Asian Journal of Multidisciplinary Studies
Vol. 1, No. 2, (2018)
ISSN 2651-6691 (Print)
ISSN 2651-6705 (Online)

Automated Testing Tools for Mobile Applications: A Literature Review

Jayson E. Tamayo
College of Computing Sciences
Pangasinan State University
jayson.tamayo2@gmail.com

Abstract *- The rapid growth of demand for mobility made the mobile application development a fast-emerging area in software development. As years pass by, the product of software development has become increasingly complex. This makes test activities even more important and challenging. Tom Taulli, a Forbes contributor, denotes the importance of mobile application testing. He stated that users are more likely to abandon an application if it has functionality problems. Although there are several testing tools already available in the market, there were no empirical studies found that explore and analyze these tools in detail. Hence, this paper aims to discuss and provide a comparison of different cross-platform automated testing tools for mobile applications particularly for Android, iOS and Windows Mobile. Most of the papers were addressing test cases for app's function and behavior. The least number of papers aims to test security loopholes on mobile applications. It is worth noting that the papers focused on applications running on Android devices, there were no tools that address applications for other platforms such as iOS and Windows Phone.*

Keywords: automated testing tool, mobile application

1 INTRODUCTION

Industry analyst firm Gartner predicts that by 2022, 70 percent of software interactions will occur on mobile devices. Many organizations embrace the fact that mobile application for their business is an excellent way to keep their consumers engaged as mobility becomes the gateway to digital business. This drives a strong demand for mobile application development. Today, mobile applications have become more complex and aside from the development challenges, testing has also been an important area to look into since this process ensures the mobile application's stability and robustness [1]. According to Tom Taulli, a Forbes contributor, denotes the importance of mobile application testing. He stated that users are more likely to abandon an application if it has functionality problems.

Many mobile applications have been developed in critical areas such as transportation [2][3][4][5][6][7][8][9], healthcare [10][11][12][13][14][15][16][17], banking [18][19][20] and education [21][22][23][24][25][26]. Therefore, testing is a fundamental life-cycle activity, with a huge economical and societal impact if not performed adequately [27].

The paper of Gunasekaran [28] surveyed five different automated testing tools for mobile applications. These tools are Robotium, Ranorex, Appium, MonkeyTalk and UIAutomator. The paper, however, failed to introduce automated testing tools for all major mobile platforms namely: Android, iOS and Windows Mobile. The paper also failed to categorize these tools according to the software engineering issues for mobile application development [29] namely: performance and reliability, function and behavior and security.

This paper aims to discuss several automated testing tools for mobile applications running on Android, iOS and Windows Mobile. Each automated testing tool shall also be categorized and discussed according to what specific software engineering issue it is trying to address.

The rest of the paper is organized as follows: section 2 discusses the methodology to be used. In section 3, results will be discussed. Finally, section 4 concludes the paper and presents the future work.

2 METHODOLOGY

This paper uses the Systematic Literature Review (SLR) method in undertaking a systematic literature review. By complying to the systematic procedure defined by the said research method, this paper can provide a more objective process in selecting relevant and note-worthy studies. The major steps in SLR include the following: (1) defining a research question, (2) search strategy for selecting studies and (3) management of studies.

Using the SLR methodology, the author should be able to define a research question that is anchored to purpose of the literature review. The author should also be able to plan for the search strategy and specify the steps needed. Lastly, the author should be able to manage the studies, filtering the irrelevant studies and selecting the pilot studies to be evaluated.

2.1 Defining a research question

This paper aims to identify automated testing tools for mobile applications and defining a research question is the initial step. The research question will be the basis for the search strategy and the selection of the pilot studies to be evaluated.

2.2 Planning a search strategy

The initial step in planning a search strategy is selecting the input data source. In this paper, ACM Digital Library and Google Scholar will be used as sources for the relevant studies. ACM Digital Library and Google Scholar have been chosen as the sources because these are the most comprehensive database of full-text articles covering computing and information technology. The second step in our search strategy is to construct a query based on the research question. Keywords should be chosen carefully to maintain the proper balance between specificity and generality.

2.3 Managing the studies

After running the query in the ACM Digital Library and Google Scholar, peer-reviewed journals will be obtained. But there is a need for each of the study to be assessed for its actual relevance through inclusion criteria. Table 1 shows the inclusion criteria.

Table 1: Inclusion Criteria

No.	Criterion	Description
1	It should be written in English.	There are some studies that are written in other language. They have provided English title and abstract so these papers will show up in the search results. Only

		studies written in English will be included.
2	It should be peer-reviewed.	To ensure the quality of this systematic literature review, only peer-reviewed studies will be included.
3	The publication date must not be earlier than 2013.	To ensure that only up-to-date energy-efficiency solutions are included, only studies that were published in the year 2013 onwards are selected.

To furtherly filter the researches and articles, abstract and conclusion of each study are carefully examined. After selecting the pilot studies to be evaluated, the studies will be ordered and arranged according to these three areas in which automated testing tools for mobile applications are trying to address [33].

2.3.1 Performance and Reliability

Performance is a very critical area where mobile applications should be tested [30]. When testing the performance, it is usually measured in the following categories: (1) Device Performance, (2) Server/API Performance, and (3) Network Performance.

Zhang et. Al. [30], furtherly subcategorized the three (3) areas. For the device performance:

(1) Application startup – this is the amount of time being spent by the app when loading for the first time

(2) Battery usage – the amount of battery consumed by the application while running

(3) Memory consumption – the amount of memory being consumed by the application while running

(4) Hardware/Software variations – the application should be running smoothly regardless of hardware and software conditions

(5) App in background – when the application is put in background and back to foreground the same state should be retrieved

For Server/API Performance:

(1) Data transfer from and to the server – the application must handle data transfers efficiently

(2) Number of API calls – the less the number of API calls in the application is better

(3) Server Downtime – the application must be able to use a locally native database when server is unreachable

For Network Performance:

(1) Jitters – in case of network jitters, the application must handle the scrambled data efficiently

(2) Packet loss – in case of a complete packet loss, the app should be able to resend the request

(3) Network speed – the app should be able to run even
the network speed is slow

2.3.2 Function and Behavior

Function and behavior testing is an essential part of mobile applications testing. Functionality testing is used to validate the display of the application's content. Applications content can be images, texts and graphics. It also verifies the behavior of the application under all circumstances.

2.3.3 Security

Concerning the privacy of personal and business information stored on mobile devices, Security testing include inspection of inscription/decryption techniques used for sensitive data. This testing also checks the protection against security attacks that may come from several means of communication like SMS, MMS, WiFi, and Bluetooth, or from exploited software vulnerabilities from both the web browser and operating system.

3 RESULTS AND DISCUSSION

This section will discuss the results of each step in the SLR methodology and later part will discuss the selected pilot studies according to the three areas of categorization.

3.1 Research question defined

This paper aims to answer the following question: What are the automated testing tools for mobile applications?

3.2 Results of the search strategy

Keywords were constructed from the research question. These keywords will be used in the search query in ACM Digital Library. The following search query will be used: "*automated testing tool for mobile application*". Table 2 shows the number of search results per source:

Table 2: Number of search results per source

Search query	Number of results (ACM Digital Library)	Number of results (Google Scholar)
automated testing tool for mobile application	411,941	951,000

3.3 Managing the studies

The search result for the first query has been furtherly refined by publication year (>= 2013). Table 3 shows the number of search results for the given query.

Table 3: Search result for the query with publication year is not earlier than 2013

Search query	Number of results (ACM Digital Library)	Number of results (Google Scholar)
automated testing tool for mobile application	79,645	16,300

To furtherly filter the results, advanced search feature has been used. The first where clause will be on the Title field that matches all (compared to matches any from the previous query) of the following words or phrases: "automated testing tool for mobile application". The next where clause will be on the field of Publication Year, this is set to on or after (>=) 2015. The full query syntax is as follows:

"query": { acmdlTitle:(+ automated + testing + tool + mobile + application*) }*

"filter": {"publicationYear":{ "gte":2013 }}, {owners.owner=HOSTED}

The above query resulted to fewer matches. Table 4 shows the number of search results from the query above.

Table 4: Search result for the query with publication year is not earlier than 2013 and query keywords matching the title

Search query	Number of results (ACM Digital Library)	Number of results (Google Scholar)
automated testing tool for mobile application	186	41

To furtherly filter the results and finally select the pilot studies, abstract and conclusion were read to verify and assess the paper's relevance to the research question. Table 5 shows the final list of pilot studies to be evaluated.

Table 5: Final list of researches with publication year

No.	Research Title	Publication Year
1	MT4A: A No-Programming Test Automation Framework for Android Applications [31]	2016
2	Graph-Aided Directed Testing of Android Applications for Checking Runtime Privacy Behaviours [33]	2016
3	CrashScope: A Practical Tool for Automated Testing of Android Applications [34]	2017
4	Poster: Framework for Automated Power Estimation of Android Applications [36]	2013
5	DroidFuzzer: Fuzzing the Android Apps with Intent-Filter Tag [38]	2017
6	Security Testing of the Communication among Android Applications [39]	2013
7	Cloud-Based Mobile App Testing Framework: Architecture, Implementation and Execution [40]	2014
8	Fully Automated UI Testing System for Large-scale Android Apps Using Multiple Devices [41]	2017
9	Automating UI Tests for Mobile Applications with Formal Gesture Descriptions [42]	2014
10	An Automated Testing Approach for Inter-application Security in Android [43]	2014
11	Systematic Exploration of Android Apps' Events for Automated Testing [44]	2016
12	UX Suite: A Touch Sensor Evaluation Platform [45]	2016

Additionally, the studies were categorized according to the main three areas of categorization. Table 5 shows the categorized studies:

Table 5: Categorized researches

Area	Studies
Performance and reliability	[31][34][36][38][40][42][44][45]

Function and behavior	[31][41][34][38][40]
security	[33][39][43]

The next sections will discuss the existing automated tools for testing mobile applications categorized by: performance and reliability, function and behavior and security.

3.3.1 Performance and Reliability

MT4A is a no programming test automation framework for Android applications created by Coelho, et. A. [31]. The said tool are made for individuals with no programming knowledge to be involved in testing Android applications. The proposed framework is mainly focused on testing all types of sensors for applications that use them. Their paper showed that users can use the framework without any prior knowledge to programming and that they can execute the tests with a similar time compared with people with programming skills. The authors also conducted a sample test case. The app which was tested is a fall detection app. The tool was able to test if the device was really able to detect a fall or not. However, this tool can only run tests on a single device at a time. There is no option to run the same test in multiple devices at the same time. The tool does not also offer a collaborative creation and editing of tests.

The tool presented by Moran, et. Al. [34], aims to better support developers in mobile testing tasks. The automated tool is called CrashScope. This tool explores a given Android app using systematic input generation, according to several strategies informed by static and dynamic analyses, with the intrinsic goal of triggering crashes. When a crash is detected, CrashScope generates an augmented crash report containing screenshots, detailed crash reproduction steps, the captured exception stack trace, and a fully replayable script that automatically reproduces the crash on a target device. In their paper, the tool was able to uncover crashes about as many crashes detected by some paid testing tools. The tool was also able to record startup time of the APK being tested. The tool is also able to test the app contextually (e.g. turning mobile data).

In the paper of Lee and Kim [36], they proposed a framework for automated power estimation for Android apps. The key features of our framework are as follows. First, in contrast to existing testing tools, our framework allows market curator to test apps without source code. Our framework decompiles app installers (.apk files) using APK Tools to obtain manifest.xml. Then, test scenarios are generated which are composed of the android component (activity, service, broadcast receiver, and content provider) and permission data by parsing manifest.xml file. Energy consumption is estimated while running apps with the scenarios. The process is performed as a black-box testing with the generated scenario and Monkey, which comes with the Android SDK. Second, our framework enables market curator to automate estimation of application power consumption. Energy consumption is measured from the current sensor on the phone or computed from the pre-generated power on/off).

3.3.2 Function and Behavior

The CrashScope [34] tool which was also discussed above, has a GUI ripping engine which enables the tool to explore an app. It is able to identify GUI or screen hierarchy, clickable, long-clickable components of the app and text entry controls. Text entry from the user is a major part of functionality in many Android apps, the GUI Ripping Engine employs a unique text input generation mechanism. It detects the type of text expected (e.g., phone number, email address, age) by a text field, by querying the keyboard type associated with the text field using the "adb shell dumpsys input_method" command. Once the type of expected input is detected, it employs two heuristics to generate text inputs: expected and unexpected. The expected heuristic generates a string within keyboard parameters without any punctuation or special characters, whereas the unexpected heuristic generates random strings with all of the allowable special characters for a given keyboard type.

In the paper of Ye, et. Al., they proposed a fuzz testing method to do the black box test of Android apps and implemented an automated tool called DroidFuzzer. The core of the tool is the variation module which takes the normal data as the input and generates abnormal data as test cases. Then the test cases are sent to the target app and the process is monitored for bug information. Typical MIME data types are used to generate test cases.

Villanes Rojas, et. Al. [40] proposed a cloud-based mobile app testing framework called AM-TaaS. The AM-TaaS framework is focused on the use of emulated devices due to the high cost to acquire real devices and to perform the test on many devices and automated test scripts as a way to help on the time of execution of each test. The tool aims to test if the defined functions of the app are working correctly as intended.

When compared to other testing tools discussed above, AutoClicker which was presented by Ki, et. Al. [41], is the most suited for large-scale Android app testing on multiple devices. AutoClicker, a fully automated UI testing system for large-scale Android apps using multiple devices. It provides a way to quickly and easily verify that a large number of Android apps behave correctly at runtime in a repeatable manner.

While other mobile testing tools were focused on touch and press simulations, Hesenius, et. Al. [42] proposed an extension to the Calabash testing framework allowing for test automation for gesture-based mobile applications.

3.3.3 Security

The paper of Keng, et. Al. [33], they presented MAMBA, a directed testing system for checking privacy in Android apps. MAMBA performs path searches of user events in control-flow graphs of callbacks generated from static analysis of app bytecode. Based on the paths found, it builds test cases comprised of user events that can trigger the executions of the apps and quickly direct the apps' activity transitions from the starting activity towards target activities of interest, revealing potential accesses to privacy-sensitive data in the apps. MAMBA's backend testing engine then simulates the executions of the apps following the generated test cases to check actual runtime behavior of the apps that may leak users' private data. By instrumenting privacy access/leak detectors during testing, the tool was able to verify from test logs that almost half of target activities accessed user privacy data, and 26.7% of target activities leaked privacy data to the network.

In the paper of Avancini and Ceccato [39], they presented a novel approach to test Android applications. An automatic test case generation is proposed to spot mismatches between the intended behavior declared by an application and the observed functionalities implemented in its code. Their approach managed to detect mismatches in three real world and largely used Android applications and the tool automatically generate JUnit test cases to reproduce potential bugs. However, this tool only supports Eclipse-created projects.

While other papers proposed tools to check if a mobile application is secured, Guo, et. Al. [43] introduced an inter-application security mechanism in Intent category components (Activity, Broadcast Receiver and Service) and Content Provider. A novel approach is presented, including a detection module and a checking module, to detect the vulnerabilities existing in Android inter-application components.

4 CONCLUSION

In this paper, several automated testing tools for mobile applications were discussed. The papers were subcategorized by (1) Performance and Reliability, (2) Function and Behavior, and (3) Security. Most of the papers found were addressing the test cases for performance and reliability. While five (5) papers were addressing test cases for app's function and behavior. The least number of papers aims to test security loopholes on mobile applications. It is worth noting that the papers focused on applications running on Android devices, there were no tools that address applications for other platforms such as iOS and Windows Phone.

REFERENCES

[1] Nagowah, L., & Sowamber, G. (2012, June). A novel approach of automation testing on mobile devices. In Computer & Information Science (ICCIS), 2012 International Conference on (Vol. 2, pp. 924-930). IEEE.

[2] Froehlich, J., Dillahunt, T., Klasnja, P., Mankoff, J., Consolvo, S., Harrison, B., & Landay, J. A. (2009, April). UbiGreen: investigating a mobile tool for tracking and supporting green transportation habits. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1043-1052). ACM.

[3] Rayle, L., Shaheen, S., Chan, N., Dai, D., & Cervero, R. (2014). App-based, on-demand ride services: Comparing taxi and ridesourcing trips and user characteristics in san francisco university of california transportation center (uctc). University of California, Berkeley, United States Rogers, B.(2015) The social costs of Uber. James E. Beasley School of Law, Temple University, Philadelphia, United States.

[4] Wang, T., Cardone, G., Corradi, A., Torresani, L., & Campbell, A. T. (2012, February). WalkSafe: a pedestrian safety app for mobile phone users who walk and talk while crossing roads. In Proceedings of the twelfth workshop on mobile computing systems & applications (p. 5). ACM.

[5] Gavin, M., Ghosh, B., Pakrashi, V., Barton, J., O'Flynn, B., & Lawson, A. (2011). A cycle route planner mobile-app for Dublin city. In Irish Transportation Research Network Conference (ITRN2011), 31 Aug-1 Sep 2011, University College Cork, Cork, Ireland.. Irish Transportation Research Network.

[6] Stockx, T., Hecht, B., & Schöning, J. (2014, November). SubwayPS: towards smartphone positioning in underground public transportation systems. In Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (pp. 93-102). ACM.

[7] Vieira, V., Salgado, A. C., Tedesco, P., Times, V., Ferraz, C., Huzita, E., ... & Steinmacher, I. (2012). The UbiBus project: Using context and ubiquitous computing to build advanced public transportation systems to support bus passengers. Anais do VIII Simpósio Brasileiro de Sistemas de Informação, 7.

[8] Navarro, K. F., Gay, V., Golliard, L., Johnston, B., Leijdekkers, P., Vaughan, E., ... & Williams, M. A. (2013, October). SocialCycle what can a mobile app do to encourage cycling?. In Local Computer Networks Workshops (LCN Workshops), 2013 IEEE 38th Conference on (pp. 24-30). IEEE.

[9] Broll, G., Cao, H., Ebben, P., Holleis, P., Jacobs, K., Koolwaaij, J., ... & Souville, B. (2012, December). Tripzoom: an app to improve your mobility behavior. In

Proceedings of the 11th international conference on mobile and ubiquitous multimedia (p. 57). ACM.

[10] Boulos, M. N. K., Brewer, A. C., Karimkhani, C., Buller, D. B., & Dellavalle, R. P. (2014). Mobile medical and health apps: state of the art, concerns, regulatory control and certification. Online journal of public health informatics, 5(3), 229.

[11] Turner-McGrievy, G. M., Beets, M. W., Moore, J. B., Kaczynski, A. T., Barr-Anderson, D. J., & Tate, D. F. (2013). Comparison of traditional versus mobile app self-monitoring of physical activity and dietary intake among overweight adults participating in an mHealth weight loss program. Journal of the American Medical Informatics Association, 20(3), 513-518.

[12] Akinyele, J. A., Pagano, M. W., Green, M. D., Lehmann, C. U., Peterson, Z. N., & Rubin, A. D. (2011, October). Securing electronic medical records using attribute-based encryption on mobile devices. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices (pp. 75-86). ACM.

[13] Aungst, T. D. (2013). Medical applications for pharmacists using mobile devices. Annals of Pharmacotherapy, 47(7-8), 1088-1095.

[14] Payne, K. F. B., Wharrad, H., & Watts, K. (2012). Smartphone and medical related App use among medical students and junior doctors in the United Kingdom (UK): a regional survey. BMC medical informatics and decision making, 12(1), 121.

[15] Aungst, T. D., Clauson, K. A., Misra, S., Lewis, T. L., & Husain, I. (2014). How to identify, assess and utilise mobile medical applications in clinical practice. International journal of clinical practice, 68(2), 155-162.

[16] Krebs, P., & Duncan, D. T. (2015). Health app use among US mobile phone owners: a national survey. JMIR mHealth and uHealth, 3(4).

[17] Semple, J. L., Sharpe, S., Murnaghan, M. L., Theodoropoulos, J., & Metcalfe, K. A. (2015). Using a mobile app for monitoring post-operative quality of recovery of patients at home: a feasibility study. JMIR mHealth and uHealth, 3(1).

[18] de Reuver, M., Verschuur, E., Nikayin, F., Cerpa, N., & Bouwman, H. (2015). Collective action for mobile payment platforms: A case study on collaboration issues between banks and telecom operators. Electronic Commerce Research and Applications, 14(5), 331-344.

[19] Bons, R. W., Alt, R., Lee, H. G., & Weber, B. (2012). Banking in the Internet and mobile era. Electronic Markets, 22(4), 197-202.

[20] Jones, W. (2014). M-commerce: Building the opportunity for banks. Journal of Payments Strategy & Systems, 8(3), 300-306.

[21] Chen, B., & Denoyelles, A. (2013). Exploring students’ mobile learning practices in higher education. Educause Review, 7.

[22] Shuler, C. (2009). iLearn: A content analysis of the iTunes app Store's education section. onference P, 149.

[23] Rossing, J. P., Miller, W. M., Cecil, A. K., & Stamper, S. E. (2012). iLearning: The future of higher education? Student perceptions on learning with mobile tablets. Journal of the Scholarship of Teaching and Learning, 12(2), 1-26.

[24] Vázquez-Cano, E. (2014). Mobile distance learning with smartphones and apps in higher education. Educational Sciences: Theory and Practice, 14(4), 1505-1520.

[25] Mehdipour, Y., & Zerehkafi, H. (2013). Mobile learning for education: Benefits and challenges. International Journal of Computational Engineering Research, 3(6), 93-101.

[26] Hsu, Y. C., Rice, K., & Dawley, L. (2012). Empowering educators with Google's Android App Inventor: An online workshop in mobile app design. British Journal of Educational Technology, 43(1).

[27] Tassey, G. (2002). The economic impacts of inadequate infrastructure for software testing. National Institute of Standards and Technology, RTI Project, 7007(011).

[28] Gunasekaran, S., & Bargavi, V. (2015). Survey on automation testing tools for mobile applications. International Journal of Advanced Engineering Research and Science, 2(11), 2349-6495.

[29] Wasserman, A. I. (2010, November). Software engineering issues for mobile application development. In Proceedings of the FSE/SDP workshop on Future of software engineering research (pp. 397-400). ACM.

[30] Zhang, Dongsong, and Boonlit Adipat. "Challenges, methodologies, and issues in the usability testing of mobile applications." International journal of human-computer interaction 18.3 (2005): 293-308.

[31] Coelho, T., Lima, B., & Faria, J. P. (2016, November). MT4A: a no-programming test automation framework for Android applications. In Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation (pp. 59-65). ACM.

[32] Holzmann, C., & Hutflesz, P. (2014, December). Multivariate Testing of Native Mobile Applications. In Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia (pp. 85-94). ACM.

[33] Keng, J. C. J., Jiang, L., Wee, T. K., & Balan, R. K. (2016, May). Graph-aided directed testing of android applications for checking runtime privacy behaviours. In Automation of Software Test (AST), 2016 IEEE/ACM 11th International Workshop in (pp. 57-63). IEEE.

[34] Moran, K., Linares-Vásquez, M., Bernal-Cárdenas, C., Vendome, C., & Poshyvanyk, D. (2017, May). Crashscope: A practical tool for automated testing of android applications. In Software Engineering Companion (ICSE-C), 2017 IEEE/ACM 39th International Conference on (pp. 15-18). IEEE.

[35] Coles, H., Laurent, T., Henard, C., Papadakis, M., & Ventresque, A. (2016, July). PIT: a practical mutation testing tool for java. In Proceedings of the 25th International Symposium on Software Testing and Analysis (pp. 449-452). ACM.

[36] Lee, J., & Kim, H. (2013, June). Framework for automated power estimation of android applications. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services (pp. 541-542). ACM.

[37] de Cleva Farto, G., & Endo, A. T. (2017, September). Reuse of model-based tests in mobile apps. In Proceedings of the 31st Brazilian Symposium on Software Engineering (pp. 184-193). ACM.

[38] Ye, H., Cheng, S., Zhang, L., & Jiang, F. (2013, December). Droidfuzzer: Fuzzing the android apps with intent-filter tag. In Proceedings of International Conference on Advances in Mobile Computing & Multimedia (p. 68). ACM.

[39] Avancini, A., & Ceccato, M. (2013, May). Security testing of the communication among Android applications. In Proceedings of the 8th International Workshop on Automation of Software Test (pp. 57-63). IEEE Press.

[40] Rojas, I. K. V., Meireles, S., & Dias-Neto, A. C. (2016, September). Cloud-Based Mobile App Testing Framework: Architecture, Implementation and Execution. In Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing (p. 10). ACM.

[41] Ki, T., Simeonov, A., Park, C. M., Dantu, K., Ko, S. Y., & Ziarek, L. (2017, June). Fully Automated UI Testing System for Large-scale Android Apps Using Multiple Devices. In Proceedings of the 15th Annual International Conference on Mobile Systems, Applications, and Services (pp. 185-185). ACM.

[42] Hesenius, M., Griebe, T., Gries, S., & Gruhn, V. (2014, September). Automating UI tests for mobile applications with formal gesture descriptions. In Proceedings of the 16th international conference on Human-computer interaction with mobile devices & services (pp. 213-222). ACM.

[43] Guo, C., Xu, J., Yang, H., Zeng, Y., & Xing, S. (2014, May). An automated testing approach for inter-application security in Android. In Proceedings of the 9th

International Workshop on Automation of Software Test (pp. 8-14). ACM.

[44] Salihu, I. A., & Ibrahim, R. (2016, November). Systematic Exploration of Android Apps' Events for Automated Testing. In Proceedings of the 14th International Conference on Advances in Mobile Computing and Multi Media (pp. 50-54). ACM.