Testing-as-a-Service for Mobile Applications: State-of-the-Art Survey

Oleksii Starov[1], Sergiy Vilkomir[2], Anatoliy Gorbenko[3], Vyacheslav Kharchenko[3]

[1]Computer Science Department, State University of New York at Stony Brook, USA
ostarov@cs.stonybrook.edu
[2]Department of Computer Science, East Carolina University, USA
vilkomirs@ecu.edu
[3]Department of Computer Systems and Networks (503)
National Aerospace University, Kharkiv, Ukraine
A.Gorbenko@csn.khai.edu, V.Kharchenko@khai.edu

Abstract. The paper provides an introduction to the main challenges in mobile applications testing. In the paper we investigate the state-of-the-art mobile testing technologies and overview related research works in the area. We discuss general questions of cloud testing and examine a set of existing cloud services and *testing-as-a-service* resources facilitating testing of mobile applications and covering a large range of the specific mobile testing features.

Keywords. Mobile application, software testing, cloud services

1 Introduction

Mobile development is characterized by a variety of applications with different quality requirements. Online application stores, like the Apple App Store and Google Play, offer thousands of market-oriented apps—mobile games, utilities, navigators, social networks, and clients for web resources. At the same time, the interest in critical mobile applications is growing. For instance, online banking has evolved into mobile banking, mobile social alerts are widely used to report accidents or warn about hurricanes [1], and special apps exist to monitor traffic [2] and help cardiac patients [3]. Augmented reality apps are used for complex navigation and involve a variety of sensors. A new trend is to use smartphones as components for mobile cyber-physical systems because the powerful hardware has a variety of sensors. Mobile applications are even being considered to support processes at such critical facilities as nuclear power plants [4]. These trends require high levels of reliability and quality for mobile software systems. They affect testing, in particular, and the whole mobile development process in general. Too often, the mobile development process ends with

the submission of a social application to an online store. The aim is to gain a wider audience of users in a shorter time, but this does not guarantee the quality of the product and non-critical bugs are usually accepted. Some surveys have confirmed that mobile developers usually deal with small apps and do not adhere to a formal development process [5]. In contrast, a totally different approach is required for critical or business-critical mobile applications, including mobile clients for trustworthy enterprise systems and solutions; for example, Facebook's iOS app is crucial for maintaining the company's profile and reputation and thus was rebuilt to overcome the poor quality of the first version.

To guarantee these mobile applications' reliability and security, sufficient testing is required on a variety of heterogeneous devices as well as on different OS. Android development is the most representative example of how different applications should function amid a plethora of hardware-software combinations [6]. Adequately testing all of these platforms is too expensive—perhaps impossible—especially for small resource-constrained mobile development companies.

Mobile development has a set of distinctive challenges and features. Mobile application testing has some similarities to website testing as both involve validation in many environments (smartphones and browsers, respectively). The general requirements for both types of testing are similar: applications should function correctly, efficiently, and be reliable and secure in all environments. However, mobile testing presents new activities and requires more effort because it includes web applications that work within mobile browsers or hybrid variants wrapped in native code [5]. This testing also involves a large number of possible combinations of mobile devices and OS. Finally, mobile testing involves the use of actual hardware and so testers need additional knowledge and skills such as build installation and crash-log retrieving. Advanced mobile software processes typically work according to the Agile-based methodology [7] and include usage of build distribution services to assist in testing, analytical services for maintenance during production, and services to obtain a wider range of mobile devices for testing. These services create a large set of *testing-as-a-service* (TaaS) resources, or supporting web-applications, that use cloud benefits to facilitate the testing of mobile applications and cover a large range of the specific mobile testing needs. These cloud solutions make mobile testers more effective because they provide complex infrastructure and/or services that are not feasible within small developer companies. The dominant type of such cloud services is a "device cloud," i.e., a service that provides hosting of remote mobile devices and running of tests in the cloud. Existing commercial variants of such platforms became an inspiration for the current study.

The rest of the paper is organized as follows. In the second and third sections we discuss cloud-based testing services and features of mobile applications testing. In Section 4 we systematize existing cloud-based services for mobile application testing including device clouds, application lifecycle management services and discuss techniques for test automation. Additional standalone tools for mobile application testing are described in the Section 5. Finally, the 6th section provides an overview of the modern research studies in mobile testing making emphasis on combinatorial testing techniques in Section 7.

2 Cloud-Based Testing

Many research papers have stated that testing extensively migrates to the cloud nowadays [8–12]. Reviews and classifications of testing cloud services include solutions for web systems and mobile development [13, 14]. Cloud benefits are used not only to support performance, load, or reliability testing of websites, but also to assist with providing required hardware resources (i.e., remote smartphones) for different needs for mobile testing. Cloud-based mobile testing is a young but very topical issue [15].

The database at the Cyber Security and Information Systems Information Analysis Center provides a large list of cloud testing references [16]. Technical and research issues about testing over the cloud are analyzed in [17] and [18] respectively.

In this work we use the term “cloud service” as the most general understanding of cloud computing, i.e., cloud service is a software tool or hardware resource that is delivered over the Internet. The definition means that we also take into account such web resources as build distribution solutions and online issue tracking systems. The term “device cloud” (i.e., mobile device cloud or cloud of devices) will also be used, pointing to both the cloud service’s nature and the many geographically dispersed devices.

Many specialized studies exist regarding the general architecture and construction of cloud and distributed systems [19, 20], including providing service through application programming interfaces (APIs). Technical issues for the tests on the cloud are discussed in [18], including Hadoop usage for test distribution. Device clouds (services that provide hosting of smartphones and run tests on multiple remote real devices) require special algorithms for effective test distribution to make overall test execution time as minimal as possible.

3 Mobile Application Testing

Mobile development has a set of distinctive features and the following specific challenges can be mentioned [5]: support of many hardware and software platforms, correct work with a variety of sensors, interconnections with other applications, high requirements for users’ experiences and the quality of the user interface, and the existence of web mobile and hybrid applications that incorporate all of these challenges to web development.

Mobile applications are popular among startups and approaches for quick prototyping to evaluate the concept of an application are now in high demand. All of these features contribute to the complexity and specifics of mobile testing [6, 21]. As for mobile testing in this work, I mean comprehensive testing of a mobile system that includes the testing of mobile apps as well as mobile operation systems (OS) and the related hardware. Different investigations have pointed to the required mobility of the apps in terms of their ability to function in different environments and configurations as the root challenge of testing [21].

uTest published The Essential Guide to Mobile App Testing [6], a book that comprehensively and coherently describes challenges and techniques in mobile application testing. A lot of research exists about automation and facilitation of the testing process, including leveraging of cloud abilities [10, 22–26]. Companies that provide cloud services for mobile testing (cloud of devices) usually assist their customers with a set of guides [27, 28].

Examples of testing matrixes to cover all smartphone models or OS versions generate an enormous number of combinations [6]. The issue is significant for the Android platform because of its representatively large number of supported devices with different characteristics (e.g., screen resolution, size of memory, and set of sensors). The problem is compounded by the fact that a smartphone simulator or an emulator cannot fully substitute for the hardware [6]. At the same time, the development for different mobile platforms looks similar. Platforms have similar developer websites with necessary documentation, examples, and suggested patterns. The principles of the application life cycle are similar, for instance, comparing Android to the Windows Phone 7 [29].

Many software development companies are interested in the mobile market and many mobile platforms now exist: Android, iOS, Windows Phone, Symbian, etc. New ones appear regularly like the recent Ubuntu Mobile OS [30]. According to Gartner [31], Android devices have most of the market and Forbes says that the Android platform aims to meet enterprise requirements in the near future [32]. Previous research on the bug statistics for the Android OS [33] proved that the Android (with Symbian) has effectively organized an open-sourced bug-tracking system that deals with bugs and makes the platform better. The total number of applications in Google Play (www.appbrain.com/stats/) is now more than 850,000 and is increasing steadily. The open source nature of Android makes it popular among the scientific community, and many examples of research studies targeted at the Android system can be found.

4 Mobile Testing Services

To facilitate mobile testing, various cloud benefits are used and different TaaS, or supporting services, exist. Figure 1 provides references to them, along with mapping to correspondent testing stages. The presented types of testing were partially taken from a diagram on Perfecto Mobile's guide that shows the demanded device allocation during different Application Lifecycle Management (ALM) stages [27]. The diagram was extended by adding conceptualizations as a separate ALM activity, plus concept, security, and user experience (UX) testing, as well as highlighting test activities such as test planning, management, and issue tracking that are all specific to real-life mobile development.

The set of cloud services for mobile testing can be divided into three types: device clouds (mobile cloud platforms), services to support ALM, and tools to provide processing according to some testing techniques. The following sub-sections describe each type separately.

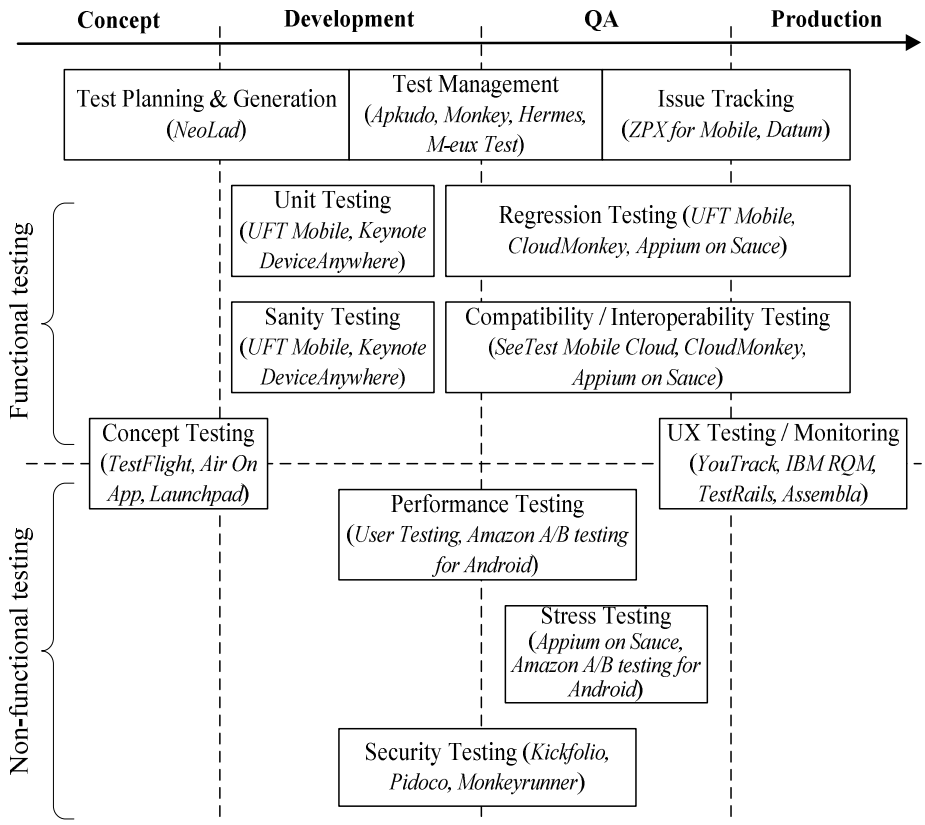

Fig. 1. Test Stages, Activities and Mobile Testing Services.

4.1 Device Clouds

The majority of cloud services for mobile testing serves as a "cloud of devices" and provides remote access to smartphones in the cloud in order to accomplish testing, in other words, provides device hosting. Such services usually aid mobile developers in using remote smartphones as real devices for manual testing (interactive testing through a web interface), recording of scripts, and automatic running of tests on a range of models.

For instance, *Perfecto Mobile* service (www.perfectomobile.com) provides all of this functionality representing different modern hardware and software mobile platforms (Android, iOS, Windows Phone, and Symbian) and can be integrated with HP UFT (QTP) or MS Team Foundation Server. Devices available in the system have different parameters, for example, testing different types of Internet connections is possible. The service works with two kinds of test scripts: QTP and the Perfecto Mobile Application. Perfecto Mobile is only a public service, but UFT Mobile can also be deployed as a private cloud. UFT Mobile provides automated functional

testing and special solutions for realistic mobile performance testing (e.g., LoadRunner and Performance Center).

Keynote DeviceAnywhere (www.keynotedeviceanywhere.com) is a similar service that provides online manual and automated testing of a mobile app on a variety of devices. It can be integrated with existing ALM through HP QTP, IBM RQM or special Java APIs.

The *SOASTA* service (www.soasta.com) provides two advanced solutions: TouchTest test automation for multi-touch, gesture-based applications and CloudTest for scalable mobile application testing (performance or load-testing with millions of geographically distributed emulated users). TouchTest scripts can be recorded and performed against user's own device. Users can control test devices via IP addresses.

The *Cigniti device cloud* (www.cigniti.com) provides remote access to a variety of mobile devices via own proprietary mobile test automation framework, with test accelerators for test automation and performance testing. Cigniti is suitable for network carrier testing.

SeeTest by Experitest (experitest.com) provides device cloud that can be deployed as a private platform within an organization. Test automation facilities include test script recording/performing on real devices or emulators and integration with HP UFT (QTP), TestComplete, C#, RFT, Java, Perl, Python. SeeTest also provides manual testing tools.

The *CloudMonkey* service (www.gorillalogic.com) runs MonkeyTalk scripts across many Android emulators and iOS simulators. Screenshot reports are positioned as the base testing results. CloudMonkey test jobs can be integrated with continuous integration (CI) servers like Jenkins.

The *Appium on Sauce* service (saucelabs.com) covers two functionalities: iOS device hosting and easy CI. The latter means that it can be used as a build server and testers do not need to set up developer environment on local machines. Test automation is implemented with Selenium, and interactive testing is only possible for web mobile applications. Appium can be deployed privately.

The *TestDroid* Cloud (testdroid.com) is a device cloud service oriented towards Android apps testing that uses the TestDroid AppCrawler engine to verify application devices' compatibility. TestDroid Recorder can be used to generate reusable Android JUnit test cases. Test results consist of screenshots and device logs. A tester can compare screenshots to check for GUI bugs. TestDroid can also be integrated with Jenkins or leveraged through REST APIs.

The *Scirocco* Cloud (www.scirocco-cloud.com) has all of the functionality of a device cloud, except of script recording. It supports only the Android platform and provides manual access to remote devices through its HTML5 web interface. Test automation is done by using one of three drivers: AndroidDriver, Monkeyrunner, or NativeDriver. Results are provided as a set of screenshots to compare.

The *LessPainfull* device cloud (www.lesspainful.com) is oriented for Android and iOS apps testing. As a test automation engine, it uses Calabash for Cucumber and accepts Cucumber-based test scripts. LessPainfull provides two options: private cloud tailored for single customer and shared cloud with devices common for several customers.

TestQuest (www.bsquare.com) is a distributed framework for deployment within an organization. It is oriented towards Android application testing and can be integrated with MS Visual Studio.

The *ZPX* service (www.zaptechnologies.com) provides device hosting and mobile test automation in the cloud and is compatible with HP ALM products.

Jamo (www.jamosolutions.com) provides a set of tools to perform remote and scheduled testing on a device. For instance, Wanconnector in combination with Remote Device Screen provides access to a device within different geographical locations. The M-eux Test tool supports web application testing.

Apkudo's Device Analytics (www.apkudo.com) provide some elements of multidirectional testing by testing devices (e.g., new smartphone models) against the top 200 apps from the market. Similar services are available for smartphone hardware testing, but these have no relation to mobile apps like Datum (www.metricowireless.com) that provides verification of calls, data quality, and video quality. Apkudo also offers free public and fully automated stress testing of the Android applications on the big range of models using the Monkey tool.

Table 1 summarizes the device clouds mentioned above and a comparison based on supported mobile platforms, types of testing, and delivery type of cloud solution.

Table 1. List of Device Clouds.

Cloud Service	Supported Platforms			Types of Testing	Delivery Type	
	Android	iOS	Other		Public	Private
Apkudo	+			Stress (automated), New device approval	+	
Appium on Sauce		+		Manual for web applications, Automated	+	+
Cigniti	+	+	+	Automated, Interoperability, Performance, Network	+	
CloudMonkey	+	+		Automated, UI-oriented	+	+
DeviceAnywhere	+	+	+	Manual, Automated, Monitoring, Coverage	+	+
Jamo	+	+	+	Automated		+
Perfecto Mobile	+	+	+	Manual, Automated, Performance, Monitoring	+	
Scirocco Cloud	+			Manual, Automated	+	
SeeTest	+	+	+	Manual, Automated, On a new devices		+
SOASTA	+	+	+	Manual, Automated, Load, Performance, Gesture-based	+	+
TestDroid Cloud	+			Automated, UI-oriented, On a new devices	+	+
UFT Mobile	+	+	+	Automated, Load, Performance, Monitoring		+
Zap-Fix	+	+	+	Automated		+

Manual testing means the remote operation of a device via a web interface, and automated testing incorporates functional and regression testing and different kinds of automation. All device clouds provide compatibility testing as intended. Public cloud means service with shared devices, while a private cloud means an infrastructure allocated to a single user or a system to be deployed on a user-developer's site.

Two known research attempts within universities to create and investigate test-bed cloud solutions for mobile development are *SmartLab* [28] and the *Android Tactical Application Assessment and Knowledge* (ATAACK) Cloud [29]. Both are distributed systems that connect a set of mobile devices under the Android OS for application investigation, development, and testing.

The *SmartLab* is an experimental test-bed being developed at the University of Cyprus. It provides more than 40 connected Android smartphones plus emulated devices, but not many details are described or known.

The *ATAACK* Cloud is new joint project for Virginia Tech, the University of Maryland, and Vanderbilt University, with the support and funding by Air Force Research Laboratories. Its goal is large-scale mobile application testing and investigations.

These research studies consider device clouds with several smartphones connected to one computer (vertical) and several computers with connected smartphones (horizontal) scaling of devices, i.e., fully distributed systems, and how to provide access and testing.

Many studies regarding less-scaled test frameworks for distributed mobile testing [30] that are not cloud services and many tools for vertical-scaled test automation only exist, but their reviews are beyond the scope of this chapter.

All services mentioned in this section appear in Figure 3 with the following logistics: services that support the running of unit tests listed under "unit testing," services that support online manual testing listed under "sanity testing," references to script automation techniques of these services listed under "regression testing," all cloud devices listed under "interoperability/compatibility testing," and references to special integrated non-functional test approaches of these services.

4.2 Application Lifecycle Management Services

The application lifecycle management of mobile applications has own specifications and many cloud services exist that support test-related activities within ALM. Several examples of these cloud services are listed below.

1. Mobile developers, like all software developers, use issue tracking systems, e.g., with Agile-oriented plugins, more complex solutions like *IBM Rational Quality Manager*, or test management systems like *TestRails*. Some of these are integrated with software configuration management and facilitate code reviews or code style checks. A review of similar tools and solutions is not the goal of this work, so Figure 3 shows only several basic examples.

2. Mobile testing involves the use of actual hardware and so testers need additional knowledge and skills, such as build installation or crash-log retrieving. To facilitate beta build distribution activities, many cloud services exist. Some of them provide

functions for test team management (e.g. *HokeyApp*, hockeyapp.net) or build provisioning and deployment to the store (*AirOnApp* for iOS, www.aironapp.com). *TestFlight* service (testflightapp.com) helps to deal with the iOS build management and distributes them via email between separated testers. It provides an easy application installation on a real device, i.e., by a tap on the link in an email opened on a smartphone. A similar service for Android is *Launchpad* (launchpadapp.com). The *HokeyApp* (hockeyapp.net) build distribution provides extended functionality to collect live crash reports, feedback from users, and analysis of resulting test coverage. Usage of these services for build distribution can be integrated along with the continuous integration process of the company (e.g., via job scripts for the Jenkins build server).

3. User experience testing and monitoring of an app in production are required activities within mobile testing. Several analytics services gather usage statistics and these can be incorporated in a mobile application. *Perfecto Mobile* (www.perfectomobile.com) service also provides some solutions for monitoring performance. The following two services incorporate user experience testing in the build distribution facilities. The *UserTesting* service (www.usertesting.com) provides many real users who will examine an app and provide feedback about their experience with the app and thoughts about it. The *Amazon A/B testing for Android* (developer.amazon.com) provides a service that distributes two builds that differ in some features between two unique groups of users. Then it provides measurements and results about which feature is more successful.

4. Mobile development is very popular among startups and usually requires rapid prototyping for concept feasibility evaluation. Thus such services exist like *FluidUI* (www.fluidui.com) to easily create interactive prototypes, or *Kickfolio* (kickfolio.com) to share an app demo, or *Pidoco* (pidoco.com) to create realistic mockups. All of these are needed to test the concept and idea of the app (i.e., if it can hit the market) at a minimal expense.

4.3 Device Cloud-Based Testing Techniques

Device clouds provide different techniques for test automation (recording, distribution, and execution). This includes unit tests and GUI-based testing. Examples of approaches are standard Android SDK tools *Monkeyrunner* and *Monkey* (developer.android.com), special solutions like *SOASTA TouchTest*, and solutions based on object recognition (e.g., *Eggplant* automation based on VNC technology, www.testplant.com).

Test automation has its own weak sides, and according to experts in the field, cannot serve as a total substitution for manual testing. The issue that was noticed during the analysis of cloud test automation was the delivery of the test input data to mobile sensors (GPS, accelerometer, camera, etc.). While solutions to send dummy GPS coordinates exist, situation with a photo camera is more complicated because it requires the simultaneous changing of a picture (preferable physically in front of a camera) while performing a script. A variety of mobile apps use a camera as a part of their key functionality (e.g., shopping apps and QR code readers), and proper testing

requires test cases with snapshots from different distances, angles, lights, etc. Other problematic aspects of automation are the sophisticated (approximate) screenshots comparisons, executions of direct device-to-device communication during the test, and others.

Device clouds provide compatibility, interoperability, and regression testing. Many services provide embedded tools to support performance monitoring and load testing (*Perfecto Mobile*, www.perfectomobile.com, *SOASTA*, www.soasta.com, *Cigniti Mobile Testing*, www.cigniti.com) or even automated stress testing on a variety of devices (e.g. *Apkudo*, www.apkudo.com).

There are special cloud services that aid with mobile performance and load testing. For instance, *SandStrom* (sandstorm.impetus.com) can be used for load testing of web mobile applications and *NeoLoad* (www.neotys.com) focuses on load testing of back-end servers by emulating typical mobile devices working in parallel and sending appropriate content to the server.

There are also standalone solutions for test techniques applications like performance frame counters on Windows Phone Emulator that theoretically can be leveraged in a cloud.

Security testing is mainly presented by static check techniques. *Checkmarks* (www.checkmarx.com) provides scanning of source code and supports Android and iOS applications. *Mobile App Security and Privacy Analysis* by Veracode (www.veracode.com) scans and evaluates binary files for vulnerabilities and can be leveraged through APIs.

Another type of services exists based on experts. For instance, *uTest* experts will assist with mobile security testing by manual penetration and using internal static and dynamic security testing solutions (www.utest.com). At the same time, research papers about novelty mobile security testing approaches exist (that potentially can be leveraged by some cloud services) [31], but they are out of the scope of this review.

Concept testing, UX testing, and monitoring techniques were comprehensively described in section 4.2 as parts of services that support ALM.

Mobile testing services should incorporate test planning and test generation techniques. *Keynote DeviceAnywhere Test Planner* (www.keynotedeviceanywhere.com) provides a coverage calculation for smartphone models to test that can be considered as application of combinatorial testing techniques, but it can be extended by using *pairwise*, *t-way*, or other approaches. *HokeyApp* only provides test coverage monitoring and analytics, i.e., the matrix of the devices and languages that were tested. *Cigniti Test Advisory Services* and *TestRails* provide more high-level test planning and control facilities.

The situation with cloud services for mobile testing is changing extremely rapidly: new ones appear and old ones get new functionalities. Thus, it is hard to guarantee that the provided list of tools and services is exhaustive, but it can serve as a useful baseline.

5 Standalone Tools for Mobile Application Testing

Any mobile platform has a correspondent software development kit (SDK) for app developers. Usually the producers of mobile platforms provide developers with a debugger, emulator or simulator, plugin for popular IDE, etc. The toolsets for Android, iOS, or Windows Phone development are very similar. Each platform also provides similar development support. For instance, web developer portals provide similar guidelines on how to use the available tools. In this section, we describe the most important standard tools (i.e., available from SDK) for Android app testing and several third-party extensions or analogues.

The basic tool for working with Android devices is *Android Debug Bridge* (ADB), which is a command-line utility to control Android devices. Device detection, debugging, execution of shell commands, and access to a device's file system is possible by using ADB. A high-level development environment like Eclipse (with the Android Development Tools plugin installed) implicitly uses ADB to install and debug builds within a connected device.

Android SDK provides two special tools for the GUI-based automated testing of applications. The first is *UI/Application Exerciser Monkey* (developer.android.com) for GUI stress testing, which generates a set of pseudo-random user events and sends them to an Android device. Previously, the *Apkudo* service (www.apkudo.com) was mentioned to provide a cloud of devices for long-term stress testing of an app using *Monkey*. It shows the statuses of the application being tested on each device, i.e., it either crashed after a sequence of random events or it is still running. Crash logs and other supporting information are provided.

A more advanced tool for automated testing provided by SDK is *Monkeyrunner* (developer.android.com), which runs on test scripts written in Python with several special classes available to provide support of touch, press, type, drag events, shell commands, intent invocations, app installations, and removal. Functionality is sufficient for basic GUI-based automation. So the following two strategies of interaction with interface components can be used:

(i) dynamic coordinates calculation (screen sizes can be dynamically retrieved);

(ii) and components enumeration through focus change.

At the same time a tester who writes test scripts should remember to put in appropriate delays (or special workarounds) between long-term events or actions and the results check. Monkeyrunner is suitable for screenshot analysis, as it provides methods to take screenshots during test script checkpoints and compare them. Thread-safeness is not guaranteed, but test scripts can include efficient simultaneous launches on several connected devices (and thus screenshots can be taken from several smartphones at the same time).

An *AndroidViewClient* extension (can be downloaded from Github) exists for Monkeyrunner that enables more high-level test scripts, particularly to address UI components in a test script by name or text. But this library only supports "rooted" devices with ViewServer installed or newer devices with Android's *UIAutomator* (Android API 16 and greater). *UIAutomator* is part of the Android SDK revision 21 and up and comes with the UIAutomatorViewer tool that lists all the UI objects.

Robotium (code.google.com) is another popular engine for the automated testing of Android applications. It is an extension of the Android test framework (JUnit tests for Androip applications) used to write easy and powerful automatic black-box tests. Similarly, the Robolectric is based on JUnit 4 and runs Android tests directly on the JVM. Both of these tools point to another direction, i.e., the application of unit tests for mobile testing and even GUI-testing.

Other test automation solutions exist. Previously, several cloud services that provide a run of tests on multiple real devices were mentioned as having their own solutions for test automation. For instance, *LessPainful* (www.lesspainful.com) accepts test scripts written in Cucumber using *Calabash-Android* (github.com/calabash/). All of the aforementioned test automation drivers can be used for cloud-based testing of mobile systems. One of considered enhancements is to provide users with a choice of test scripts to use. The principles of usage are similar to Monkeyrunner, so it does not require a lot of work to integrate another driver like Robotium.

6 Research Studies in Mobile Testing

In the Table 2 we summarize recent research studies in the field of mobile testing. Each of them concerns a testing aspect that can be used in the cloud. For instance, many of the research studies deal with test automation, and theoretically, any service like device clouds can use described approaches as the test automation driver. In the same way, such extensions like test generation or static analysis can serve as an additional functionality integrated within any cloud service to facilitate mobile testing. Table 2 shows research areas and contributions for papers and highlights the year of release and the targeted mobile platforms. We can conclude that the popularity of mobile testing continues to grow and touches all possible aspects from effective test generation and design to execution and monitoring. At the same time, Android became the most popular platform under study. An open-source nature, prevalence in the market, support of an enormous number of devices, and ease of development (no provisions or jailbreaks are needed as in the case of iOS)—all make it the choice of researchers. These listed studies are potential directions for implementation of the integrated cloud services for mobile applications testing. They do not discuss cloud solutions for mobile testing, but instead present actual issues and techniques and describe possible supporting functionality.

7 Combinatorial Testing

Application of combinatorial approaches to mobile testing can aid in dealing with large amounts of different combinations of hardware and software parameters that should be covered by the tests. Coverage calculation is a crucial activity within mobile testing. So far, there are nine families of Android OS presented in the market (not counting lower sub-versions and correspondent builds without Google APIs), four types of screen resolutions (small, normal, large, and extra), and four levels of screen density.

Table 2. Researches in Mobile Testing.

Year	Ref.	Mobile Platform	Research Area	Contribution
2012	[32]	Multi (J2ME)	Automation of mobile app testing	Framework that does not require a device under testing to be connected to a computer
	[31]	Android	Whitebox automated security testing of mobile apps	Fuzz test generation approach/testbed for emulation in the cloud
	[33]	Android	Automatic categorization of mobile apps	New method for categorizing Android applications through machine-learning techniques (while accepting malicious apps into the market)
	[34]	Android	GUI-based unit testing of mobile apps	Framework to test applications from GUI
	[35]	Android	Testing mobile apps through symbolic execution	Application of symbolic execution to generate test cases for mobile apps
	[36]	Android	Verification of touch screen devices	Test environment and supporting Android app to test touch screens
	[37]	Android	Automated mobile app testing through GUI-ripping	Technique and real-life case study of bug detection
2011	[38]	Android	GUI crawling-based testing of mobile apps	Technique for rapid crash testing and regression testing
	[39]	Multi	Model-driven approach for automating mobile app testing	Tool suite to apply Domain-Specific Modeling Language
	[40]	Android	Automation of mobile app testing	Review of the Android Instrumen-tation and the Positron frameworks
	[41]	Android	Automation of mobile app testing	Approach to use the Monkey tool in conjunction with JUnit
	[42]	Android (Dalvik)	Automated privacy testing of mobile apps	Automated privacy validation system to analyze apps (while they are accepted into the market)
	[43]	Multi (Android)	Automation of service-oriented mobile app testing	Approach for decentralized testing automation and test distribution
	[44]	Android	Model-based GUI testing of mobile apps	Extensive case study
	[45]	Multi	Automated test case design strategies for mobile apps	Comprehensive review of challenges and correspondent techniques
	[46]	Android	Static analysis of mobile apps	Extensions to Julia to provide formally correct analysis of mobile apps
	[47]	Android	GUI unit-testing of mobile apps	Techniques to assess the validity of the GUI code
2010	[48]	Multi (Android)	Adaptive random testing of mobile apps	Test case generation technique
2009	[49]	Windows Mobile	Automated GUI stress testing of mobile apps	Review/automated GUI stress testing tool
	[50]	J2ME	Automation of mobile app testing	Tool for testing mobile device applications
	[51]	Multi	Automation of mobile app testing	SOA based framework for mobile app testing

Other parameters like type of Internet connection (WiFi, 3G, or 4G), size of RAM, vendor, and a processor's characteristics should also be taken into account to provide adequate coverage during testing.

Many combinatorial testing materials can be found on the corresponding webpage of the National Institute of Standards and Technology (NIST) [52]. One of the simplest and easiest ways to implement combinatorial approaches is the Base Choice [53]. The idea is to create a base test case that represents the most important (common or popular) value for each parameter, and then create others by varying the value of only one parameter at a time. The base test case can be created using statistics, especially in case of mobile testing (i.e., what screen resolution is the most spread or what vendor shares the best part of the market). Pair-wise [54] and t-wise (t-way) [54] testing are the most common and powerful combinatorial testing approaches. According to the t-wise testing approach, for each subset of t input parameters of a system, every combination of valid values of these parameters should be covered by at least one test case. In pair-wise testing, which is a case of t-wise testing with t equals 2. The idea behind the t-wise approach is that the faults in the software are more likely triggered by a small number of input parameters, with the benefits being that t-wise testing providing reasonable coverage of software input space while using a small number of test cases. For example, if there are 15 Boolean input variables, the total number of various input combinations is 215 or 32,768. However, it takes only 10 input combinations (as pair-wise test cases) to cover all of the different values for each pair of input variables.

Some examples of combinatorial tests based on different configurations of Android application can be found in [56]. Other similar techniques, including t-wise testing [57], MC/DC [58], and RC/DC [59] testing criteria are also worth to be mentioned. The *ACTS tool* (csrc.nist.gov) created by the NIST and the *ALLPAIRS* (www.satisfice.com) provide engines to calculate different combinatorial strategies and perform combinatorial testing.

8 Conclusions

Ensuring quality of modern mobile applications is complicated by a variety of mobile hardware and software platforms, variety of sensors, network interfaces, existence of web mobile and hybrid applications, and also high user's expectations. This is why thorough testing of mobile applications is of a great importance for both developers and consumers of these products.

Nowadays, testing extensively migrates to the clouds allowing to support team work, shorten testing time, and to reduce development costs, that is especially important for many startup companies. In the paper we have described a set of cloud services for mobile testing that can be divided into three types: (i) device clouds (mobile cloud platforms), (ii) services to support application lifecycle management, and (iii) tools to provide processing according to some testing techniques. Mobile testing over a cloud is an extremely important activity that is very hard to research. As it was described above, a lot of cloud services exist that fulfill the initial testers' needs,

but a scalable platform for effective crowdsourcing in mobile testing supporting multidirectional testing and flexible integration of many different testing services and techniques is still of a great demand.

9 References

1. White, J., Clarke, S., Doughtery, B., Thompson, C. & Shmidt, D.: R&D Challenges and Solutions for Mobile Cyber-Physical Applications and Supporting Internet Services. Springer Journal of Internet Services and Applications, 1(1), 45–56 (2010)
2. Work, D.B. & Bayen, A.M.: Impacts of the Mobile Internet on Transportation Cyberphysical Systems: Traffic Monitoring using Smartphones. In: National Workshop for Research on High-Confidence Transportation Cyber-Physical Systems: Automotive, Aviation and Rail, Washington, DC, USA (2008)
3. Leijdekkers, P. & Gay, V.: Personal Heart Monitoring and Rehabilitation System using Smart Phones. In: Intern. Conf. on Mobile Business, Copenhagen, Denmark (2006)
4. Moser, K.: Improving Work Processes for Nuclear Plants. In: American Nuclear Society Utility Working Conf., Hollywood, Florida, USA (2012)
5. Wasserman, A.: Software engineering issues for mobile application development. In: Workshop on Future of Software Engineering Research at the 18th Int. Symposium on Foundations of Software Engineering (ACM SIGSOFT), Santa Fe, USA, pp. 397–400 (2010)
6. The Essential Guide to Mobile App Testing, `http://www.utest.com/landing-blog/essential-guide-mobile-app-testing`
7. Holler, R.: Mobile Application Development: A Natural Fit with Agile Methodologies, `http://www.versionone.com/pdf/mobiledevelopment.pdf`
8. Vilkomir, S.: Cloud Testing: A State-of-the-Art Review, Information & Security: An International Journal, 28(2), No. 17, 213–222 (2012)
9. Tilley. S., Parveen, T.: Software Testing in the Cloud: Perspectives on an Emerging Discipline, IGI Global (2012).
10. Tsai, W., Chen, X., Liu, L., Zhao, Y., Tang, L. & Zhao, W.: Testing as a service over cloud. In: 5th IEEE Int. Symposium on Service Oriented System Engineering, pp. 181–188 (2010)
11. Kalliosaari, L., Taipale, O. & Smolander, K.: Testing in the Cloud: Exploring the Practice. IEEE Software, 29(2), 46–51 (2012)
12. Weidong, F., Yong, X.: Cloud testing: The next generation test technology. In: 10th Int. Conf. Electronic Measurement & Instruments, Chengdu, China, pp. 291–295 (2011)
13. Inçki, K., Ari, I., Soze, H.: A Survey of Software Testing in the Cloud. In: IEEE 6th Int. Conf. on Software Security and Reliability Companion, pp. 18–23 (2012)
14. Priyanka, Chana, I., Rana, A.: Empirical evaluation of cloud-based testing techniques: a systematic review. ACM SIGSOFT Software Engineering Notes archive, 37(3), 1–9 (2012)
15. Mote, D.: Cloud based Testing Mobile Apps. In: 2nd IndicThreads.com Conference on Software Quality, Pune, India (2011)
16. Cloud Testing: Database of Cyber Security and Information Systems Information Analysis Center, `https://sw.thecsiac.com/databases/url/key/7848/8764/8765#.USGPb-h8vDm`
17. Tilley, S., Parveen, T.: Software Testing in the Cloud: Migration & Execution. Springer Briefs in Computer Science (2012)
18. Riungu, L.M., Taipale, O., Smolander, K.: Research Issues for Software Testing in the Cloud. In: IEEE 2nd Int. Conf. Cloud Computing Technology and Science, pp. 557–564. (2010)

19. Rhoton, J., Haukioja, R.: Cloud Computing Architected: Solution Design Handbook. Recursive (2011)
20. Coulouris, G., Dollimore, J., Kindberg, T., Blair, G.: Distributed Systems: Concepts and Design. Addison-Wesley (2011)
21. Muccini, H., Francesco, A., Esposito, P.: Software testing of mobile applications: Challenges and future research directions. In: 7th Int. Workshop on Automation of Software Test (2012)
22. Franke, D., Weise, C.: Providing a Software Quality Framework for Testing of Mobile Applications. In: IEEE 4th Int. Conf. on Software Testing, Verification and Validation, Berlin, Germany, pp. 431–434 (2011)
23. Milano D.: Android Application Testing Guide. Publishing Ltd. (2011)
24. Frederick G.. Lal, R.: Testing a Mobile Web Site. Beginning Smartphone App Development -- Part IV. Apress (2009)
25. Dantas, V., Marinho, F., Da Costa, A., Andrade, R.: Testing requirements for mobile applications. In: 24th Int. Symposium on Computer and Information Sciences (2009)
26. Test Strategies for Smartphones and Mobile Devices, http://www.macadamian.com/images/uploads/whitepapers/MobileTestStrategies_Aug2010.pdf
27. Make Your Mobile Testing Solution Enterprise-Ready, http://www.perfectomobile.com/portal/cms/resources/enterprise-ready_white-paper
28. Konstantinidis, A., Costa, C., Larkou, G., Zeinalipour-Yazti, D.: Demo: a programming cloud of smartphones. In: 10th Int. Conf. on Mobile Systems, Applications, and Services, pp. 465–466 (2012)
29. Turner, H., White, J., Reed, J., Galindo, J., Porter, A., Marathe, M, Vullikanti, A., Gokhale, A.: Building a Cloud-Based Mobile Application Testbed. IGI Global (2012)
30. She, S., Sivapalan, S., Warren, I.: Hermes: A Tool for Testing Mobile Device Applications. In: Software Engineering Conf., Queensland, Australia (2009l)
31. Mahmood, R., Esfahani, N., Kacem, T., Mirzaei, N., Malek, S., Stavrou A.: A whitebox approach for automated security testing of Android applications on the cloud. In: 7th Int. Workshop on Automation of Software Test, pp. 22–28 (2012)
32. Nagowah, L., Sowamber, G.: A Novel Approach of Automation Testing on Mobile Devices. In: Int. Conf. on Computer & Information Science, Vol. 2, pp. 924–930 (2012)
33. Sanz, B., Santos, I., Laorden, C., Ugarte-Pedrero, X., Bringas, P.: On the Automatic Categorisation of Android Applications. In: 9th Annual IEEE Consumer Communications and Networking Conf. – Security and Content Protections (2012)
34. Allevato, A., Edwards, S.: RoboLIFT: simple GUI-based unit testing of student-written android applications. In: 43rd ACM Technical Symposium on Computer Science Education, p. 670 (2012)
35. Mirzaei, N., Malek, S., Păsăreanu, C., Esfahani, N., Mahmood, R.: Testing Android Apps Through Symbolic Execution. ACM SIGSOFT Software Engineering Notes archive, 37(6), 1–5 (2012)
36. Zivkov, D.: Touch screen mobile application as part of testing and verification system. In: 35th Int. Convention MIPRO, pp. 892–895 (2012)
37. Amalfitano, D., Fasolino, A., Tramontana, P., De Carmine, S.: Using GUI ripping for automated testing of Android application. In: 27th IEEE/ACM Int. Conf. on Automated Software Engineering, Germany (2012)
38. Amalfitano, D., Fasolino, A.R., Tramontana, P.: A GUI Crawling-Based Technique for Android Mobile Application Testing. In: Software Testing, Verification and Validation Workshops, pp. 252–261 (2011)

39. Ridene, Y., Barbier, F.: A model-driven approach for automating mobile applications testing. In: 5th European Conf. on Software Architecture: Companion (2011)
40. Kropp, M., Morales, P.: Automated GUI Testing on the Android Platform. IMVS Fokus Report, 4(1) (2010)
41. Hu, C., Neamtiu, I.: Automating GUI testing for Android applications. In: 6th Int. Workshop on Automation of Software Test, pp. 77–83 (2011)
42. Gilbert, P., Chun, B., Cox, L., Jung, J.: Automating Privacy Testing of Smartphone Applications. Technical Report CS-2011-02 (2011)
43. Edmondson, J., Gokhale, A., Sandeep Neema: Automating Testing of Service-oriented Mobile Applications with Distributed Knowledge and Reasoning. In: Service-Oriented Computing and Applications, pp. 1–4 (2011)
44. Takala, T., Katara, M., Harty, J.: Experiences of System-Level Model-Based GUI Testing of an Android Application. In: Software Testing, Verification and Validation, pp. 377–386 (2011)
45. Selvam, R., Karthikeyani V.: Mobile Software Testing – Automated Test Case Design Strategies. Int. J. on Computer Science and Engineering (2011)
46. Payet, E., Spoto, F. Static Analysis of Android Programs, In: Automated Deduction CADE-23, LNCS, vol. 6803, pp. 439–445 (2011)
47. Sadeh, B., Ørbekk, K., Eide, M., Gjerde, N., Tønnesland, T., Gopalakrishnan, S.: Towards Unit Testing of User Interface Code for Android Mobile Applications. Communications in Computer and Information Science, 181(1), 163–175 (2011)
48. Liu, Z., Gao, X., Long, X.: Adaptive random testing of mobile application. In: Computer Engineering and Technology, vol., pp. 297–301 (2010)
49. Abdallah, N., Ramakrishnan, S.: Automated Stress Testing of Windows Mobile GUI Applications. In: 20th Int. Symposium on Software Reliability Engineering (2009)
50. Sivapalan, S., Warren, I.: Hermes: A Tool for Testing Mobile Device Applications. In: Software Engineering Conference, Australia (2009)
51. Zhi-fang Liu, Bin Liu, Xiao-peng Gao: SOA based mobile application software test framework. In: 8th Int. Conf. Reliability, Maintainability and Safety, pp. 765–769 (2009)
52. Combinatorial Methods in Software Testing, `http://csrc.nist.gov/groups/SNS/acts/`
53. Grindal, M., Offutt, J., Andler, S.F.: Combination Testing Strategies: a Survey. Software Testing, Verification and Reliability, 15(3), 167–199 (2005)
54. Kuhn, D.R., Lei, Y.,Kacker, R.: Practical Combinatorial Testing - Beyond Pairwise. IEEE IT Professional, 6, 19–23 (2008)
55. Maximoff, J.R., Trela, M.D., Kuhn, D.R., Kacker, R.: A Method for Analyzing System State-space Coverage within a t-Wise Testing Framework. In: IEEE Int. Systems Conf., San Diego (2010)
56. Kuhn, D.R., Kacker, R.N., Lei, Y.: Practical Combinatorial Testing. NIST Special Publication, 10, 13–15 (2010)
57. Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., Lawrence, J.: IPOG: A General Strategy for T-Way Software Testing. In: IEEE Engineering of Computer Based Systems Conf., pp. 549–556 (2007)
58. Chilenski, J.J., Miller, S.: Applicability of Modified Condition/Decision Coverage to Software Testing. Software Engineering J., 9, 193–200 (1994)
59. Vilkomir, S., Bowen, J.P.: From MC/DC to RC/DC: Formalization and Analysis of Control-flow Testing Criteria. Formal Aspects of Computing, 18(1), 42–62 (2006)