A Comparative Analysis of Quality Assurance of Mobile Applications using Automated Testing Tools

Haneen Anjum
Department of CS & SE,
International Islamic University,
Islamabad, Pakistan

Maham Khan
Department of CS & SE,
International Islamic University,
Islamabad, Pakistan

Zainab Shahid
Department of CS & SE,
International Islamic University,
Islamabad, Pakistan

Muhammad Imran Babar
Army Public College of
Management & Sciences, Pakistan
Corresponding Author

Saima Chaudhry
Department of CS & SE,
International Islamic University,
Islamabad, Pakistan

Furkh Zeshan
Department of CS, COMSATS,
Lahore,
Pakistan

Muhammad Jehanzeb
Army Public College of
Management & Sciences,
Pakistan

Summiyah Sultana
Department of CS & SE,
International Islamic University,
Islamabad, Pakistan

Shahid Nazir Bhatti
Department of SE, Bahria
University, Islamabad,
Pakistan

***Abstract*—Use of mobile applications are trending these days due to adoption of handheld mobile devices with operating systems such as Android, iOS and Windows. Delivering quality mobile apps is as important as in any other web or desktop application. Simplification and ease of quality assurance or evaluation in mobile devices is achieved by using automated testing tools. These tools have been evaluated for their features, platforms, code coverage, and efficiency. However, they have not been evaluated and compared to each other for different quality attributes they can enhance in the apps under test. This research study aims to evaluate different testing tools focusing on identifying quality factors they aid to achieve in the apps under test. Furthermore, it aims to measure overall trends of essential quality factors achieved using automated testing tools. The findings of this study are beneficial to the practitioners and researchers. The practitioners need to look up for specific tools which aid them to assure the desired quality factors in the apps under test. The researchers may base their studies on the findings of this study to propose solutions or revise existing tools in order to achieve maximum number of critical quality attributes in the app under test. This study revealed that the trend of automated testing is high on usability, correctness and robustness. Moreover, the trend is average on testability and performance. However, for assurance of extensibility, maintainability, scalability, and platform compatibility, only a few tools are available.**

Keywords*—Mobile application; quality assurance; automated testing; testing tools

I. INTRODUCTION

Software testing enables the software testers to detect defects in the software and remove them to ultimately achieve improved software quality. Recently software testing became wide-spread and critical among software development companies. Software testing can be performed either manually or automatically. Manual testing is to manually write the test cases and executing them without using any tool. In manual testing a tester performs the testing through carefully navigating through the different interfaces of the system under test, testing with different values of inputs, recording and comparing the observed results with the expected results of the tests.

Automated testing is done with the help of an automated testing tool. The automated testing tool provides a computer-controlled testing rather than manually. The testing tool executes the test cases to test the performance and functionality of the software under test. The aim of automated testing is to reduce the required human effort as in manual testing but it does not remove the need of manual testing at all [1]. Mobile platforms are being adopted worldwide because of a variety of software being offered to users in those handheld and portable devices. Testing is being used as a quality assurance technique for mobile apps too [2].

Several tools are proposed and implemented for this purpose. These tools have already been already been evaluated and compared for their unique features, supported platforms, code coverage, and efficiency. However, existing automated testing tools of mobile applications have not been evaluated and compared for different quality attributes they can enhance in apps under test. Therefore, two research objectives are formulated for this study that is: 1) to evaluate different testing tools of mobile apps focusing on identifying quality factors they aid to achieve in the apps under test; 2) to measure overall trends of essential quality factors achieved in the mobile apps under test using automated testing tools. In this paper, we have evaluated and compared automated testing tools for adding or enhancing valuable quality factors in mobile applications under test. The findings and result of this study are beneficial to the

practitioners as well as the researchers. The list of quality factors to be achieved varies among apps. The testing of different apps requires selection of different tools. Therefore, the practitioners may need to look up for tools which aid them to assure the desired quality factors in a particular App Under Test (AUT). The researchers who are interested in proposing the tools and techniques for testing of mobile apps may need to consider the quality factors highlighted in this study. Moreover, they can begin their own research study on the basis of these tools to propose merged, revised and enhanced solutions for achieving the maximum number of quality attributes in the AUT.

The rest of this paper is structured as follows: Section 2 gives a comprehensive knowledge about the background concepts of manual and automated software testing. Section 3 describes methodology that we used to achieve our research objectives. Section 4 presents description of a number of automated testing tools for mobile applications. Section 5 presents comparative study. Section 6 presents our findings and discussion. Finally, Section 7 concludes the paper.

II. BACKGROUND

Success of any software product is determined by the quality of that software. This gives software quality assurance a great opportunity in software industry and customer satisfaction drives it. To develop a product of good quality and without any defects within the cost and time constraints have become critical. Implementing such products, with minimum or no bugs is a difficult task. This is the reason that the concept of software testing has got its existence [3]. In software industry, testing of software has become an extensive and vital phase of SDLC. It also provides final evaluation of other activities such as requirements specification, software design, and coding [4].

Software testing is an activity, which is performed to evaluate correctness and functionality of software for assuring fulfillment of user requirements and expected quality [5]. IEEE defines software testing as the process to evaluate the system or its components manually or by automated means to determine whether it fulfills the user requirements or to find the difference among actual result and expected result [6]. Hence, the software testing is to execute a software to identify defects or any missing features that were expected by the user requirements. Software testing results in improved quality and effectiveness of the software system, if it is executed appropriately. Detecting the defects in a software and removing those defects before the release of software leads to reduced maintenance cost.

All the activities of software testing can be conducted by two means: automated testing and manual testing. Manual testing is the fundamental software testing. It is conducted manually through moving about in the software application. A test plan or test cases are followed for manual testing. Test cases describe the complete test scenario in terms of actions to be performed during testing. On the other hand, in automated testing, the testing is conducted through some testing tool without the navigating through the different parts of the application manually.

Initially, manual testing was only performed. Because of human error, few defects may be ignored or unidentified through manual testing. So, through manual testing better quality of a software system cannot be ensured. To overcome this lack in manual testing, automated testing has evolved. The automated testing is helpful in quicker testing process. Recently automated testing got more attention and many testers prefer to use automated testing for the variety of software systems [7]. The basic element behind automated testing is the automated testing tool that is used to conduct the tests.

A. Software Testing

Normally software testing is considered as an activity for detection of defects whereas there are different reasons behind conduction of software testing. Improved software quality is one of the major reasons. Software quality is improved by ensuring that the software product fulfils the user requirements and expectations. Smooth functioning of the software system can be ensured through testing. The software developing industries spend most of their time and cost on software testing during the SDLC [8]. If the testing is done early in the SDLC to prevent the occurrence of defects, it reduces the time and cost spent whereas, if the defects are detected in later stages, then the time to market and cost rises significantly. Therefore, performing testing throughout the SDLC is a better practice to detect the defects of the software. It is less expensive to remove the defects earlier, even before the release of the software [9].

Software testing aims to evaluate the capabilities of an application or the software and verify that it fulfils the quality principles such as reliability, portability, efficiency, security, usability, etc. Through testing all these principles should also be verified and ensured [10]. There are two main objectives of software testing. First, the detection of errors or defects. Second, preventing the number of occurrences of defects in the software system, that results in overall improved efficiency of the system.

B. Manual Software Testing

Manual testing is the simplest level of testing in which the tests are executed as per test cases and by directly interacting with the software. In this testing, the tester prepares the test cases. Test cases, are the explanations of the features and the expected results of the software under test, and are written in simple natural language. The process of manual testing becomes too much time-taking as it requires all the activities to be performed manually. Though, manual testing is preferred in case of some complex systems where a few critical defects can only be discovered while testing manually. During manual testing the tester interacts with the system under test as the end user of that software would, and ensures the effectiveness of the system by navigating through the software [11]. Manual testing have the following drawbacks [12]:

- Time-taking
- Requires more testers
- Less accurate results
- Testing multiple features in parallel, not possible
- Lack of reusability of tests

- Lack of test completeness.

C. Automated Software Testing

As the automated software testing got popular in software industries, the testing process become more effective. Automated software testing helps in easily executing various tests like performance testing and regression testing. The difficult testing activities got easier than before, as the automated testing evolved and improved, because it conducts the test for various datasets and the tests can be executed repeatedly without human involvement [1]. Automated software testing requires a little primary investment for the software but that doesn't have much economical effect as it results in reduced human efforts required for testing [13]. The automated software testing can be performed in various phases: preparation of test plan or developing the test cases, selecting the testing tool, creation of the test script and finally executing the test by using the automated testing tool and the script.

The main objective of automating software testing is to reduce the testing effort, time and cost. Testing automation results in improved efficiency, whereas reduction in human involvement in testing process. Automated testing supports the reusability of test scripts, using the testing tool, for different upgrades of the system under test [1]. Automated software testing simplifies the testing process and results in reduced maintenance cost of the software [10]. Automated testing has the following benefits [7]:

- Simplified regression testing
- Tests are repeatable and reusable
- Reduces time and cost
- Performance testing is possible due to simultaneous testing.

Automated testing has the following drawbacks [12]:

- It is more expensive
- All areas cannot be automated
- Manual testing cannot be fully discarded.

D. Manual vs Automated Software Testing

Table 1 illustrates the differences between manual and automated software testing [1], [7], [12].

III. Methodology

For mobile applications, nine essential software quality factors, as described in Table 2, are selected. These factors are the most significant quality attributes not only in software and web based applications, but also the mobile apps must conform to these quality requirements. Firstly, all industry-dominant and proposed mobile apps testing tools are identified from existing literature from 2010 to 2017. Secondly, each of these tools is studied in order to extract its features. Thirdly, for each tool, the quality factors it may aid to achieve in AUT are derived on the basis of its features and characteristics. All the derived and implied quality factors for each tool form a subset of the set of factors mentioned in Table 2. The tools are compared on the basis of their quality factors in Section 5. Moreover, for each tool, the derivation of the quality factors is also justified based on its features and characteristics. The summarized results of this comparative study are presented graphically in Section 6 to show an overall trend of quality factors achieved using automated testing.

TABLE. I. Differences Between Manual and Automated Software Testing

Manual Testing	Automated Testing
1. Time Consuming	Time Efficient
2. More human effort is required.	One-time human effort for creating the test scripts is enough.
3. Not accurate, due to room for human errors	More accuracy as less space for human error
4. Test cases cannot be reused	Supports reusability of test cases
5. More effective for functional testing and exploratory testing	Effective for regression testing, load testing & performance testing
6. Reduced short term cost (no automated testing tool is required) while increased long term cost (maintenance).	Increased short term cost (automated testing tool) while reduced long term cost (maintenance).

TABLE. II. Software Quality Factors for Comparative Analysis

Software Quality Factors	Description
Extensibility	Ability of software components to be added, modified and removed easily without badly effecting existing system. Flexibility is its category focused on ability of components to be added easily.
Maintainability	Maintainability is ability to make change for error corrections, supported by defined interfaces, documentations, comments in code.
Performance	Performance is related to acceptable response time.
Scalability	Ability to respond in an acceptable time in increased load or stress.
Robustness	Robustness is the ability of software to keep working and remain available in failure states by backup plans, data and hardware.
Usability	Usability is the ability of user to easily interact with the system using the user interface.
Platform compatibility	Software should run on several platforms like operating systems, browsers etc.
Testability	Testability refers to maximum and efficient code coverage by testing.
Correctness	Correctness is software should conform to with requirements or specifications.

IV. AUTOMATED TESTING TOOLS FOR MOBILE APPLICATIONS

In Software Development, Mobile Applications Development is a prominent area which is emerging rapidly. Therefore, testing also becomes significant in this area. Many tools are available for supporting different types and levels of testing in platforms like Android and iOS [14]. Following are some noteworthy tools that are being used in Software industry for their strong testing support for mobile apps. Robotium is one of the UI automation frameworks used for android systems. It is available free of cost in the market and can be used by enterprises and individuals as well. It assists the test case developers in writing functional, acceptance and system test scenarios, spanning a range of android activities. It is a Java based tool while JUnit test framework is a part of it as well. It is made to make it easy for test case developers to write robust and powerful automatic black box test cases. This tool cannot be used for Web or Flash apps [14].

Renorex is a testing tool and framework that supports the scriptless way of working and coding capabilities. This tool is mainly used for GUI supports in mobile and web apps. It offers a fast and intuitive way to write test cases as functions used in SUT. It gives some extra ability for creation of a robust regression testing. It supports cross browser testing too. The Renorex studio IDE delivers a feature 'click and go Function' in order to ensure the reusability of test actions and various UI element with the team of technical skill levels [12]. Appium is another cross-platform testing tool that allows test case developers to write test for multiple platforms such as iOS and android, using a single API. It enables code to be reused among iOS and android test suites. It is an open source tool used for web app and hybrid application of automating native mobile on both the iOS and android platforms, where the native apps can be written using android SDK or iOS [14].

MonkeyTalk is an open source tool used for functional testing. It is simple to use and powerful tool for testing mobile applications. This tool works with a range of real devices and emulators. It tests from a simple 'smoke test' to the sophisticated test suites such as data driven test suites. The tests are created for iOS and android if the parameterized tests are used. MonkeyTalk IDE is an eclipse based tool for recording, playing, editing and managing the functional test suites for iOS and Android applications that runs on emulators, simulators and devices [14]. UIAutomator is one of the testing frameworks provided by Google's Android. The tests run by this framework ensures an application to meet the functional requirements and it achieves a fine standard quality so that it can be successfully adopted by android users. It allows to run the tests reliable, fast, and repeatable manner [14].

Reran is a record and replay tool for smartphones that have Android operating system. It captures input event sent from the phone to the OS of a user session and after that allows the sequence of events to be sent into the phone programmatically at high level. Reran captures the low level events and replays them that are triggered on the phone, which allows it to capture and playback GUI events such as touchscreen gestures, and input sensors on device [15]. EvoDroid is used to test system of Android apps. It combines two techniques 1) to identify parts of the code open to be searched independently an android-specific program analysis; 2) an algorithm performs search step by step under the given info. Its main goal is to look for test cases that amplify code coverage [16]. MobiGUITAR models the state of the app's GUI, which helps us more accurately model mobile apps' state-sensitive behaviour. On the basis of state machine, it makes new test adequacy criteria. This test generation technique uses the models and criteria to generate test cases automatically. It delivers fully automatic testing that works on security policies of smartphone platforms [2].

Dynodroid automatically generates inputs to Android apps. It is capable of generating both UI inputs (e.g., touchscreen taps and gestures) and system inputs (e.g., simulating incoming SMS messages). It allows interleaving inputs from machine and human. Through a sequence of events it interacts with its environment. Dynodroid is an observe-select-execute cycle, it observes which events are important to current state, selects those events, and execute those events to make a new state in which it repeats this process [17]. FSMdroid is a guided approach to GUI testing of Android apps. Its basic idea is to 1) construct an initial stochastic model for the app under test; 2) iteratively mutate the stochastic model and derive tests. Compared with the traditional model-based testing approaches, it enhances the diversity of test sequences by 85%, but reduces the number of them by 54%. It first uses static analysis to identify UI events which can be missed during dynamic analysis [18]. Table 3 summarizes general information about above testing tools i.e. their support for testing types or levels, platform. According to Table 3, 90% of the tools support automated testing of Android apps. However, 20% of the tools support testing of the iOS apps.

TABLE. III. AUTOMATED MOBILE APPLICATIONS TESTING TOOL

Testing Tool	Testing Type	Platform
Dynodroid	Event driven testing	Android
Evodroid	System testing	Android
FSM Droid	GUI testing	Android
MobiGUITAR	GUI testing	Android
Renorax	Compatibility testing	C#, Python, VB.net
Reran	GUI, system, stress, and security testing	Android
Robotium	GUI, system, functional, and acceptance testing	Android
Appium	GUI and functional testing	Android, IOS
MonkeyTalk	Compatibility and functional testing	Android, IOS
UIAutomator	Functional and GUI testing	Android

V. Comparative Analysis of Software Testing Tools

The purpose of testing is to ensure that software meets its functional requirements and it is of desired or standard quality so that it is accepted and adopted by the user for its intended use [14]. Aforementioned tools are proficient in one or more from functional testing, system testing, code coverage and user interface testing, etc. of mobile applications. This section presents their comparative analysis on the basis of quality factors from Table 2 they test and thus enhance in mobile apps under test.

Dynodroid smartly plays the role of user of mobile app under test by generating input events automatically [17], thus giving an illusion of actual interaction of user with the application in expected environment. For each auto generated user event, this tool observes reaction of the application to further generate next possible event that could be performed by the user [17]. This proficiency makes Dynodroid fit for evaluating mobile applications for their usability. The test reports can help front-end developers to improve usability by reshaping the possible interaction with user while still fulfilling his needs. Furthermore, it allows tester's intervention at any stage for entering relevant and intelligent input in any sequence of events [17] to evaluate correctness of application. Results of these customized tests reveal the level of correctness achieved in application so that further conformance to requirements can be achieved. Studies have proved that this tool also finds bugs [17] that may crash the application, which are corrected by developers. Thus, this tool contributes to reliability and robustness of solution just tested. If the promised percentage of source code is covered under tests [17], then it shows that the testability of software is achieved. If there is less source code coverage, then the application has not attained the quality factor of testability.

Evodroid aims to perform system testing of mobile applications [16]. System testing exercises application for its overall behaviour to check correctness, so as to state that application fits for its intended user. It offers much higher code coverage [16] which can easily evaluate testability in an application. Despite of higher coverage of code being offered, if not a good percentage of code is being covered, then the application's design must be modified to reduce testing effort. So, in complicated solutions, other quality factors like correctness, robustness, maintainability, etc. can be evaluated after deployment also. Evodroid effectively provides features of deploying, maintaining, and enhancing mobile applications [16]. Thus, it adds to correctness, flexibility, and maintainability of apps by following its methods and tips of utilizing these features. FSMDroid focuses on Graphical User Interface (GUI) testing [18]. GUI is the interaction point between user and system. When GUI is tested for prompting input, displaying output and scenarios of erroneous inputs from user, it ultimately gives good evaluation of usability and accessibility of application's features under test. It also evaluates testability as it also offers high coverage of code [18]. It also reveals fatal bugs in code [18], which must be solved with proper handling of exceptional error scenarios in code. In this way, it contributes to robustness of application under test. It helps to make GUI models which consume minimum events [18], thus improving performance of application by avoiding duplication and complex GUI events sequences.

MobiGUITAR helps to model state of an application's GUI to test behaviour at a particular sensitive state of GUI [2]. This feature lets testers monitor correctness of an application by mapping response or behaviour with GUI events and states. This tool is proficient in finding concurrency error [2] that may lead to severe concurrency issues, fatal errors, and crashes. It highlights other logical errors [2] too. All these errors are fixed for achieving robustness, fault tolerance and reliability in application being tested. It adds to testability also by its acceptable code coverage [2]. Renorax performs platform compatibility testing [12] which assures that the software is of good quality in terms of its diverse usage on a variety of famous platforms and configurations like operating systems, browsers, web programming languages, etc. It also adds features for supporting further addition and enhancement [12] thus adding flexibility factor for easy maintenance and updates of the software.

Reran tests applications which take user inputs from device sensors and sophisticated GUI operations [15] like zoom, tap, swipe, etc. Reran evaluates the usability of application with all complex application and system level events from rich controls of GUI and sensors. Such application should perform with greater accuracy and precision of time [15] due to sudden inputs from sensors. Reran evaluates the performance and efficiency of application by its strong testing support. Bugs indicated during debugging [15] and test results are corrected by developers which ultimately adds to correctness, performance and robustness of the application under test. It also performs stress testing [15] which evaluates the scalability of application for achieving optimum quality under stress or load conditions. It also catches security related bugs caught after invalid user inputs or malicious plugins [15] to give clues to developers to not leave any vulnerability and make the app and its data secure.

Robotium also supports a good evaluation of usability by performing tests on rich GUI controls of a touch screen mobile device [14]. Test results of function, system and acceptance testing [14] on Robotium allows developers to improve correctness and performance of applications to an optimum level. Appium focuses on testing interaction of user with the content of mobile web applications. Automated test cases are configurable with Safari and Chrome web browsers [14]. Test results are used to evaluate correctness and user experience with the mobile web application in terms of usability or accessibility of the features.

If an application is platform independent or cross-platform, it means it is applicable for a diverse use on different operating systems. It is a plus point to check quality factor of platform compatibility. MonkeyTalk serves this purpose to perform tests for mobile application's compatibility with iOS and Android by offering cross platform testing [14]. It creates test scripts to perform functionality tests against action of each user interface event or command [14]. UIAutomator ensures that mobile app under test is a quality app considering factors of correctness and usability by performing UI functional testing. It automates user test cases to reflect user experience and correctness of

behaviour against input entry and events in asynchronous GUIs like dialogs, alerts, etc. also. This tool is proficient in automating functional UI tests even on two or more devices. [14]

VI. Findings and Discussions

According to Table 4, Evodroid and Renorax aid to achieve quality factors 'extensibility' and 'maintainability'. 'Performance' of the AUT can be enhanced by using three tools i.e. FSM Droid, Reran, and Robotium. Among all the tools, only Reran aims to achieve 'scalability' of the AUT. 'Robustness' can be achieved by five tools i.e. Dynodroid, Evodroid, FSM Droid, MobiGUITAR and Reran. Many of the tools assure 'usability' of the AUT i.e. Dynodroid, FSM Droid, Reran, Robotium, Appium, and UIAutomator. The 'platform compatibility' testing is supported by only two tools, namely, Renorax and MonkeyTalk. 'Testability' of the app can be verified and enhanced using four tools, namely, Dynodroid, Evodroid, FSM Droid, and MobiGUITAR. Most of the tools, namely, Dynodroid, Evodroid, MobiGUITAR, Reran, Robotium, Appium, MonkeyTalk, and UIAutomator assure the 'correctness' of the AUT.

Fig. 1 presents a graph showing the results of this comparative study. Ten dominant automated testing tools for mobile applications are considered for this study. Each tool focused one or more quality factors to achieve or enhance quality of apps under test. Moreover, Fig. 1 shows an overall trend of quality factors achieved by using automated testing. The quality factor of 'correctness' will be achieved using almost every automated testing tool. 'Usability' is also a major aspect of mobile apps which can be evaluated and achieved by using approximately 60% of the available software testing tools. Approximately 50% of these tools focus on achieving desired or optimum level of 'robustness' of mobile apps. A close to average percentage of testing tools attain quality factors of 'testability' and 'performance' in the app under test.

TABLE. IV. Quality Factors Achieved by Automated Testing Of Mobile Apps

Software Testing Tools	Software Quality Factors								
	Extensibility	Maintainability	Performance	Scalability	Robustness	Usability	Platform compatibility	Testability	Correctness
Dynodroid					✓	✓		✓	✓
Evodroid	✓	✓			✓			✓	✓
FSM Droid			✓		✓	✓		✓	
MobiGUITAR					✓			✓	✓
Renorax	✓	✓					✓		
Reran			✓	✓	✓	✓			✓
Robotium			✓			✓			✓
Appium						✓			✓
MonkeyTalk							✓		✓
UIAutomator						✓			✓

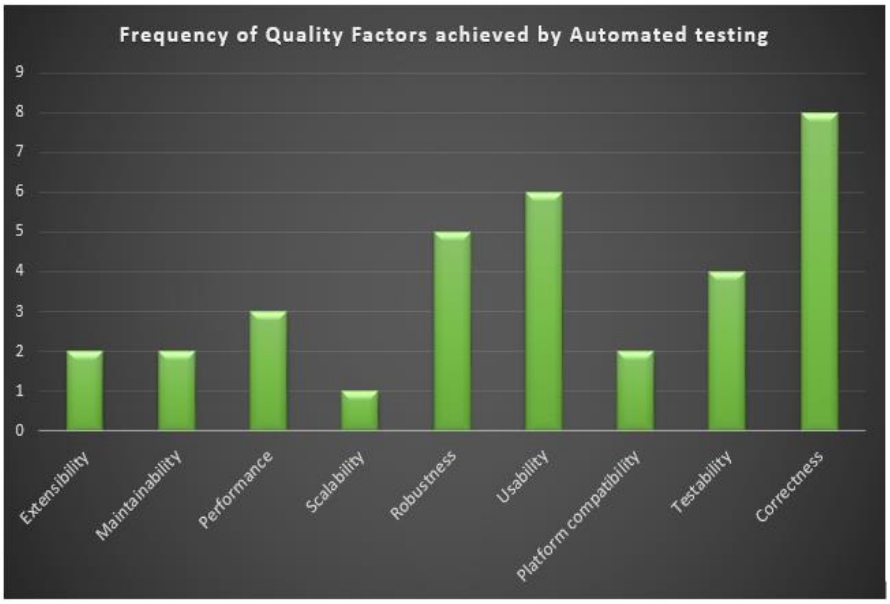

Fig. 1. Frequency of software quality factors achieved using automated testing tools for mobile applications.

A lesser percentage of tools are observed for other important quality attributes like extensibility, maintainability, scalability, and platform compatibility. Reran and Evodroid are better than other tools because they tend to achieve five out of nine quality factors. The rest of the eight tools help to achieve less than five quality attributes in the AUT. Therefore, it is recommendable that for an AUT, more than one tool should be used to assure all the critical quality factors.

There is no automated testing tool or solution for mobile apps which tests for all possible quality factors that are mentioned in Table 2. Most testing tools cover only usability, correctness and robustness, which are desired by almost every mobile app. To support incremental development with testing, and post deployment maintainability and flexibility, only a few tools serve this purpose. Therefore, trend of automated testing is high on usability, correctness and robustness, average on testability and performance, and lesser on extensibility, maintainability, scalability, and platform compatibility.

VII. Conclusion and Future Work

There is no mobile app testing tool which tests for all possible quality factors. Most testing tools cover only usability, correctness and robustness, which are desired by almost every mobile app. To support incremental development with testing, and post deployment maintainability and flexibility, only a few tools serve this purpose. Trend of automated testing is high on usability, correctness and robustness, average on of testability and performance, and lesser on extensibility, maintainability, scalability, and platform compatibility. In automated testing of mobile applications, further research can be done to propose automated mobile apps testing tool that aims to achieve all quality factors mentioned in Table 1. A similar analysis can be made by considering testing tools for other mobile operating systems as well like windows. A comparative analysis can also be done on quality of apps of different mobile operating systems based on automated testing tools of each platform.

Several tools are proposed and implemented for testing of mobile apps. In this research study, these tools are evaluated focusing on identifying the quality factors they aid to achieve in the apps under test. Moreover, overall trends of essential quality factors achieved using automated testing tools are measured. This study revealed that the automated testing provides best support for assurance of usability, correctness and robustness. An average number of tools aid to assure testability and performance. However, for assurance of extensibility, maintainability, scalability, and platform compatibility, only a few tools are available. In automated testing of mobile applications, further research can be done to propose automated mobile apps testing tool which aims to achieve all quality factors mentioned in Table 2. A similar analysis can be made by considering testing tools for other mobile operating systems too, e.g., windows. A comparative analysis can also be done on quality of apps of different mobile operating systems based on automated testing tools of each platform. Moreover, on the basis of the tools identified from this study, revised and enhanced solutions can be proposed for achieving the maximum number of quality attributes in the AUT.

Acknowledgement

Special thanks to International Islamic University, Islamabad Pakistan and Army Public College of Management & Sciences, Rawalpindi, Pakistan for providing support in order to complete this research.

References

[1] P. Rathi and V. Mehra, “Analysis of Automation and Manual Testing Using Software Testing Tool,” 2015.

[2] D. Amalfitano, et al., “MobiGUITAR: Automated model-based testing of mobile apps,” IEEE Software, vol. 32, pp. 53-59, 2015.

[3] X. Wang and G. He, “The research of data-driven testing based on QTP,” in 2014 9th International Conference on Computer Science & Education, 2014.

[4] M. Monier and M. M. El-mahdy, “Evaluation of automated web testing tools,” International Journal of Computer Applications Technology and Research, vol. 4, 2015.

[5] K. M. Mustafa, et al., “Classification of software testing tools based on the software testing methods,” in Proceedings of the International Conference on Computer and Electrical Engineering (ICCEE’09), 2009, pp. 229-233.

[6] G. Saini and K. Rai, "Software Testing Techniques for Test Cases Generation," International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, 2013.

[7] N. Islam, "A Comparative Study of Automated Software Testing Tools," 2016.

[8] D. Shikha and K. Bahl, “Software Testing Tools & Techniques for Web Applications.”

[9] S. Jagannatha, et al., “Comparative Study on Automation Testing using Selenium Testing Framework and QTP,” 2014.

[10] S. Sharma and M. VISHAWJYOTI, “STUDY AND ANALYSIS OF AUTOMATION TESTING TECHNIQUES,” Journal of Global Research in Computer Science, vol. 3, pp. 36-43, 2013.

[11] H. Kaur and G. Gupta, “Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete,” Harpreet Kaur et al Int. Journal of Engineering Research and Applications ISSN, pp. 2248-9622, 2013.

[12] A. Jain, et al., “A Comparison of RANOREX and QTP Automated Testing Tools and their impact on Software Testing,” IJEMS, vol. 1, pp. 8-12, 2014.

[13] T. Xie, “Improving effectiveness of automated software testing in the absence of specifications,” in 2006 22nd IEEE International Conference on Software Maintenance, 2006, pp. 355-359.

[14] S. Gunasekaran and V. Bargavi, “Survey on Automation Testing Tools for Mobile Applications,” International Journal of Advanced Engineering Research and Science (IJAERS), vol. 2, pp. 36-41, 2015.

[15] L. Gomez, et al., “Reran: Timing-and touch-sensitive record and replay for android,” in 2013 35th International Conference on Software Engineering (ICSE), 2013, pp. 72-81.

[16] R. Mahmood, et al., “Evodroid: Segmented evolutionary testing of android apps”, in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014, pp. 599-609.

[17] A. Machiry, et al., “Dynodroid: An input generation system for android apps,” in Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, 2013, pp. 224-234.

[18] T. Su, “FSMdroid: guided GUI testing of android apps,” in Proceedings of the 38th International Conference on Software Engineering Companion, 2016, pp. 689-691.