Journal of Systems and Software, Volume 117, July 2016, pp 334–356, (c) 2016 Elsevier.

A Systematic Mapping Study of Mobile Application Testing Techniques

Samer Zein[1], Norsaremah Salleh[1], John Grundy[2]

[1]Department of Computer Science,
Kulliyyah of Information & Communication Technology, IIUM, Malaysia
[1]samer.m.zain@gmail.com, [1]norsaremah@iium.edu.my

[2]School of Software and Electrical Engineering
Swinburne University of Technology
jgrundy@swin.edu.au

ABSTRACT

The importance of mobile application specific testing techniques and methods has been attracting much attention of software engineers over the past few years. This is due to the fact that mobile applications are different than traditional web and desktop applications, and more and more they are moving to being used in critical domains. Mobile applications require a different approach to application quality and dependability and require an effective testing approach to build high quality and more reliable software. We performed a systematic mapping study to categorize and to structure the research evidence that has been published in the area of mobile application testing techniques and challenges that they have reported. Seventy nine (79) empirical studies are mapped to a classification schema. Several research gaps are identified and specific key testing issues for practitioners are identified: there is a need for eliciting testing requirements early during development process; the need to conduct research in real-world development environments; specific testing techniques targeting application life-cycle conformance and mobile services testing; and comparative studies for security and usability testing.

Keywords

Systematic mapping, mobile application testing, mobile testing, software testing.

1. INTRODUCTION

Smartphones, also known as Smart mobile terminals, are high-end mobile phones that are built on mobile operating systems and offer advanced computing and connectivity. Modern smartphones have stronger processors, growing memories, high resolution touch-screens, richer sensors, GPS, high-speed data access through wi-fi and so forth (Canfora et al., 2013) (Lu et al., 2012). Due to the fact that much more computing power has been incorporated into smartphones and mobile devices in the past few years, they have become very commonly used in everyday life. Mobile applications, also known as mobile apps, are software applications that are developed to run on smartphones and mobile devices. Compared to desktop and web applications, mobile applications have to deal with specific challenges. For instance, mobile applications have to process inputs from users as well as inputs from constantly changing contexts. Additionally, smartphones and mobile devices are still limited in their resources compared to modern personal computers and laptops. Further, there is a large diversity of mobile operating systems, and the same operating system gets upgraded regularly and in relatively short time periods (Zhifang et al., 2010b).

Mobile applications nowadays are not developed to only serve the entertainment sector, but also target safety and time critical domains such as payment systems, m-government, military and mobile health initiatives to mention a few (Muccini et al., 2012) (Payet and Spoto, 2012). As mobile applications have been developed to address more and more critical domains, they are not only becoming more complex to develop, but also more difficult to test and to validate (Nagowah and Sowamber, 2012). According to (Muccini et al., 2012) there are several open research issues regarding testing of mobile and smart-phone software applications. Among these issues is that mobile applications are inherently different from traditional software applications and therefore require specialized testing techniques and methods.

As far as we are aware, there are currently no available comprehensive systematic review studies in the area of mobile and smart-phone application testing. Our initial informal literature searches found very little evidence on mobile applications testing; this also provided motivation for conducting a rigorous systematic mapping study e.g. (Muccini et al., 2012), (Harrison et al., 2013). Inspired by a study on research directions for mobile application testing (Muccini et al., 2012), **this study provides a comprehensive and in-depth mapping study using a well-defined methodology to build a new classification scheme and structures the research area of mobile application testing**. Additionally, our mapping study collects, interprets and analyzes all related evidence for empirical studies addressing challenges, approaches, methods or techniques of testing mobile and smart-phone applications.

This study also aims to highlight important research gaps in the areas of mobile application testing. A total of 79 studies (see Appendix A for the list of included studies) were selected for our mapping study after going through three (3) filtration steps. We present the synthesis of evidence based on FIVE (5) classification sub-categories: i) usability testing, ii) test automation; iii) context-awareness, iv) security and v) general category. Several research gaps are also reported and discussed.

The remainder of this paper is organized as follows: Section 2 presents the motivation and the overview of related work for this study. Section 3 describes briefly the methodology of our mapping study. Section 4 presents the results from the mapping study followed by a discussion in Section 5. Finally, Section 6 concludes our work.

2. MOTIVATION & RELATED WORK

During our search of the literature, we found one systematic mapping study as well as several informal reviews within the area of mobile and smart phone application testing. The systematic mapping study presented by (Méndez-Porras et al., 2015) structures studies under testing approaches, testing techniques and empirical assessments. However, the study focus is limited only at the test automation area. Further, their study does not include clear inclusion/exclusion criteria, making it subjective to biased selection. In contrast our study is comprehensive as it focuses on several areas of interest such as, test automation, usability, context-awareness, and security testing. Additionally, our study has a well defined protocol and investigates important issues of mobile application testing such as life cycle conformance testing, mobile services testing, and testing metrics.

A study conducted by (Muccini et al., 2012) applies an informal review process to answer research questions regarding mobile application testing challenges and consequently suggests further research directions. The study defines two types of mobile applications then discusses thoroughly peculiarities of these applications and how these peculiarities derive specialized research on mobile application testing. The study also identifies several research gaps in the areas of mobile services testing, test automation and test integration for mobile applications.

Another study by (Harrison et al., 2013) conducted a small and informal literature review in the specific area of mobile usability models. The study argues that most existing prominent usability models for mobile applications are incomplete since they only focus on three usability attributes; effectiveness, efficiency and satisfaction; and neglect other important usability attributes such as cognitive overload. In order to address this issue, the study proposes a new usability model known as PACMAD (People At the Center of Mobile Application Development). According to the study, PACMAD is a more comprehensive model as it contains important attributes from different usability models. In order to evaluate PACMAD model, the study conducted a literature review and compiled a set of usability studies to examine which of the usability attributes defined in PACMAD were used by those studies.

In another relatively old review study conducted by (Looije et al., 2007), a review of research done on usability of maps on mobile devices is discussed. Their study focuses on reviewing the research done to solve technical, environmental and social challenges of mobile maps application usage.

The study by (Joorabchi et al., 2013) conducts a qualitative research approach based on grounded theory to gain an understanding of real challenges faced by real world mobile application developers of different mobile platforms. Their study provides an interesting overview of current challenges faced by developers such as building native mobile apps for different platforms, slow emulators, and lack of analysis, monitoring and testing tools. To elaborate more in the area of testing, the study reports that manual testing is the most prevalent practice compared to automatic testing. Further, test engineers have to conduct separate testing processes for each platform. Additionally, most unit testing frameworks do not provide interfaces to mobile-specific capabilities, such as GPS and sensors.

In a study by (Gao et al., 2014a), a comprehensive discussion of mobile application (native and web-based) testing infrastructures and related testing approaches are discussed in details. In their study, the authors discuss specific mobile application testing requirements as well as available testing infrastructures such as emulation, device, cloud and crowd based. Then the advantages and limitations of each infrastructure is discussed and analysed. Their study also provides a discussion of available state-of-the-art tools and processes for testing native and web-based mobile applications. The paper concludes with a brief discussion of challenges, issues and needs for mobile application test engineers.

In another study by (Gao et al., 2014b) an informative tutorial and discussion on mobile testing as a service (MTaaS) is presented. This study proposes TaaS (Testing as a Service) infrastructure to support cloud-based mobile testing in two different approaches: i) mobile device test cloud and ii) emulation-based test cloud. The main objective of the study is to address three (3) major testing challenges in the area of mobile applications: i) high costs in current mobile testing environments; ii) lack of mobile scalability support; and iii) the complexity and harness due to diversity in mobile devices, platforms and environments.

A recent study by (Starov et al., 2015) conducted a survey to report a set of cloud services for mobile testing. The set of cloud services described by the study is divided into three (3) types: i) device clouds; ii) services to support application development lifecycle management; and iii) testing tools categorized based on testing techniques. The study argues that mobile testing over a cloud is very important, at the same time, hard to research. The study concludes that even though there are a lot of cloud services available that fulfill testers' initial needs, but still there is a need for a scalable platform for effective crowdsourcing in mobile testing to support multidirectional testing as well as flexible integration of different testing techniques and services.

Initial attempts at a literature search found no comprehensive and convincing studies on systematic mapping in the area of mobile application testing, which encouraged us to perform such a formal and in-depth mapping review. We also found a wide variety of studies reporting mobile testing tools and methods (e.g. (Looije et al., 2007), (Harrison et al., 2013), (Muccini et al., 2012)), but few that apply a rigorous empirical approach. In order to provide a wide overview of empirical studies in the area of mobile application testing, the present study applies a systematic mapping methodology to build a classification scheme, to identify and analyze evidence for challenges, techniques and methods that have been previously published. Analyzing all related evidence for mobile application testing challenges and techniques is therefore needed in order to identify possible research gaps and to suggest further studies such as systematic literature reviews.

3. METHOD

This section describes the systematic mapping method that we applied in this study. The details of review planning and conduct are also discussed in this section.

In this study, our research methodology for a systematic mapping was based on the guidelines provided by (Petersen et al., 2008) and (Kitchenham and Charters, 2007). This review is also inspired by other systematic mapping studies (Bailey et al., 2007), (Mujtaba et al., 2008), more specifically in the area of data synthesis and analysis. Such review normally leads to provide a coarse-grained overview for field area and to provide a baseline to suggest areas for further research (Petersen et al., 2008).

According to (Petersen et al., 2008), a systematic mapping process consists of five (5) discrete phases (see Figure 1). The first phase is defining research questions. The second phase is conducting the search. In this phase a researcher specifies a search strategy and selects primary studies. The third phase is screening of papers. The fourth phase is keywording of abstracts. During this phase a researcher builds a classification scheme. The last phase is data extraction and mapping process. During this phase, the relevant articles are mapped into the scheme and this involves the process of data extraction. A summary of the materials used in this study is put online[1]. The steps in this mapping process may seem to be sequential, but it is important to clarify that some of them are iterative. For instance, the classification scheme evolves and gets updates through the process since new terms are added or merged while going through the included papers.

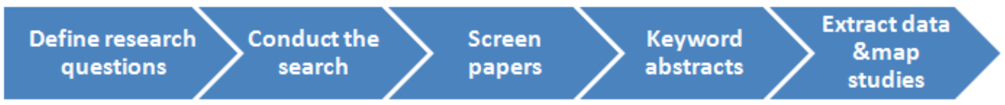

Figure 1: the systematic mapping process

3.1 Research Questions

This study tries to build a classification scheme through identification of all related evidence and knowledge gained from empirical studies of mobile and smart-phone application testing techniques. Further, this study aims to identify research gaps and outstanding challenges and to suggest where future research fits to best extend the current body of knowledge. Hence, we need to identify the contributions of studies on mobile application testing reported to date. As highlighted by (Muccini et al., 2012), it is necessary to identify the peculiarities of mobile application testing due to the diversity of mobile platforms and features of mobile devices. Due to the lack of reviews related to this area, we focus this mapping study on empirical studies on mobile application testing techniques, methods or approaches. To achieve the above aims, the following primary and sub research questions (RQs) were specified:

Primary RQ: What are the studies that empirically investigate mobile and smart phone application testing techniques and challenges?

sub-RQ1: What research approaches do these studies apply and what contribution facets do they provide?

sub-RQ2: What kind of applications (industrial or simple) do these studies use in order to evaluate their solutions?

sub-RQ3: Which journals and conferences included papers on mobile application testing?

[1] https://sites.google.com/site/mobileappsms2/home/resources

3.2 Sources of Evidence

The present study was performed at the International Islamic University of Malaysia (IIUM), consequently, the sources of information were restricted to available resources subscribed by the IIUM library. The primary search process involved the use of standard online databases that index Computer Science and ICT related literature. These include: *IEEExplore*, *ACM Digital Library*, *Scopus*, *SpringerLink*, *ScienceDirect* and *ProQuest*.

3.3 Search Strategy

In this review, we included empirical studies of both qualitative and quantitative approaches. Such studies need to be directly related to mobile and smart-phone applications focusing on testing techniques, challenges, methods or approaches.

We adopted the strategy to construct the search string as suggested by Kitchenham and Charters (2007):

- Search for synonyms and alternative keywords.
- Use Boolean OR to incorporate alternate spellings and synonyms.
- Use Boolean AND to link major terms together.

In our preliminary search, it took several tries to construct the right search strings due to the fact that the term **mobile** is a generic term and it is connected to different research areas such as robotics, vehicles and other unrelated engineering terms. In each try, the search string was evaluated based on how much the returned studies were relevant to our focus area, i.e., mobile application testing. Additionally, and based on our experience and initial research review, we used 10 studies as a second criteria to examine the quality of our search strings. The final search string chosen was the one that could return results that are most relevant to our area of focus and also the one that returned the maximum number of the previously known ten (10) studies. For instance, the search string of Try2 in Table 1 was excluded because the results were not much relevant to our area of focus as compared to search string of Try5.

The search terms were mainly driven by the research questions (see Table 1). The terms "mobile application", "testing", and "challenges" represented the main terms. Additionally, we aggregated additional terms as synonyms such as "verification", "fault", "approach" and "limitation" to make the search broader and to ensure that we cover larger area.

Further, and based on previous knowledge and previously known studies in this field, the term **context-aware** was notably found in many existing research studies on mobile application testing (Amalfitano et al., 2013) (Bo et al., 2011) (Wang, 2008). This is due to the fact that context-awareness is one of the most compelling peculiarities of mobile applications (Muccini et al., 2012) and that a considerable number of studies discuss the testing challenges of this peculiarity. Hence, we incorporated the term "context-aware" in our search string to ensure that search results contain more relevant studies. In addition, and since several studies are published using the term mobile "app" instead of "application", the term "**app**" was incorporated into our search string as can be seen at Table 1 Try5. For instance, even though search string of Try4 returned all 10 studies, but still it did not return studies that contain "**app**" keyword. Accordingly, search string of Try5 was select in this mapping study.

The online database IEEExplore was used to pilot search strings against the 10 previously known studies. Search strings had to be considerably strict since that when we used generic terms such as "mobile application testing", we ended up with thousands of hits. Table 1 shows piloted search strings, number of studies missed and returned results from IEEExplore. Hence, after the pilot evaluation, search string of Try5 was chosen.

Table 1: Search strings piloted on IEEExplore

# Try	Search string:	# studies missed	Returned results
Try1	((mobile) AND (application OR software) AND (context aware OR context awareness OR adaptive) AND (testing OR verification) AND (technique OR approach OR method OR challenge OR limitation))	7	377
Try2	((mobile) AND (application OR software) AND (testing OR verification) AND (technique OR approach OR method OR challenge OR limitation))	1	766
Try3	((("mobile application" OR "mobile software" OR "context-aware") AND (testing OR verification OR fault) AND (technique OR approach OR method OR challenge OR limitation)))	1	657
Try4	(((("mobile application" OR "mobile applications" OR "context-aware") AND (testing OR verification OR fault) AND (technique OR approach OR method OR challenge OR limitation))))	0	819
Try5	(((("mobile application" OR "mobile applications" OR “mobile apps” OR "context-aware") AND (testing OR verification OR fault) AND (technique OR approach OR method OR challenge OR limitation))))	0	917

3.4 Study Selection Criteria

The main focus of our mapping study is based on identification of empirical studies (both qualitative and quantitative) in the area of mobile application testing. According to Perry et al. (2000), empirical studies can take many forms and are not only realized as experiments, but also as case studies, surveys and prototyping exercises as well. Further, empirical studies usually involve the steps of formulating a hypothesis, observing a situation, analyzing the data and drawing conclusions. According to (Fenton et al., 1994), in order for a software engineering research to obtain a more solid scientific foundation, it should be based on empirical evaluation and data. A research simply based on anecdotes, gut feeling, opinion or flawed research is not of a recognized scientific value (Fenton et al., 1997). Therefore, selection criteria were defined during the review planning stage to avoid bias. In our mapping study, we considered a study to be empirical if the proposed solution is backed up with empirical evaluation and data. For example, if one paper is proposing a new testing approach or method, it should contain evidence or data that supports the proposed approach. Studies that are not backed up with empirical data or merely presenting opinion without any supporting evidence are not included in this review.

As suggested by (Kitchenham and Charters, 2007), the selection criteria were piloted on known studies, and consequently, were refined and enhanced. In this mapping study, we applied the following inclusion criteria:

- Studies must be directly related to software testing techniques, approaches, challenges or limitations for applications running on mobile phone devices, smart phones or PDAs. Such techniques and approaches should be applied during the software development process.
- Studies must provide empirical data or supporting evidence (i.e. containing empirical quantitative or qualitative data).

The following were the exclusion criteria used to exclude irrelevant studies:

- Studies related to testing embedded systems in general, and not running on mobile devices.
- Studies related to mobile communication infrastructure, mobile hardware, or robotics.

- Studies related to other software development phases such as analysis, design or implementation and not related to testing.
- Studies that merely present opinion without any supporting empirical evidence.

As outlined above, we excluded studies that discuss methods and approaches related to software development phases of mobile applications other than testing such as development and design. Other excluded studies discuss hardware and communication infrastructure and the remaining excluded studies proposes solutions provided as methods, frameworks and models without empirical data or experimental evaluation.

3.5 Study Selection Process

The search process performed on all databases was based on the advanced search feature provided by the online databases. The search string was applied using advanced command search feature and set to include meta-data of studies. Additionally, initial dates were not specified or restricted during the search, i.e., we did not define any lower bound date to ensure wide coverage of search. However, the search process was restricted for studies related to computer science field. This restriction is due to the fact that the term **mobile** is commonly used in other engineering disciplines. The literature search covered studies published up until **2015**.

The study selection process was iterative and incremental where each paper went through three different filtration steps (see Figure 2). Initial phase was related to searching the database using search string. Then in the first phase, resulting papers were filtered based on their title and abstract. In this step, papers' titles and abstracts that are not related to testing of software applications running on mobile or smart-phones were excluded.

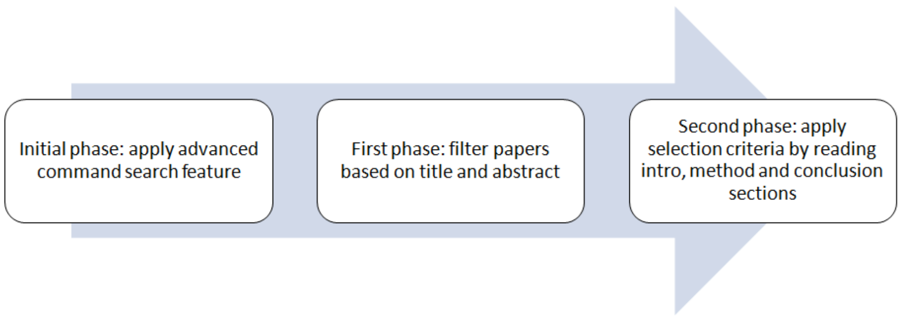
Initial phase: apply advanced command search feature
First phase: filter papers based on title and abstract
Second phase: apply selection criteria by reading intro, method and conclusion sections

Figure 2: selection process

In the second phase, filtration was based on applying selection criteria by reading a selected paper's introduction, methodology and conclusion. From the remaining papers of phase one, papers were excluded either because they were not empirical, they did not confirm to study selection criteria identified in Section 3.4, or because papers were duplicates of other papers. When duplicate papers were found (at second phase), i.e. similar paper appears in more than one venue, the most comprehensive version of the paper was selected. The final filtration step was based on complete and thorough reading of remaining papers.

3.6 Keywording of Abstracts (Classification Scheme)

The purpose of keywording is to reduce the time needed to build a classification scheme and to ensure that the scheme takes into account existing studies. We applied a thematic analysis approach which identifies, analyzes and reports themes within data (Braun and Clarke, 2006). In general, the keywording process was inspired by (Petersen et al., 2008) and consisted of two phases and was applied to the final set of included papers. In the first phase, the main researcher (the first author) read abstracts of selected papers and looked for sentences and concepts that reflected the investigated problem as well as the contribution and area of focus of papers. When the abstracts were of poor quality or too short to allow convincing keywords to be chosen, the researcher reads the introduction and conclusions as well.

In the second phase, and based on the thematic analysis approach, the set of keywords from different papers were combined together to form a high-level understanding about the nature and contribution of the research. This led into identifying a set of topics (sub-categories) for the classification scheme. When a final set of keywords was chosen, they were clustered and used to form the map categories.

During the first phase of keywording process, there were lots of concepts reflecting the different investigated problems and contributions of included papers. Examples of such concepts were model-based testing, test case generation, usability data analysis, automated collection of usability data, context events, malware detection, etc. This resulted in a relatively large number of concepts due to the diversity of problems investigated and contributions in the included papers. Thus, during the second phase the resulting concepts from the first phase were grouped together based on the area of focus for each paper. The topics of "usability testing", "test automation", "context-awareness", and "security testing" were carefully chosen as a higher level of concepts that best fit our included papers and became the main category in our resulting classification scheme. Resulting classification scheme will be presented in section 4.2.

3.7 Data Extraction and Mapping of Studies

The main aim of this phase was to map identified studies into the classification scheme and extract relevant data to answer our research questions. That is, after having the classification scheme in place, the relevant studies were sorted into the scheme. Data was inserted into tables, and frequencies of publications for each category were calculated. The EndNote citation management tool was used to record and manage papers' citations. This included authors' names, publication year, source, and title among others. Additionally, data extraction form was designed to extract data based on the research questions. Extracted data here reflected contribution facets, research approaches used in the study, challenges addressed; testing techniques applied, methodology, study setting (i.e. whether the study based its solution on real-world development team's needs or not), the specific mobile apps testing topic (e.g. mobile services, or testing conformance of mobile life cycle models). The extracted data was collated and stored using spreadsheets and the frequencies of publication were calculated.

The challenges recorded are related to major problems addressed by a study. If there is more than one challenge, they were ordered according to their appearance sequence in that study. Techniques applied, on the other hand, represent the special technical approaches or methods that authors applied to solve their study problem. Finally, the solution methodology summarizes the steps of how techniques were applied to solve a problem. The overall classification scheme and resulting data extracted helped in providing deeper understanding and enabled us into identifying research gaps.

3.7.1 Validity control

The first author was responsible in reading and completing the extraction form for each of the primary studies included in the systematic mapping study. The second and third authors on the other hand provided detailed feedback on the study protocol to minimize any possible bias. In order to reduce the bias during study selection and data extraction phases, the second author performed random analysis of 10% of included studies independently. The results were compared in a meeting and no significant anomalies were evident. We did not measure inter-rater agreement since our review meeting aimed to reach an absolute consensus on the sample used.

4. RESULTS

4.1 Search results

It was apparent beforehand that searching for and retrieving empirical studies for mobile or smart-phone application testing techniques/approaches needs careful construction of search strings. As noted earlier, the term mobile is used in different engineering disciplines. This explains why there were several tries to pilot search strings and compare results with previously known studies. However, we believe that our search string is considered reliable because it contains the term "mobile application/app testing" and that almost all related articles are categorized under this term.

Initially our initial search results returned 7356 studies from all sources. Afterwards, the three filtration steps were applied as discussed in section 3.5. Table 2 shows online databases searched, initial search results, and the number of remaining studies after applying each filtration step (see Figure 2). In total, 79 studies were included after applying filtration steps and inclusion/exclusion criteria (see Appendix A for the list of included studies). Out of the 79 studies, 36 (45.5%) came from IEEExplore and 30 studies (38%) came from ACM Digital Library, two (2) studies from SpringerLink, two (2) from ProQuest, three (3) studies from ScienceDirect, and six (6) from Scopus. The distribution of included studies over publication years can be seen in Figure 3. It can be seen that the earliest study was published in 2005 and year 2015 is the year where most included studies were published.

Table 2: Remaining studies after each filtration step

Online Database	Search Results	Phase1	Phase2
IEEExplore	917	59	36
ACM Digital Library	2330	54	30
Science Direct	698	7	3
Springer Link	1453	11	2
ProQuest	62	6	2
Scopus	1896	28	6
Total	**7356**	**165**	**79**

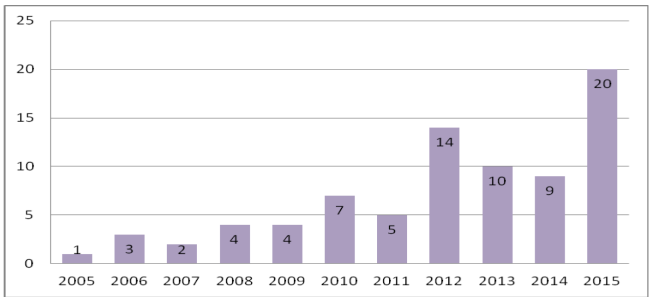

Figure 3: Studies per publication year

4.2 Classification Scheme

The classification scheme we used consists of FOUR (4) main categories: i) Structure of the topic or evidence; ii) Contribution facets; iii) Objects involved in the study (i.e., the type of applications used for evaluations); and iv) Research facets. In the first category (structure of the topic), and based on the thematic analysis, we grouped the papers into five topics (sub-categories): *usability testing*, *test automation, context-awareness testing, security testing* and *testing in general*. These topics were constructed by investigating the main focus area that each paper addresses as described in section 3.6. It was very apparent during the phase "keywording of abstracts" that test automation, usability testing, security testing, and context-awareness are the main areas and focus of research under which, relevant studies can be grouped. For the rest of the studies that did not belong to these four types, they were put under "general" topic.

The second classification which is inspired by (Shahrokni and Feldt, 2013), investigates the contribution facets. As suggested by (Shahrokni and Feldt, 2013), the contribution facet criterion structures the studies in the final set into specific contribution type, namely *framework*, *method*, *tool*, *evaluation* and *metrics*. A framework is a detailed method that covers wide purpose by focusing on several research questions and areas. In contrast, a method has a specific goal and narrow purpose or research question (Shahrokni and Feldt, 2013). Studies where the tools represent major topic were classified under tool contribution facet. Metrics on the other hand measure important variables in software testing. Finally an evaluation contribution facet represents studies that evaluate systems or methods.

The third classification category is "objects involved in the study" which reflects answer for **sub-RQ2** and represents the type of applications (industrial/simple) used for evaluating proposed study solution. In this criterion, **simple (toy) application** represents a special purpose small application built to evaluate the case study. On the other hand, **industrial** represents commercial and real world application used to evaluate the case study. We were motivated to study the context of the studies to investigate two aspects: first, it is important to measure how included studies evaluate their solutions and to what extent. Evaluating using real world applications can ensure that the proposed solution is trustworthy and reliable (Shahrokni and Feldt, 2013). Secondly, and since that mobile application development field is relatively new, we believe that studies should shed some light on problems and challenges faced by real world development teams to see how teams currently approach mobile application development. Such insight would help reveal real problems and thus produce solutions that could solve real problems.

The fourth classification is the "research facet", which is inspired from (Petersen et al., 2008). In this criterion, we choose the existing types of research approaches as suggested by (Wieringa et al., 2006):

- **Validation research:** the techniques investigated are novel and have not yet been applied in practice.
- **Evaluation research:** techniques are implemented in practice and an evaluation of the technique is available. This kind of research shows how the technique is implemented in practice along with its benefits and drawbacks.
- **Experience papers:** these papers show how something was done in practice as a result of personal experience of the author.

4.3 Answering the research question

RQ: What are the studies that empirically investigate mobile and smart phone application testing techniques and challenges?

The studies included in this mapping study were categorized and grouped according to the classification scheme described in section 4.2. The main category to structure the topic included: *usability testing*, *test automation*, *context-awareness*, *security testing*, and *testing in general* topics (sub-categories). Studies that have presented challenges not related to any of the first four topics are put into a general topic. Studies with clear contribution are discussed in more details in the following sections. Out of 79 studies, nineteen (19) studies came under usability testing, twenty nine (29) studies under test automation, eight (8) studies under context-awareness, eight(8) studies under security testing and fifteen (15) studies under general testing topic. Table 3 shows the included studies for each topic. It is noticeable that most of the studies published were related to test automation 37% (i.e. 29 out of 79).

Table 3: Studies under each topic (sub-category)

Category	Studies (S)	Total # studies
Usability Testing	S2, S9, S19, S21, S22, S23, S24, S25, S26, S28, S33, S35, S36, S37, S40, S43, S64, S69, S77	19
Test Automation	S1, S3, S6, S11, S12, S20, S16, S31, S32, S41, S42, S45, S52, S53, S55, S56, S57, S58, S59, S61, S62, S63, S68, S70, S71, S72, S73, S78, S79	29
Context-Awareness	S5, S7, S8, S10, S17, S39, S54, S74	8
Security	S46, S47, S48, S49, S50, S51, S60, S75	8
General Category	S4, S13, S14, S15, S16, S18,S27, S29, S34, S38, S44, S65, S66, S67, S76	15

4.3.1 Usability Testing

According to (Harrison et al., 2013), in the context of mobile applications, usability is represented in terms of three attributes; effectiveness, efficiency, satisfaction and cognitive load. Additionally, and as explained by (Bruegge and Dutoit, 2004), the goal of usability testing is to find errors in the user interface of an application. According to our study findings, there are a considerable number of published studies on usability testing. Out of the 79 included studies, 19 studies were related to usability testing and validation (see Table 3).

The study presented by (Balagtas-Fernandez and Hussmann, 2009) [S2] addresses the challenges of usability analysis and evaluation of mobile applications because of restrictions of device, and lack of supporting tools. They have developed a framework that is based on a logging technique. Through this technique, the study of usability for mobile applications running on a device can be simplified. Another contribution by (Ravindranath et al., 2012) [S21] discusses how to identify critical user transactions when program is running in the wild. They have developed a tool that instruments mobile application binaries to automatically identify the critical path in user transactions.

A recent study by (Flood et al., 2012) [S35] evaluates the usability of spreadsheets for mobile applications. Their study provide lessons learned and usability guidelines derived from laboratory usability testing of mobile spreadsheet applications based on video recording technique.

The challenge of evaluating mobile user interfaces for usability is the main focus of study by (Lettner and Holzmann, 2012) [S23]. This study presents a novel approach and toolkit for automated and unsupervised evaluation for mobile applications that is able to trace any user interaction during the entire lifecycle of an application.

Based on the hypothesis that agile methodologies share crucial needs with usability engineering and mobile applications development general requirements, (Losada et al., 2012) [S25] applied usability engineering in the agile methodology called InterMod on mobile application development. InterMod technique includes the use of questionnaires, interviews, observations and user test through paper prototypes.

A case study presented by (Pham et al., 2010) [S26] examines the usability evaluation of MobiTOP mobile application in the context of a travel companion for tourists. Participants agreed that the features in MobiTOP are generally usable as a content sharing tool. Another case study by (Huhn et al., 2012) [S28] with a similar focus, contributes to this line of research by presenting a user experience study on mobile advertising with a novel CAVE-smartphone interface. Two experiments were conducted to evaluate the intrusiveness of a mobile location-based advertising app in a virtual supermarket.

The study by (Oyomno et al., 2013) [S33] discussed a usability study on Mobile Electronic Personality Version 2 (ME2.0) which is a context-aware service personalizing mobile application. According to this study, and in order to guarantee the effectiveness of ME2.0 in privacy preservation, the User Interface (UI), the User Experience (UX) and usability need to efficient and meaningful. Additionally, the design and implementation of context-aware mobile applications that manage users' personalization attributes can be a daunting task especially when neglecting users' perspectives. To address these issues from different user perspectives, the study conducts several usability studies centered on the themes of effectiveness, efficiency, learn-ability, memorizability, error-rate, and scope.

(Kronbauer et al., 2012) [S36] report that there is a lack of approaches reported in the literature for evaluating mobile application usability. This includes the use of quantitative data (metrics), subjective evaluation (users' impressions) and context data. The study presents a proposal for a hybrid usability evaluation of smart-phone applications, which is composed by a model and an infrastructure that implements it.

The empirical study by (Billi et al., 2010) [S37] applies techniques of early assessment and ad-hoc mobile oriented methods to evaluate the usability and accessibility of mobile applications. A case study by (Biel et al., 2010) [S40] designed a method that aligns the inspection method "*Software ArchitecTure analysis of Usability Requirements realizatioN*" (SATURN) and a mobile usability evaluation in the form of a user test. The study also proposes to use mobile context factors and requirements as a common basis for both inspection and user test.

A field study approach is applied by (Bjornestad et al., 2011) [S9] to investigates the usability of location-based news service for mobile phones. Through their study, a system to support location-based news is developed consisting of authoring tool for journalists and a reader tool for mobile phones with web browsers. The investigation was done using qualitative and quantitative data from a field experiment. The study concludes that respondents found both the software and journalistic concept easy to understand.

A case study by (Fetaji et al., 2008) [S19] addresses the problem of lack of research about efficiency, effectiveness and usability of mobile learning or m-learning systems. The study also discusses the usability of a learning environment and proposes a strategy on how to implement a successful and usable m-learning environment. The proposed strategy is based on incorporating a qualitative approach in order to gather better qualitative information for the usability and benefits of the environment. Further, the strategy includes the user-centered design approach in which end users are included in the design of application user interfaces from the beginning.

(Canfora et al., 2013) [S43], developed a platform named ATE for supporting the design and automatic execution of user experience tests for Android applications in order to evaluate how end users perceive the responsiveness and speed of such applications. Methodology employed in this case study is made of three distinct steps: In the first step, the developed platform is used to define, execute and evaluate user experience tests of smartphones. During the case study, they used two versions of smartphones with different processing capabilities. They also developed three different types of user profiles: normal, smart and business users. In the second step, a demographic analysis of real users using interviews and observations was used to collect real data for comparison. In the final step, they performed a comparison with three well known tools. The study argues that ATE produces user experience estimates that are comparable to those reported by humans.

In another study by (Borys and Milosz, 2015), the authors discuss the setup and results of quasi-real settings of mobile usability test using mobile eye-tracking glasses. The focus of experiment is to evaluate the usability of mobile application called Sale Force Automation in terms of its basic functionality. The study concludes that it is possible to approximate the real conditions of application usage while still having complete control over it. Further, the application of eye-tracker technique enabled accurate data gathering as well as detecting a whole range of usability problems.

The study by (Masood and Thigambaram, 2015) investigates the usability of mobile educational applications for children age between 4 and 5 years. The study uses eye-tracking technique and is based on children's mental model as well as the quality of their learning experience. The study also provides a set of principles for user interface design and guidelines for developers when developing mobile educational applications.

Another study by (Wei et al., 2015) focuses on library mobile application of Chongqing University in order to provide recommendations for improving the user experience of application users. The methodology of the study is based on pre-test questionnaires, accomplished tasks, and post-tests surveys. The study concludes that the library application was effective; however, the efficiency of the application needs more improvements in terms of clarity and usefulness.

Additionally, the studies under usability testing sub-category can be further classified based on their area of focus. More specifically, these studies can be classified under the sub-categories of (i) specific domain usability; (ii) during development; and (iii) general. Table 4 shows the further classification of usability studies. In "specific domain usability", the studies focus on usability evaluation for specific domain (e.g. location based news and spreadsheets applications). On the other hand, "development solutions" represents studies providing usability solutions that aid developers during the construction and evaluation of mobile applications in terms of saving time and effort. Finally, and since the remaining studies belong to a variety of other focus areas, they were put under "general" sub-category.

Table 4: Further classification of usability studies based on focus area

	Study	Application domain
	S9	News Reader
	S19	Learning through mobile
	S24	News application
	S26	Location-based annotation system
Specific domain usability	S28	Advertizing systems
	S33	Mobile electronic personality
	S35	Spreadsheets applications
	S64	Sale force automation
	S69	Mobile educational applications
	S77	Mobile library application
Development solutions	S2, S21, S23, S25	
General	S22, S36, S37, S40, S43	

4.3.2 Test Automation

Test automation refers to the use of one piece of software to test another piece of software (Crispin and Gregory, 2008). With the help of automation tools, test engineers can keep pace with development team, maintain agility and save testers from routine, time consuming and error prone manual testing activities (Crispin and Gregory, 2008).

We found 29 studies that have reported evidence on test automation of mobile applications S1, S3, S6, S11, S12, S20, S16, S31, S32, S41, S42, S45, S52, S53, S55, S56, S57, S58, S59, S61, S62, S63, S68, S70, S71, S72, S73, S78, and S79.

Automatic testing of Android mobile applications is explored by (Amalfitano et al., 2011) [S1]. In this study, the authors present a technique and a tool to perform rapid crash testing, regression testing and automatic generation of test cases. In another study by (Nagowah and Sowamber, 2012) [S3], a framework is presented to automate software test on the mobile device itself rather than using the emulator. This is due to the fact that running automated tests on emulators may compromise the reliability of the test since those emulators are not the actual devices, and may not reflect actual reliable results.

An approach presented by (Edmondson et al., 2011) [S6] combines portable operating system libraries with knowledge and reasoning. This approach will eventually leverage the best features of centralized and decentralized testing infrastructures to support both heterogeneous systems and distributed control by reasoning on distributed testing events.

A distributed client/server testing tool is proposed by (She et al., 2009) [S11] to address the challenges of heterogeneity of mobile devices and their limited resources. The study also presents partially implemented tool writing, executing and reporting of tests. Another study by (Jiang et al., 2007) [S12] proposes a tool for automatic black-box testing of mobile applications. Additionally, the study adopts a sensitive-event based approach to simplify the design of test cases and enhance their efficiency.

The study by (Liang et al., 2014) [S57] focuses on the problem that test inputs have to be run with a large variety of contexts. Consequently, the study presents a testing tool called *Caiipa* that is based on cloud service technology for testing mobile applications over an expandable mobile context space. The study also includes techniques to make mobile applications testing more traceable to quickly locate failure scenarios for each application. Another testing tool called *AppDoctor* is discussed in the study by (Hu et al., 2014) [S59]. AppDoctor applies the techniques of approximate execution and action slicing which enable the tool to run much faster than real execution and expose software bugs. The challenge of how to systematically explore Android applications is discussed by (Azim and Neamtiu, 2013) [S56]. In their study, the authors argue that relying on end users to perform systematic exploration is not effective. The authors present a novel approach to perform a systematic exploration for Android applications that is based on static taint-style dataflow analysis without the need for the application source code. In their approach, a high-level control flow graph is produced capturing legal transitions between activities. Later on, this graph is explored by a strategy called targeted exploration that allows direct and fast exploration of activities.

The challenge of how to automatically generate sequences of test inputs for Android applications is discussed by (Choi et al., 2013) [S58]. In their study, they propose an automated technique called *Swift Hand* that is based on machine learning to produce sequences of test inputs that enable visiting unexplored states of the application. A key feature of their technique is that it avoids restarting the application which is a relatively an expensive operation.

A study by (Amalfitano et al., 2012) [S20] introduces an automated technique based on a user-interface driven "ripper". This technique automatically explores an application's user interface with the aim of exercising the application in a structured manner. Another study by (Kaasila et al., 2012) [31] reveals an online platform for conducting scripted user interface tests on a variety of Android physical handsets. This study was performed as an attempt to address the challenge of comprehensive testing of interactive applications running on multiple versions of Android operating system. An interesting finding of this study is that it can reveal common issues and problems such as that applications fail to install on certain handsets and mistakes in Android application manifest files.

(Zhifang et al., 2010a) [S30] introduces the idea of constructing a testing framework employing techniques from MobileTest tool, service-oriented architecture (SOA), image comparison based testing and optical character recognition (OCR). (Puhakka and Palola, 2006) [S32] discuss the new testing needs of beyond 3G (B3G) applications and presents an experimental system for automating testing of B3G mobile applications that supports application testing in multiple mobile phones at the same time.

(Lu et al., 2012) [S41] argue that mobile applications are different from traditional web and desktop applications due to physical constraints of mobile devices as well as new features of mobile operating systems which in total impose unique challenges when testing these applications. Consequently the study proposes a method for automatic testing for Android applications based on functional testing through application activities. The method is based on a model for application activities and a special algorithm to generate test cases. The method is implemented by extending open source tools Robotium (2014b) and Mokeyrunner (2014a).

The empirical study conducted by (Song et al., 2011) [S42] addresses the problem of having several platforms for mobile applications which in turn requires test engineers to spend much effort and time to test their application on each platform. The study aims at developing an integrated test automation framework through which implementations can be tested on mutable heterogeneous platforms effectively. This is based on the idea of describing test cases in a high level language without having to generate test code manually. Although platforms are different, but still, common events, such as touch, drag, scroll, etc. can be extracted to generate independent test cases. The study argues that by automating this part, the cost of testing can be reduced.

The study by (Zhang and Elbaum, 2014) [S45] focuses on the important problem of validating code for exceptional behavior handling, especially when dealing with external resources that may be noisy and unreliable. The study suggests an automated approach that addresses this challenge by performing a systematic amplification of the application space by manipulating the behavior of external resources. Additionally, the study provides an assessment of the cost-effectiveness of the approach by testing it on eight real-world Android applications.

In another study by (Costa et al., 2014) [S52], the authors assess the feasibility of using the Pattern Based GUI Testing (PBGT) approach to test mobile applications. PBGT is based on the concept of User Interface Test Patterns to test recurrent behavior. Since PBGT was developed with web applications in mind, their study describes the adaptations and updates the PBGT should undergo to test mobile applications.

The study by (Tao and Gao, 2014) [S55] focuses on the problem that existing test models rarely target the test modeling and analysis for mobile environment contexts such as mobile platforms, web browsers, different technologies, device gestures, APIs, etc. Consequently, and in order to better achieve effective test automation, the paper provides an approach to modeling mobile test environments based on a Mobile Test Environment Semantic Tree (MTEst). Based on MTEst model, the paper discusses test complexity evaluation methods for test environment.

(Morgado et al., 2014) [S53] uses the techniques of reverse engineering and behavioral patterns to test mobile applications. Their testing approach is based on automatically identifying and testing behavior that is common in mobile applications. They also present a tool that automatically identifies patterns in the behavior of the application and then applies associated tests for each identified pattern.

In the study at (Villanes et al., 2015)[S61] proposes a testing framework called Automated Mobile testing as a service which provides automated tests for mobile applications. The framework is mainly based on cloud technology and emulates mobile devices using virtual machines and cloud infrastructure. The study focuses on the criterion of OTA Install (automated installation of mobile applications on devices). The study concludes through experiments that 100% of the emulated devices could be tested using test cases of their framework.

The study by (Wen et al., 2015)[S62] addresses the challenge of automatically testing complex Android GUI applications and maintaining efficiency. Thus the study proposes a parallel testing platform which performs GUI testing based on master/slave model. The authors argue that their testing platform can increase testing efficiency and mitigate the tedious testing process.

The study by (Zhauniarovich et al., 2015)[S63] investigates the problem of measuring code coverage for mobile applications when the source code is absent. The study argues that current test frameworks do not provide statistics or coarse-grained reports when measuring code coverage. Thus, the study introduces a new framework called BBoxTester that is able to generate detailed code coverage reports as well as uniform coverage metrics without the need for application source code.

The use of Model-Based testing in the construction and implementation of automated tests for Android applications is investigated by (de Cleva Farto and Endo, 2015) [S68]. The study investigates the applicability, current state-of-the-art and challenges faced when adopting model based testing techniques. The study concludes that model-based testing can be used to test Android mobile applications and that it does provide advantages such as automatic generation of test cases, fault detection, improve test quality and time consumed, and evolution of test models.

A new testing technique that is search-based is introduced by (Amalfitano et al., 2015a) [S70]. The study argues that record/replay, random, model-learning and model based techniques do not produce test cases that are effective and efficient. To address this challenge, the study presents a search-based technique that is based on genetic and hill climbing techniques.

The challenge of improving test cases' quality and effectiveness is investigated by the study at (Adamsen et al., 2015) [S71]. The study realizes the problem of having manually written test cases not focusing on unusual events. Additionally, automated generation of test cases does not focus on intended functionality of the application. Consequently, the study proposes a new testing methodology by leveraging existing test cases by systematically so as to expose unexpected events to surface. The study concludes that real-world mobile applications are often fragile to unexpected events.

The problem of insufficient testing techniques or tools that can handle inter application communication is investigated by (Kumar et al., 2015) [S72]. The study proposes a conceptual model to represent inter application communication at a higher level as well as a technique to generate test cases from the model. The study argues that the conceptual model can be applied during different stages of mobile application development such as analysis and testing stages.

In another study by (Hu et al., 2015) [S73] addresses the challenge of recording and replaying sensor and network input, and inter application communications using intents. The study introduces a stream-oriented record and replay approach that is capable of recording above events while maintaining high accuracy and low overhead. The study claims that proposed testing approach is capable of replaying high-throughput and time sensitive applications such as video/audio recognition.

The problem of how to model the state of mobile application GUI as well as application state-sensitive behavior is investigated by (Amalfitano et al., 2015b) [S78]. In their study, the authors introduce a GUI testing framework for Android called MobiGUITAR. The framework is capable of addressing the above challenges as well as applying new test adequacy criteria based on state machines. According to the study, MobiGUITAR employs new test case generation technique and provides fully automated testing that works with mobile platform security.

Finally, the study by (Griebe et al., 2015) [S79] provides an extension to the testing framework Calabash allowing to integrate sensor information into user acceptance tests that are written in Gherkin. The study also introduces a simulation engine that can feed artificial sensor data to application under test.

Table 5 shows detailed classification to approaches for test automation studies. We found that majority of the studies (8 out of 19) applied model-based testing approach in their evaluation.

Table 5: Test automation papers classified according to test approaches:

Testing approach	**Study**
Model-based	S1, S20, S41, S52, S55, S68, S72, S78
Data-driven	S3
Portable operating system libraries with knowledge and reasoning	S6
Black box	S11, S12, S63
Sensitive-event based	S30
Scripted user interface	S31, S32, S42
Exhaustive test amplification	S45
Reverse engineering	S53
Static taint-style dataflow analysis, depth-first	S56

exploration	
Contextual fuzzing	S57
Machine learning	S58
Approximate execution	S59
Automated mobile testing as a service	S61
Parallel GUI testing based on master-slave model	S62
Search based	S70
Systematic exploration of test suites	S71
Sensor and event-stream based approach	S73
Sensor simulation	S79

4.3.3 Context-Awareness

In context-aware mobile applications, the application is aware of the computing environment in which it runs, and adapts to its changes in contexts such as user, time or physical ones. Further, contexts can be categorized into two groups: human factors such as user, social environment and task. The other group is physical environments such as location, infrastructure and physical conditions (Muccini et al., 2012). Based on the data extraction, eight (8) out of 79 studies were specifically related to context-aware mobile applications (S5, S7, S8, S10, S17, S39, S54, and S74).

The first study that has investigated about context-awareness issue on mobile applications was published in 2005. In the study, (Ryan and Rossi, 2005) [S17] define and empirically evaluate metrics to capture software, resource utilization and performance attributes for the purpose of modeling their impact on context-aware mobile applications. Additionally, the study introduces a suite of metrics to model the impact of software code attributes upon performance and resource utilization.

(Zhimin et al., 2007) [S5] reports the challenges of validating context-aware applications for pervasive software. The study introduces an approach for identifying context-aware break points and systematically changes the context-aware data fed to application to expose failures.

In another study, (Wang, 2008) [S39] addresses the problems of orthogonal input space, intrinsically noisy data, continuous and indirect input feeding and continuous adaptations. The study also identifies context-aware program points where context changes may be relevant. Additionally, control program execution identifies two classes of adaptation fault patterns and analyzes a system's model of adaptation rules to detect such faults rank statements based on their sensitivity to context changes.

(Sama et al., 2010) [S39] investigate the problem of exposing context-aware mobile apps faults that cannot be exposed using regular testing techniques. Consequently, their study defines and applies a new model for the detection of faults of incorrect adaptation logic, asynchronous updating of context information and defines algorithms to automatically detect such faults. Additionally, the study proposed a new model of adaptive behavior named "Adaptive finite state machine". This new model can detect faults caused by both erroneous adaptation logic and asynchronous refresh of context information.

In another study by (Bo et al., 2011) [S10], the authors address the problem of exposing faults of buggy context providers and propose a fault tolerant design to make the mobile application immune to buggy context providers bugs. The authors apply a statistical fault localization framework targeting at bugs caused by context provider faults.

The study by (Amalfitano et al., 2013) [S7] focuses on the problem of testing mobile application taking into consideration context and context-related events. The study presents approaches based on the definition of reusable event patterns for the manual and automatic generation of test cases for mobile application testing.

In another study by (Yu et al., 2014) [S54], the authors propose to use a sorted biographical reaction system (BRS) to model context-aware environments. In their study, test cases are generated by tracing the interactions between BRS model and the middleware model. In order to decrease the number of test cases the authors propose a bi-graphical pattern flow testing strategy. Their testing approach is validated on sample airport application.

The study by (Vieira et al., 2015)[S74] reviews the challenges of testing context aware mobile applications and presents a new approach for a context simulator. The context simulator supports modeling and simulation of context in various levels such as physical and logical scenarios. Further, the context simulator can generate test cases and enables the execution of such test cases for several context sources.

A summary of challenges addressed by papers listed in this section can be seen in Table 6:

Table 6: Summary of challenges addressed by context-awareness studies.

Study	Challenge(s) addressed
S5	Improving the test suite.
S7	Testing mobile applications as event-driven systems
S8	Detecting faults of erroneous adaptation logic and asynchronous updating of context information.
S10	Detecting buggy context providers.
S17	Defining metrics for resource utilization and performance attributes.
S39	Context changes, intrinsic adaptation mechanisms, implicit reliance on variable context values.
S54	Modeling context-aware environments and improving test suites.
S74	Simulating context environments.

4.3.4 Security Testing

We found eight (8) studies under the category of mobile applications security testing. The studies are S46, S47, S48, S49, S50, S51, S60, and S75.

(Johnson et al., 2013) [S46] discusses the cyber threats emerging from new smart devices capabilities and the online market applications for mobile devices. The study continues and presents a special framework from exposing the functionality of mobile applications using dynamic and static program analysis techniques to retrieve all program execution paths. In another study by (Salva and Zafimiharisoa, 2013) [S47], the authors propose a model-based security testing approach to detect data vulnerabilities in Android applications. In their method, they apply vulnerability patterns on Android inter-application messaging mechanisms and generate test cases. (Lu et al., 2013) [S48] discusses Android malware detection that can monitor various features obtained from Android devices and then applies machine learning technology to detect malicious mobile applications. The study also applies Bayesian Classification method along with Chi-square filtering test. The study by (Chan et al., 2012) [S50] addresses the security problem of vulnerable Android applications that may leak their capabilities to other applications. The study presents a software testing tool called *DroidChecker* that uses inter-procedural control graph flow searching along with static taint checking to detect vulnerable data paths in Android applications.

The study by (Avancini and Ceccato, 2013) [S49] also addresses the problem of inter-application communication security threats and proposes a method to automatically generate test cases along with adequacy criterion for Android applications through the application of static and dynamic program analysis Additionally, the study by (Guo et al., 2014) [S51] also addresses the security problem of Android inter-application messaging using static and dynamic program analysis to detect security vulnerabilities in the messaging system. It can be seen that the three studies [S49], [S50] and [S51] address nearly the same problem area and apply nearly the same solution approach. However, these studies surprisingly do not reference each other.

The study by (Knorr and Aspinall, 2015) [S60] proposes a new security testing method for Android m-Health applications that is based threat analysis to detect possible attack scenarios and vulnerabilities. The study reports a number of serious vulnerabilities in hypertension and diabetes Android applications.

In another study by (Hay et al., 2015) [S75], the authors a comprehensive testing algorithm for detecting Android inter-application communication vulnerabilities. The study also provides a catalog of 8 concrete vulnerability types that can potentially result from unsafe handling of incoming inter-application communication messages.

A summary of challenges addressed of security category papers can be seen in Table 7:

Table 7: Summary of challenges for security papers.

Studies	Challenge(s) addressed
S49, S50, S51, S75	Testing inter-application communications.
S46	Exploring all application execution paths including libraries.
S47	Detecting data vulnerabilities for Android applications.
S48	Detecting Android malware applications.
S60	Testing security of Android health applications.

4.3.5 Testing in General

This section contains discussion of studies with clear contributions but do not have a certain focus either on usability, automation, security or context-awareness testing. We identified fifteen (15) studies which can be classified under this general category (S4, S13, S14, S15, S16, S18,S27, S29, S34, S38, S44, S65, S66, S67, and S76).

Modeling an application from a black-box perspective is discussed by (Zhifang et al., 2010b) [S4]. Their study also proposes a distance metric and technique to generate test cases. In another paper presented by (Heejin et al., 2009) [S14], a method and a tool to support performance testing are proposed to address the challenge of performance testing for mobile applications. The tool utilizes a database established through benchmark testing in emulator-based test environment at the unit test level.

In a study presented by (Franke et al., 2012b) [S18] the authors have tested the conformance of mobile applications to its application life cycle properties. Their paper presents a unit-testing based approach that triggers life-cycle callback methods. The same authors extended their work through another study (Franke et al., 2012a) [S15] that applies approaches of testing mobile applications to the testing of mobile services. In the study they present how to reverse engineer the service life cycles of mobile platforms to develop a service life cycle model that is complete and correct by applying the same techniques and methods.

(Zhi-fang et al., 2009) [S16] argue that test frameworks development should be based on generality, reusability and scalability. They further introduce the service-oriented architecture into this area, and adopt COM (Common Object

Model) technique, aims at designing mobile application software test framework. In another study, (Delamaro et al., 2006) [S29] presented a strategy to support coverage testing for mobile device software in such a way that the applications can be tested not only on emulators, but also on their real target mobile devices. This could be achieved with the aid of structural coverage assessment.

A case study by (Merwe et al., 2012) [S34] describes the development of Android application verification tool that is built on Java Pathfinder, a Java model checking engine. The tool provides a simplified model of the Android framework on which an Android application can run. It then allows the user to write special script that simulates user events to drive the application flow.

The challenge of client-server performance for mobile applications is discussed by (Briseno and Vincent, 2008) [S13]. The investigation is based on web server time response, execution platform and network method. In this study, two web servers are tested: Apache and Sun Java System Web Server. The study highlights the importance of testing performance not only using emulators, but also on real mobile devices. Additionally, the study argues that the Java Sun Web Server was faster than the Apache Server in all the tests.

A case study by (Sa and Carrico, 2008) [S27] discusses the difficulties that emerged through the data gathering, prototyping and evaluation stages in designing mobile applications. They also emphasize the absence of adequate techniques and methods to support mobile applications development activities.

(Starov, 2013) [S38] presents an analysis of existing cloud services for mobile testing and addresses their weaknesses. Methods applied in this study are Crowdsourcing, testing as a service, multi-directional testing, and flexible integration of test techniques. Finally, in a recent study by (Vilkomir and Amstutz, 2014) [S44], the authors recognize the problem of selecting the optimal set of mobile devices when testing mobile applications. Testing mobile applications on emulators can never be enough since those emulators cannot simulate many of the real peculiarities of real mobile devices. However, with the fact that there are over 11,000 different mobile devices available in the market, selecting optimal set of these devices for testing can be non-trivial task. The study suggests an approach based on combinatorial methods to provide coverage for device characteristics. The study also incorporates Each Choice and Pair-wise techniques (Grindal et al., 2005; Kuhn et al., 2008).

The study by (Zhang et al., 2015) [S65] presents a compatibility testing method based on a statistical approach for testing mobile applications on mobile devices. The testing method takes into consideration both the large diversity of mobile devices and maintaining a low testing cost. The study provides a solution to generate an optimized compatibility test sequence for mobile applications using the K-Means statistical algorithm.

The challenge of testing location-based function services is investigated by (Aktouf et al., 2015) [S66]. The study proposes a new test model and test coverage criteria. The study applies a case-study research method on location-based application called Waze.

In another study by (Vilkomir et al., 2015) [S67], the authors evaluate methods for selecting mobile devices such as tablets and smart phones when testing mobile applications. The study argues that there are specific software faults that are only found in certain devices. The study focuses on the problem of determining how many devices must be tested and which methods of device selection are most successful in order to detect the device-specific faults. The study concludes that most successful approach was the coverage of different types of Android operating systems and provides recommendations and guidelines to increase effectiveness and decrease cost of testing.

Finally, the study at (Ahmed et al., 2015) [S76] provides a testing approach based on adapting the two testing frameworks Reweb and Testweb by providing an adaptation model. The study also addresses the problem of reducing the redundancy in test cases by refactoring source code before test case generation.

Additionally, the papers included in this section can be further classified based on the area of focus as can be seen in Table 8.

Table 8: Classifying studies of general testing category based on area of focus:

Study	Area of focus
S4	Black box test case generation.
S13,S14	Performance testing.
S15,S18	Lifecycle conformance testing.
S16	Testing mobile applications using SOA (Service-Oriented Architecture).
S27	Prototype-based testing.
S29	Testing on real mobile devices.
S34	Adapting Java Path-Finder testing method for mobile applications.
S38	Testing mobile applications using cloud services.
S44,S67, S65	Compatibility testing.
S66	Testing location-based mobile applications.
S76	Adapting web application testing techniques for mobile applications.

4.4 Research Approaches and Contribution Facets

Sub-RQ1: What research approaches do these studies apply and what contribution facets do they provide?

Contribution facets, inspired from (Shahrokni and Feldt, 2013), were classified into metrics, evaluations, methods, tools and frameworks, as shown in Table 9. Metrics is a contribution type that provides guidelines for measuring different aspects of application testing. An evaluation represents an evaluation or assessment of a method using software application or evaluation of a software application using a method. For example, (Franke et al., 2012a) [S15] evaluated techniques of life cycle conformance for mobile applications on mobile services. Methods have a specific goal and research question. An example of a method is the one presented by (Heejin et al., 2009) [S14] to perform performance testing at unit testing level. A tool represents specific software for certain purpose to assist test engineers in their work. Example of such tools is the one presented by (Amalfitano et al., 2011) [S1] that is based on a crawler that automatically builds a model from GUI and generates test cases. Finally, a framework is a detailed method or technique that has a wide purpose and focuses on several research questions (Shahrokni and Feldt, 2013).

We found fourteen (14) studies presenting frameworks for testing mobile and smartphones application ([S3], [S11], [S30], [S2], [S23], [S16], [S38 [S10], [S43], [S42], [S46], [S61], [S62], and [S63]. An example of such a framework is the one suggested by (Nagowah and Sowamber, 2012) [S3] to automate tests on the device itself and not on emulator. In our mapping study, we found that most of the contributions facets (38 out of 79, 48%) were provided as methods. On the other hand, contribution facet in terms of metrics was the least studied (only 1 study). Figure 4 shows the distribution of contribution facets, while Table 9 specifies contribution facet for each study. Further, Table 11 presents a summary of testing tools found in included studies.

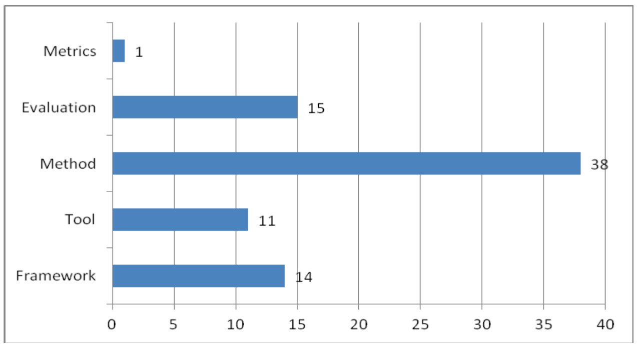

Figure 4: Contribution facets

Table 9: Contribution facet for each study

Types of contribution facet	**Studies (S)**	**Description**
Metrics	S17	Model the impact of software code attributes upon performance and resource utilization in terms of memory, network and CPU usage in context-aware mobile apps
Evaluation	S22	Realizing usability heuristics appropriate for mobile computing.
	S24	Presents findings on usability issues and enhancements requested by users.
	S25	Apply usability engineering in the agile methodology called *InterMod* on mobile development.
	S26	Usability evaluation of MobiTOP application was conducted in the context of a travel companion for tourists.
	S28	Presents a UX case study on mobile advertising with a novel CAVE-smartphone interface
	S33	A usability study on Mobile Electronic Personality Version 2 and conclusions on key issues related to user needs, based on user interviews, surveys, prototypes and field evaluations.
	S35	Presents lessons learned and usability guidelines derived from laboratory usability testing of mobile spreadsheet applications
	S36	Automatically monitor and collect context data and usability metrics, how those data can be processed for analysis support and how users' impressions can be collected.
	S13	Investigation of performance factors such as server response and network connection speed.
	S15	Reverse engineer the service life cycles of mobile platforms.
	S64	Conducts a mobile usability test using mobile eye-tracking glasses through quasi-real conditions.
	S67	Conducts an in-depth evaluation for methods used to select mobile devices for testing.
	S68	Evaluates the use of model-based testing in the construction and implementation of automated tests to verify functional requirements for mobile applications.
	S69	Investigates the importance of usability in user interfaces design of mobile educational applications.
	S77	Investigates the usability of mobile library application at the university of Chongqing.
Method	S6	Combines portable operating system libraries with knowledge and reasoning to support both heterogeneous systems and distributed control.
	S9	An evaluation method based on a field experiment where a group of end users tested the mobile application. Qualitative and quantitative data about end users' impressions were collected and analyzed.
	S19	Proposes a method on how to implement a successful usable m-learning environment
	S37	The study presents a unified methodology for evaluating mobile applications which is based on two fundamental principles: early assessment and the usage of

		ad-hoc and mobile oriented methods.
	S40	Designed a method that aligns the inspection method “Software ArchitecTure analysis of Usability Requirements realizatioN” SATURN with mobile usability evaluation in the form of a user tests.
	S14	Supports performance testing utilizing a database established through benchmark testing in emulator-based test environment at the unit test level.
	S18	Identifies life cycle dependent properties in the application specification, and derives assertion-based test cases for validating the conformance of the properties.
	S27	Presents an evaluation method based on evaluation sessions that consist of various scenarios of diverse situations where users moved through locations and contexts. The sessions were split into two subsets. First one that required the presence of an evaluator using Wizard-of-Oz approach. And another subset without the evaluator.
	S29	Presents a strategy to aid coverage testing for mobile applications in such a way that the applications can be tested not only on emulators, but also on their real target mobile devices.
	S7	Presents testing techniques that take into account both user events and context events. The proposed techniques are based on manual definition of reusable event patterns that include context events.
	S8	Defines and applys a new model for the detection of faults of incorrect adaptation logic and asynchronous updating of context information. Define algorithms to automatically detect such faults
	S5	Identifys context-aware break points by systematically changing the context-aware data fed to app to expose failures.
	S39	Analyzes context-aware applications and identify the main difficulties in validating them then developed validation techniques which address the identified limitations.
	S4	A testing approach based on modeling the application from a black-box perspective and presenting a distance metric for test cases of mobile applications. Additionally the study proposes am ART test case generation technique.
	S41	An automatic functional testing for Android applications based on a model for activity page
	S44	An approach based on combinatorial methods for coverage of each device characteristics.
	S45	An approach that validates code handling exceptional behavior for mobile applications. The approach is based on systematic amplification of the program space explored by a test by manipulating the behavior of external resources.
	S47	A method based on a model-based security testing approach to attempt to detect data vulnerabilities in Android applications.
	S48	Proposes a new Android malware detection method based on machine learning technology.
	S49	Presents a testing approach to test communication among mobile applications with a test case generation strategy and testing adequacy criterion for Android applications.
	S51	Presents a compositional testing approach based on static and dynamic testing techniques to detect security vulnerabilities caused by messaging between components.
	S52	Presents a testing approach based on Pattern Based GUI testing which is a model based as well.
	S53	An approach for testing mobile applications based on reverse engineering and behavioral patterns.
	S54	A testing approach based on extending a biographical sorting predicate logic as constraints to create a meta-model, then using that meta model to build data-model to model context aware environments.
	S55	An approach to modeling mobile test environments based on Mobile Test Environment Semantic Tree.
	S56	Presents an approach that allows substantial Android apps to be systematically

		explored while running on actual devices.
	S58	Presents an automated technique that is based on machine learning that is capable of generating inputs sequences of test inputs for Android applications.
	S60	Presents a testing method based on threat analysis considering possible attack scenarios and vulnerabilities specific to mHealth applications.
	S65	Proposes an optimized compatibility testing strategy based on statistical approach to reduce test costs and improve test efficiency.
	S66	Proposes a new testing method to address the challenges of testing for mobile location-based function services mobile applications.
	S70	Proposes a new search-based test automation technique that is based on genetic and hill climbing techniques.
	S71	Presents a new testing methodology that aims at systematically exploring and executing test suites to expose adverse conditions.
	S72	Proposes a conceptual model to represent inter-component communication at higher abstraction level and a technique to derive test cases from the model.
	S73	Proposes a novel stream-oriented record and reply testing approach capable of recording sensor and network input, event schedules and inter application communication.
	S74	Presents a testing approach based on simulating physical and logical context situations for mobile applications.
	S75	Presents a testing approach to detect Android inter-application communication vulnerabilities.
	S76	Presents an adaptation model for testing mobile applications based on Reweb and Testweb testing frameworks.
	S79	Presents a testing approach to integrate sensor information into test suites using a sensor simulation engine to enable automatic test case execution.
Tool		See Table 11
Framework	S3	Implementing test automation on the device itself, not on an emulator.
	S11	Framework for writing, executing and reporting tests on mobile devices.
	S30	Presents scalable and pervasive testing framework for constantly changing mobile applications, based on SOA architecture.
	S2	A methodology and framework to help developers in preparing mobile app for usability analysis and automation of manual tasks.
	S23	Automated and unsupervised evaluation of mobile applications that is able to trace any user interaction during the entire lifecycle of an application.
	S38	A scalable and light weight framework for effective research crowdsourcing in mobile testing.
	S10	A localization framework targeting at the bugs in the mobile application, especially the buggy context provider faults.
	S43	Supporting the design and automatic execution of UX tests for Android applications.
	S42	an integrated test automation framework by which implementations on multiple heterogeneous platforms can be tested efficiently.
	S16	Presents a flexible and expandable test automation framework based on service-oriented architecture and COM technique.
	S46	Presents a framework based on a combination of static and dynamic program analysis to expose the functionality of mobile application along with all available execution paths.
	S61	Proposes a framework called Automated Mobile Testing as a Service which offers automated mobile application tests.
	S62	Describes a parallel GUI testing technique called PATS that performs GUI testing based on master/slave model.
	S63	Presents a framework that is capable of generating code coverage reports and produce uniform coverage metrics in testing without the need for source code.

Research approaches can be summarized in Table 10. Most of the studies were conducted using a validation research approach (75%). On the other hand, the number of studies performing evaluation research is relatively small (19%). Accordingly, this implies that there is a need for more evaluation research that can evaluate how much effective these new testing methods really are.

Table 10: Research approach facets

Research approach	Study (S)	# of studies
Validation research	S1, S3, S6, S11, S12, S20, S30, S42, S45, S19, S22, S23, S40, S43, S4, S14, S15, S16, S18, S29, S34, S44, S7, S10, S8, S5, S17, S39, S41, S2, S37, S46, S47, S48, S49, S50, S51, S52, S53, S54, S55, S56, S57, S58, S59, S60, S61, S62, S63, S65, S66, S70, S71, S72, S73, S74, S75, S78, S79	59
Evaluation research	S31, S9, S21, S24, S26, S28, S33, S38, S36, S64, S67, S68, S69, S76, S77	15
Experience papers	S32, S25, S35, S13, S27	5

A presentation of studies offering software testing tools is provided at Table 11 showing the name of the tool along with the targeted platform/operating system. It can be seen here that most testing tools are targeting Android platform (64%) and little studies are targeting other successful prevalent platforms such as Apple iOS and Microsoft Phone.

Table 11: A summary of testing tools reported.

Study	Study (S)	Tool Name	Platform	Description
(Amalfitano et al., 2011)	S1	Android Automatic Testing Tool	Android	A crawler-based tool that automatically simulates real user events on user interface and generates a GUI model. The GUI model is used later on to generate test cases for regression testing.
(Jiang et al., 2007)	S12	MobileTest	Symbian	Based on sensitive-event based approach and is targeted to perform automatic black box testing.
(Amalfitano et al., 2012)	S20	AndroidRipper	Android	Based on automated ripper technique that automatically explores the application GUI to exercise the application in structured manner.
(Kaasila et al., 2012)	S31	TestDroid	Android	Based on an online platform for conducting scripted user interface parallel tests on several Android devices.
(Puhakka and Palola, 2006)	S32	Not defined.	Symbian	General testing tool that supports testing of mobile applications on several devices at same time.
(Ravindranath et al., 2012)	S21	AppInsight	Silverlight	A tool that monitors application performance in the wild and reports critical paths and user transactions.
(Merwe et al., 2012)	S34	JPF-Android	Android	Built on Java Pathfinder, allows users to script events to drive the application and can detect deadlocks and runtime exceptions.
(Chan et al., 2012)	S50	DroidChecker	Android	An Android application analyzing tool based on inter-procedural flow graph searching and static taint checking to detect vulnerable data paths.
(Liang et al., 2014)	S57	Caiipa	Windows Phone	A cloud service for testing mobile apps offering expandable mobile context space in a scalable way.

(Hu et al., 2014)	S59	AppDoctor	Android	A system based on approximate execution to test mobile apps against many system and user actions.
(Amalfitano et al., 2015b)	S78	MobiGUITAR	Android	Automates GUI-driven testing of Android applications.

A bubble plot showing counts of papers of our main category against contribution facets can be seen in Figure 5. It can be seen that there is a lack of studies offering metrics in almost all main categories. On the other hand, there are comprehensive studies in the categories of usability and automation testing (see Figure 5).

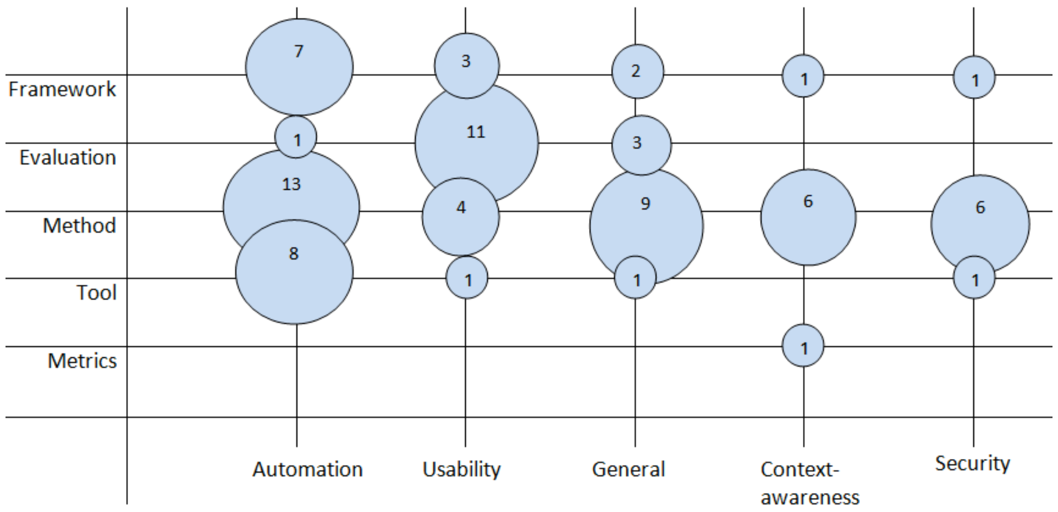

Figure 5: Bubble plot for main categories against contribution facets

Additionally, we found that studies offering contribution facet of type *Frameworks* (14 studies) can be further classified according to what they offer in their solution frameworks. More specifically, we found that these studies offer a combination of tool/API, methods/guideline, models, metrics and algorithms. Table 12 provides a comparison of these studies.

Table 12: Comparison of testing frameworks

	Author(s)	Studies (S)	Tool/API	Methods/ Guideline	Models	Metrics	Algorithms
Test automation	(Nagowah and Sowamber, 2012)	S3	✓	✓			
	(Zhifang et al., 2010a)	S30	✓	✓	✓		✓
	(She et al., 2009)	S11	✓	✓			
	(Song et al., 2011)	S42	✓	✓	✓		
	(Villanes et al., 2015)	S61	✓	✓			
	(Wen et al., 2015)	S62	✓	✓	✓		
	(Zhauniarovich et al., 2015)	S63	✓	✓		✓	
Usability testing	(Balagtas-Fernandez and Hussmann, 2009),	S2	✓	✓	✓	✓	

	(Lettner and Holzmann, 2012)	**S23**	✓	✓	✓	✓	
	(Canfora et al., 2013)	**S43**	✓			✓	
Context-awareness	**(Bo et al., 2011)**	**S10**		✓		✓	
Security	**(Johnson et al., 2013)**	**S46**	✓	✓			
General	**(Zhi-fang et al., 2009)**	**S16**	✓	✓	✓		
	(Starov, 2013)	**S38**	✓	✓	✓		

4.5 Object Involved in the Study and Research Context

Sub-RQ2: What kind of applications (industrial or simple) do these studies use in order to evaluate their solutions?

Most of the studies under the category of usability testing (17 studies) had their evaluation done on real world mobile applications. For the rest of the categories: twenty (20) studies under test automation, eight (8) studies under security testing, ten (10) studies under general and four (4) studies under the category of context-awareness were conducted on real industrial applications. Examples on real world applications and context are China Telecommunication Lab (Jiang et al., 2007) [S12] and Word Press for Android (Amalfitano et al., 2012) [S20]. The rest of the studies had their evaluation done on a toy application or on a considerably simple application developed for the sole purpose of the experiment. Table 13 shows the objects involved in each study.

Table 13: Objects involved in each study

Category	**List of Studies**	**Evaluation object**
Test Automation	S1, S3, S30, S32, S41, S42, S53, S68, S79	Simple
	S6, S11,S12, S20, S31, S45, S52, S55, S56, S57, S58, S59, S61, S62, S63, S70, S71, S72, S73, S78	Industrial
Usability Testing	S2, S19	Simple
	S9, S21, S22, S23, S24, S25, S26, S28, S33, S35, S36, S37, S40, S43, S64, S69, S77	Industrial
Security Testing	S46, S47, S48, S49, S50, S51, S60, S75	Industrial
General	S14, S29, S34, S15, S76	Simple
	S4, S13, S16, S18, S27, S38, S44, S65, S66, S67	Industrial
Context awareness	S8, S17, S39, S54	Simple
	S7, S10, S5, S74	Industrial

Additionally, three studies (Balagtas-Fernandez and Hussmann, 2009) [S2], (Bertini et al., 2006) [S22], (Losada et al., 2012) [S25] under the category of usability testing were conducted under a real development team setting. In these studies, either the evaluation of the proposed solution was done using real-world application developers, or the problem itself had partially emerged from the needs of real-world teams and experience.

4.6 Publication Fora

Sub-RQ3: Which journals and conferences included papers on mobile and smart phone application testing?

Tables 14 and 15 show the list of journals and conferences for each of the studies included in this mapping study. Among the 79 studies, 13 came from journals, 2 from dissertations ((Starov, 2013) [S38], (Wang, 2008) [S39]) and the rest (64) from conferences. We found that the conference papers were published in a total of 39 conference proceedings. These data show that research on mobile application testing appears in diverse conference venues including workshops.

Table 14: List of journals

Study (S)	Journal Name	# of studies
S34	ACM SIGSOFT Software Engineering Notes	1
S45	ACM Transactions on Software Engineering and Methodology (TOSEM)	1
S56, S58	ACM SIGPLAN Notices	2
S40	Journal of Systems and Software	1
S33	Personal Ubiquitous Computing	1
S8	IEEE Transactions on Software Engineering,	1
S37	Universal Access in the Information Society	1
S68	Electronic Notes in Theoretical Computer Science	1
S69	Procedia-Social and Behavioral Sciences	1
S76	growth	1
S77	Library Hi Tech	1
S78	IEEE Software	1

Table 15: List of Conferences

Study (S)	Conference Name	# studies
S10	International Conference on Advanced Computer Control (ICACC)	1
S20,	International Conference on Automated Software Engineering (ASE)	1
S3, S2	International Conference on Computer & Information Science (ICCIS)	2
S4	International Conference on Computer Engineering and Technology (ICCET)	1
S15	International Workshop on Database and Expert Systems Applications (DEXA)	1
S19	International Conference on Information Technology Interfaces	1
S13	International Conference Information Technology	1
S9	IFIP International Conference on New Technologies, Mobility and Security (NTMS	1
S23	International Conference on Advances in Mobile Computing Multimedia	1
S27	International conference on Human computer interaction with mobile devices and services	1
S21	USENIX conference on Operating Systems Design and Implementation	1
S31	International Conference on Mobile and Ubiquitous Multimedia	1
S25	International Conference on Interacción Persona-Ordenador	1
S29, S30, S12, S49, S51	International workshop on Automation of software test	5
S32	International conference on Mobile technology, applications & systems	1
S24	Workshop on Mobile video delivery	1
S35, S36	International Conference on Human-Centered Software Engineering	2
S26	International Conference on Advances in Mobile Computing and Multimedia	1
S28	Conference on Human Factors in Computing Systems (SIGCHI)	1

S22	Conference on Advanced visual interfaces	1
S16	International Conference on Reliability, Maintainability and Safety. (ICRMS)	1
S14	IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI)	1
S6	IEEE International Conference on Service-Oriented Computing and Applications (SOCA)	1
S42	ACIS International Symposium on Software and Network Engineering (SSNE)	1
S11	Australian Software Engineering Conference, ASWEC	1
S5	International Conference on Software Engineering (ICSE)	1
S17	IEEE International Symposium on Software Metrics	1
S18, S43	IEEE International Conference on Software Testing, Verification and Validation (ICST)	2
S1, S7, S44, S60	IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)	4
S41	Third World Congress on Software Engineering	1
S46	Reliability and Maintainability Symposium (RAMS)	1
S47	IEEE Information Security of South Africa	1
S48	IEEE Third International Conference on Instrumentation, Measurement, Computer, Communication and Control (IMCCC)	1
S50	Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM	1
S52, S53	IEEE International Conference on the Quality of Information and Communications Technology (QUATIC)	2
S54	IEEE International Conference on Software Security and Reliability (SERE)	1
S55	Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing	1
S57	International conference on Mobile computing and networking	1
S59	European Conference on Computer Systems	1
S61	IEEE World Congress on Services (SERVICES), 2015	1
S62	IEEE 39th Annual Computer Software and Applications Conference (COMPSAC), 2015	1
S63	10th International Conference on Availability, Reliability and Security (ARES), 2015	1
S64	8th International Conference on Human System Interactions (HSI), 2015	1
S65, S66	IEEE Symposium on Service-Oriented System Engineering 2015	2
S67, S72	International Conference on Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM	2
S70	3rd International Workshop on Software Development Lifecycle for Mobile	1
S71, S75	Proceedings of the 2015 International Symposium on Software Testing and Analysis	2
S73	Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications	1
S74	Proceedings of the 30th Annual ACM Symposium on Applied Computing	1

4.7 Suggested Areas for Further Research

In this section, we present possible areas that require further research based on the findings from our mapping study. The suggested research gaps presented herein are categorized based on the classification structure we used in our mapping study. The suggested areas for further research are based on our interpretation of the findings and data as well as number of studies found in each area.

- **Test automation:** in this category, we suggest four (4) further research needs:

  (i) **Eliciting testing requirements during requirements analysis phase:** Among the 79 studies included in this systematic mapping study, only two studies ((Franke et al., 2012a) [S15] and (Franke et al., 2012b) [S18]) had a small consideration to this important issue. In these two studies, the solution they propose is based on the application requirements specification to derive assertions in an assertion-based testing approach. However, this step has to be done manually by the developer.

  (ii) **Life-cycle conformance testing:** Only two studies ((Franke et al., 2012b) [S18], (Franke et al., 2012a) [S15]) propose and evaluate approaches to test conformance to life cycle models. The first study ((Franke et al., 2012b) [S18]) focuses on mobile applications, while the second study ((Franke et al., 2012a) [S15]) targets mobile services. Both studies highlight the importance of life cycle model awareness in order to produce dependable mobile application and services. Their testing approach, which is based on case study demonstration, first identifies the most common and important scenarios that represent common transitions of life-cycle states an application can go through. Life-cycle callback methods are then derived using unit-testing framework. Finally, they identify life-cycle dependent properties and derive assertion based test cases. Consequently, we suggest an automated tool that can check application program to see if it complies with lifecycle models and rules.

  (iii) **Mobile services testing:** We found only one study ((Franke et al., 2012a) [S15]) addresses testing of mobile services in the specific area of service life cycle models. In this study, the first part presents how to reverse engineer the service life cycles of mobile platforms. The second part applies test approaches from mobile application life cycle conformance to services. In an informal review, (Muccini et al., 2012) highlights that mobile services testing is an area that requires further research. In our mapping study, we found only one study that addresses mobile services testing, and thus further research in this topic is required.

  (iv) **Real world contexts:** We believe that mobile application automation testing tools and frameworks could be better evaluated by real world development teams and developers. Since these tools and frameworks aim at simplifying developers' manual tasks, they have to be evaluated by real developer on real world application in order to get more convincing feedback.

- **Metrics:** We only found one study by (Ryan and Rossi, 2005) that is concerned with software testing metrics for mobile applications. Accordingly we highlight it as a research gap.

- **Usability testing:** It can be seen in this mapping study that there is relatively large number (16) of studies in the area of usability testing, most of which do validation of their solutions using real world applications. We suggest a comparative study that compares the effectiveness of different approaches suggested by these studies.

- **Security testing:** most of the studies under the category of security testing address the problem of mobile application inter-communication threats, more specifically for Android applications. Here we also suggest a comparative study to evaluate the effectiveness of these different approaches. Additionally, we suggest that future studies should consider inter-communication threats for other platforms such as iOS.

5. Discussion

The purpose of this systematic mapping study was to build a classification scheme and to collect, interpret and analyze all evidence related to mobile and smart-phone application testing techniques, approaches and challenges. Currently, there are no comprehensive and convincing systematic mapping studies done in this important and constantly evolving area. Accordingly, a thorough and unbiased systematic mapping review could contribute to the body of knowledge of mobile application testing studies.

This study indicates several research gaps that acquire further research and investigation. First, the studies that were categorized under test automation and general testing categories (see Table 13) identify problems and, consequently propose solutions that are based on laboratory environments. If these studies however, were based on identification of problems observed in **real development environments,** and later on, applied and evaluated their solutions in such environments, the scientific value of these studies would have been more convincing and comprehensive. It is true however, that some of these studies (see Table 13) used industrial applications to evaluate their solutions. However, these applications are mostly simple (text editor, email composer, etc.) and do not represent real-world and complex mobile applications. We highly recommend that evaluation of test automation tools should be applied on more complex applications that represent different aspects such as context-awareness and mission critical systems. Further, they did not base their studies on the needs of real-world development and testing teams.

Secondly, there is a lack of studies addressing the challenge of **eliciting testing requirements** that are related to mobile application testing peculiarities. As discussed in Section 4.7, only two studies ((Franke et al., 2012b) [S18] and(Franke et al., 2012a) [S15]) had a considerably minor awareness in this area. For instance, an interesting challenge could be on how to elicit testing requirements related to mobile application life cycle properties from requirements specifications. This finding interestingly, corroborates the finding reached by (Shahrokni and Feldt, 2013). Further, it is important to note that test engineers should derive testing requirements for their application early during requirements elicitation phase (Crispin and Gregory, 2008). Test engineers should be aware on how to elicit specific testing requirements in the areas of life-cycle conformance and context-awareness for example so that they can choose appropriate testing techniques and tools.

To elaborate more, except the study by (Franke et al., 2012a) [S15], there are no other empirical studies addressing testing challenges of **mobile services testing**. Mobile services are currently targeting time and safety critical contexts such as abnormal and disaster management situations, and hence, requires a high availability and reliability of such services. Additionally, there is one study offering software testing metric for mobile applications.

Further, and as shown in Section *4.7*, only two empirical studies so far ([S15] and [S18]) address how to test conformance of mobile application to **life cycle models**. It is important to note though that these two studies are very recent and their approaches are basic in the sense that most of the critical steps proposed are manual and depend on the developer or tester's way of thinking and perception on problems in hand. Consequently, lots of further research can be done in this challenging area to provide a better solution or maybe automated solution to such tasks. In addition, most of the software testing tools provided target the Android platform. Consequently, more studies should target other prevalent platforms such as Apple iOS and Windows Phone as well.

Finally, and based on our observation that most research approaches are applied using validation research, we believe that test engineers would find it difficult to choose among test automation tools and techniques available under the category of test automation, usability and security testing. This is due to the fact that several test methods and techniques exist with no clear road-map available for test engineers guiding them on which tool to choose or technique to apply. We recommend future studies that can compare such testing solutions using evaluation research, and evaluate them on several complex and real world mobile applications.

5.1 Threats to Validity

In this mapping study, several factors need to be taken into account when generalizing the results. First, during the process of identifying the relevant literature, we only considered articles published electronically. This however, may have neglected studies that might have appeared in journals or conference proceedings that were not published online. But since mobile and smart phone applications are considered to be relatively young (earliest included study in this study published in 2005), we believe that it is unlikely that such studies are not available online.

Other key threats to validity of the results are related to bias in the selection of studies and inaccurate data extraction. Several search strings were tested in order to choose the most appropriate search string. However, it is not possible to guarantee that all relevant studies were returned and there is a slight risk that some studies were omitted due to search terms. Nonetheless, we have piloted the selection of search terms and this could help minimize the possibility of missing important evidence.

We excluded work that was not empirically tested i.e. only empirical studies were included in our study. This will have excluded those works that do not have an empirical testing component to validate the papers' experiments and conclusions. Our rationale was that studies with an empirical component provide some level of validation of the claimed benefits and limitations of the testing approach discussed.

Regarding data extraction process, it may have been negatively impacted by bias when selecting articles. This is mainly due to the fact that data extraction was performed by a single researcher (the first author). In order to validate the data extraction, the second author randomly selected 10% of the selected studies and completed the extraction form. We compared the results in a meeting and discussed if there is any discrepancies found until a consensus is met.

The presentation of results might have been affected by the structure of classification scheme. In this mapping study, the classification scheme developed consists of four main categories (i.e. structure of the topic, contribution facets, objects involved in the study, and the research facet). Such categorization emerged from existing guidelines and relevant secondary studies (e.g. Shahrokni and Feldt, 2013 and Petersen et al., 2008) and this might potentially neglects the analysis of other relevant attributes possibly appeared in the primary studies.

6. Conclusions and Future Research

Based on the findings from our systematic mapping study, this paper presents a review of the state of knowledge of empirical studies in the field of mobile and smart-phone application testing. A total of 7356 studies from six online databases were analyzed and went through three filtration steps. A total of 79 studies were finally included and mapped to our classification scheme in our study based on predefined inclusion/exclusion criteria.

The classification scheme contained four categories: first one is the main category to structure the topic and containing test automation, usability testing, context-awareness testing, security and general testing sub-categories. These sub-categories represent the main focus area in which the studies addressed. Then at each topic (sub-category), the techniques or methods applied by each study were discussed briefly. Second classification represents the contribution facet, namely framework, method, tool, evaluation and metrics. The third classification represents objects involved in the study as simple evaluation application or real world industrial application. The fourth classification represents the research facet or research approaches applied in the paper. These include validation, evaluation and experience studies.

Several research gaps were identified as described in the Discussion section. First, among the studies under the categories of test automation and testing in general, very little base their investigation on real-world mobile application development environments. Secondly, there is a lack of studies that focus on eliciting testing requirements in the requirements engineering phase. Additionally, only one study was reported to investigate testing of mobile services as well as one study for testing metrics. In addition, more studies should address the important issues of conformance to life cycle models, mobile services and mobile testing metrics.

It is also noted that practitioners will find it difficult to choose from several testing techniques and automation tools and that there is a need for a clear road-map to guide them. Additionally we recommend on future studies that can systematically compare proposed testing solutions under categories of automation, usability and security using real-world and complex applications. In our future work we will conduct a study targeting real-world mobile application development environments to elicit specific testing needs and peculiarities. Consequently, a test framework will be developed incorporating solutions to address specific research gaps identified in this mapping study.

Acknowledgment

This research was funded by the Ministry of Higher Education Malaysia under FRGS research grant (FRGS14-125-0366).

APPENDIX A

LIST OF INCLUDED STUDIES

The references listed below correspond to those prefaced with the letter “S” throughout the paper.

S1	Amalfitano, D., Fasolino, A.R., Tramontana, P., 2011. A GUI Crawling-Based Technique for Android Mobile Application Testing, Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, pp. 252-261.
S2	Balagtas-Fernandez, F., Hussmann, H., 2009. A Methodology and Framework to Simplify Usability Analysis of Mobile Applications, Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on, pp. 520-524
S3	Nagowah, L., Sowamber, G., 2012. A novel approach of automation testing on mobile devices, Computer & Information Science (ICCIS), 2012 International Conference on, pp. 924-930
S4	Zhifang, L., Xiaopeng, G., Xiang, L., 2010b. Adaptive random testing of mobile application, Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, pp. V2-297-V292-301
S5	Zhimin, W., Elbaum, S., Rosenblum, D.S., 2007. Automated Generation of Context-Aware Tests, Software Engineering, 2007. ICSE 2007. 29th International Conference on, pp. 406-415
S6	Edmondson, J., Gokhale, A., Sandeep, N., 2011. Automating testing of service-oriented mobile applications with distributed knowledge and reasoning, Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on, pp. 1-4
S7	Amalfitano, D., Fasolino, A.R., Tramontana, P., Amatucci, N., 2013. Considering Context Events in Event-Based Testing of Mobile Applications, Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, pp. 126-133
S8	Sama, M., Elbaum, S., Raimondi, F., Rosenblum, D.S., Zhimin, W., 2010. Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification. Software Engineering, IEEE Transactions on 36, 644-661
S9	Bjornestad, S., Tessem, B., Nyre, L., 2011. Design and Evaluation of a Location-Based Mobile News Reader, New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, pp.

	1-4.
S10	Bo, J., Xiang, L., Xiaopeng, G., Zhifang, L., Chan, W.K., 2011. FLOMA: Statistical fault localization for mobile embedded system, Advanced Computer Control (ICACC), 2011 3rd International Conference on, pp. 396-400.
S11	She, S., Sivapalan, S., Warren, I., 2009. Hermes: A Tool for Testing Mobile Device Applications, Software Engineering Conference, 2009. ASWEC '09. Australian, pp. 121-130.
S12	Jiang, B., Long, X., Gao, X., 2007. MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices, Automation of Software Test , 2007. AST '07. Second International Workshop on, pp. 8-8.
S13	Briseno, M.V., Vincent, P., 2008. Observations on performance of client-server mobile applications, Information Technology, 2008. IT 2008. 1st International Conference on, pp. 1-4.
S14	Heejin, K., Byoungju, C., Wong, W.E., 2009. Performance Testing of Mobile Applications at the Unit Test Level, Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. Third IEEE International Conference on, pp. 171-180.
S15	Franke, D., Elsemann, C., Kowalewski, S., 2012a. Reverse Engineering and Testing Service Life Cycles of Mobile Platforms, Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on, pp. 16-20.
S16	Zhi-fang, L., Bin, L., Xiao-peng, G., 2009. SOA based mobile application software test framework, Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on, pp. 765-769.
S17	Ryan, C., Rossi, P., 2005. Software, performance and resource utilisation metrics for context-aware mobile applications, Software Metrics, 2005. 11th IEEE International Symposium, pp. 10 pp.-12.
S18	Franke, D., Kowalewski, S., Weise, C., Prakobkosol, N., 2012b. Testing Conformance of Life Cycle Dependent Properties of Mobile Applications, Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on, pp. 241-250.
S19	Fetaji, M., Dika, Z., Fetaji, B., 2008. Usability testing and evaluation of a mobile software solution: A case study, Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on, pp. 501-506.
S20	Amalfitano, D., Fasolino, A.R., Tramontana, P., De Carmine, S., Memon, A.M., 2012. Using GUI ripping for automated testing of Android applications, Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on, pp. 258-261.
S21	Ravindranath, L., Padhye, J., Agarwal, S., Mahajan, R., Obermiller, I., Shayandeh, S., 2012. AppInsight: mobile app performance monitoring in the wild, Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation. USENIX Association, Hollywood, CA, USA, pp. 107-120.
S22	Bertini, E., Gabrielli, S., Kimani, S., 2006. Appropriating and assessing heuristics for mobile computing, Proceedings of the working conference on Advanced visual interfaces. ACM, Venezia, Italy, pp. 119-126.
S23	Lettner, F., Holzmann, C., 2012. Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications, Proceedings of the 10th International Conference on Advances in Mobile Computing and Multimedia. ACM, Bali, Indonesia, pp. 118-127.
S24	Vaataja, H., Mannisto, A., 2010. Bottlenecks, usability issues and development needs in creating and delivering news videos with smart phones, Proceedings of the 3rd workshop on Mobile video delivery. ACM, Firenze, Italy, pp. 45-50.
S25	Losada, B., Urretavizcaya, M., Lopez, J., Castro, I., 2012. Combining InterMod agile methodology with usability engineering in a mobile application development, Proceedings of the 13th International Conference on Interacci&oacute;n Persona-Ordenador. ACM, Elche, Spain, pp. 1-8.
S26	Pham, T.P., Razikin, K., Goh, D.H.-L., Kim, T.N.Q., Quach, H.N.H., Theng, Y.-L., Chua, A.Y.K., Lim, E.-P., 2010. Investigating the usability of a mobile location-based annotation system, Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia. ACM, Paris, France, pp. 313-320.
S27	Sa, M., Carrico, L., 2008. Lessons from early stages design of mobile applications, Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. ACM, Amsterdam, The Netherlands, pp. 127-136.
S28	Huhn, A., Khan, V., Lucero, A., Ketelaar, P., 2012. On the use of virtual environments for the evaluation of location-based applications, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Austin, Texas, USA, pp. 2569-2578.

S29	Delamaro, M.E., Vincenzi, A.M.R., Maldonado, J.C., 2006. A strategy to perform coverage testing of mobile applications, Proceedings of the 2006 international workshop on Automation of software test. ACM, Shanghai, China, pp. 118-124.
S30	Zhifang, L., Bin, L., Xiaopeng, G., 2010a. Test automation on mobile device, Proceedings of the 5th Workshop on Automation of Software Test. ACM, Cape Town, South Africa, pp. 1-7.
S31	Kaasila, J., Ferreira, D., Kostakos, V., Ojala, T., 2012. Testdroid: automated remote UI testing on Android, Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia. ACM, Ulm, Germany, pp. 1-4.
S32	Puhakka, T., Palola, M., 2006. Towards automating testing of communicational B3G applications, Proceedings of the 3rd international conference on Mobile technology, applications &#38; systems. ACM, Bangkok, Thailand, p. 27.
S33	Oyomno, W., Jappinen, P., Kerttula, E., Heikkinen, K., 2013. Usability study of ME2.0. Personal Ubiquitous Comput. 17, 305-319.
S34	Merwe, H.v.d., Merwe, B.v.d., Visser, W., 2012. Verifying android applications using Java PathFinder. SIGSOFT Softw. Eng. Notes 37, 1-5.
S35	Flood, D., Harrison, R., Iacob, C., 2012. Lessons learned from evaluating the usability of mobile spreadsheet applications, Proceedings of the 4th international conference on Human-Centered Software Engineering. Springer-Verlag, Toulouse, France, pp. 315-322.
S36	Kronbauer, A.H., Santos, C.A.S., Vieira, V., 2012. Smartphone applications usability evaluation: a hybrid model and its implementation, Proceedings of the 4th international conference on Human-Centered Software Engineering. Springer-Verlag, Toulouse, France, pp. 146-163.
S37	Billi, M., Burzagli, L., Catarci, T., Santucci, G., Bertini, E., Gabbanini, F., Palchetti, E., 2010. A unified methodology for the evaluation of accessibility and usability of mobile applications. Univ Access Inf Soc 9, 337-356.
S38	Starov, O., 2013. Cloud platform for research crowdsourcing in mobile testing. East Carolina University, Ann Arbor, p. 12.
S39	Wang, Z., 2008. Validating context-aware applications. The University of Nebraska - Lincoln, Ann Arbor, p. 186.
S40	Biel, B., Grill, T., Gruhn, V., 2010. Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. Journal of Systems and Software 83, 2031-2044.
S41	Lu, L., Hong, Y., Huang, Y., Su, K., Yan, Y., 2012. Activity page based functional test automation for android application, Third World Congress on Software Engineering. IEEE, pp. 37-40.
S42	Song, H., Ryoo, S., Kim, J.H., 2011. An integrated test automation framework for testing on heterogeneous mobile platforms, Software and Network Engineering (SSNE), 2011 First ACIS International Symposium. IEEE, pp. 141-145.
S43	Canfora, G., Mercaldo, F., Visaggio, C.A., D'Angelo, M., Furno, A., Manganelli, C., 2013. A case study of automating user experience-oriented performance testing on smartphones, Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference, pp. 66-69.
S44	Vilkomir, S., Amstutz, B., 2014. Using Combinatorial Approaches for Testing Mobile Applications, Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on. IEEE, pp. 78-83.
S45	Zhang, P., Elbaum, S., 2014. Amplifying Tests to Validate Exception Handling Code: An Extended Study in the Mobile Application Domain. ACM Transactions on Software Engineering and Methodology (TOSEM) 23, 32.
S46	Johnson, R., Wang, Z., Stavrou, A., Voas, J., 2013. Exposing software security and availability risks for commercial mobile devices, Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual. IEEE, pp. 1-7
S47	Salva, S., Zafimiharisoa, S.R., 2013. Data vulnerability detection by security testing for Android applications, Information Security for South Africa, 2013. IEEE, pp. 1-8.
S48	Lu, Y., Zulie, P., Jingju, L., Yi, S., 2013. Android Malware Detection Technology Based on Improved Bayesian Classification, Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2013 Third International Conference on. IEEE, pp. 1338-1341

S49	Avancini, A., Ceccato, M., 2013. Security testing of the communication among Android applications, Proceedings of the 8th International Workshop on Automation of Software Test. IEEE Press, pp. 57-63.
S50	Chan, P.P., Hui, L.C., Yiu, S.-M., 2012. Droidchecker: analyzing android applications for capability leak, Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, pp. 125-136.
S51	Guo, C., Xu, J., Yang, H., Zeng, Y., Xing, S., 2014. An automated testing approach for inter-application security in Android, Proceedings of the 9th International Workshop on Automation of Software Test. ACM, pp. 8-14.
S52	Costa, P., Paiva, A.C., Nabuco, M., 2014. Pattern Based GUI testing for Mobile Applications, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC). IEEE, pp. 66-74.
S53	Morgado, I.C., Paiva, A.C., Faria, J.P., 2014. Automated Pattern-Based Testing of Mobile Applications, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC). IEEE, pp. 294-299.
S54	Yu, L., Tsai, W.T., Jiang, Y., Gao, J., 2014. Generating test cases for context-aware applications using bigraphs, Software Security and Reliability (SERE), 2014 Eighth International Conference on. IEEE, pp. 137-146.
S55	Tao, C., Gao, J., 2014. Modeling mobile application test platform and environment: testing criteria and complexity analysis, Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing. ACM, pp. 28-33.
S56	Azim, T., Neamtiu, I., 2013. Targeted and depth-first exploration for systematic testing of android apps. ACM SIGPLAN Notices 48, 641-660.
S57	Liang, C.-J.M., Lane, N.D., Brouwers, N., Zhang, L., Karlsson, B.F., Liu, H., Liu, Y., Tang, J., Shan, X., Chandra, R., 2014. Caiipa: Automated large-scale mobile app testing through contextual fuzzing, Proceedings of the 20th annual international conference on Mobile computing and networking. ACM, pp. 519-530.
S58	Choi, W., Necula, G., Sen, K., 2013. Guided gui testing of android apps with minimal restart and approximate learning, ACM SIGPLAN Notices. ACM, pp. 623-640.
S59	Hu, G., Yuan, X., Tang, Y., Yang, J., 2014. Efficiently, effectively detecting mobile app bugs with AppDoctor, Proceedings of the Ninth European Conference on Computer Systems. ACM, p. 18.
S60	Knorr, K., Aspinall, D., 2015. Security testing for Android mHealth apps, Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on. IEEE, pp. 1-8.
S61	Villanes, I.K., Bezerra Costa, E.A., Dias-Neto, A.C., 2015. Automated Mobile Testing as a Service (AM-TaaS), Services (SERVICES), 2015 IEEE World Congress on. IEEE, pp. 79-86.
S62	Wen, H.-L., Lin, C.-H., Hsieh, T.-H., Yang, C.-Z., 2015. PATS: A Parallel GUI Testing Framework for Android Applications, Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE, pp. 210-215.
S63	Zhauniarovich, Y., Philippov, A., Gadyatskaya, O., Crispo, B., Massacci, F., 2015. Towards Black Box Testing of Android Apps, Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, pp. 501-510
S64	Borys, M., Milosz, M., 2015. Mobile application usability testing in quasi-real conditions, Human System Interactions (HSI), 2015 8th International Conference on. IEEE, pp. 381-387.
S65	Zhang, T., Gao, J., Cheng, J., Uehara, T., 2015. Compatibility Testing Service for Mobile Applications, 2015 IEEE Symposium on Service-Oriented System Engineering
S66	Aktouf, O.E.K., Tao, Z., Gao, J., Uehara, T., 2015. Testing Location-Based Function Services for Mobile Applications, Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on, pp. 308-314.
S67	Vilkomir, S., Marszalkowski, K., Perry, C., Mahendrakar, S., 2015. Effectiveness of multi-device testing mobile applications, Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on. IEEE, pp. 44-47.
S68	de Cleva Farto, G., Endo, A.T., 2015. Evaluating the Model-Based Testing Approach in the Context of Mobile Applications. Electronic Notes in Theoretical Computer Science 314, 3-21.
S69	Masood, M., Thigambaram, M., 2015. The Usability of Mobile Applications for Pre-schoolers. Procedia-Social and Behavioral Sciences 197, 1818-1826.
S70	Amalfitano, D., Amatucci, N., Fasolino, A.R., Tramontana, P., 2015. AGRippin: a novel search based testing technique for Android applications, Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile. ACM, pp. 5-12.
S71	Adamsen, C.Q., Mezzetti, G., Møller, A., 2015. Systematic execution of Android test suites in adverse conditions, Proceedings of the 2015 International Symposium on Software Testing and Analysis. ACM, pp. 83-93.
S72	Kumar, A., Lee, S., Lee, W.J., 2015. Modeling and test case generation of inter-component communication in android, Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM

	International Conference on. IEEE, pp. 113-116.
S73	Hu, Y., Azim, T., Neamtiu, I., 2015. Versatile yet lightweight record-and-replay for Android, Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. ACM, pp. 349-366.
S74	Vieira, V., Holl, K., Hassel, M., 2015. A context simulator as testing support for mobile apps, Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, pp. 535-541.
S75	Hay, R., Tripp, O., Pistoia, M., 2015. Dynamic detection of inter-application communication vulnerabilities in Android, Proceedings of the 2015 International Symposium on Software Testing and Analysis. ACM, pp. 118-128.
S76	Ahmed, M., Ibrahim, R., Ibrahim, N., 2015. An Adaptation Model for Android Application Testing with Refactoring. growth 9.
S77	Wei, Q., Chang, Z., Cheng, Q., 2015. Usability study of the mobile library App: an example from Chongqing University. Library Hi Tech 33, 340-355.
S78	Amalfitano, D., Fasolino, A.R., Tramontana, P., Ta, B.D., Memon, A.M., 2015b. MobiGUITAR: Automated Model-Based Testing of Mobile Apps. IEEE Software 32, 53-59.
S79	Griebe, T., Hesenius, M., Gruhn, V., 2015. Towards Automated UI-Tests for Sensor-Based Mobile Applications, in: Fujita, H., Guizzi, G. (Eds.), Intelligent Software Methodologies, Tools and Techniques. Springer International Publishing, pp. 3-17.

REFERENCES

2014a. Monkey Talk. (21/12/2014), Retrieved from https://www.cloudmonkeymobile.com/monkeytalk

2014b. Robotium. (21/12/2014), Retrieved from https://code.google.com/p/robotium/

Adamsen, C.Q., Mezzetti, G., Møller, A., 2015. Systematic execution of Android test suites in adverse conditions, Proceedings of the 2015 International Symposium on Software Testing and Analysis. ACM, pp. 83-93.

Ahmed, M., Ibrahim, R., Ibrahim, N., 2015. An Adaptation Model for Android Application Testing with Refactoring. growth 9.

Aktouf, O.E.K., Tao, Z., Gao, J., Uehara, T., 2015. Testing Location-Based Function Services for Mobile Applications, Service-Oriented System Engineering (SOSE), 2015 IEEE Symposium on, pp. 308-314.

Amalfitano, D., Amatucci, N., Fasolino, A.R., Tramontana, P., 2015a. AGRippin: a novel search based testing technique for Android applications, Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile. ACM, pp. 5-12.

Amalfitano, D., Fasolino, A.R., Tramontana, P., 2011. A GUI Crawling-Based Technique for Android Mobile Application Testing, Software Testing, Verification and Validation Workshops (ICSTW), 2011 IEEE Fourth International Conference on, pp. 252-261.

Amalfitano, D., Fasolino, A.R., Tramontana, P., Amatucci, N., 2013. Considering Context Events in Event-Based Testing of Mobile Applications, Software Testing, Verification and Validation Workshops (ICSTW), 2013 IEEE Sixth International Conference on, pp. 126-133.

Amalfitano, D., Fasolino, A.R., Tramontana, P., De Carmine, S., Memon, A.M., 2012. Using GUI ripping for automated testing of Android applications, Automated Software Engineering (ASE), 2012 Proceedings of the 27th IEEE/ACM International Conference on, pp. 258-261.

Amalfitano, D., Fasolino, A.R., Tramontana, P., Ta, B.D., Memon, A.M., 2015b. MobiGUITAR: Automated Model-Based Testing of Mobile Apps. IEEE Software 32, 53-59.

Avancini, A., Ceccato, M., 2013. Security testing of the communication among Android applications, Proceedings of the 8th International Workshop on Automation of Software Test. IEEE Press, pp. 57-63.

Azim, T., Neamtiu, I., 2013. Targeted and depth-first exploration for systematic testing of android apps. ACM SIGPLAN Notices 48, 641-660.

Bailey, J., Budgen, D., Turner, M., Kitchenham, B., Brereton, P., Linkman, S.G., 2007. Evidence relating to Object-Oriented software design: A survey, ESEM. Citeseer, pp. 482-484.

Balagtas-Fernandez, F., Hussmann, H., 2009. A Methodology and Framework to Simplify Usability Analysis of Mobile Applications, Automated Software Engineering, 2009. ASE '09. 24th IEEE/ACM International Conference on, pp. 520-524.

Bertini, E., Gabrielli, S., Kimani, S., 2006. Appropriating and assessing heuristics for mobile computing, Proceedings of the working conference on Advanced visual interfaces. ACM, Venezia, Italy, pp. 119-126.

Biel, B., Grill, T., Gruhn, V., 2010. Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application. Journal of Systems and Software 83, 2031-2044.

Billi, M., Burzagli, L., Catarci, T., Santucci, G., Bertini, E., Gabbanini, F., Palchetti, E., 2010. A unified methodology for the evaluation of accessibility and usability of mobile applications. Univ Access Inf Soc 9, 337-356.

Bjornestad, S., Tessem, B., Nyre, L., 2011. Design and Evaluation of a Location-Based Mobile News Reader, New Technologies, Mobility and Security (NTMS), 2011 4th IFIP International Conference on, pp. 1-4.

Bo, J., Xiang, L., Xiaopeng, G., Zhifang, L., Chan, W.K., 2011. FLOMA: Statistical fault localization for mobile embedded system, Advanced Computer Control (ICACC), 2011 3rd International Conference on, pp. 396-400.

Borys, M., Milosz, M., 2015. Mobile application usability testing in quasi-real conditions, Human System Interactions (HSI), 2015 8th International Conference on. IEEE, pp. 381-387.

Braun, V., Clarke, V., 2006. Using thematic analysis in psychology. Qualitative research in psychology 3, 77-101.

Briseno, M.V., Vincent, P., 2008. Observations on performance of client-server mobile applications, Information Technology, 2008. IT 2008. 1st International Conference on, pp. 1-4.

Bruegge, B., Dutoit, A.H., 2004. Object-Oriented Software Engineering Using UML, Patterns and Java-(Required). Prentice Hall.

Canfora, G., Mercaldo, F., Visaggio, C.A., D'Angelo, M., Furno, A., Manganelli, C., 2013. A case study of automating user experience-oriented performance testing on smartphones, Software Testing, Verification and Validation (ICST), 2013 IEEE Sixth International Conference, pp. 66-69.

Chan, P.P., Hui, L.C., Yiu, S.-M., 2012. Droidchecker: analyzing android applications for capability leak, Proceedings of the fifth ACM conference on Security and Privacy in Wireless and Mobile Networks. ACM, pp. 125-136.

Choi, W., Necula, G., Sen, K., 2013. Guided gui testing of android apps with minimal restart and approximate learning, ACM SIGPLAN Notices. ACM, pp. 623-640.

Costa, P., Paiva, A.C., Nabuco, M., 2014. Pattern Based GUI testing for Mobile Applications, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC). IEEE, pp. 66-74.

Crispin, L., Gregory, J., 2008. Agile Testing: A Practical Guide for Testers and Agile Teams. Pearson Education.

de Cleva Farto, G., Endo, A.T., 2015. Evaluating the Model-Based Testing Approach in the Context of Mobile Applications. Electronic Notes in Theoretical Computer Science 314, 3-21.

Delamaro, M.E., Vincenzi, A.M.R., Maldonado, J.C., 2006. A strategy to perform coverage testing of mobile applications, Proceedings of the 2006 international workshop on Automation of software test. ACM, Shanghai, China, pp. 118-124.

Edmondson, J., Gokhale, A., Sandeep, N., 2011. Automating testing of service-oriented mobile applications with distributed knowledge and reasoning, Service-Oriented Computing and Applications (SOCA), 2011 IEEE International Conference on, pp. 1-4.

Fenton, N., PFIEEGER, S.L., Glass, R.L., 1997. Science and substance: A challenge to software engineers. Applying Software Metrics 46, 6.

Fenton, N., Pfleeger, S.L., Glass, R.L., 1994. Science and substance: a challenge to software engineers. Software, IEEE 11, 86-95.

Fetaji, M., Dika, Z., Fetaji, B., 2008. Usability testing and evaluation of a mobile software solution: A case study, Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on, pp. 501-506.

Flood, D., Harrison, R., Iacob, C., 2012. Lessons learned from evaluating the usability of mobile spreadsheet applications, Proceedings of the 4th international conference on Human-Centered Software Engineering. Springer-Verlag, Toulouse, France, pp. 315-322.

Franke, D., Elsemann, C., Kowalewski, S., 2012a. Reverse Engineering and Testing Service Life Cycles of Mobile Platforms, Database and Expert Systems Applications (DEXA), 2012 23rd International Workshop on, pp. 16-20.

Franke, D., Kowalewski, S., Weise, C., Prakobkosol, N., 2012b. Testing Conformance of Life Cycle Dependent Properties of Mobile Applications, Software Testing, Verification and Validation (ICST), 2012 IEEE Fifth International Conference on, pp. 241-250.

Gao, J., Bai, X., Tsai, W.-T., Uehara, T., 2014a. Mobile application testing: a tutorial. Computer, 46-55.

Gao, J., Tsai, W.-T., Paul, R., Bai, X., Uehara, T., 2014b. Mobile Testing-as-a-Service (MTaaS)--Infrastructures, Issues, Solutions and Needs, High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on. IEEE, pp. 158-167.

Griebe, T., Hesenius, M., Gruhn, V., 2015. Towards Automated UI-Tests for Sensor-Based Mobile Applications, in: Fujita, H., Guizzi, G. (Eds.), Intelligent Software Methodologies, Tools and Techniques. Springer International Publishing, pp. 3-17.

Grindal, M., Offutt, J., Andler, S.F., 2005. Combination testing strategies: a survey. Software Testing, Verification and Reliability 15, 167-199.

Guo, C., Xu, J., Yang, H., Zeng, Y., Xing, S., 2014. An automated testing approach for inter-application security in Android, Proceedings of the 9th International Workshop on Automation of Software Test. ACM, pp. 8-14.

Harrison, R., Flood, D., Duce, D., 2013. Usability of mobile applications: literature review and rationale for a new usability model. J Interact Sci 1, 1-16.

Hay, R., Tripp, O., Pistoia, M., 2015. Dynamic detection of inter-application communication vulnerabilities in Android, Proceedings of the 2015 International Symposium on Software Testing and Analysis. ACM, pp. 118-128.

Heejin, K., Byoungju, C., Wong, W.E., 2009. Performance Testing of Mobile Applications at the Unit Test Level, Secure Software Integration and Reliability Improvement, 2009. SSIRI 2009. Third IEEE International Conference on, pp. 171-180.

Hu, G., Yuan, X., Tang, Y., Yang, J., 2014. Efficiently, effectively detecting mobile app bugs with AppDoctor, Proceedings of the Ninth European Conference on Computer Systems. ACM, p. 18.

Hu, Y., Azim, T., Neamtiu, I., 2015. Versatile yet lightweight record-and-replay for Android, Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications. ACM, pp. 349-366.

Huhn, A., Khan, V., Lucero, A., Ketelaar, P., 2012. On the use of virtual environments for the evaluation of location-based applications, Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. ACM, Austin, Texas, USA, pp. 2569-2578.

Jiang, B., Long, X., Gao, X., 2007. MobileTest: A Tool Supporting Automatic Black Box Test for Software on Smart Mobile Devices, Automation of Software Test , 2007. AST '07. Second International Workshop on, pp. 8-8.

Johnson, R., Wang, Z., Stavrou, A., Voas, J., 2013. Exposing software security and availability risks for commercial mobile devices, Reliability and Maintainability Symposium (RAMS), 2013 Proceedings-Annual. IEEE, pp. 1-7.

Joorabchi, M.E., Mesbah, A., Kruchten, P., 2013. Real challenges in mobile app development, Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on. IEEE, pp. 15-24.

Kaasila, J., Ferreira, D., Kostakos, V., Ojala, T., 2012. Testdroid: automated remote UI testing on Android, Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia. ACM, Ulm, Germany, pp. 1-4.

Kitchenham, B., Charters, S., 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering. School of Computer Science and Mathematics, Keele University.

Knorr, K., Aspinall, D., 2015. Security testing for Android mHealth apps, Software Testing, Verification and Validation Workshops (ICSTW), 2015 IEEE Eighth International Conference on. IEEE, pp. 1-8.

Kronbauer, A.H., Santos, C.A.S., Vieira, V., 2012. Smartphone applications usability evaluation: a hybrid model and its implementation, Proceedings of the 4th international conference on Human-Centered Software Engineering. Springer-Verlag, Toulouse, France, pp. 146-163.

Kuhn, R., Lei, Y., Kacker, R., 2008. Practical combinatorial testing: Beyond pairwise. IT Professional 10, 19-23.

Kumar, A., Lee, S., Lee, W.J., 2015. Modeling and test case generation of inter-component communication in android, Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on. IEEE, pp. 113-116.
Lettner, F., Holzmann, C., 2012. Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications, Proceedings of the 10th International Conference on Advances in Mobile Computing and Multimedia. ACM, Bali, Indonesia, pp. 118-127.
Liang, C.-J.M., Lane, N.D., Brouwers, N., Zhang, L., Karlsson, B.F., Liu, H., Liu, Y., Tang, J., Shan, X., Chandra, R., 2014. Caiipa: Automated large-scale mobile app testing through contextual fuzzing, Proceedings of the 20th annual international conference on Mobile computing and networking. ACM, pp. 519-530.
Looije, R., Brake, G.M.t., Neerincx, M.A., 2007. Usability engineering for mobile maps, Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology. ACM, Singapore, pp. 532-539.
Losada, B., Urretavizcaya, M., Lopez, J., Castro, I., 2012. Combining InterMod agile methodology with usability engineering in a mobile application development, Proceedings of the 13th International Conference on Interacci&oacute;n Persona-Ordenador. ACM, Elche, Spain, pp. 1-8.
Lu, L., Hong, Y., Huang, Y., Su, K., Yan, Y., 2012. Activity page based functional test automation for android application, Third World Congress on Software Engineering. IEEE, pp. 37-40.
Lu, Y., Zulie, P., Jingju, L., Yi, S., 2013. Android Malware Detection Technology Based on Improved Bayesian Classification, Instrumentation, Measurement, Computer, Communication and Control (IMCCC), 2013 Third International Conference on. IEEE, pp. 1338-1341.
Masood, M., Thigambaram, M., 2015. The Usability of Mobile Applications for Pre-schoolers. Procedia-Social and Behavioral Sciences 197, 1818-1826.
Méndez-Porras, A., Quesada-López, C., Jenkins, M., 2015. Automated testing of mobile applications: A systematic map and review, CIBSE 2015 - XVIII Ibero-American Conference on Software Engineering, pp. 195-208.
Merwe, H.v.d., Merwe, B.v.d., Visser, W., 2012. Verifying android applications using Java PathFinder. SIGSOFT Softw. Eng. Notes 37, 1-5.
Morgado, I.C., Paiva, A.C., Faria, J.P., 2014. Automated Pattern-Based Testing of Mobile Applications, 2014 9th International Conference on the Quality of Information and Communications Technology (QUATIC). IEEE, pp. 294-299.
Muccini, H., Di Francesco, A., Esposito, P., 2012. Software testing of mobile applications: Challenges and future research directions, Automation of Software Test (AST), 2012 7th International Workshop on, pp. 29-35.
Mujtaba, S., Petersen, K., Feldt, R., Mattsson, M., 2008. Software product line variability: A systematic mapping study. School of Engineering, Blekinge Inst. of Technology.
Nagowah, L., Sowamber, G., 2012. A novel approach of automation testing on mobile devices, Computer & Information Science (ICCIS), 2012 International Conference on, pp. 924-930.
Oyomno, W., Jappinen, P., Kerttula, E., Heikkinen, K., 2013. Usability study of ME2.0. Personal Ubiquitous Comput. 17, 305-319.
Payet, É., Spoto, F., 2012. Static analysis of Android programs. Information and Software Technology 54, 1192-1201.
Perry, D.E., Porter, A.A., Votta, L.G., 2000. Empirical studies of software engineering: a roadmap, Proceedings of the conference on The future of Software engineering. ACM, pp. 345-355.
Petersen, K., Feldt, R., Mujtaba, S., Mattsson, M., 2008. Systematic mapping studies in software engineering, 12th International Conference on Evaluation and Assessment in Software Engineering, p. 1.
Pham, T.P., Razikin, K., Goh, D.H.-L., Kim, T.N.Q., Quach, H.N.H., Theng, Y.-L., Chua, A.Y.K., Lim, E.-P., 2010. Investigating the usability of a mobile location-based annotation system, Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia. ACM, Paris, France, pp. 313-320.
Puhakka, T., Palola, M., 2006. Towards automating testing of communicational B3G applications, Proceedings of the 3rd international conference on Mobile technology, applications &#38; systems. ACM, Bangkok, Thailand, p. 27.
Ravindranath, L., Padhye, J., Agarwal, S., Mahajan, R., Obermiller, I., Shayandeh, S., 2012. AppInsight: mobile app performance monitoring in the wild, Proceedings of the 10th USENIX conference on Operating Systems Design and Implementation. USENIX Association, Hollywood, CA, USA, pp. 107-120.
Ryan, C., Rossi, P., 2005. Software, performance and resource utilisation metrics for context-aware mobile applications, Software Metrics, 2005. 11th IEEE International Symposium, pp. 10 pp.-12.
Sa, M., Carrico, L., 2008. Lessons from early stages design of mobile applications, Proceedings of the 10th international conference on Human computer interaction with mobile devices and services. ACM, Amsterdam, The Netherlands, pp. 127-136.
Salva, S., Zafimiharisoa, S.R., 2013. Data vulnerability detection by security testing for Android applications, Information Security for South Africa, 2013. IEEE, pp. 1-8.
Sama, M., Elbaum, S., Raimondi, F., Rosenblum, D.S., Zhimin, W., 2010. Context-Aware Adaptive Applications: Fault Patterns and Their Automated Identification. Software Engineering, IEEE Transactions on 36, 644-661.
Shahrokni, A., Feldt, R., 2013. A systematic review of software robustness. Information and Software Technology 55, 1-17.
She, S., Sivapalan, S., Warren, I., 2009. Hermes: A Tool for Testing Mobile Device Applications, Software Engineering Conference, 2009. ASWEC '09. Australian, pp. 121-130.
Song, H., Ryoo, S., Kim, J.H., 2011. An integrated test automation framework for testing on heterogeneous mobile platforms, Software and Network Engineering (SSNE), 2011 First ACIS International Symposium. IEEE, pp. 141-145.
Starov, O., 2013. Cloud platform for research crowdsourcing in mobile testing. East Carolina University, Ann Arbor, p. 12.
Starov, O., Vilkomir, S., Gorbenko, A., Kharchenko, V., 2015. Testing-as-a-Service for Mobile Applications: State-of-the-Art Survey, Dependability Problems of Complex Information Systems. Springer, pp. 55-71.
Tao, C., Gao, J., 2014. Modeling mobile application test platform and environment: testing criteria and complexity analysis, Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing. ACM, pp. 28-33.
Vieira, V., Holl, K., Hassel, M., 2015. A context simulator as testing support for mobile apps, Proceedings of the 30th Annual ACM Symposium on Applied Computing. ACM, pp. 535-541.

Vilkomir, S., Amstutz, B., 2014. Using Combinatorial Approaches for Testing Mobile Applications, Software Testing, Verification and Validation Workshops (ICSTW), 2014 IEEE Seventh International Conference on. IEEE, pp. 78-83.

Vilkomir, S., Marszalkowski, K., Perry, C., Mahendrakar, S., 2015. Effectiveness of multi-device testing mobile applications, Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on. IEEE, pp. 44-47.

Villanes, I.K., Bezerra Costa, E.A., Dias-Neto, A.C., 2015. Automated Mobile Testing as a Service (AM-TaaS), Services (SERVICES), 2015 IEEE World Congress on. IEEE, pp. 79-86.

Wang, Z., 2008. Validating context-aware applications. The University of Nebraska - Lincoln, Ann Arbor, p. 186.

Wei, Q., Chang, Z., Cheng, Q., 2015. Usability study of the mobile library App: an example from Chongqing University. Library Hi Tech 33, 340-355.

Wen, H.-L., Lin, C.-H., Hsieh, T.-H., Yang, C.-Z., 2015. PATS: A Parallel GUI Testing Framework for Android Applications, Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual. IEEE, pp. 210-215.

Wieringa, R., Maiden, N., Mead, N., Rolland, C., 2006. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. Requirements Engineering 11, 102-107.

Yu, L., Tsai, W.T., Jiang, Y., Gao, J., 2014. Generating test cases for context-aware applications using bigraphs, Software Security and Reliability (SERE), 2014 Eighth International Conference on. IEEE, pp. 137-146.

Zhang, P., Elbaum, S., 2014. Amplifying Tests to Validate Exception Handling Code: An Extended Study in the Mobile Application Domain. ACM Transactions on Software Engineering and Methodology (TOSEM) 23, 32.

Zhang, T., Gao, J., Cheng, J., Uehara, T., 2015. Compatibility Testing Service for Mobile Applications, 2015 IEEE Symposium on Service-Oriented System Engineering.

Zhauniarovich, Y., Philippov, A., Gadyatskaya, O., Crispo, B., Massacci, F., 2015. Towards Black Box Testing of Android Apps, Availability, Reliability and Security (ARES), 2015 10th International Conference on. IEEE, pp. 501-510.

Zhi-fang, L., Bin, L., Xiao-peng, G., 2009. SOA based mobile application software test framework, Reliability, Maintainability and Safety, 2009. ICRMS 2009. 8th International Conference on, pp. 765-769.

Zhifang, L., Bin, L., Xiaopeng, G., 2010a. Test automation on mobile device, Proceedings of the 5th Workshop on Automation of Software Test. ACM, Cape Town, South Africa, pp. 1-7.

Zhifang, L., Xiaopeng, G., Xiang, L., 2010b. Adaptive random testing of mobile application, Computer Engineering and Technology (ICCET), 2010 2nd International Conference on, pp. V2-297-V292-301.

Zhimin, W., Elbaum, S., Rosenblum, D.S., 2007. Automated Generation of Context-Aware Tests, Software Engineering, 2007. ICSE 2007. 29th International Conference on, pp. 406-415.

Samer Zein received his MSc degree in Software Engineering from Northumbria University, United Kingdom. He is a PhD student at the Computer Science Department at International Islamic University Malaysia (IIUM). Samer has also worked in the software industry for ten years on projects of different sizes and domains. His research interests are mobile application testing techniques and automation as well as software engineering in practice.

Norsaremah Salleh is an Assistant Professor and the former Head of Computer Science Department at International Islamic University Malaysia (IIUM). She received her PhD in Computer Science from the University of Auckland, New Zealand. Her research interests include the areas of empirical software engineering, evidence-based software engineering, computer science/software engineering education, and social network sites research. Prior to joining academic, she has worked in the manufacturing industry nearly five years as analyst programmer.

John Grundy is Professor of Software Engineering and Dean of the School of Software and Electrical Engineering at Swinburne University of Technology. Previously he was Head of Computer Science and Software Engineering at Swinburne, Head of Department for Electrical and Computer Engineering at the University of Auckland, New Zealand, and Director of Software Engineering, University of Auckland . He has published widely in the areas of automated software engineering, model-driven engineering, visual languages, software architecture and software methods and tools. He is Associate Editor-in-Chief for IEEE Transactions on Software Engineering, and Associate Editor for IEEE Software and Automated Software Engineering. He is currently the CORE Australasia President, is on the Australian Research Council College of Experts, and is on the Steering Committee for the IEEE/ACM International Conference on Automated Software Engineering. He is a Fellow of Automated Software Engineering and Fellow of Engineers Australia.