (P6) that were considered problems by most companies are not a problem for Testlio because their testers use personal devices for testing. On the other hand, they are subject to some challenges that are not relevant for any other companies. For example, since their testers are working remotely, they need more advanced UI recording tools (P44) than companies that perform testing locally. The large number of test devices (P6) is also not a problem for Wazombi who mainly provides end-to-end services that include both hardware and software development.

The lack of design principles (P36) was considered already solved by guidelines provided by mobile operating systems and cross-platform principles were not considered necessary. The problem of not being able to modify a mobile application after installing (P41) was considered solved by either automatic updates provided by app stores or specialized software that can be embedded into applications for A/B testing. The fact that testing is expected to be faster for mobile applications was not considered a big obstacle because mobile applications on average were said to contain less functionality than desktop applications.

5.2 Findings Related to RQ5

In this sub-section we present the results regarding the extent to which industry professionals interviewed in our case study consider the solutions provided in literature potentially useful in practice (cf. Table 4).

11 solutions were presented to industry professionals based on the problems that they considered relevant. None of the solutions were uniformly accepted by the companies, although solution S26 (An approach for amplifying exception handling code) was considered useful by all the companies that found it applicable.

Solution S8 (MobiBug) was presented to all companies. Respondents from Testlio, TestDevLab and Mobi Lab considered it potentially useful. Wazombi commented that since even devices of the same model don't function completely identically, a model that assumes they do might be inaccurate. MoonCascade said that nowadays OS built-in logging is already more fine-grained than stated in the article and 3rd party libraries for monitoring fault configurations exist, there-fore this solution already exists.

None of the interviewees considered solution S9 (iTest) a useful innovation. Some mentioned that a solution of this kind already exists. Others were skeptical of whether this approach would work well because people rarely give any feedback when things work (Mobi Lab) and it is difficult to ensure a full variety of user profiles in registered testers (Testlio). One company expressed that the success of this approach highly

depends on the tester incentive mechanism. Therefore, the technical solution alone does not bring much value.

Solution S10 (Symbolic execution of Android apps) was presented to two companies. TestDevLab considered it potentially useful while Wazombi said the solution would not applicable for them because it can only handle applications written in Java.

Solution S11 (JPF-Android) is not applicable to Wazombi whose applications are not Java-based. TestDevLab was hesitant about whether this would work and MoonCascade said the tool would be useful if it could emulate drivers of all kinds of sensors and developers manage to keep the tool up to date with new OS versions.

Most of the companies liked the concept of solution S16 (VALERA) and thought it would be useful. Wazombi was more skeptical due to the fact that VALERA does not record memory operations.

Solution S21 (Tool with 2 approaches for automated model-based testing) was only presented to the Testlio representative who found it useful.

Solution S23 (Unit-testing performance) was presented to Mobi Lab and TestDevLab. Mobi Lab found it useful while the latter commented that the duration of method execution can depend on things outside the developer's control, e.g. network conditions, therefore duration of execution cannot be accredited to just performance.

TestDevLab considered solution S25 (VanarSena) useful. Mobi Lab said that it is already used for Windows Phone applications.

Solution S26 (Approach for amplifying exception-handling code) was presented to 4 companies. For Wazombi, this solution wasn't applicable due to being Java-based, but the others considered it useful.

Regarding solution S28 (Knowledge base for compatibility testing) Mobi Lab thought it could work and Wazombi said it could work partly, for API version based problems. MoonCascade was skeptical about how an appropriate level of granularity could be set for recording results – if every combination of application version, device, OS version, etc. would be recorded separately then very few queries would get a reply from the database while in other cases there is a high probability of over-generalization. TestDevLab said that a solution like this is probably already integrated to some testing software.

Solution S29 (TestDroid) was not very well-received. Sob Solutions and Mobi Lab said that this solution already exists. TestDroid itself is available online and is not the only cloud-based testing platform. Testlio said that in principle the approach is plausible while TestDevLab thought it might be useful only for small teams that do not have access to an extensive set of test devices. MoonCascade was also of the opinion that for companies of significant size, it is better to have their own set of devices as cloud-based solutions are expensive, unreliable and do not have support for various test styles and frameworks.

In total, there were only 3 solutions that were considered relevant by all companies to which they were presented and who found them applicable, i.e. S10 (2 companies), S21 (1 company), and S26 (3 out of 4 companies). None of the respondents considered S9 a good solution and most were skeptical about S28 and S29.

Upon hearing the solution concepts, many interviewees expressed that the general concept of the solution is familiar to them or already exists. However, they had not marked the corresponding problems as 'Solved' in the questionnaire part of the

interview. This implies that the concepts they already knew either do not fully solve the proposed problem or the professionals have not thought about using this concept to solve the given problem. The latter is compatible with our general observation that companies seem to consider the new challenges of mobile application testing inevitable and thus do not think about the possibility to eliminate them with the help of new methods and tools. In that sense, help from the scientific community could actually help if they were considered more seriously.

Several proposed solutions were considered to be either too theoretical, general or relating to problems that were already solved by the case companies. The latter point can to some degree attributed to the fact that we included articles published from 2007, i.e. almost 10 years ago, in the literature study. However, the two least supported solution concepts, TestDroid and iTest, were published in 2014 and 2012, respectively. Therefore, either the field of mobile application testing is developing so fast that only papers published less than two years ago provide practical value for companies or research is sometimes detached from the current problems in industry.

Another observation was that additional attention could be paid to the fact that companies use different tools for developing and testing mobile applications. For example, not all Android applications are developed in Java and cloud-testing platforms would be more useful if they supported different testing frameworks.

6 Discussion of Limitations

In the following, we summarize the limitations of our literature survey and case study.

6.1 Limitations of the Literature Survey

The literature study was performed by one person (the first author) within a limited amount of time (four weeks). Due to this resource limitation a systematic literature study following the guidelines defined in [12] to the letter was not viable. Therefore, it is possible that some relevant papers were either not found or filtered out incorrectly when applying the defined search strings and inclusion/exclusion criteria. Also, there was not enough time for conducting a thorough quality assessment of the included papers. Additionally, since the information was extracted from papers by just one person and without previously specifying what constitutes a problem or solution, the analysis is bound to be somewhat subjective, although we tried to mitigate this problem by having another person (the second author) review the results of the paper identification and selection. The limitation applies to the linking of problems to their potential solutions in the problem-solution matrix. Lastly, since papers from 2007 to 2016 were used in the study, it is possible that some of the problems mentioned in the literature have been solved and thus the problems have become obsolete. Also, Android was significantly more represented than other platforms in the set of found papers and therefore many of the found problems and solutions concern mobile applications on the Android platform.

Table 4. Relevance of solutions

Solution	Company					
	Fob Solutions	Testlio	Wazombi	TestDevLab	MoonCascade	Mobi Lab
S8	No	Yes	No	Yes	Exists	Yes
S9	Exists	Partly	No	No	Exists	No
S10			n/a	Yes		
S11			n/a	Maybe	Maybe	
S16		Yes	No	Yes	Yes	Yes
S21		Yes				
S23				No		Yes
S25				Yes		Exists
S26			n/a	Yes	Yes	Yes
S28			Partly	Exists	No	Yes
S29	Exists	Yes		Partly	No	Exists

6.2 Limitations of the Case Study

In our case study, we used [13] as a guideline but due to time and resource constraints, we didn't follow all recommendations to the letter. While both testing and development oriented companies in Estonia were included in the study, the initial list of companies was compiled mostly opportunistically, i.e. where the first author had some previous knowledge and (in some cases) personal contacts. Therefore, it is likely that the participants of this study are not fully representative for all companies in Estonia doing mobile application testing. During the interviews, participants were asked to assess the potential suitability of some solutions proposed in literature. Since it would be unreasonable to expect participants to read the relevant scientific articles, we shortly explained each solution concept that the interviewees were asked to assess on the spot. As a result, our personal bias and the quality of our explanations might have had an effect on the perceived usefulness of the solutions. Due to time constraints not all potential solutions were presented. In each case, the decision of which solutions to present was made based on the prioritization of problems and using the problem-solution matrix. This creates the possibility that the set of solutions proposed to the case companies might not have been complete.

7 Summary and Conclusion

We conducted a two-staged study involving a literature survey and a case study to find answers to five research questions concerning challenges and solutions of mobile application testing as seen by researchers and industry.

In the attempt to answer RQ1 and RQ2, 49 problems and 39 potential solutions were extracted in our literature survey. These lists answer research question 1 and 2, respectively. For RQ3, the result is less clear. Even though only one paper specified that the proposed solution is already used in industry, it is likely that some of the others

are as well, considering that many authors were associated with companies active in the industry. Therefore, it can be said that the solutions are used in industry, but the extent of this usage cannot be adequately determined just based on scientific litera-ture. For RQ4, none of the problems mentioned in literature were considered uniform-ly relevant by all industry professionals. However, most companies considered frag-mentation a serious problem and usually mentioned it before being handed the ques-tionnaire. Many of the problems mentioned in literature were not considered im-portant by our case companies. Regarding RQ5, many of the solutions proposed in literature were too general, too little evaluated, or too little related to the most relevant problems, to be explained to and discussed with the professionals in sufficient detail. And of those that were presented and discussed, only a subset was uniformly consid-ered useful while others were said to already exist (i.e., are already implemented) or have significant shortcomings (and thus would not be considered for implementation.

In conclusion, research literature is addressing some problems that are considered very important by our case companies. However, there seems to be room for making research more useful for industry since many of the currently proposed solutions are considered as too much conceptual and too little practical by professionals.

Acknowledgements. We would like to thank Fob Solutions, Testlio, Mobi Lab, Wazombi, TestDevLab and MoonCascade. This research was supported by the Estonian Research Council.

References

1. Muccini, H., Di Francesco, A., Esposito, P.: Software testing of mobile applications: challenges and future research directions. In: Proceedings of AST 2012, Piscataway, NJ, USA (2012)
2. Paul, S.: Role of mobile handhelds in redefining how we work, live and experience the world around us: some challenges and opportunities. In: Proceedings of SIGCOMM 2010, New Delhi, India (2010)
3. Wasserman, A.I.: Software engineering issues for mobile application development. In: Software Engineering Issues for Mobile Application Development, Santa Fe, New Mexico, USA (2010)
4. Santos, A., Correia, I.: Mobile testing in software industry using agile: challenges and opportunities. In: Proceedings of ICST 2015, Graz, Austria (2015)
5. N. T., Did you know what was the first smartphone ever? PhoneArena, 31 July 2014. http://www.phonearena.com/news/Did-you-know-what-was-the-first-smartphone-ever_id58842. Accessed 10 May 2016
6. Apple Inc., Apple Reinvents the Phone with iPhone. http://www.apple.com/pr/library/2007/01/09Apple-Reinvents-the-Phone-with-iPhone.html. Accessed 12 May 2016
7. Statista Inc., Number of smartphones sold to end users worldwide from 2007 to 2015 (in million units). http://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/. Accessed 14 May 2016
8. Martinie, C., Palanque, P.: Design, development and evaluation challenges for future mobile user interfaces in safety-critical contexts. In: Proceedings of the 2015 Workshop on Future Mobile User Interfaces, Florence, Italy (2015)

9. Corral, L., Sillitti, A., Succi, G.: Software assurance practices for mobile applications. Computing **97**(10), 1001–1022 (2015)
10. Net Applications, Mobile/Tablet Operating System Market Share January–March 2016. https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qpsp=68&qpnp=1&qptimeframe=Q&qpmr=10&qpdt=0&qpct=3. Accessed 12 May 2016
11. Albanesius, C.: Nadella Raises Eyebrows With Plans to 'Streamline' Windows, PC Magazine, 23 July 2014. http://www.pcmag.com/article2/0,2817,2461253,00.asp. Accessed 13 May 2016
12. Kitchenham, B.A., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Technical report EBSE-2007-01, School of Computer Science and Mathematics, Keele University (2007)
13. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Softw. Eng. **14**(2), 131–164 (2009)
14. Charland, A., Leroux, B.: Mobile application development: web vs. native. Commun. ACM **54**(5), 49–53 (2011)
15. Samuel, T.: Problems and solutions in mobile application testing, MSc thesis, University of Tartu, Estonia (2016). https://comserv.cs.ut.ee/ati_thesis/datasheet.php?id=54422&year=2016
16. Zhifang, L., Bin, L., Xiaopeng, G.: Test automation on mobile device. In: Proceedings of the 5th Workshop on Automation of Software Test, Cape Town, South Africa (2010)
17. Yan, M., Sun, H., Liu, X.: ITest: testing software with mobile crowdsourcing. In: Proceedings of CrowdSoft 2014, Hong Kong, China (2014)
18. Kaasila, J., Ferreira, D., Kostakos, V., Ojala, T.: Testdroid: automated remote UI testing on android. In: Proceedings of 11th International Conference on Mobile and Ubiquitous Multimedia, Ulm, Germany (2012)
19. Ravindranath, L., Nath, S., Padhye, J., Balakrishnan, H.: Automatic and scalable fault detection for mobile applications. In: Proceeidngs of MobiSys 2014, Bretton Woods, New Hampshire, USA (2014)
20. Ham, H., Park, Y.: Designing knowledge base mobile application compatibility test system for android fragmentation. Intl. J. Softw. Eng. Appl. **8**(1), 303–314 (2014)
21. Tang, L., Yu, Z., Zhou, X., Wang, H., Becker, C.: Supporting rapid design and evaluation of pervasive applications: challenges and solutions. Pers. Ubiquit. Comput. **15**(3), 253–269 (2011)
22. Galindo, J.A., Turner, H., Benavides, D., White, J.: Testing variability-intensive systems using automated analysis: an application to Android. Softw. Qual. J. **42**(2), 365–405 (2014)
23. Baride, S., Dutta, K.: A cloud based software testing paradigm for mobile applications. SIGSOFT Softw. Eng. Notes **36**(3), 1–4 (2011)
24. Azim, T., Neamtiu, I.: Targeted and depth-first exploration for systematic testing of Android apps. ACM SIGPLAN Not. **48**(10), 641–660 (2013)
25. Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y., Octeau, D., McDaniel, P.: FLOWDROID: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for Android apps. ACM SIGPLAN Notes **49**(6), 259–269 (2014)
26. Haller, K.: Mobile testing. SIGSOFT Softw. Eng. Notes **38**(6), 1–8 (2013)
27. Bastien, J.C.: Usability testing: a review of some methodological and technical aspects of the method. Intl. J. Med. Inf. **79**(4), e18–e23 (2010)
28. Ma, X., Yan, B., Chen, G., Zhang, C., Huang, K., Drury, J., Wang, L.: Design and implementation of a toolkit for usability testing of mobile apps. Mob. Netw. Appl. **18**(1), 81–97 (2013)

29. Mirzaei, N., Malek, S., Pasareanu, C.S., Esfahani, N., Mahmood, R.: Testing Android apps through symbolic execution. SIGSOFT Softw. Eng. Notes **37**(6), 1–5 (2012)
30. Kim, H.-K.: Hybrid model based testing for mobile applications. Intl. J. Softw. Eng. Appl. **7** (3), 223–238 (2013)
31. van der Merwe, H., Tkachuk, O., van der Merwe, B., Visser, W.: Generation of library models for verification of android applications. SIGSOFT Softw. Eng. Notes **40**(1), 1–5 (2015)
32. Hussain, A., Hashim, N., Nordin, N., Tahir, H.: A metric-based evaluation model for applications on mobile phones. J. Inf. Commun. Technol. **12**(1), 55–71 (2013)
33. Koivisto, E.M.I., Suomela, R.: Using prototypes in early pervasive game development. In: Proceedings of ACM SIGGRAPH Symposium on Video Games, San Diego, California, USA (2007)
34. van der Merwe, H., Tkachuk, O., Nel, S., van der Merwe, B., Visser, W.: Environment modeling using runtime values for JPF-Android. SIGSOFT Softw. Eng. Notes **40**(6), 1–5 (2015)
35. Agarwal, S., Mahajan, R., Zheng, A., Bahl, V.: Diagnosing mobile applications in the wild. In: Proceedings of ACM SIGCOMM Workshop on Hot Topics in Networks, Monterey, California (2010)
36. Kim, H., Choi, B., Yoon, S.: Performance testing based on test-driven development for mobile applications. In: Proceedings of ICUIMC 2009, Suwon, South Korea, (2009)
37. Ceccato, M., Avancini, A.: Security testing of the communication among Android applications. In: Proceedings of AST 2013, San Francisco, CA, USA (2013)
38. Adamsen, C.Q., Mezzetti, G., Moller, A.: Systematic execution of android test suites in adverse conditions. In: Proceedings of ISSTA 2015, Baltimore, MD, USA (2015)
39. Zhang, P., Elbaum, S.: Amplifying tests to validate exception handling code: an extended study in the mobile application domain. ACM Trans. Softw. Eng. Methodol. **23**(4), 32:1–32:28 (2014)
40. De Cleva Farto, G., Endo, A.: Evaluating the model-based testing approach in the context of mobile applications. Electron. Notes Theor. Comput. Sci. **314**, 3–21 (2015)
41. Zapata, B.C., Fernandez-Aleman, J.L., Idri, A., Toval, A.: Empirical studies on usability of mHealth apps: a systematic literature review. J. Med. Syst. **39**(2), 1–19 (2015)
42. Diewald, S., Geilhof, B., Siegrist, M., Lindemann, P., Koelle, M., Halle, M., Kranz, M.: Mobile AgeCI: potential challenges in the development and evaluation of mobile applications for elderly people. In: Computer Aided Systems Theory – EUROCAST 2015, Las Palmas, Spain (2015)
43. Oulasvirta, A.: Rethinking experimental designs for field evaluations. IEEE Pervasive Comput. **11**(4), 60–67 (2012)
44. Biel, B., Grill, T., Gruhn, V.: Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. J. Syst. Softw. **83** (11), 2031–2044 (2010)
45. Rapp, A., Cena, F., Gena, C., Marcengo, A., Console, L.: Using game mechanics for field evaluation of prototype social applications: a novel methodology. Behav. Inf. Technol. **35** (3), 184–195 (2015)
46. Billi, M., Burzagli, L., Catarci, T., Santucci, G., Bertini, E., Gabbanini, F., Palchetti, E.: A unified methodology for the evaluation of accessibility and usability of mobile applications. Univ. Access Inf. Soc. **9**(4), 337–356 (2010)
47. Nascimento, L.H.D., Machado, P.D.: An experimental evaluation of approaches to feature testing in the mobile phone applications domain. In: Proceedings of DOSTA 2007: in Conjunction with the 6th ESEC/FSE Joint Meeting, Dubrovnik, Croatia (2007)

48. van der Merwe, H., van der Merwe, B., Visser, W.: Verifying android applications using Java pathfinder. SIGSOFT Softw. Eng. Notes **37**(6), 1–5 (2012)
49. Salva, S., Zafimiharisoa, S.R.: APSET, an Android aPplication SEcurity Testing tool for detecting intent-based vulnerabilities. Intl. J. Softw. Tools Technol. Transfer **17**, 201–221 (2015)
50. Aranha, E., Borba, P.: Estimating manual test execution effort and capacity based on execution points. Intl. J. Comput. Appl. **31**(3), 167–172 (2009)
51. Serra, L.C., Carvalho, L.P., Ferreira, L.P., Vaz, J.B.S., Freire, A.P.: Accessibility evaluation of e-government mobile applications in Brazil. Procedia Comp. Sci. **37**, 348–357 (2015)
52. Morgado, I.C., Paiva, A.C.R.: Test patterns for android mobile applications. In: Proceedings of the 20th European Conference on Pattern Languages of Programs, Kaufbeuren, Germany (2015)
53. Wang, X.S., Balasubramanian, A., Krishnamurthy, A., Wetherall, D.: Demystifying page load performance with WProf. In: Proceedings of the 10th USENIX Conference on Networked Systems Design and Implementation, Lombard, IL (2013)
54. Hutflesz, P., Holzmann, C.: Multivariate testing of native mobile applications. In: Proceedings of MoMM 2014, Kaohsiung, Taiwan (2014)
55. Choi, W., Necula, G., Sen, K.: Guided GUI testing of Android apps with minimal restart and approximate learning. ACM SIGPLAN Not. **48**(10), 623–639 (2013)
56. Hu, Y., Azim, T., Neamtiu, I.: Versatile yet lightweight record-and-replay for Android. In: Proceedings of 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, vol. 50(10), pp. 349–366 (2015)
57. Gao, J., Tsai, W.-T., Paul, R., Bai, X., Uehara, T.: Mobile testing-as-a-service (MTaaS) - infrastructures, issues, solutions and needs. In: Proceedings of 2014 IEEE 15th International Symposium on High-Assurance Systems Engineering, Miami, FL, USA (2014)
58. Costa, P., Paiva, A.C.R., Nabuco, M.: Pattern based GUI testing for mobile applications. In: 9th International Conference on the Quality of Information and Communications, Guimaraes, Portugal (2014)
59. Bo, J., Xiang, L., Xiaopeng, G.: MobileTest: a tool supporting automatic black box test for software on smart mobile devices. In: Proceedings of AST 2007, Washington, DC, USA (2007)
60. Google, Performance focus. http://developer.android.com/about/versions/lollipop.html#Perf. Accessed 8 Apr 2016
61. Amalfitano, D., Fasolino, A.R., Tramontana, P., Ta, B.D., Memon, A.M.: MobiGUITAR: automated model-based testing of mobile apps. IEEE Softw. **32**(5), 53–59 (2015)
62. Amalfitano, D., Amatucci, N., Fasolino, A.R., Tramontana, P.: AGRippin: a novel search based testing technique for android applications. In: Proceedings of 3rd International Workshop on Software Development Lifecycle for Mobile, Bergamo, Italy (2015)
63. Amalfitano, D., Amatucci, N., Fasolino, A.R., Tramontana, P., Kowalczyk, E., Memon, A. M.: Exploiting the saturation effect in automatic random testing of android applications. In: Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems, Florence, Italy (2015)
64. Amalfitano, D., Fasolino, A.R., Tramontana, P., De Carmine, S., Memon, A.M.: Using GUI ripping for automated testing of android applications. In: Proceedings of ASE 2012, Essen, Germany (2012)
65. Bergvall-Kareborn, B., Larsson, S.: A case study of real-world testing. In: Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia, Umeå, Sweden (2008)
66. Guo, C., Xu, J., Yang, H., Zeng, Y., Xing, S.: An automated testing approach for inter-application security in android. In: Proceedings of AST 2014, Hyderabad, India (2014)