iOS Applications Testing

Ivans Kulesovs

Faculty of Computing
University of Latvia
Riga, Latvia

Enterprise 2.0 Department
C.T.Co Ltd.
Riga, Latvia

ivans.kulesovs@gmail.com

***Abstract*—** *Mobile applications conquer the world, but iOS devices hold the major share of tablets market among the corporate workers. This study aims to identify the aspects (i.e. features and/ or limitations) that influence the testing of the native iOS applications. The aspects related to general mobile applications testing are identified through the systematic literature review of academic sources. iOS applications testing aspects are identified through the review of non-academic (multivocal) literature sources. The identified aspects are merged and discussed in detail using the reviewed sources and based on the author's professional experience in iOS applications testing. The references to credible sources are provided in order to support the professional experience findings. The study eliminates the gap that exists in the academic world in regards to iOS applications testing. The practitioners are also encouraged to fulfill their iOS applications testing strategies with the identified aspects.*

Keywords—iOS application; mobile application; testing; quality; verification and validation; systematic literature review;

I. INTRODUCTION

According to Clearwater Technology Mobile Computing Sector Report[1], the mobile computing industry is expected to be worth almost US $330 billion by 2015. According to the same study, 67 % of corporations allow workers to use the tablets, but 51 % of corporations even buy the tablets for them. iOS devices hold 75 % of the tablets market share among the corporate workers.

With the growth of platform abilities applications become more complex[2] to satisfy the increasing user needs. The increased complexity means that there are many aspects that should be taken into consideration when testing mobile applications. Mobile workers mostly use native business applications on their devices; otherwise there would not be such a dominant position of the single operating system. That is why iOS native applications are the subject of the main interest for this study.

Despite the fact that the topic being hot, there are only some academic studies [1] – [3] performed that systemize the generic aspects that should be taken into consideration when testing the mobile applications without specifying the platform. Other studies - [4] and [5] that include the clear distinction between the platforms, concentrate on some narrow topic. On the other side, there are different iOS testing checklists, mind maps, blogs etc. available in the internet. This motivates the author to perform the systematic literature review of academic literature in the field of mobile testing and perform the literature review of the available non-academic (or multivocal, as per [6]) sources in the field of iOS testing.

It was decided to concentrate both reviews on aspects of manual testing of such quality characteristics as functional suitability, performance efficiency, compatibility, reliability, maintainability, and portability according to ISO/IEC 25010 [7]. That is why the test automation, security, and usability testing are out of scope (except parts that are closely related to or are on the border line with the quality characteristics mentioned above).

The following research question was formulated:

RQ: *Which aspects (i.e. features and/ or limitations) influence the testing of functional suitability, performance efficiency, compatibility, reliability, maintainability, and portability of the iOS native business applications?*

The results of both reviews are merged in order to answer the research question. The goal of the study is to eliminate the gap that currently exists between academic and non-academic sources in the field of iOS applications testing, as well as to provide the sufficient details for practitioners to make their iOS applications testing strategy more complete and solid.

The paper is organized as follows: in Section II the research methodology is described; in Section III the merged results gathered through both reviews are presented; in Section IV the details of the identified iOS testing aspects are discussed, and conclusions based on the findings are presented in Section V.

II. RESEARCH METHODOLOGY

The systematic literature review (SLR) of the academic sources was performed in order to gain the aspects of the mobile applications testing. The multivocal literature review (MLR) was performed in order to gain the exclusive aspects of iOS applications testing. Fig. 1 shows the stages of sources selection for the whole review process applied in this paper.

The procedure described by Kitchenham and Charters [8] was followed in order to conduct the systematic literature review. The qualitative review approach was applied in order to include a rigor into the systematic review of multivocal literature as suggested by Ogawa and Malen [6]. They define the multivocal sources as accessible, but non-academic writings on the topic.

Author thanks C.T.Co Ltd. (http://mobile.ctco.eu/) for allowing and stimulating the conduction of the research based on the real software projects.

1 - http://www.clearwatercf.com/documents/library/Mobile_Report_FINAL.pdf

2 - http://pages.crittercism.com/rs/crittercism/images/crittercism-mobile-benchmarks.pdf

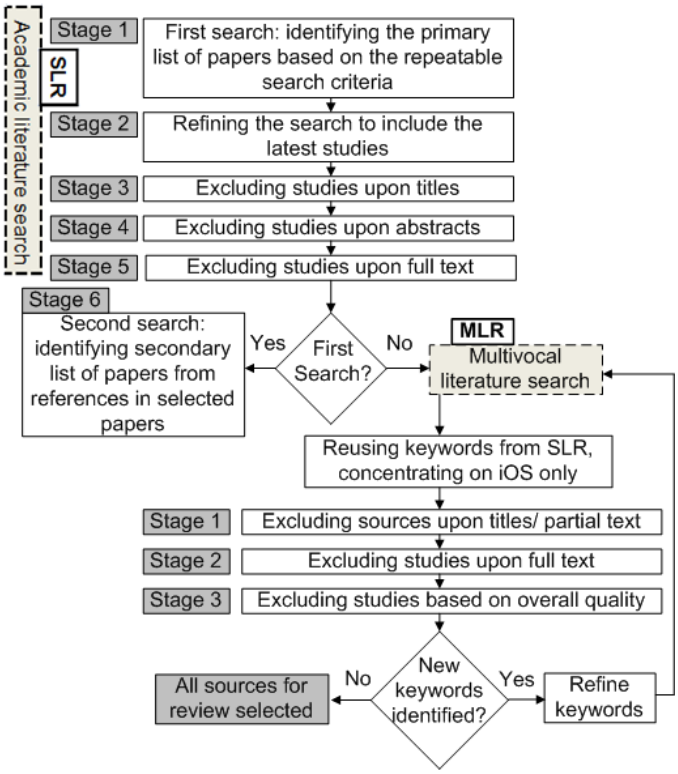

Fig. 1. *Process of sources selection for SLR and MLR*

A. *Systematic Literature Review*

The search of academic literature for SLR was performed in two iterations. The first iteration was executed using the databases and search criteria that are described below. 12 papers were selected as relevant to answer RQ. The second iteration was executed based on the references in the papers selected after the first iteration. Some relevant sources were found, but they appeared to be non peer-reviewed. The details of each iteration can be found in Table 1.

TABLE I. NUMBER OF PAPERS LEFT AFTER EXCLUSION/ INCLUSION DURING EACH SLR STAGE

Iteration	**Stage**	**Number of Academic Works after Stage**
1	1. Initial repeatable search (duplicates removed)	946
	2. Refined search to include works from 2014	772
	3. Exclusion upon titles	33
	4. Exclusion upon abstract	18
	5. Exclusion upon full text	12
2	6. Secondary search based on references in selected results	0
	Total:	12

1) Databases. The following databases were used to search the keywords described in the *Search Keywords* section: IEEE Xplore (IEEE further in the text), ACM Digital Library (ACM), and Springer Links (Springer).

2) Search Keywords. Appropriate keywords were searched in metadata. Due to the search engines differences, metadata should be treated as a search within the title, OR abstract, OR keywords for ACM and as a search within the title only for Springer, while IEEE has an option to search within all metadata at once.

Because preliminary search of keywords “iOS application” and “testing” or “iOS application” and “quality” returned small amount of results, the keyword “iOS” was substituted with “mobile”. It was also given a try to shorten the word “application” to “app”. The following search string was used: (("iOS apps" OR "iOS applications") OR ("iPhone OS apps" OR "iPhone OS applications") OR ("mobile apps" OR "mobile applications")) AND ("quality" OR "testing" OR “verification” OR “validation”).

3) Inclusion/ Exclusion Criteria. Only peer-reviewed papers in English were selected. There was no limitation given on the type of the source (i.e. journals, conference proceedings etc.). Papers starting from the year 2007 were chosen, because it is the year when iOS (iPhone OS at that time) was released. The year 2013 was chosen as the last year of publication for the search results repeatability. The search was also refined by adding papers from the year 2014 in order not to miss the latest available information.

Irrelevant papers were excluded upon title, then upon abstract, and then upon full text. The main credit was given to the papers that offered some categorization or general overview of mobile applications testing. Papers that mention only the specific testing type of mobile applications (i.e. unit testing, security, usability etc.) or that are related to test automation were excluded from the results after additional acquaintance with abstract because they do not focus on the aspects asked in RQ. Only non-shortened version of papers were included if two versions of the same paper for different occasions (e.g. conference proceedings and magazine) were identified.

3) Data Extraction and Synthesis. The data extraction phase involved the extraction of aspects and categories of aspects related to RQ from the selected studies. The categories of multiple non-overlapping aspects are mentioned in some papers, while the detailed description of aspects from single category is mentioned in others. The data synthesis phase includes the merge of aspects from the different papers that appeared to have the same meaning. In order to make the data more usable the aspects were divided between 4 large clusters: *Environment*, *Application Lifecycle*, *Inside the Application*, and (functional or performance aspects of) *UI/ UX*.

B. *Multivocal Literature Review*

1) Data Sources and Search Strategy. Sources for MLR were searched in Google (http://www.google.com/). The combination of the same keywords as for SLR, excluding the “mobile applications” OR “mobile apps” part, was used for the first search iteration. The keyword “checklist” was added for the second iteration. The first 50 relevant articles per iteration (see Appendix B) based on the Google ranking algorithm were taken for subsequent analysis.

TABLE II. NUMBER OF PAPERS LEFT AFTER EXCLUSION DURING EACH MLR STAGE

Iteration	Initial	Stage 1	Stage 2	Stage 3
Iteration 1	50	20	5	5
Iteration 2	50	20	5	4
Total				9

2) Inclusion/ Exclusion Criteria. The sources were excluded during three stages by evaluating 1) Title/ partial text available in the search results; 2) full text; 3) overall quality. The sources related to iOS testing only were included into the final results, i.e. the sources containing only information about general mobile testing aspects were excluded. The sources on security or unit testing, as well as the sources on testing automation were excluded as well. Duplicates were excluded upon title during the first exclusion stage. Some sources were excluded on the second stage because they directly referred to other sources found.

3) Data Extraction and Synthesis. The data extraction phase involved the extraction of aspects and aspects categories asked in RQ. Some sources already contain categorized lists of aspects while other are materials written in narrative. The data synthesis phase includes the merge of aspects from the selected sources. The identified aspects were divided between the same clusters as done for SLR.

III. RESULTS

A. Summary of Reviews

Despite the fact that the search criteria for SLR includes studies starting from 2007, the first selected study was published in 2009 [S2], but the most productive years are 2012 (5 studies: [S3], [S4], [S9], [S11], and [S12]) and 2013 (3 studies: [S1], [S8], and [S10]). 2 studies [S5] and [S6] were published in 2011, and 1 study [S7] was published in 2014. 2 studies [S3] and [S5] are related to narrow topic of mobile application lifecycle, 1 study [S10] is related to user complaints about iOS applications, and other 9 sources [S1], [S2], [S4], [S6] – [S9], [S11], and [S12] are related to general testing of mobile applications.

Between the sources selected through MLR, 7 sources [M23], [M43], [M45], [M49], [M78], [M95], and [M97] were published in 2013, and 1 source was published in 2012 [M56] and in 2014 [M5]. 5 sources [M5], [M43], [M45], [M49], and [M56] are blog posts, 2 sources [M23] and [M97] are testing checklists, 1 source [M78] is a white paper, and 1 source [M95] is a mind map. All the blog posts describe the testing only of one or some aspects, while other sources try to cover the whole iOS testing field.

B. Aspects of iOS Applications Testing

The aspects that influence the testing of iOS applications gathered through SLR and MLR are shown in Table 3. If a source is referred in the table before the details of an aspect, it means that aspect is just mentioned in the source without pointing the details that are related to iOS applications testing.

TABLE III. ASPECTS OF IOS APPLICATIONS TESTING

Environment		
Hardware		
Devices	iPad, iPhone, iPod [M23, M78, M95, M97].	Screen size, resolution & pixel ratio, processing efficiency, memory, storage capacity; [S2, S6, S7, S8, S9, S10, S11, S12, M78, M95, M97] motion activities [M95, M97].
Simulator	[S2, S6, S8, S9, S12, M5]	
External Accessories	Headphones [S1, S4, M97], keyboard [S1, S4]; wired/ unwired [S1, S4].	
Operating System		
OS Variety	[S7, S8, S9, S12, M78, M95, M97]	OS upgrade [S8]
Restrictions and Privacy Settings	[S2, S7, S8, S10] Safari, Camera, Siri, IAP (in-app purchase), Location Services, Contacts, Calendars, Photos, Social Networking, Microphone, Motion Activities, Cellular Data Use, Background App Refresh [M23, M49, M95].	
Resources		
Limitations	Lack of storage, amount of memory, running out of battery, processing capabilities. [S2, S3, S4, S8, S10, S11, S12]	
Consumption	Memory consumption, battery consumption. [S2, S4, S11, S10, S12, M23]	
Connectivity		
Network Types	Wifi, Cellular networks; [S2, S4, S7, S12, M45, M97] Bluetooth [S2, S4, S12, M97]; Airplane mode. [M95, M97]	
Network Conditions	[S1, S2, S4, S10, S11, S12] Strong/ no/ poor connection; connection loss [M23, M45, M49, M95]. Ask for connection [S2].	
Internalization		
Region Formats	[S11] Date format, hour format [M95, M97]	
Date/ Time Settings	Switching between time zones, system time too fast/ too slow. [M95]	
Application Lifecycle		
Installing and Launching	[S11, M23, M97]	
Background	[S3, S4, S5, M23, M95]	
Crash	[S8, S10, M78, M95]	
Low-Memory Warnings	[S3, S4, S5, S6, M95]	
Interruptions	[S2, S3, S4, S5, S11] Call/ SMS [S1, M23]; push notifications [M23, M95], system alerts [S1]; GPS signal [S1]; audio/ video [M23, M95, M97].	
Application Update	[S8, M95, M97]	
Inside the Application		
Keyboard	[S2] Extended keyboard [M49].	
Data Import/ Export	Email; Bluetooth/ network (peer to peer) [M23, M95].	
Logging/ Analytics	[M95, M97]	
In-App Purchases	[M95]	
Web View	[S7, M95]	
UI/ UX		
Gestures	[S7, M95]	
Smooth Animation	[S10, M95]	
Pull to Refresh	[M95]	
Orientation	Portrait, landscape. [S2, S4, M95, M97]	
Half Pixels	[M95]	
Localization	[S8, M97] Native characters and special symbols [M23].	
Accessibility	VoiceOver, accessibility zoom etc. [M43, M56, M78, M95, M97]	

There are three types of iOS devices: iPad, iPhone, and iPod mentioned in [M23], [M78], [M95], and [M97] that have different screen size, resolution & pixel ratio, processing efficiency, memory, and storage capacity, as per [S2], [S6], [S7], [S8] – [S12], [M78], [M95], and [M97]. It is claimed in [S2] that functionalities, usability issues in the interface design, and user behavior "to be tested in emulator", while other sources [S6], [S8] – [S10], and [M5] state that almost everything should be tested on the real device to get the reliable test results. There are also different types of the external accessories, both wired and wireless [S1, S4], like headphones [S1, S4, M97] and keyboard [S1, S4] that can be connected to the device.

It is claimed in many sources [S7] – [S9], [S12], [M78], [M95], and [M97] that the variety of operating systems (OS) is an important testing aspect, while OS upgrade is mentioned explicitly only in [S8]. It is possible to set the restrictions on the usage of different hardware or OEM software completely or for the specific application within the iOS [M23, M49, M95].

Mobile devices have limited power, processing, and memory resource [S2 – S4, S8, S10 – S12]. Thus resources consumption efficiency plays an important role in application success [S2, S4, S10 – S12, M23]. Applications should also be checked on different networks, i.e. strong WiFi connection, cellular network (LTE, 3G, EDGE), and in Airplane mode [S2, S4, S7, S12, M45, M49, M95, M97]. Different network conditions (e.g. slow connection, packets loss etc.) should be taken into consideration as well [M95]. Different regional settings, like data and time formats [M95, M97], as well as time zones [M97] are also the subject of interest.

iOS application lifecycle consists of several phases, and there are specific conditions that can uniquely influence application's behavior while being in the definite phase. An application can be just installed and launched for the first time [S11, M23, M97], work in foreground, stay in background [S3 –S5, M23, M95], receive memory warnings [S3 – S6, M95], be interrupted by a call or SMS [S1, M23], system alert [S1], push notification [M23, M95], GPS signal [S1], or audio/ video from another application [M23, M95, M97]. It can even crash [S8, S10, M78, M95]. Or it can also be updated to the next version [S8, M95, M97].

[M49] warns about the need to check an extended (Chinese) on-screen keyboard, while [S2] mentions on-screen keyboard as a generic aspect that should be taken into consideration. According to [M23] and [M95] data can be shared via email or Bluetooth, or another network between the applications. According to [M95] and [M97] it is necessary to check application's logging and analytics features. Testing of In-App Purchase component is mentioned in [M95]. Testing of Web View component is mentioned both in [S7] and [M95].

An application can be manipulated with a variety of gestures [S7, M95]. When animated transitions occur, they must run smoothly [S10, M95] irrespectively of the task executed in parallel. Testing for half pixels glitches and testing of Pull to Refresh feature are mentioned in [M95]. The necessity of checking the application both in portrait and landscape is noticed in [S2], [S4], [M95], and [M97]. The importance of localization testing is mentioned in [S8] and [M97]. [M23] identifies the need for testing of native characters and special symbols. It should also be checked that application works as designed when accessibility features of OS are enabled [M43, M56, M78, M95, M97].

IV. Discussion and Implications

Despite the fact that the Results section shows the identified aspects of iOS applications testing gathered through SLR and MLR, the author feels the necessity to discuss the details of identified aspects. There are also some aspects that are known to the author (like iAd), but they are missing in the reviewed literature. Some of the details are provided in the reviewed sources. Others are added based on the author's more than three years of professional experience of leading more than 20 iOS applications testing projects for several Global Fortune 500[3] and other multinational companies, giving the references to iOS Developer Library[4] or other credible sources where possible.

A. Hardware

1) Devices. While there are three types of iOS devices, business applications are mostly developed for iPads[5], and sometimes have reduced iPhone versions[6]. iPods generally are out of scope.

iPad 1st generation devices, as well as iPhone 3G and iPhone 3GS are not taken into consideration anymore when new applications for iOS are developed. Only iPhone 3GS has limited support by iOS 6 (the latest iOS version at the moment of writing is iOS7), but both other mentioned devices already are not[7].

iPad 2[8] and iPad mini[9] both have non retina display (i.e. a display with lower pixel density than the latest iOS devices) and generally the same hardware options. They are the least powerful iPad devices that support the latest iOS version. Special checks that application design fits the small screen of the device and that every UI control can be easily interacted with should be performed on iPad mini.

iPad 4[10] and iPad 3[11] both have retina displays, but iPad 4 is more powerful than iPad 3. Generally, it is enough to have only one device of any generation to cover this category of devices.

iPad Air[12] and iPad mini retina[13] both have new GPU (but still retina display) and new M7 64-bit core processor that has built-in hardware for motion activities like *accelerometer*, *gyroscope*, and *compass.*

iPhone 4[14] and higher all have retina displays. iPhone 4S[15] has a faster dual core processor in comparison with iPhone 4. iPhone 5[16] and iPhone 5C[17] are both packed with even faster next generation processor. iPhone 5S[18] is packed with already mentioned M7 64-bit core processor.

Despite the fact that iPhone 5th generation devices have larger screen size in comparison with iPhone 4th generation devices, applications designed for iPhone 4th generation

3 - http://money.cnn.com/magazines/fortune/global500/index.html
4 - https://developer.apple.com/library/
5 - http://www.apple.com/ipad/business/
6 - http://www.apple.com/iphone/business/
7 - http://support.apple.com/kb/ht5457
8 - http://support.apple.com/kb/sp622
9 - http://support.apple.com/kb/SP661
10 - http://support.apple.com/kb/sp662
11 - http://support.apple.com/kb/sp647
12 - http://support.apple.com/kb/SP692
13 - http://support.apple.com/kb/SP693
14 - http://support.apple.com/kb/sp587
15 - http://support.apple.com/kb/sp643
16 - http://support.apple.com/kb/sp655
17 - http://support.apple.com/kb/SP684
18 - http://support.apple.com/kb/SP685

devices can still run on iPhone 5th generation devices, but there are black bars above and below application content, unless properly named and to a larger screen accordingly sized launch image is provided.[19]

Generally speaking, one device from each generation would be enough to cover the whole set of iPhones, in case of application under test does not rely on the specific function of the device like motion activity or finger print of iPhone 5S or Siri (advanced voice control) that is available only starting from iPhone 4S.

2) Device vs. Simulator. The author's professional experience supports the statement expressed in [S6], [S8] – [S10], and [M5] that for achieving good quality of the application, it should be tested on the device rather than on the simulator, because testing results can vary. It also should be taken into consideration that application can behave differently when it is built in debug, not in release mode.[20]

3) External Accessories. There are different kinds of accessories, both wired and wireless [S1, S4], that can be attached to the device: headphones [S1, S4, M97], keyboard [S1, S4], stylus etc. It can occur that an application handles the inputs and outputs from/ to external accessories in a different way than it does without them, or it does not handle them at all.[21] External accessories from different manufacturers can behave differently, e.g. styluses from different manufacturers can have different configurations inside the application in order to handle the palm (interaction) rejection etc.[22]

B. Operating System

1) iOS Variety. Release of the new iOS version almost always leads to the major retesting cycle for the non-trivial applications. New Xcode[23] version (that includes new version of iOS SDK and compiler)[24] is shipped together with the new iOS version. Thus, there can be completely different test results when the same code is built by the different Xcode versions.

The following update strategy is followed by the development organizations which the author works or worked for when the new iOS version is released:

1. Current application version built by previous the Xcode is checked on the new iOS version (preliminary checks are done already on Beta or GM versions).
2. Major failures, if any, are fixed, and the application is released with remark that it supports the latest iOS version.
3. More thorough testing cycle follows when the application current version is built by the new Xcode afterwards.

It is possible to leave the application version built by the previous Xcode for some period of time if, for example, active development currently is not planned. But here is a list of situations when developers are forced to rebuild the application with the new version of Xcode:

- New iOS version does not support the methods that were previously deprecated, but still used in the application; new supported methods that substitute the deprecated ones are available with the new Xcode, e.g. detection of UDID.[25]
- Apple announces that all the new applications or application updates submitted to the App Store must be optimized for new iOS and built with the latest Xcode.[26]
- Application should be redesigned for the marketing purposes, because of the iOS redesign (as it occurred with iOS 7[27]), but new UI is achieved using the latest Xcode.

It is worth mentioning that devices with the previous iOS version should always be available and handled carefully in case some of the applications developed within the organization still support it. It should not be forgotten that there is no official way to install any previous major iOS version after the release of the latest major iOS version[28]. It should be taken into consideration that not all users update iOS version as soon as it is released[29], but can continue to use the previous one for quite a long period of time. From the author's experience, it is especially applicable for enterprise users – they update iOS version only after the enterprise infrastructure that supports the latest iOS version is ready.

2) Restrictions and Privacy Settings. In iOS a user can set different restrictions, both system and application wise, on the usage of different hardware or OEM software. For example, it is possible to restrict the usage of Safari, Camera, Siri, IAP (In-App Purchase), Location Services, Contacts, Calendars, Photos, Social Networking, Microphone, Motion Activities, Cellular Data Use, Background App Refresh etc. [M23, M49, M95] The application should handle cases when it tries to access the restricted item. The user should also be warned about the restriction and instructed how to remove it[30] or offered to remove the restriction within the application if it is possible.

C. Resources.

1) Limitations and Consumption. Due to the fact that a mobile device has more limited storage, memory, power, and processing capabilities than an ordinary PC [S2 – S4, S8, S10 – S12], examination of how the applications handle these limits and operate within these limits are of special interest. The application should check for the free storage availability when the new data is added/ downloaded. Otherwise, from the author's experience, the user will not be able to operate with

19 - https://developer.apple.com/library/ios/documentation/iPhone/Conceptual/iPhoneOSProgrammingGuide/AdvancedAppTricks/AdvancedAppTricks.html
20 - https://developer.apple.com/library/ios/documentation/ToolsLanguages/Conceptual/Xcode_Overview/DebugYourApp/DebugYourApp.html
21 - https://developer.apple.com/library/ios/featuredarticles/ExternalAccessoryPT/Articles/MonitoringEvents.html
22 - http://www.apartmenttherapy.com/tablet-stylus-test-lab-comparison-of-pencil-intuos-pogo-connect-jot-script-tech-test-lab-reviews-196850
23 - https://developer.apple.com/xcode/
24 - https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/WhatsNewXcode/00-Introduction/Introduction.html
25 - https://developer.apple.com/library/ios/documentation/uikit/reference/UIDevice_Class/DeprecationAppendix/AppendixADeprecatedAPI.html
26 - https://developer.apple.com/news/?id=12172013a
27 - http://www.apple.com/pr/library/2013/09/10iOS-7-With-Completely-Redesigned-User-Interface-Great-New-Features-Available-September-18.html
28 - http://www.itproportal.com/2013/09/29/why-apple-wont-allow-you-to-downgrade-your-iphone-from-ios-7-to-ios-6/
29 - http://appleinsider.com/articles/13/12/31/ios-7-now-installed-on-78-of-active-apple-handheld-devices
30 - http://support.apple.com/kb/ht4213

the data that already is inside the application due to crashes. The application should be checked for efficient battery consumption as well [S2, S4, S10 – S12, M23]. It can be verified using Xcode Instruments tools[31]. Battery usage logging can also be enabled on the device that is provisioned for the development[32]. Instruments tools can also be used for profiling the efficiency of memory and processor resource utilization.

D. Connectivity

1) Network Types. During Alpha testing the application is mostly checked in the laboratory environment [S2]: on the strong WiFi connection and in the Airplane mode. The working on the cellular data (LTE, 3G, EDGE) should be checked as well [S2, S4, S7, S12, M45, M97], especially if the application utilizes a lot of traffic. The user, at least, should be warned when large data synchronization occurs on the cellular network.

2) Network Conditions. There are different network conditions possible [S1, S2, S4, S10 – S12, M23, M45, M49, M95] (e.g. slow connection, packets loss, etc.). It should be checked if under these conditions:

- The application handles different network conditions on the first launch.[M95]
- Proper error messages are shown on timeouts and other network errors. [M95]
- The interaction with UI (i.e. the main thread) is not blocked. [M95]
- The corrupted data is not stored, or at least can be redownloaded.

For simulating different network conditions Apple Network Link Conditioner can be used [M45]. This tool is a part of Xcode Developer Tools[33] and can simulate network conditions on the device if the network connection from Mac is shared. It can also be enabled directly on the device that is provisioned for development.

Sometimes it is also necessary to check a poor connection or a connection loss/ switching in the real world. From the author's experience, the most common cases that should not be simulated, but should be checked in the field are:

- The traffic loss while the device "thinks" that it is still connected to the network (e.g. entering the elevator or walking outside the network coverage).
- Switching from WiFi to the cellular network and vice versa, switching from one WiFi access point to another, switching between different cellular network types.
- Only cellular network conditions (e.g. inbound/ outbound connection speed, packet loss ratio etc.) can be simulated, but the device will still think that it is on WiFi. Thus, the real cellular network should be used to check the cellular network specific functionality of the application.
- The situation when the device is not connected to any network should be checked separately to make sure that this condition is treated the same way as the Airplane mode.

E. Internalization

1) Region Formats. Applications should be tested using different region formats [S11] that have different hour format (24 or 12) [M95, M97] and different coma separators. For example, German Switzerland and United States regions cover these both differences. From the author's experience, the specific Arabic and Israel region formats should be explicitly tested if the application's functionality is directly related to the calendar and weekend days.

2) Date/ Time Settings. When the application receives updates from backend, and especially when creation/ update timestamps for items are visible (but the same also applies for locally created items), it is necessary to check how the application behaves with different time settings [M95]:

- When switching between time zones.
- When the system time is too fast or too slow.

Besides checking that functionality works properly itself, it is necessary to check that relative times are properly calculated [M95].

F. Application Lifecycle

1) Installing and Launching. The application should be installed both on the device that already contained some version of the application and on the clean device after the factory reset. The user should be warned through a message or a progress bar in case the access to the application functionality is given in more than 5 seconds after launching. [M23]

2) Background. The background mode is one of the major cycles of iOS application lifecycle. If the application cannot be sent to the background in approximately 5 seconds, then iOS kills it. The same is applicable when going back to the foreground.[34] That is why it is necessary to check that the application changes the state in sufficient amount of time even with the large amounts of data inside. The application can also perform the refreshes in the background using the special multitasking feature provided in iOS 7 or if it uses the Location Services, plays audible content in backround, etc.[34] It should be checked that all the data is preserved [M97], but specific data is updated and is not corrupted after the application is returned to the foreground. All the animations should be restarted as well – it does not occur automatically.

3) Locked Device. Apple warns that improper design or implementation of cryptographic operations can introduce performance or battery life problems. Locking the device with passcode can influence the applications that can operate in background. What is more, the device denies the access to the keychain and files.[35] From the author's experience, the incidents including data loss and crashes can occur if the application needs access to the keychain during the background activity, but the situation when the keychain is not

31 - https://developer.apple.com/library/ios/documentation/AnalysisTools/Reference/Instruments_User_Reference/Introduction/Introduction.html
32 - https://developer.apple.com/library/ios/recipes/xcode_help-devices_organizer/articles/provision_device_for_development-generic.html
33 - https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/WhyNetworkingIsHard/WhyNetworkingIsHard.html
34 - https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/ManagingYourApplicationsFlow/ManagingYourApplicationsFlow.html
35 - https://www.apple.com/ipad/business/docs/iOS_Security_Oct12.pdf

available is not handled properly. It usually takes a long time to isolate the cause of such incidents. It is easy to crash the application just by frequent locking with a passcode and unlocking the device if the file data protection strategy is poorly designed.

4) Crash. There is an option to use crash reports [M95] when the tester cannot reproduce the exact steps that led to the crash. Some crash reports if symbolicated (i.e. converted to the proper stacktrace using debug symbols of the build)[36] can give a hint on the exact scenario that led to crash. Others are not useful if the crash occurred not in the application, but in iOS itself.

5) Low-Memory Warning. When iOS needs more memory, it unloads applications that are currently in the background [S4, S5, M95]. Prior to iOS 6, if the application needed even more memory it could unload cashed images (if cashing was performed) and not visible views of the currently running application. In such cases it was possible to see only the placeholders of images or the application could even crash if unloaded data reload was not properly handled during the further navigation activities. Now developers must handle actions to perform when memory warning is received completely on their own.[37, 38] If the application utilizes a lot of memory (usually it means that there are memory leaks in the application) then it can be fully unloaded from the device memory by iOS itself.[38] Low-memory warnings can be simulated by Xcode Instruments (but only for the simulator).[38,39] From the author's experience, they can be easily reproduced on the device when many heavy pages are loaded in Safari or when photos or videos are made while the application under test is running on the background. Working with very large data or quick and frequent refreshes of data in UI collections can cause low-memory warnings when the application under test is running on the foreground.

6) Interruptions. The application should preserve its state and should not freeze if it receives an incoming call or SMS [S1, M23], system alert [S1], or local, or push [M23, M95] notification while being in the foreground, especially when activities occur on the main thread.

It is possible to open the application through the push notification if it is received when the application is in the background or closed. Different navigation start points should be checked in case the application also does some navigation actions inside itself on confirming the push notification. The application icon badge update should also be checked including the case when several updates are received in a row.[40, 41]

For applications that play audio/video it should be checked that other audio/ video streams are paused on in-application audio stream start. It should be checked if audio continues to play or not when the application is in the background (to play or not - it depends on the requirements). [M23, M95, M97] It is worth mentioning that audio/ video inside the Web Views is handled in a different way than audio/ video played natively.[42]

7) Application Update. The migration process of the application from the previous versions should be tested before the new version of the application that will be available to the final user is released. [M95, M97] After the application is updated from the previous version it should be checked that:

- The data is not corrupted.[43]
- The user preferences stay in place. [M95]
- Saved credentials are still there. [M95]
- Previously registered push notifications are still received.[43] [M95]

The updates should be performed using the different possible paths starting from the very first application release [M95]. From the author's experience, in some cases (e.g. due to the requirements change, or the incomplete data model design during the first release) data model changes can be so significant that users are asked to perform the backup of their data and to perform the clean install of the application. Encrypted data migration is also the subject of interest. When there is a backend server and it is updated as well, it is necessary to check the old application versions on the new server version if there is no mechanism that does not allow connecting to the server with the old versions of the application.

G. *Inside the Application*

1) Keyboard. Editable UI elements should be focused through auto scroll after onscreen keyboard appears. In practice, it is often forgotten to check how they behave in case of split, undocked, extended [M49] or external keyboard [S1, S4], only docked and merged onscreen keyboard is taken into account. From the author's experience, non standard keyboard appearances often influence the usability of those editable elements that are placed near the screen bottom border.

2) Data Import/ Export. Many applications support different file formats that they can operate with. There are different ways how supported file formats can be imported into or exported from the application. They are:

- Open In from email, web browser, or other applications;[44]
- via Air Drop;[45]
- via Email;[M23, M95]
- via iTunes;[46]
- via Photos application/ Camera;[47]
- via Bluetooth/ network (peer to peer); [M23, M95]
- import (download and open) from URL.[48]

36- https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/AppDistributionGuide/AnalyzingCrashReports/AnalyzingCrashReports.html
37 - https://developer.apple.com/library/ios/documentation/uikit/reference/UIViewController_Class/Reference/Reference.html
38 - https://developer.apple.com/library/ios/documentation/iphone/conceptual/iphoneosprogrammingguide/PerformanceTuning/PerformanceTuning.html
39 - https://developer.apple.com/library/ios/documentation/IDEs/Conceptual/iOS_Simulator_Guide/InteractingwiththeiOSSimulator/InteractingwiththeiOSSimulator.html
40 - https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Introduction.html
41- https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/IPhoneOSClientImp.html
42 - https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/00_Introduction.html
43 - https://developer.apple.com/library/ios/technotes/tn2285/_index.html
44 - https://developer.apple.com/library/ios/qa/qa1587/_index.html
45 - http://support.apple.com/kb/ht5887
46 - http://www.apple.com/itunes/
47 - https://developer.apple.com/library/ios/documentation/AudioVideo/Conceptual/CameraAndPhotoLib_TopicsForIOS/Introduction/Introduction.html
48 - https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/URLLoadingSystem/Tasks/UsingNSURLDownload.html

It should be checked that the application handles (i.e. is registered to open and can open[49]) supported file formats in non case sensitive manner.[50] Naming of the exported data should be verified as well.

Files can be sent via email from the application. In most cases, iOS native email client is used for this purpose. It should be checked that there are default to, subject, and body set on email creation. The application should also properly handle the case when there is no email account configured. [M95]

From the author's experience, there are not many problems encountered when images are imported from the Photos application using the native view controller. But in case when custom view controller is used, it should be more strictly checked how it is synchronized with Photos application. The robustness of the Camera component usage is also the subject of worries. The Camera component tests should include the device orientation change, rotation lock, background, etc., i.e. the same aspects that should be checked for every mobile application.

3) Logging/ Analytics. Public analytics engines are often used for collecting crash reports, feature usage statistics and other logs for further development activities and testing thoroughness prioritization [M95, M97]. Analytics is mostly used for publically available applications without own backend server. Based on the author's experience, if analytics is used then the main points that should be checked are:

- Analytics gathering should handle situations when the data is not available or has another format than expected. It is better to send the wrong one or no statistics than to break the UX.
- The statistics should not be sent via cellular networks. In most cases only WiFi connection should be used.
- The analytics should not gather the data about the user without his/ her permission. The user should be warned about how and where the data will be used.[51]
- Collecting the data should not break the UX in any other way.

Enterprise applications can have other, more strict and extensive rules for logging depending on the corporate policy. Own logging protocols are used in such cases.

4) In-App Purchase. In-App Purchase (IAP) is a business model that allows the user to buy virtual or digital consumables, non-consumables, and subscriptions within the application that is distributed via Apple App Store. It should be checked that the purchased items are available on all the devices that are registered for the particular user, and that purchases are restored after the application reinstall, clean install, and iOS update or clean install.[52]

IAP products can be tested using special test users on Apple test environments. It is also possible to test auto-renewable subscriptions on these environments, because they have compressed durations for testing purposes.[53]

IAP password cashing system setting is of the special interest. The password can be saved for 15 minutes or asked each time the user makes any IAP.[54] The application should be checked for handling both options. [M95]

5) iAd. iAd is Apple's platform that allows to "generate revenue and promote … apps" by showing an advertisement within the applications.[55] Test advertisements, including the erroneous one can be sent "over local networks or USB using iAd Producer, or over the carrier network using Apple's test servers".[56] There are two types of advertisement available: banner views and full-screen advertisements.[57] Apple suggests checking that the application shows only fully loaded advertisements. The application should pause other activities when the user begins the interaction with a banner and should restart them when the user finishes (or system cancels) the interaction with a banner. Advertisements should appear quickly and response to the device orientation changes.[57]

6) Web View. Web View is a part of WebKit. Web Views are used to represent the web content inside the native mobile applications.[58] The native application is called a hybrid when most of the data inside it is represented using Web Views.[59] Web Views are often used in order to open different file formats[60]or to login to the different content providers. It should be checked that the links inside the Web Views are opened in the way they are designed to (they are opened in the same view by default[61], but they often should be opened in a default browser, for example). From the author's experience, unnecessary scroll bars and bouncing effects should be eliminated if any.

H. UI/ UX

1) Gestures. The application can be manipulated with a variety of gestures like tap, double tap, touch and hold, pinch, pan, swipe, etc. It should be checked that gestures bring the same user experience as suggested in iOS Human Interface Guidelines.[62] The applications made by Apple can be used for reference. Based on the author's practice, if some elements on the definite application screen support non-trivial gestures other than single tap, other screen elements around should be checked for interaction using the same non-trivial gestures. It also should be verified that unexpected interactions with multiple UI elements at once are not allowed, because such actions often lead to crash.

It is worth mentioning that minimal suggested tappable area is 44 x 44 px.[63]

2) Smooth Animation. The animation is used to improve UX when the application responds to the user actions or when it provides the user with a feedback about the occurring on the screen. But they should not be "excessive or gratuitous" otherwise they "can obstruct app flow, decrease performance, and distract users from their task".[64] The animation should be

49 - https://developer.apple.com/library/ios/documentation/filemanagement/conceptual/documentinteraction_topicsforios/Introduction/Introduction.html
50 - https://developer.apple.com/library/ios/qa/qa1697/_index.html
51 - https://developer.apple.com/appstore/resources/approval/guidelines.html
52 - https://developer.apple.com/in-app-purchase/In-App-Purchase-Guidelines.pdf
53 - https://developer.apple.com/library/ios/documentation/LanguagesUtilities/Conceptual/iTunesConnectInAppPurchase_Guide/Chapters/TestingInAppPurchases.html
54 - http://support.apple.com/kb/ht6088
55- https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/iAd_Guide/Introduction/Introduction.html
56 - http://support.apple.com/kb/HT5245
57 - https://developer.apple.com/library/ios/documentation/userexperience/conceptual/iAd_Guide/TestingiAdApplications/TestingiAdApplications.html
58 - https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/DisplayWebContent/DisplayWebContent.html
59 - http://blogs.telerik.com/appbuilder/posts/12-06-14/what-is-a-hybrid-mobile-app-
60 - https://developer.apple.com/library/IOs/qa/qa1630/_index.html
61 - https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/DisplayWebContent/Tasks/SimpleBrowsing.html
62 - https://developer.apple.com/library/ios/documentation/userexperience/conceptual/MobileHIG/InteractivityInput.html
63 - https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/LayoutandAppearance.html
64 - https://developer.apple.com/library/ios/documentation/userexperience/conceptual/mobilehig/Animation.html

smooth irrespectively of the currently running background tasks, thus the author recommends to test different animated transitions for smoothness while heavy background tasks occur.

3) Pull to Refresh. Pull to Refresh [M95] feature is a very common user experience mechanism that is used for performing delta data loads in mobile applications.[65] It should be verified that Pull to Refresh mechanism loads only the new data, not the whole available data set, and that already loaded data is persisted in case of the Pull to Refresh update failed. It also should be checked how it behaves when the current data/ time and/ or data settings (e.g. format and zone) are changed.

4) Orientation. The application should be checked in both orientations if applicable [S2, S4, M95, M97].[66] Based on the author's experience, it can occur that UI elements are wrongly placed on the orientation change, and the application can crash when the user interacts with misplaced elements (it often occurs with popovers[67]). The application can also crash when the device is rotated during the execution of heavy operations. Executing the actions after the rotation with the rotation lock option enabled is also the subject of interest, because there are several ways how the device orientation can be checked and how the device orientation change is detected by the application.[66, 68]

5) Half Pixels. Sometimes there are half-pixels [M95] and other unexpected blurs[69] noticed when using the application. They occur when UI elements are scaled or when their size and origin are calculated, but not rounded to the whole pixels. The same applies to the fonts. These UI glitches are more visible on the non-retina displays and are often inspected in practice using the 3-fingers accessibility zoom[70].

6) Localization. The following should be checked in case the application supports localizations:

- Localized text in images.[M95]
- Localized translated text fits the available area. [M95]
- The same text is localized in exactly the same way when used in different parts of the application.
- Right-to-left text input and alignment [M95] for Arabic and Hebrew languages.
- Native and special characters:
    - persistence in a database or a file;
    - printing and display [M23];
    - writing to log;
    - handling both by the client and the server.

7) Accessibility. There are plenty of accessibility features available in iOS [M43, M56, M78, M95, M97], i.e. VoiceOver, accessibility zoom, bold text, invert colors etc.[70] They all change the way how the system and the applications look and respond to the gestures. Thus it should be checked that enabling the accessibility features of the system does not break the application.

V. Conclusions

The literature review of both academic and multivocal literature was performed. The majority of the sources selected for the review, both academic and multivocal, were published during the last three years period.

The results of SLR are mostly related to general mobile applications testing aspects like limited resource utilization, orientations, localizations etc., while the results of MLR provided the needed details of iOS application testing aspects (like definite restrictions and privacy settings, iOS accessibility features, etc.), as well as identified some new aspects like IAP, date/ time settings etc. The identified aspects were divided between 4 large clusters: *Environment*, *Application Lifecycle*, *Inside the Application*, and (functional or performance aspects of) *UI/ UX*. The details of each aspect were discussed based on the selected sources and the author's professional experience giving the appropriate references to Apple Developers Library[4] and other credible sources. Some aspects that were not identified through literature reviews, but are known to the author (iAd, update of Xcode, AirDrop etc.) were discussed as well.

The author concludes that the study eliminates the gap that existed in the academic world in regards to the identification and detailed description of iOS application testing aspects. These details should also be useful for practitioners who want to make their iOS testing strategy more solid and complete.

Acknowledgment

Author thanks Darja Smite and Vladislavs Simanovics for their valuable reviews.

65 - http://patft.uspto.gov/netacgi/nph-Parser?Sect1=PTO1&Sect2=HITOFF&d=PALL&p=1&u=%2Fnetahtml%2FPTO%2Fsrchnum.htm&r=1&f=G&l=50&s1=8,448,084.PN.&OS=PN/8,448,084&RS=PN/8,448,084

66 - https://developer.apple.com/library/ios/featuredarticles/ViewControllerPGforiPhoneOS/RespondingtoDeviceOrientationChanges/RespondingtoDeviceOrientationChanges.html

67 - https://developer.apple.com/library/ios/documentation/UIKit/Reference/UIPopoverController_class/Reference/Reference.html

68 - https://developer.apple.com/library/ios/qa/qa1688/_index.html

69 - https://developer.apple.com/library/mac/documentation/userexperience/conceptual/applehiguidelines/IconsImages/IconsImages.html

70 - http://support.apple.com/kb/HT5018

APPENDIX A. SELECTED ACADEMIC LITERATURE SOURCES

Source	Id
D. Amalfitano, A.R. Fasolino, P. Tramontana, and N. Amatucci, "Considering context events in event-based testing of mobile applications", in *Proc. IEEE 6th Int. Conf. Softw. Testing, Verification and Validation Workshops*, Luxembourg, ICSTW, 2013, pp.126-133.	S1
V.L.L. Dantas, F.G. Marinho, A.L. da Costa, R.M.C. Andrade, "Testing requirements for mobile applications", in *Proc. 24th Int. Symp. Comput. and Inform. Sci.*, Guzelyurt, ISCIS, 2009, pp. 555-560.	S2
D. Franke, S. Kowalewski, C. Weise, and N. Prakobkosol, "Testing conformance of lifecycle-dependent properties of mobile applications", in *Proc. 5th Int. Conf. Softw. Testing, Verification and Validation*, Montreal, ICST, 2012, pp. 241 – 250.	S3
D. Franke, S. Kowalewski, C. Weise, "A mobile software quality model", in *Proc. 12th Int. Conf. Quality Softw.*, Xi'an, Shaanxi, QSIC, 2012, pp. 154 – 157.	S4
D. Franke, C. Elsemann S. Kowalewski, C. Weise, "Reverse engineering of mobile application lifecycles", in *Proc. 18th Work. Conf. Reverse Eng.*, Limerick, WCRE, 2011, pp. 283 – 292.	S5
D. Franke, C. Weise, "Providing a software quality framework for testing of mobile applications", in *Proc. 4th Int. Conf. Softw. Testing, Verification and Validation*, Berlin, ICST, 2011, pp. 431 - 434.	S6
J. Gao, B. Xiaoying, T. Wei-Tek, T. Uehara, "Mobile application testing: a tutorial", *IEEE Computer*, vol. 47, no. 2, pp. 46-55, Feb. 2014.	S7
K. Haller, "Mobile testing", *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 6, pp. 1-8, Nov. 2013.	S8
H.-K. Kim, "Mobile applications software testing methodology", *Commun. in Comput. and Inform. Sci.*, vol. 342, 2012, pp. 158-166.	S9
H. Khalid, "On identifying user complaints of iOS apps", in *Proc. 35th Int. Conf. Softw. Eng.*, San Francisco, CA, ICSE, 2013, pp. 1474 – 1476.	S10
E. H. Marinho, R.F. Resende, "Quality factors in development best practices for mobile applications", in *Proc. Computational Sci. and Its App.*, Salvador de Bahia, Brazil, ICCSA, 2012, pp. 632-645.	S11
H. Muccini, F. Di Antonio, and P. Esposito, "Software testing of mobile applications: challenges and future research directions", in *Proc. IEEE 7th Int. Workshop Automation of Softw. Test*, Zurich, AST, 2012, pp. 29–35.	S12

APPENDIX B. SELECTED MULTIVOCAL SOURCES

The full list of the reviewed multivocal sources with indicating the exclusion phase can be found here: https://dspace.lu.lv/dspace/bitstream/handle/7/2739/iOS Applications Testing - Multivocal Sources.pdf

I[a]	Source title	Source URL	Id
1	Testing iOS Applications	http://blog.smartbear.com/mobile/testing-ios-applications/	M5
1	Testing Criteria for iOS Apps - App Quality Alliance	http://www.appqualityalliance.org/files/AQuA_testing_criteria_for_iOS_for_v1.0%20final%2022_oct_2013.pdf	
1	Tips For Accessibility Testing Of iOS Apps | Pat's Tapestry	http://patstapestry.wordpress.com/2013/05/24/tips-for-accessibility-testing-of-ios-apps/	M43
1	Testing iOS Apps for Tough Network Conditions | Nearsoft	http://nearsoft.com/blog/testing-ios-apps-for-tough-network-conditions/	M45
1	TestElf Blog — We Find These Common Bugs When Testing iOS Apps	http://blog.testelf.com/post/56341438836/we-find-these-common-bugs-when-testing-ios-apps	M49
2	iOS Accessibility - A Useful Guide For Testing | Rosie Land	http://www.rosiesherry.com/2012/09/02/ios-accessibility-a-useful-guide-for-testing/	M56
2	The Essential Guide to iPhone & iPad App Testing	http://go.utest.com/rs/utest1/images/uTest_Whitepaper_The_Essential_Guide_to_iOS_App_Testing.pdf	M78
2	iOS Testing mind map 1.2 – Now with more stuff | Neglected Potential	http://www.neglectedpotential.com/2013/10/ios-testing-mind-map-1-2/	M95
2	iOS Devices | Dave Addey	http://daveaddey.com/?cat=16 (http://daveaddey.com/postfiles/AgantReleaseChecklist2013.pdf)	M97

[a] *I – iteration.*

REFERENCES

[1] H. Muccini, F. Di Antonio, and P. Esposito, "Software testing of mobile applications: challenges and future research directions", in *Proc. IEEE 7th Int. Workshop Automation of Softw. Test*, Zurich, AST, 2012, pp. 29–35.

[2] V.L.L. Dantas, F.G. Marinho, A.L. da Costa, R.M.C. Andrade, "Testing requirements for mobile applications", in *Proc. 24th Int. Symp. Comput. and Inform. Sci.*, Guzelyurt, ISCIS, 2009, pp. 555-560.

[3] J. Gao, B. Xiaoying, T. Wei-Tek, T. Uehara, "Mobile application testing: a tutorial", *IEEE Computer*, vol. 47, no. 2, pp. 46-55, Feb. 2014.

[4] D. Franke, S. Kowalewski, C. Weise, and N. Prakobkosol, "Testing conformance of lifecycle-dependent properties of mobile applications", in *Proc. 5th Int. Conf. Softw. Testing, Verification and Validation*, Montreal, ICST, 2012, pp. 241 – 250.

[5] D. Franke, C. Elsemann S. Kowalewski, C. Weise, "Reverse engineering of mobile application lifecycles", in *Proc. 18th Work. Conf. Reverse Eng.*, Limerick, WCRE, 2011, pp. 283 – 292.

[6] R.T. Ogawa, B. Malen, "Towards rigor in reviews of multivocal literatures: applying the exploratory case study method", *Review of Educ. Research*, vol. 61, no. 3, pp. 265–286, Fall 1991.

[7] *Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models*, ISO/IEC 25010, 2011.

[8] B. Kitchenham, S. Charters, „Guidelines for performing Systematic Literature Reviews in Software Engineering", EBSE Tech. Rep., 2007, vers. 2.3.