Cloud Testing for Mobile Software Systems: Concept and Prototyping

Sergiy Vilkomir
East Carolina University
58 PUBLICATIONS **497** CITATIONS

Vyacheslav Kharchenko
KhAI - Aerospace university
102 PUBLICATIONS **173** CITATIONS

Cloud Testing for Mobile Software Systems
Concept and Prototyping

Oleksii Starov[1], Sergiy Vilkomir[1] and Vyacheslav Kharchenko[2]
[1]*Department of Computer Science, East Carolina University, Greenville, NC, U.S.A.*
[2]*Department of Computer Systems & Networks, National Aerospace University KhAI, Kharkiv, Ukraine*
{starovo12, vilkomirs}@ecu.edu, v.kharchenko@khai.edu

Keywords: Cloud Testing, Mobile Systems, Testing Framework, TaaS, Android.

Abstract: This paper describes an approach for increasing the effectiveness of mobile software system testing. A Cloud Testing of Mobile Systems (CTOMS) framework is presented in the form of a cloud service that provides the ability to run tests on a variety of remote mobile devices. This framework is based on a heterogeneous networked system that connects operational computers, mobile devices, and databases with software applications. Our research focuses on building a concept and a prototype of CTOMS that supports testing Android mobile applications in the cloud. CTOMS allows multidirectional testing, providing the opportunities to test an application on different devices and/or operating system (OS) versions and new device models for their compatibility with the newest OS versions and the most popular applications. Another new aspect is to embed the test model, specifically the appropriate testing techniques for mobile development, within the framework. For users, this model will provide suggestions from CTOMS about the test methods, criteria, coverage, and possible test cases. These suggestions are based on available configurations, statistics, and resource constraints.

1 INTRODUCTION

Mobile developments are presented nowadays as a variety of different applications with different quality requirements. Interest in critical mobile applications that require high-level reliability and security is growing rapidly. For instance, mobile applications have become commonplace for online banking (Bank of America, 2013), and some researchers are discussing the use of smartphones and tablets at nuclear power plants (Moser, 2012). A new trend is to use smartphones as components for mobile cyber-physical systems because the powerful hardware has a variety of sensors (White et al., 2010). Examples of such systems include mobile applications for notifications about hurricanes (Carr, 2012), monitoring cardiac patients (Leijdekkers, 2006), and traffic monitoring (Work and Bayen, 2008).

To guarantee the 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 (uTest, 2013). Adequately testing all of these platforms is too expensive—perhaps impossible—especially for small resource-constrained mobile development companies.

This paper describes a framework that facilitates the testing of mobile applications. The idea is to create a cloud service that provides the ability to run tests on a variety of remote mobile devices (i.e., smartphones). The proposed framework is based on a heterogeneous networked system that connects operational computers, mobile devices, and databases with applications. This framework is presented as a combination of hardware (smartphones) and software (applications) that allows for different testing directions. For instance, it is possible to test a new smartphone model for its compatibility with mobile applications and to test a new application on different smartphone models. This framework—CTOMS—serves as Testing as a Service (TaaS) for mobile development.

This paper is organized as follows. Section 2 discusses the current state-of-the-art cloud testing and the testing of mobile systems and applications. Section 3 presents CTOMS, including its ability for multidirectional testing, and the use of testing models that allow users to choose specific testing methods for their systems. The issues of security and performance testing using CTOMS are also analyzed and described. A high-level CTOMS architecture is suggested in Section 4. Elements of prototyping and feasibility studies of the framework are described in Section 5. Section 6 concludes the paper and proposes the directions for future work.

2 RELATED WORKS

A variety of cloud testing services exist to facilitate software testing (Inçki et al., 2012); (Vilkomir, 2012); (Tilley and Parveen, 2012), including testing mobile applications (Priyanka et al., 2012). Several of these services provide remote access to connected smartphones in order to accomplish their testing (Perfecto Mobile, 2013); (Keynote Device Anywhere, 2013). Of these, the Perfecto Mobile service provides a well-suited example of optimum core functionality for our proposed CTOMS framework. Specifically, Perfecto Mobile aids mobile developers in using remote smartphones for manual testing, recording of scripts, and automatic running of tests on a range of models. The CTOMS framework offers two key enhancements over existing testing services: multidirectional testing capabilities and integration of the testing model into the functionality of the service. Apkudo's device analytics (Rowinski, 2012) provide some elements of multidirectional testing by testing devices on the top 200 apps from the market. Keynote DeviceAnywhere Test Planner provides elements of testing techniques by suggesting a set of devices to use for testing. While these services do not offer the total range of desired comprehensive functionalities compared with CTOMS, they point to the interest in this testing direction.

A scientific testbed for a cloud solution is discussed in Konstantinidis et al. (2012). Usage of a set of plug-ins for different testing methods as a part of the cloud testing framework is considered in Jenkins et al. (2011). Cloud testing distributed systems is analyzed in Tilley and Parveen (2012), Rhoton and Haukioja (2011), and Coulouris et al. (2011).

Many testing approaches were investigated specifically for Android applications, including automation of testing and GUI testing (Hu and Neamtiu, 2011); (Ridene and Barbier, 2011), testing frameworks that use distributed networked solutions (Ridene and Barbier, 2011); (Mahmood et al., 2012), security testing (Mahmood et al., 2012), and others.

Many investigations justify Android selection as a base mobile platform for the CTOMS prototype. Thus, according to Gartner, Android devices hold the best part of the market (Haselton, 2012), and forecasts by *Forbes* indicate that the Android platform will advance to meet enterprise requirements soon (Fidelman, 2012). Previous research regarding bugs statistics in Android OS (Maji, 2010) proves that Android has effectively organized an open-sourced bug-tracking system that can be helpful in further investigations.

3 THE CTOMS CONCEPT

3.1 Multidirectional Testing

This paper aims to generalize the concept of the cloud service that provides mobile devices for testing. The CTOMS framework extends the typical functionality and provides the ability to test OS version updates and new hardware devices against most popular or/and important applications. It is important to be confident that the legacy mobile applications will still work properly in a new environment.

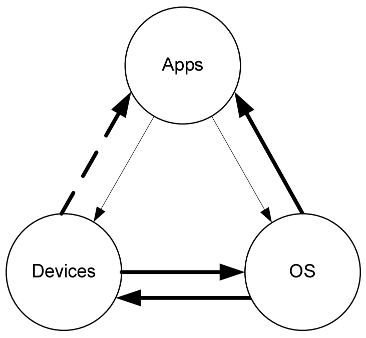

Figure 1: Multidirectional testing.

Figure 1 illustrates the concept of multidirectional testing. It shows the three main types of objects in the system: applications (apps), devices (hardware), and versions of OS. CTOMS provides the ability to test each side on/against others; in other words, it provides multidirectional testing from all possible perspectives. All use cases are in high demand. Simple lines show existing services. Dashed lines show partially new use cases; see Rowinski (2012) for an analogue of testing new devices against applications. Bold arrows show totally new functionality.

The current study considers cloud solutions for the following scenarios:

1. Application developers can test a product on different devices and/or OS versions.
2. OS developers can test new versions of an OS on a set of modern devices and the most popular apps to ensure compatibility.
3. Hardware developers can test new device models for their compatibility with the newest OS versions and the most popular applications.

The innovative aspect of our approach is that it provides testing of new OS version against the top popular applications (and test cases used for their development). The relevance of such functionality becomes obvious if we consider how rapidly new versions of iOS or Android systems are released. The same acute situation applies to hardware, thus the device fragmentation testing matrix for Android development can have nearly 4,000 separate Android device models (uTest, 2013).

In general, the need to test OS or hardware against applications is driven by critical systems that contain mobile applications. In such systems, it is very important to guarantee the dependability of crucial applications with a newer version of the OS or a new device. Critical mobile applications must still work properly after OS updates and provide the same reliability, or else the resulting faults can be very expensive. The presence of all testing perspectives in the CTOMS framework provides the ability to comprehensively test mobile systems because the hardware components are also key ones for them.

Android development is the most representative example of the problem of diverse configurations to be tested. The variety of heterogeneous devices, OS versions, screen resolutions, and other parameters is significant. The popularity of the Android operating system necessarily makes the question of cloud testing extremely important. As such, this research focuses first on building a prototype of the CTOMS that supports Android testing. The following sections offer solutions from both the architecture and implementation points of view. The proposed CTOMS framework also takes into consideration the possibility of support for different kinds of devices: from smartphones to mobile robots and from microcontroller-based embedded systems to field-programmable hardware (Kharchenko et al., 2009).

3.2 The CTOMS Structure

The conceptual structure of the CTOMS framework contains several layers of mechanisms and functionalities. Each should be analyzed, architected, and implemented in a final comprehensive system. Figure 2 illustrates these layers and their connections with possible use cases:

1. The contributor of the device only invests in hardware by connecting it to the cloud system.
2. The application developer uploads the application under test (source codes or binary), specifies test cases (automated scripts or unit tests), gets results as pass/fail statistics or screenshots for checkpoints, manually accesses selected devices for debugging, etc.

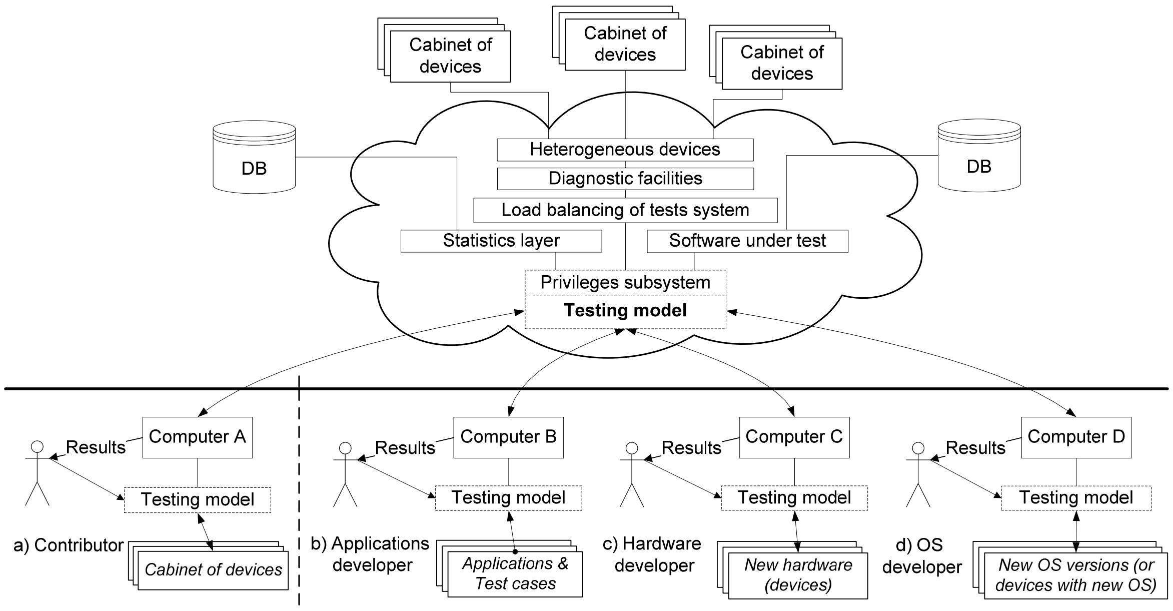

Figure 2: The CTOMS structure and use cases.

3. The application developer uploads the application under test (source codes or binary), specifies test cases (automated scripts or unit tests), gets results as pass/fail statistics or screenshots for checkpoints, manually accesses selected devices for debugging, etc.
4. The device producers test new devices against the base of apps and OS software in CTOMS. Test scripts available in the system for chosen apps can be used. Producers can also specify their own test scenarios.
5. The OS producer connects the devices with new OS versions to the cloud or uploads update packages to the database. Then the new OS versions are tested with apps/scripts/ devices in the system. OS producers can also specify their own test scenarios.

Figure 2 shows that each user can provide a "testing model" while using the framework. For application developers, this means specifying a test strategy, namely, what tests need to be performed on what devices and how. For example, they can specify their own test scripts; select devices, OS versions, coverage, methods to check if a test is passed, etc. For other users, this means not only specification of test strategy or rules of testing (in case of contributor), but also how to perform the testing on connected devices. For example, the settings can be made for which or how many applications to test. The technical interface of the system for contributors, as well as OS and hardware producers, is almost the same.

The innovative feature of CTOMS is its testing model that serves as an internal mechanism and additional service for users (layer inside cloud on the Figure 2). This aspect is described in more details in the next section.

The databases in CTOMS store the software (applications and OS versions), the testing results, the statistical information about testing, and user information for granting privileges based on the billing and for providing multi-tenancy, etc.

This work with devices requires the development of three additional layers: load balancing of tests execution, diagnostic facilities, and heterogeneous device connections. All of these layers (inside the cloud in Figure 2) should be investigated in accordance with earlier specified new use cases, along with the creation of corresponding methodologies. Load balancing of tests execution system means algorithms to distribute test cases between connected smartphones in optimal ways with respect to time of operation and wait time of other users. Simultaneously, general scalability should also be taken into account.

The main high-level architecture solutions for the challenges above are described in Section 4.

3.3 Testing Model

Another new aspect proposed in this study is to embed the test model, i.e., appropriate testing techniques for mobile development within the cloud framework. Specifically, pair-wise testing (Kuhn et al., 2009) is considered for this purpose. The pair-wise testing combinatorial approach aids in dealing with large amounts of different combinations of hardware and software parameters that should be covered by the tests. Coverage evaluation is a crucial activity within mobile testing. According to the Android Developers website (Android Developers, 2013), there are nine families of Android OS present in the market (not counting subversions and builds without Google APIs), four types of screen resolution (small, normal, large, extra), and four levels of screen density. Other parameters, such as the type of Internet connection (WiFi, 3G), size of ram, vendor, and processor's characteristics, should also be taken into consideration n order to provide adequate coverage during testing. Some examples of combinatorial tests based on different configurations of Android application can be found in Kuhn et al. (2010). Other techniques, including t-wise testing (Lei et al., 2007), MC/DC (Chilenski and Miller, 1994), and RC/DC (Vilkomir and Bowen, 2006) testing criteria, are also considered for integration with CTOMS.

From a user's point of view, the embedding of a testing model operates as suggestions provided by CTOMS: namely, suggestions relating to what hardware-software configurations need to be tested, the testing criterion to choose, the minimal test coverage required, the risk statistics about particular device configurations, etc. As a result, a user can choose an appropriate testing model for a given situation in terms of desired budget, time, requirements, etc.

A testing model must also provide the general organization of testing (i.e., launching tests, storing results, etc.). This provides an opportunity to collect statistics in the system, and for instance, to advise a user that a particular device caused the main part of defects during other similar applications testing.

CTOMS can also be considered for providing reliability, performance, and security testing. In the context of security testing, the following variants of additional services are proposed:

- Implementation of different kinds of static analysis.
- Whitebox approaches based on decompiling Java classes (Mahmood et al., 2012).
- Model-driven approaches for security and language-based security analysis.
- Automated stress security testing.

For performance testing, it is proposed to use frame rate counters similar to Windows phone Emulator (Windows Phone Dev Center, 2013).

For reliability testing, detailed usage of statistics is proposed in conjunction with the long-term performing of tests.

A detailed design of the CTOMS framework aims to provide the ability to extend functionality with such kinds of testing in a way similar to plug-ins. The global goal is to create a comprehensive testing environment.

4 ARCHITECTURE OF CTOMS

The architecture of a networked system, such as CTOMS, can vary significantly in its level of complexity. For example, the size of the desired distributed solution is one dependent factor. CTOMS can be architected as a single PC computer with connected smartphones, or as a comprehensive cloud solution that operates hundreds of such PC nodes. To achieve all of its goals, CTOMS should be implemented from a large-scale perspective and must correspond to all cloud computing features, such as service delivery, scaling, virtualization, elasticity, multi-tenancy, load balancing, universal access, etc. The cloud type can also differ from a public SaaS (Software as a Service) solution to a private IaaS (Infrastructure as a Service), with the ability to provide a low-level configuration.

In this paper, trade-off solutions for CTOMS complexity are suggested. The solutions include leveraging public cloud providers, such as Google App Engine (GAE) and Amazon AWS to create a master application (layer) of the system in PaaS (Platform as a Service). Computers with connected mobile devices serve as “leaf nodes” and create a slave layer of the system. They perform actual testing on smartphones, collect results, and interact with a master application through the Internet.

Figure 3 illustrates the high-level architecture of CTOMS. It shows three layers of the system interacting with each other. The separation of the master layer as a public PaaS application gives the ability to easily achieve both horizontal and vertical scaling (layer 1 in Figure 3). The master layer does not depend on the other layers, e.g., leaf nodes. Besides, it also can leverage the highly scalable databases and data storage provided by PaaS. The master application has a comprehensive web interface.

Communication between subsystems through the Internet (web services) provides a convenient scaling of layer 3. Leaf computers with smartphones can be easily connected to a master application in the cloud. They could also operate emulators (virtual Android devices marked with a dashed line). The interface for working with emulators and devices is the same.

The middle layer 2 is optional. Its goal is to leverage embedded Hadoop facilities of chosen PaaS, particularly, MapReduce functionality. The

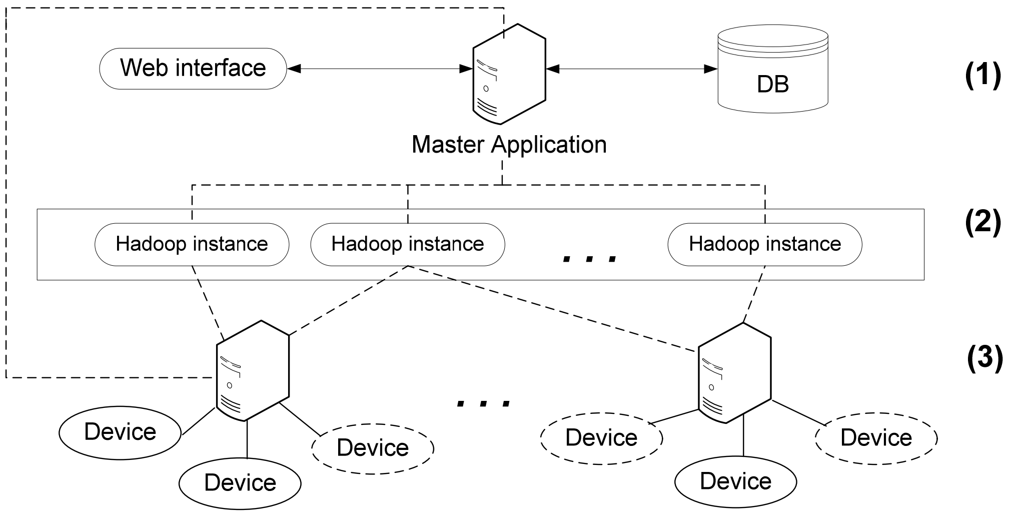

Figure 3: A variant of the CTOMS architecture.

aim is to organize test distribution and results gathering by the master application. The algorithms for effective load balancing of tests execution still are performed by the master application.

5 IMPLEMENTATION OF THE CTOMS PROTOTYPE

The current study provides a prototyping of the CTOMS according to a specified distributed architecture of the desired cloud system. The first and third layers, according to Figure 3, were implemented as the first prototype. The resulting system consists of the two web Java applications: the master and the leaf node.

The created master application is placed in a Google App Engine cloud (Google App Engine, 2013), and, from a user's point of view, looks like a web interface to upload Android applications' binaries and test scripts and get testing results.

The MonkeyRunner tool (MonkeyRunner, 2013) provided with Android SDK (Android SDK, 2013) is used to perform and automate actual testing on mobile devices. This was the easiest choice for acquiring a useful system that reflects all the required characteristics of similar testing services. Our aim was not to create a new and better way of testing automation or manual remote access to smartphones, but to create a cloud system to accommodate other improvements. Thus, the current CTOMS prototype accepts scripts for the MonkeyRunner tool and provides as testing results a set of screenshots-checkpoints taken on different devices to compare by user-tester. Then the user can specify which screenshot shows a bug.

The following associated issues were elaborated: scripts parallelization to run on several devices simultaneously; automation of statistics of fail/pass checks based on heuristic images comparison; and a possibility to embed useful supporting frameworks as AndroidViewClient (AndroidViewClient, 2013) and ViewServer (ViewServer, 2013).

Users can also run the client application on their own computers to perform local testing. The functionalities are the same (except for the absence of integrated testing techniques). In this case, the computer plays the role of the leaf node.

Functionality of multidirectional testing is provided in the CTOMS prototype. It gives an opportunity to test a connected device against applications and corresponding tests within the database of the system. Then a user can compare screenshots taken on the new device (or device with installed new OS version) to nominal screenshots taken on trustworthy old models. This new device can be connected to the system from a user's site and serve in the private (not shared) mode. At the same time, MonkeyRunner tests do not require uploading a binary (APK file) and can simulate common user interactions with Android OS.

The ACTS tool by the National Institute of Standards and Technology (NIST, 2013) is used to integrate a combinatorial testing model in CTOMS. A correspondent interface is provided through the master application.

All interconnections of the master application and leaf nodes were implemented through RESTful web services. This gives the ability to publicly expose them as Application Programming Interfaces (Jacobson et al., 2011).

Integration of static code analyzers to provide quality indicators and reveal security vulnerabilities was considered, but because such functionality can serve as a fully separated service, it was moved to further versions.

6 CONCLUSIONS

The CTOMS cloud framework is presented here as a way to increase the quality of testing mobile applications. The main concepts, architecture, and prototype implementation are described. Integrating a testing model in CTOMS allows researchers to apply different testing techniques and provide performance and security testing of mobile applications. The current prototype provides ability of functional testing and detecting user interface related bugs (e.g., layout or graphic performance issues).

A variety of extensions to the framework are possible on conceptual and implementation levels. The future work will focus on case studies and experiments using CTOMS. In particular, using possible crowdsourcing benefits provided by CTOMS, we are going to investigate dependency between bugs in mobile applications and updates of OS, and show testing effectiveness of such a cloud of devices. Development experience and preliminary analysis of reviews and bug reports in Google Play indicate high clustering of defects for a specific OS version or smartphone model. For instance, based on the latest reviews of Twitter Android app (Twitter, 2013), we can conclude that users have problems with notifications on different devices under Android 4.0.1. Some statistics on causes of mobile app crashes (including crashes by OS version), provided by Crittercism (Crittercism, 2013), can be found in Geron (2012).

REFERENCES

Android Developers, 2013. Dashboards. Available at: http://developer.android.com/about/dashboards/

Android SDK, 2013. Available at: http://developer. android.com/sdk/

AndroidViewClient, 2013. Extension to MonkeyRunner. Available at: https://github.com/dtmilano/Android ViewClient

Bank of America, 2013. Mobile Banking. Available at: https:// www.bankofamerica.com/online-banking/ mobile.go

Carr, D. F. 2012. *Hurricane Sandy: Mobile, Social Tools Help Emergency Management,* Brainyardnews (Oct. 2012). Available at: http://www.informationweek. com/thebrainyard/news/social_media_monitoring/240 012463/hurricane-sandy-mobile-social-tools-help-emergency-management

Chilenski, J. J., Miller, S., 1994. *Applicability of Modified Condition/Decision Coverage to Software Testing*, Software Engineering Journal, Sept. 1994, 193-200.

Coulouris, G., Dollimore, J., Kindberg, T., Blair, G., 2011. *Distributed Systems: Concepts and Design*, Addison-Wesley. 5nd edition.

Crittercism, 2013. Available at: https://www. crittercism.com/

Fidelman, M., 2012. *The Latest Infographics: Mobile Business Statistics For 2012*. Available at: http:// www.forbes.com/sites/markfidelman/2012/05/02/the-latest-infographics-mobile-business-statistics-for-2012

Geron, T. 2012. *Do iOS Apps Crash More Than Android Apps? A Data Dive*. Available at: http://www.forbes.com/sites/tomiogeron/2012/02/02/d oes-ios-crash-more-than-android-a-data-dive/

Google App Engine, 2013. Developers portal. Available at: https://developers.google.com/appengine

Haselton, T., 2012. *Android Has 56.1% Of Global OS Market Share, Gartner Says*. Available at: http://www.technobuffalo.com/2012/05/16/android-has-56-1-of-global-os-market-share-gartner-says

Hu, C., Neamtiu, I., 2011. Automating GUI testing for Android applications. In *Proceedings of the AST '11, 6th International Workshop on Automation of Software Test*. ACM New York, NY, USA, 77-83.

Inçki, K., Ari, I., Sozer, H., 2012. A Survey of Software Testing in the Cloud. In *Proceedings of 2012 IEEE Sixth International Conference on Software Security and Reliability Companion*. 18-23.

Jacobson, D., Brail, G., Woods, D., 2011. *APIs: A Strategy Guide*. O'Reilly Media, Inc., 60-70.

Jenkins, W., Vilkomir, S., Sharma, P., Pirocanac, G., 2011. Framework for Testing Cloud Platforms and Infrastructures. In *Proceedings of the CSC 2011, International Conference on Cloud and Service Computing,* Hong Kong, China, Dec. 12-14, 2011, 134-140.

Keynote DeviceAnywhere, 2013. The Mobile Testing Platform. Available at: http://www.keynotedevice anywhere.com

Kharchenko, V., Siora, O., Sklyar, V., 2009. Design and testing technique of FPGA-based critical systems. In *Proceedings of CADSM 2009, 10th International Conference - The Experience of Designing and Application of CAD Systems in Microelectronics* (Polyana-Svalyava, Ukraine, Feb. 24-28, 2009, 305-314.

Konstantinidis, A., Costa, C., Larkou, G., Zeinalipour-Yazti, D., 2012. Demo: a programming cloud of smartphones. In *Proceedings of the MobiSys '12, 10th international conference on Mobile systems, applications, and services.*. ACM New York, NY, USA, 465-466.

Kuhn, R., Kacker, R., Lei, Y., Hunter, J., 2009. *Combinatorial Software Testing*, IEEE Computer. Volume 42, Number 8, Aug. 2009, 94-96.

Kuhn, R., Kacker, R. N., Lei, Y., 2010. *Practical Combinatorial Testing*, NIST Special Publication, October, 2010.

Lei, Y., Kacker, R., Kuhn, D. R., Okun, V., Lawrence, J., 2007. IPOG: A General Strategy for T-Way Software Testing. In *Proceeding of ECBS '07, IEEE Engineering of Computer Based Systems conference,* March 2007, 549-556.

Leijdekkers, Gay, V. 2006. Personal Heart Monitoring and Rehabilitation System using Smart Phones. In *Proceedings of the International Conference on Mobile Business*. Citeseer, 29.

Mahmood, R., Esfahani, N., Kacem, T., Mirzaei, N., Malek, S., Stavrou, A., 2012. A whitebox approach for automated security testing of Android applications on the cloud. In *Proceedings of AST 2012, Automation of Software Test, 7th International Workshop*.

Maji, K., 2010. Characterizing Failures in Mobile OSes: A Case Study with Android and Symbian. In *Software Reliability Engineering (ISSRE) 21st International Symposium*.

MonkeyRunner, 2013, Android SDK tool. http://developer.android.com/tools/help/monkeyrunner _concepts.html

Moser K., 2012. Improving Work Processes for Nuclear Plants (Exelon Nuclear). In *American Nuclear Society Utility Working Conference 2012*.

NIST, 2013. ACTS tool. Available at: http://csrc.nist.gov/ groups/SNS/acts/download

Perfecto Mobile, 2013. The MobileCloud Company. Available at: http://www.perfectomobile.com/

Priyanka, Chana, I., Rana, A., 2012. *Empirical evaluation of cloud-based testing techniques: a systematic review*, ACM SIGSOFT Software Engineering Notes archive. Volume 37, Issue 3, May 2012, ACM New York, NY, USA, 1-9.

Rhoton, J., Haukioja, R., 2011. *Cloud Computing Architected: Solution Design Handbook*, Publisher: Recursive, Limited, ISBN: 0956355617.

Ridene, Y., Barbier, F., 2011 A model-driven approach for automating mobile applications testing. In

Proceedings of the 5th European Conference on Software Architecture: Companion Volume Article No. 9.

Rowinski D., 2012. *Mobile Carriers and OEMs Get Android App Testing Cloud from Apkudo*, ReadWrite (February 7, 2012). Available at: http://readwrite.com/2012/02/07/mobile-carriers-and-oems-get-a

Tilley, S., Parveen, T., 2012. *Software Testing in the Cloud: Perspectives on an Emerging Discipline*, Information Science Reference, Nov. 2012.

Twitter, 2013. Application for Android. Available at: https://play.google.com/store/apps/details?id=com.twitter.android&hl=en

uTest, Inc., 2013. *The Essential Guide to Mobile App Testing (free eBook)* Available at: http://www.utest.com/landing-blog/essential-guide-mobile-app-testing.

ViewServer, 2013. Available at: https://github.com/romainguy/ViewServer

Vilkomir, S., 2012. *Cloud Testing: A State-of-the-Art Review*, Information & Security: An International Journal. Volume 28, Issue 2, Number 17, 2012, 213-222.

Vilkomir, S., Bowen, J. P., 2006. *From MC/DC to RC/DC: formalization and analysis of control-flow testing criteria*, Formal aspects of computing, Volume 18, Number 1, 2006, 42-62.

White, J., Clarke, S., Dougherty, B., Thompson, C., and Schmidt, D. 2010. R&D Challenges and Solutions for Mobile Cyber-Physical Applications and Supporting Internet Services. Springer Journal of Internet Services and Applications. Volume 1, Number 1 (2010), 45-56.

Windows Phone Dev Center, 2013. Testing Apps for Windows Phone. Available at: http://msdn.microsoft.com/library/windowsphone/develop/jj247547(v=vs.105).aspx

Work, D. B., Bayen, A. M., 2008. Impacts of the Mobile Internet on Transportation Cyberphysical Systems: Traffic Monitoring using Smartphones. In *Proceedings of National Workshop for Research on High-Confidence Transportation Cyber-Physical Systems: Automotive, Aviation and Rail*, Washington, DC, Nov. 18-20, 2008.