



Article

The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review

Natnael Gonfa Berihun ^{1,*}, Cyrille Dongmo ¹ and John Andrew Van der Poll ²

¹ Department of Computer Science, School of Computing, Science Campus, University of South Africa, Johannesburg 1709, South Africa; dongmc@unisa.ac.za

² Digital Transformation and Innovation, Graduate School of Business Leadership (SBL), Midrand Campus, University of South Africa, Midrand 1686, South Africa; vdpolja@unisa.ac.za

* Correspondence: 64066428@mylife.unisa.ac.za or natnaelgonfa4@gmail.com

Abstract: Mobile applications are developed and released to the market every day. Due to the intense usage of mobile applications, their quality matters. End users' rejection of mobile apps increases from time to time due to their low quality and lack of proper mobile testing. This indicates that the role of mobile application testing is crucial in the acceptance of a given software product. Test engineers use automation frameworks for testing their mobile applications. Automated testing brings several advantages to the development team. For example, automated checks are used for regression testing, fast execution of test scripts, and providing quick feedback for the development team. A systematic literature review has been used to identify and collect evidence on automated testing frameworks for mobile application testing. A total of 56 relevant research papers were identified that were published in prominent journals and conferences until February 2023. The results were summarized and tabulated to provide insights into the suitability of the existing automation testing framework for mobile application testing. We identified the major test concerns and test challenges in performing mobile automation testing. The results showed that the keyword-driven testing framework is the widely used approach, but recently, hybrid approaches have been adopted for mobile test automation. On the other hand, this review indicated that the existing frameworks need to be customized using reusable and domain-specific keywords to make them suitable for mobile application testing. Considering this, this study proposes an architecture, the mobile-based automation testing framework (MATF). In the future, to address the mobile application testing challenges, the authors will work on implementing the proposed framework (MATF).

Keywords: testing; automation testing; mobile automation testing; automation testing frameworks; systematic review



Citation: Berihun, N.G.; Dongmo, C.; Van der Poll, J.A. The Applicability of Automated Testing Frameworks for Mobile Application Testing: A Systematic Literature Review. *Computers* **2023**, *12*, 97. <https://doi.org/10.3390/computers12050097>

Academic Editor: Yan Liu

Received: 28 March 2023

Revised: 24 April 2023

Accepted: 28 April 2023

Published: 3 May 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

For mobile apps to be successful, they must satisfy the application usability. A mobile application should be effective and efficient. This means the application must easily and quickly provide the user with what they are looking for. Mobile apps must pass through several testing phases to meet the desired quality.

Testing mobile apps manually has its bottlenecks. This includes repetitiveness and its time-consuming nature [1–3]. Nevertheless, much of mobile app testing is still performed manually, thus prone to errors, inefficiency, and high cost. Thus, there is an urgent need for automation testing.

Automated testing lets the computer simulate what the tester does when manually running test cases [4]. As a result, large mobile app development companies are migrating to automated software testing methods. Test automation for mobile applications has increased in popularity among software developers and testers since it speeds up their work process to achieve better and more robust results [1,5,6].

Automated testing for mobile apps offers the possibility to test mobile apps quickly and effectively. This enhances the reliability and acceptability of the mobile product. Using automated testing tools, developers can develop and run test cases within a short period. The tools assist in the playback of pre-recorded and predefined actions, compare the results to the expected behavior, and provide the results to test engineers [2,7]. In addition, once the test cases are created, they can readily be repeated and extended to perform similar testing scenarios, which are hardly possible with manual testing [2].

The main contributions of this paper include:

- Analyzing the major test concerns that occurred in the mobile test automation process and providing a taxonomy of mobile automation testing research to categorize and summarize the research works;
- Examining the existing literature on test automation frameworks for mobile application testing and identifying the test challenges;
- Proposing an improved mobile automation testing framework architecture that tackles the test challenges;
- Eliciting directions for furthering the research in mobile test automation.

The rest of the paper is organized as follows: Section 2 discusses preliminary research on mobile test automation frameworks. Section 3 describes the related work and Section 4 discusses the systematic literature review methodology. In Section 5, we present the selected publications, along with a statistical analysis of those selected publications. It also introduces the data extraction strategy. Section 6 introduces the result of our systematic literature review (SLR) whereas Section 7 discusses the major findings from the review. Section 8 enumerates threats to the validity of the research. Finally, Section 9 discusses future works and concludes the research paper.

2. Preliminary Research on Mobile Test Automation

Mobile phones and mobile applications are incorporated into our day-to-day activities. End users are using several mobile apps on their phones to perform their daily routines. For mobile apps to be successful, they must satisfy the application usability. A mobile application must be effective and efficient. This means the application must provide the user with what they look for, easily and quickly. Mobile apps must pass through several testing techniques to meet the desired quality.

2.1. Mobile Automation Testing

Mobile automation testing is one of the phases performed by testers to check whether the required functionality is satisfied by the apps [7,8]. Mobile testers are now adapting automation to achieve quality and productivity. However, the era of mobile automation testing has its challenges. Some of the pitfalls of mobile automation testing include not knowing or identifying which tests can and should be automated, automating inappropriate test cases by developers, and automating tests at the wrong layer and time.

2.2. Software Quality for Mobile Apps

Successful mobile apps satisfy several quality characteristics. Some standard qualities defined in ISO/IEC9126 [9] and that are also published in articles include: usability, maintainability, reliability, security, efficiency, compatibility, and functionality [10–12].

1. **Usability:** It refers to how a product can be used to reach a specified goal. This shows to what extent the application is understandable, easy to learn, easy to use, has a minimum error rate, and overall satisfaction with the application. These are generic factors of usability that a given mobile app should fulfill [13].
2. **Maintainability:** This is a characteristic that determines the probability that a failed application is restored to its normal state within a given timeframe. A maintainable app is reasonably easy to scale and correct.

3. **Reliability:** It indicates the application's ability to function in given environmental conditions for a particular amount of time.
4. **Security:** It shows the application's ability to protect itself from hacker attacks and the techniques used to ensure the integrity and confidentiality of the data.
5. **Efficiency:** An efficient mobile app consumes fewer resources such as less loading time and consuming less power and memory during usage of the application.
6. **Compatibility:** A compatible mobile app is one that properly works across different mobile devices, OS platforms, and browsers.
7. **Functionality:** This refers to the application's ability to perform a task that meets the user's expectations.

2.3. Test Automation Frameworks

Mobile test automation is performed using a testing framework that defines the execution environment for automated tests. It is responsible for setting out the template to express expectations. Frameworks play a major role in reducing maintenance costs, and testing enhances automation experts' test speed and efficiency as well as their test accuracy. However, to achieve the quality metrics (see Section 2.2), finding a suitable mobile automation testing framework is crucial.

As per our preliminary literature review, there are different types of test automation frameworks. Each approach has benefits and limitations. Each of the frameworks is briefly discussed below.

2.3.1. Linear Automation Framework

Linear automation framework includes record and playback features that enable testers to record each testing action and then playback the script to repeat the test. This is one of the fastest ways to generate test scripts. A limitation of this framework is that the test scripts generated are not reusable since the test data are hardcoded into the script. In addition, including the script developed using this framework is not maintainable and scalable since any changes to the application induce modification and reworking of the test scripts [2,4].

2.3.2. Modular-Based Automation Framework

Modular-based testing is built on the concept of abstraction. It involves the creation of independent scripts by breaking down the application under test into several logical units and isolated modules. The modules, in turn, are used hierarchically to build large test cases. The key strategy in this framework is to build an abstraction layer so that any changes made in the individual module would not affect the other module or component. This benefits the automation testers to readily maintain and scale their test suites. The main challenge with the framework is that the test data are hardcoded into the test script and need programming skills to set up the framework [2,4].

2.3.3. Library Architecture Testing Framework

The test library architecture framework is also known as the structured scripting framework [14]. The framework is based on the module-based testing framework with some additional benefits. Instead of dividing the application under test into test scripts, it divides into procedures and functions. The framework requires the creation of libraries to store the functions which can be called by the test script whenever needed [4]. This framework introduces a high level of modularization which makes maintenance easier and cost-efficient. Test data are hardcoded within the scripts. More plans and technical expertise are needed to prepare the test scripts.

2.3.4. Data-Driven Testing Framework

Data-driven testing separates the test script logic from the test data [2,4]. These help testers store the test data in an external database. The data are stored in "key-value" pairs,

so test scripts can read and populate the necessary data. The main advantage of this framework is the reduction of the total number of scripts required to cover all possible test scenarios. This saves testers time by executing more tests faster. It also increases flexibility and maintainability. The disadvantage of the data-driven technique is the need for a highly skilled programming expert who properly manages the link between test scripts and external data sources [4,15].

2.3.5. Keyword-Driven Testing Framework

Keyword-driven testing is a system-independent framework that utilizes data-driven testing. In this approach, sets of code called **keywords**, belonging to the test script, are kept in an external data file. The keywords act as a directive that dictates the actions to be performed on the application under test. The keywords and test data are independent of the automation tool being used and are stored in a tabular structure. In addition to the advantages provided by the data-driven testing framework, the keyword-driven framework requires only minimal scripting knowledge from testers. Since a single keyword can be used across multiple test scripts, it provides high code reusability. The main challenge with this framework is its complexity and the need for a good automation expert [2,4,14,16].

2.3.6. Hybrid Test Automation Framework

The hybrid framework is a combination of any of the frameworks mentioned above [14]. The framework leverages the advantages and mitigates the weaknesses of the associated frameworks. It comes up with a flexible framework that suits the application under test.

To understand more about the existing automation testing frameworks, we have decided to conduct a systematic literature review. The focus of the systematic review is to investigate the applicability of the existing test automation frameworks for mobile test automation.

3. Related Work

Several surveys and systematic mapping studies have been conducted in mobile application testing. Some of these have attempted to provide an overview of the field via empirical studies, surveys, and systematic mapping of mobile application testing techniques.

Muccini et al. [6] indicated that testing mobile apps come with several challenges including performance, security, reliability, and energy requirements. The paper also mentioned that automation played an important role in testing mobile applications. However, they did not conduct a detailed investigation of the automation testing techniques and frameworks.

Corral et al. [17] conducted a state-of-the-art survey to discuss what kind of software assurance practices are needed for mobile apps in three levels: software development processes, software product assurance practices, and software implementation practices. According to the authors, they were able to find several practices for mobile systems. Even though they identified several software assurance practices for mobile apps, we believe a rigorous systematic literature review would further facilitate the choice of an appropriate testing technique among the existing ones.

In another study, Sahinoglu et al. [18] performed a systematic mapping of mobile applications. Their results indicate that the most frequently used test types for mobile applications are functional and usability testing. The research also showed that the top three issues addressed by the research community include test environment management, test case automation, and test case generation. The authors recognized in their conclusion that further work is needed in mobile application testing techniques as well as for building a suitable framework for it.

Zein et al. [19] reported on mobile application testing and its challenges using a systematic mapping study. The study provides comprehensive mapping to build a novel classification scheme for mobile application testing. The authors believed that there is no substantial work on this evolving topic and have provided a high-level categorization of the

techniques into four main categories: (1) structure of the topic or evidence; (2) contribution facets; (3) objects involved in the study; and (4) research facets. In each scheme, the authors discussed the methods applied in each study.

Kong et al. [20] performed a systematic literature review on the automated testing of android apps. The study identified the current trends, methodologies, and challenges faced by the android testing approaches. The study created a taxonomy of the related research exploring various testing dimensions. The dimensions include test objectives, test targets, test levels, and test techniques. The study performed a trend analysis for the taxonomy mentioned above. For instance, the trend of testing methods showed that model-based testing has the most dominant literature from the explored literature.

Singh et al. [21] implemented a test automation framework using a model-based testing approach using a manual test script. The framework is also open for integration with other automation testing frameworks such as data-driven and keyword-driven testing frameworks. The authors also emphasized that model-based testing helps in delivering test automation at a fast rate and with minimum cost.

Linares-vásquez et al. [22] conducted a survey to show the key challenges that make automated solutions ineffective. To this end, the authors defined a testing framework based on the continuous, evolutionary and large-scale principle (CEL). To enable fully automated solutions, the framework also incorporates a research agenda that describes six major topics that need to be addressed in the future: (1) improved model-based representations of mobile apps; (2) flexible open source solutions for large-scale and crowdsourced testing; (3) goal-oriented automated test case generation; (4) derivation of scalable, precise automated oracles; (5) derivation of methods to provide useful feedback for developers; and (6) mining software repositories and user reviews to drive testing.

An empirical study together with a systematic literature review was performed to investigate mobile development challenges [23]. The authors discussed the challenges involved in native, web, and hybrid mobile applications. Fragmentation, compatibility, and testing are the most reported challenges for native, web, and hybrid mobile applications, respectively. Other challenges reported include lack of expertise and tool support, change management, reuse of code, security, lack of training, and knowledge management.

Junmei et al. [24] performed research on mobile automation testing using Appium as a testing tool. The research aimed to improve the quality of mobile applications and conducted an experiment to evaluate the proposed method. The result indicates that the Appium testing tool is more efficient in the realization of mobile automation testing. The study also indicates that regression testing is more convenient and accelerates the development of mobile applications.

A more recent state-of-the-art survey conducted by Luo et al. [25] looks into context simulation methods for testing context-aware mobile applications. The authors performed an in-depth comparison of the key technical details of relevant context simulation techniques including both data-driven and model-based approaches. They concluded that testing mobile context-aware applications is costly and time-consuming. The findings of this study are in agreement with the Amalfitano research [7], who suggested that only a few research works have been conducted on context-aware systems and further work needs to be done on context-aware approaches to assist testers in different testing activities.

To summarize, most of the related work discussed above focuses on mobile application testing techniques and challenges and does not embrace the applicability of the automation testing framework for mobile application testing. Further, the studies did not propose a suitable testing framework that will address the test challenges. A summary of the limitations of the related work is shown in Table 1.

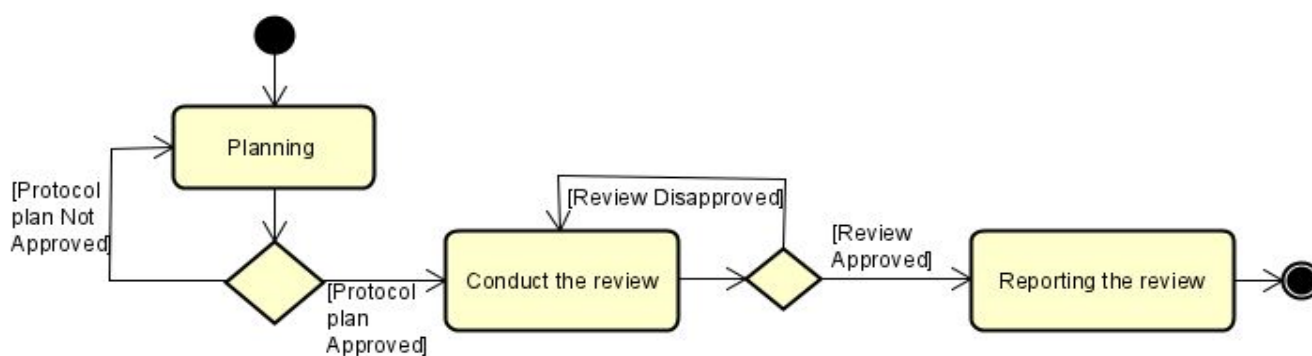
Table 1. The limitations of the related work.

Title of Paper	Limitations
Software testing of mobile applications: challenges and future research directions [6]	Did not perform a detailed investigation on automation testing techniques and frameworks.
Software assurance practices for mobile applications [17]	A rigorous systematic literature review is not conducted that aid testers to choose an appropriate testing framework.
Mobile application verification: a systematic mapping study [18]	Mobile application testing is not reviewed in detail and a suitable framework needs to be developed.
A systematic mapping study of mobile application testing techniques [19]	The authors did not propose an improved architecture for solving mobile application test challenges.
Automated testing of android apps: a systematic literature review [20]	The authors did not propose an improved architecture for solving mobile application test challenges.
Implementing test automation framework using model-based testing approach [21]	The implemented framework is not associated with the test challenges.
Continuous, evolutionary and large-scale: a new perspective for automated mobile app testing [22]	A detailed investigation on the current test frameworks is not conducted and the test challenges addressed by the framework are not discussed.
Research on mobile application automation testing technology based on Appium [24]	The study is limited to the Appium testing tool and other testing tools are not incorporated.
A survey of context simulation for testing mobile context-aware applications [25]	The study conducts a survey on testing context-aware mobile systems and did not suggest a solution for it.

4. Methodology of This Review

This work has adopted the SLR approach proposed by Kitchenham et al. [26] and can be summarized into three main phases: planning the review, conducting the review, and reporting the review.

In the planning phase, a review protocol is developed, which consists of the following elements: the research questions that the review is intended to answer, identification of the search string, and primary studies' selection criteria. Conducting the review phase involves primary studies' identification, selection, and evaluation according to the inclusion and exclusion criteria set in the review protocol. The final phase involves reporting the review findings which includes specifying dissemination mechanisms and formatting the main report [26]. Figure 1 depicts the process of SLR.

**Figure 1.** A systematic review process.

4.1. Definition of Research Scope

We defined two research objectives and two specific research questions that support the investigation of the research work.

I. Research Objectives

The current research focuses on examining the existing automation testing frameworks' suitability for mobile application testing and suggest an improved mobile testing architecture that addresses the gaps in automation testing frameworks. To this end, we identified two research objectives that guide the research work.

RO1: To assess the existing automation testing framework strength and weaknesses in testing mobile applications.

RO2: To propose an improved mobile automation testing framework that addresses the test challenges.

II. Research Questions

Following the research objectives, there are two research questions formulated in this review.

RQ1: What is the applicability of automation testing frameworks in mobile application testing?

With this research question, we intend to uncover the popularity of the existing test automation frameworks in mobile test automation with a focus on library, data-driven, keyword-driven, and hybrid frameworks. We look more in detail at how existing frameworks are adopted in mobile application testing and see which one is most popular and reliable. Concerning the applicability of the frameworks, we also look into the architecture of the frameworks, which consists of the components of the framework [2,27].

RQ2: What are the challenges of the existing automation testing frameworks for mobile application testing?

With this research question, we investigate the strength and weaknesses of the current automation frameworks. We will answer the research question by identifying the relevant test concerns or challenges in testing mobile applications.

4.2. The Search String

To form the search string, the main terms that compose the research question were identified. The terms "mobile application", "automation testing framework", "challenges", and "applicability" represented the main terms (see Table 2).

Table 2. Search keywords.

Group	Keywords
Mobile	Mobile, android, iOS
Automation testing	Automation testing, mobile automation testing, automated testing
Automation testing framework	Testing framework, automation testing framework, mobile automation testing framework
Challenge	Challenge, limitation, constraint, drawback
Framework	Frameworks, tool, model

A search by the keywords was chosen as the study selection methodology. The search terms used were constructed as suggested by Kitchenham and Charters [26]; most of the terms were generated from the research questions, and we used Monkey Learn, a keyword extractor tool, to identify the major group of keywords, then alternative terms similar to those generated from the research question were added. The final search string is created as a combination of the keywords by linking each group (search_string = group1 & group2 & group3 & group4 & group5), where each group is represented as a disjunction of its keywords (group = kw1 | kw2 | kw3 | kw4 | kw5).

The systematic review focuses on two key aspects: mobile automation testing challenges and the applicability of mobile automation testing frameworks. Since a variety of

terms may be used by authors to indicate any of these aspects, we rely on the extended set of keywords identified in Table 2.

The search keywords defined in Table 2 are applied to online databases to identify appropriate research papers. It took several tries to construct the right search strings since adding more keywords and synonyms makes the returned result irrelevant. We considered five widely used repositories for the review: IEEE Xplore, ACM Digital Library, Google Scholar, Web of Science, and Scopus. The “advanced” search functionality of the five selected online databases did not provide us with more relevant articles. The search was run on the digital repositories many times iteratively. In each iteration, the articles were filtered based on their content, which included the abstract, the methodology, and the conclusion part of the article.

Most of the relevant search strings were found in Scopus, hence it was used to pilot the search strings. Table 3 shows the piloted search strings, returned results, and relevant studies from Scopus. The returned results showed that Try5 yields precise and relevant articles, so it is selected as a search string.

Table 3. Search strings piloted on Scopus.

#Try	Search String	Returned Results	#Relevant Articles
Try1	((“automation test *”) AND (mobile * OR android OR ios) AND (applicable * OR appropriate * OR suitable *))	3	1
Try2	((“automation test *”) AND (mobile * OR android OR ios))	45	18
Try3	((“automation test *”) AND (mobile application *))	29	12
Try4	((“automation test *”) AND (mobile *) AND (framework * OR model *) AND (challenge *))	4	1
Try5	((“automation test *”) AND (mobile application test *))	29	13

The asterisk (*) is used as a wildcard symbol in a search to find content with the same prefix.

In order not to miss the relevant papers and to discard the irrelevant ones, we considered an offline searching approach for the research papers produced from the database repositories above, i.e., checking the content of the papers with relevant keywords defined in the search string. For instance, if a given research paper did not include any of the keywords mentioned in Table 2; it will be excluded from the candidate list. Furthermore, a backward snowballing technique was applied to the filtered articles to identify more relevant articles to include in the review.

4.3. Inclusion and Exclusion Criteria

The inclusion and exclusion criteria are:

Inclusion criteria:

I1: The study must be directly related to mobile automation testing, automation testing frameworks, or challenges of automation testing frameworks.

I2: Papers must provide empirical evidence to support their results (i.e., it should provide empirical qualitative or quantitative data).

I3: If more than one paper reports the same study, only the latest or mature paper is included. A mature paper is a recent high-quality paper with substantial results for the problem under study.

I4: Publications from January 2008 to February 2023 are included in the review.

Exclusion criteria:

E1: Studies related to automation testing frameworks for web applications are excluded.

E2: Research papers that are not related to mobile automation frameworks for testing mobile apps are excluded. Our search keywords indeed include broad terms such as automation testing, mobile application testing, and testing frameworks, as the aim is

to include all relevant papers that focus on mobile application testing using the current existing automation testing frameworks.

E3: Short papers/posters are excluded, mainly because such papers are often idea papers or works-in-progress, and they are not mature enough to be included in the review. In this study, papers that are explicitly stated as posters and idea papers without proper methodology are classified as short ones.

In addition to that, we have applied two filter criteria.

F1: Papers that are not written in the English language are filtered out.

F2: Secondary studies were also excluded from the review.

The inclusion and exclusion criteria are applied in the following manner:

1. E1, E2, and E3 are applied in turn to the search result to exclude irrelevant articles.
2. I1 and I2 are applied to each remaining study to include the research papers that satisfy the criteria.
3. I3 is applied to duplicate articles to include mature studies.
4. I4 is applied to include the most recent articles.

5. Primary Publication Selection

First, we conducted an abstract review and a quick full paper scan to filter out unrelated papers based on the exclusion criteria defined above. At the end of this stage, a collection of the primary publication is known. Secondly, a full evaluation of each primary publication is performed to extract all the important information from the publication that helps to answer the research questions. The extracted data are further checked to ensure their validity and consistency. Table 4 summarizes the statistics of the collected papers in the search stage.

Table 4. Summary of the selection of primary publications.

Step	Count
Repository (subject database) search without restriction	4917
After performing a manual walkthrough of the papers	345
After an in-depth review of titles/abstracts	251
After skimming/scanning full paper	61
After removing duplicate papers	56

The repository search gave us a total of 4917 research papers without restriction. Then, a manual walkthrough was conducted on the title and abstract of each publication to exclude those that correspond with the exclusion criteria. After filtering out using the exclusion criteria, the papers that satisfied the matching requirements dropped from 4917 to 345. After this step, we performed a second iteration on the abstract and introduction part of the remaining paper, leading to the exclusion of 94 papers, which made the remaining set of papers 251. Subsequently, we went through a full scan of the remaining papers to exclude 190 papers. Lastly, duplicate papers were removed to obtain 56 articles considered as relevant primary publications. The full list of investigated publications is shown in Appendix A section.

As illustrated in Figure 2, almost 50% of the selected publications are found in IEEE Xplore and Google Scholar. The rest of the papers are from ACM, Web of Science, and Scopus.

As shown in Figure 3, 30% of the 56 selected publications are journals, 63% are conference papers, and the rest 7% are symposium publications.

We have also shown the distribution of selected articles by year. As shown in Figure 4, 45% of the 56 selected articles are published from 2014–2017, early publications before 2014 account for 35%, whereas recent publications from 2018 onwards account 20% of the total publications.

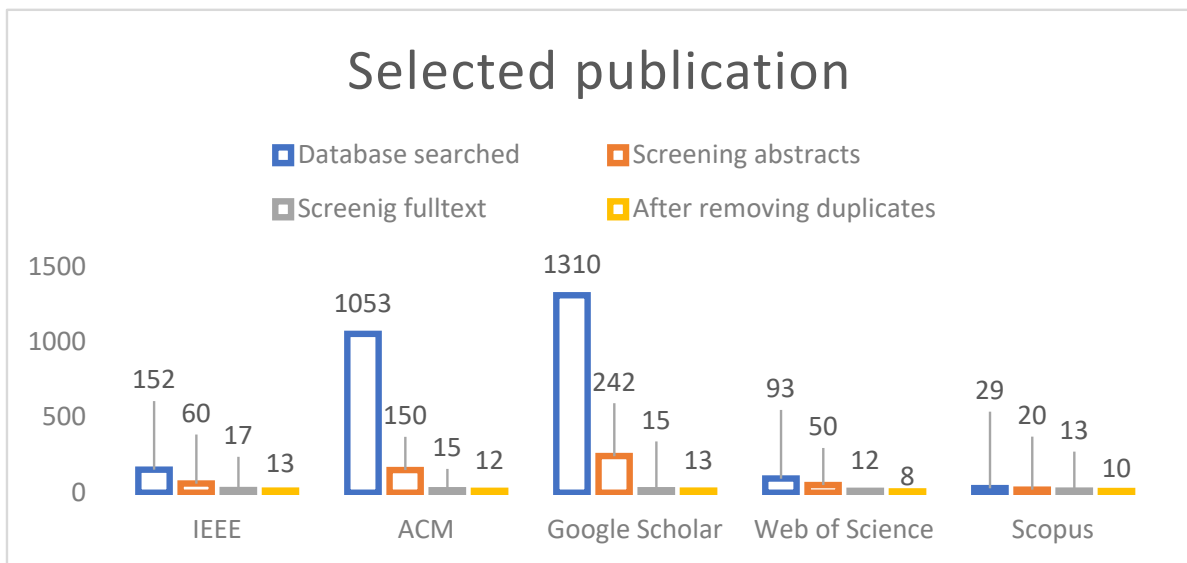


Figure 2. Frequency of selected publications from database repositories.

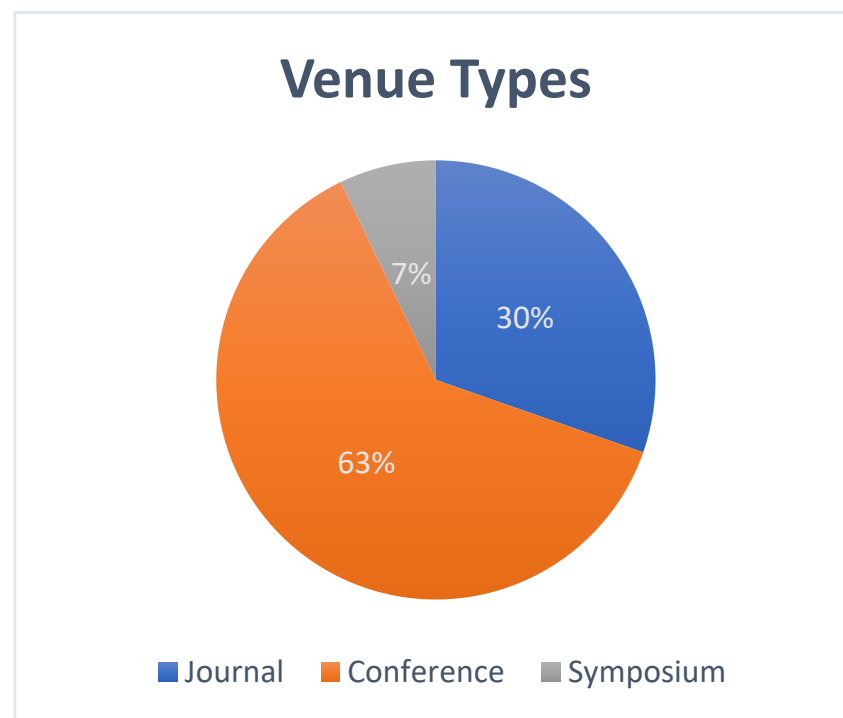


Figure 3. Distribution of primary publication venue types.

We examined the frequency of the test automation frameworks categorized under four-time intervals. Figure 5 illustrates the use of test automation frameworks in selected articles.

As illustrated in Figure 5, keyword-driven testing framework was investigated or used in the articles as a testing approach as compared with other automation testing frameworks. However, from 2016 onwards, researchers were looking for other approaches such as model-based testing and other automation testing tools. This indicates that customization of a keyword-driven testing architecture needs to be performed before applying it for mobile automation testing.

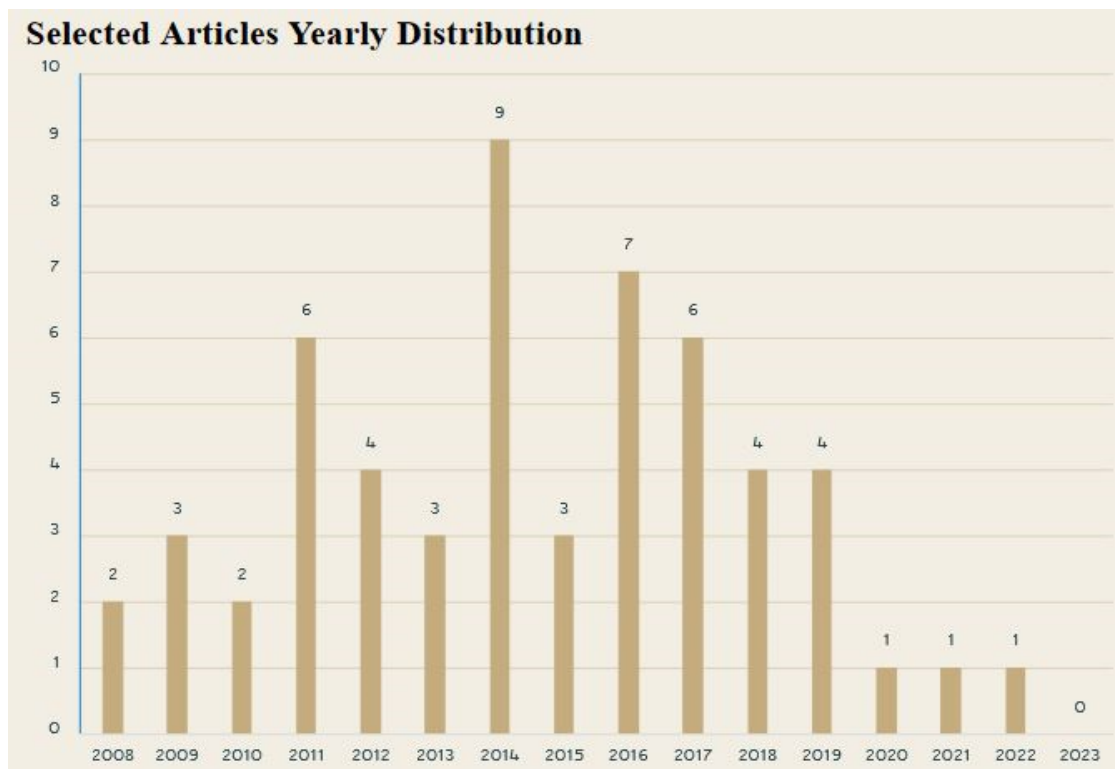


Figure 4. Yearly distribution of primary publications.

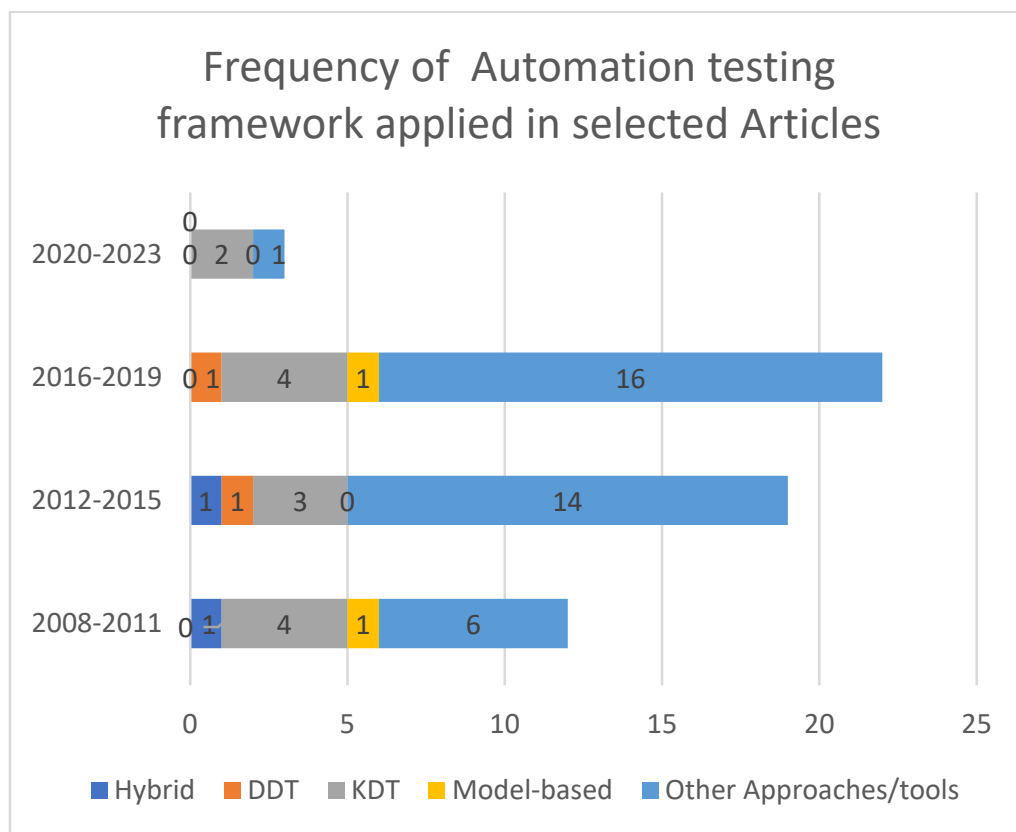


Figure 5. Frequency of test automation framework.

5.1. Data Extraction

Based on the selection method, the required data for answering the research questions were extracted. For this purpose, an extraction form was designed for the selected research papers. During this process, two types of data were extracted: data for answering the research questions and data for displaying the bibliographic information of the study. The extracted data were stored in an excel file for further analysis. Table 5 shows the data extraction form for the selected articles.

Table 5. Data extraction form.

Data Item	Description
Paper ID	The unique ID assigned for the research paper
Title	The title given for the research paper
Author(s)	The authors of the paper
Year	Year of publication of the research paper
Publication venue	The publisher's name for the study
Venue type	The type of the research article
Mobile automation techniques	Mobile automated testing techniques used in the study
Mobile testing tools	Mobile testing tools used/discussed in the paper
Framework/tool	A framework or a tool proposed in the study
Challenges	The challenge or limitation of mobile automation frameworks discussed in the study
Applicability	The degree of the applicability of the test automation framework for mobile apps

5.2. Taxonomy of MATF Research

To facilitate the data extraction process and to answer the research questions stated in Section 4.1 concretely, we found it useful to build a taxonomy of mobile automation testing. Such classification is important to gain a full understanding of the latest development in mobile automation testing framework. We adopted the taxonomy from the Kong et al. [20] research study. A general view of the taxonomy diagram, represented in Figure 6, is described in three categories, namely test objectives, test techniques, and test challenges. These categories are associated with the research questions. Test objectives or concerns and test techniques are linked with RQ1, and test challenges are linked with RQ2.

To answer RQ1, we exploited the existing automation testing framework's (Test approaches) applicability in mobile application testing. However, this will only show the frequency of the test approaches discussed in the selected articles. We also need to check if the test approaches have a role in satisfying the mobile quality factors (or test objectives). To answer RQ2, we investigated the limitations or concerns of the test approaches.

Overall, we classified the taxonomy into three dimensions.

Test Objectives: This category summarizes the test objectives. We have listed six testing objectives. These include reusability, efficiency, reliability, compatibility, scalability, performance, and functionality.

Test Techniques: This category focuses on automation testing approaches (e.g., data-driven or keyword-driven) as well as testing types (e.g., regression testing).

Test Challenges: The last category demonstrates the limitations indicated in the selected articles from the existing test automation frameworks (e.g., fragmentation, complexity).

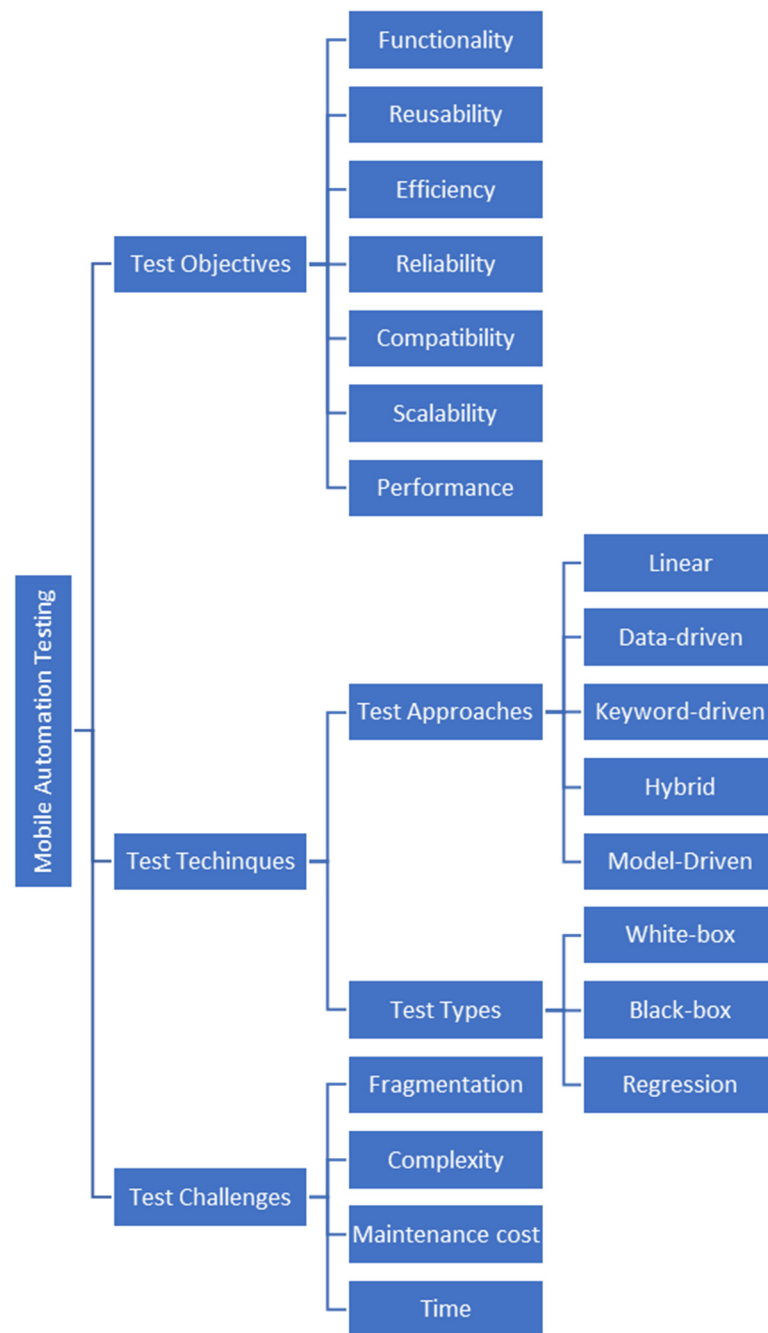


Figure 6. Taxonomy of mobile automation testing (Reproduced from [20]).

6. Review Findings

A summary of the reported findings is shown in Tables 6–8, respectively. We also stated a research question that is related to the taxonomy of mobile automation testing research.

6.1. What Are the Concerns of Automation Testing Frameworks?

Our review investigates the test objectives that test automation frameworks seek to achieve. Test objectives are factors that need to be fulfilled in the mobile automation testing process. For example, reusability is one of emphasized test objectives [28].

6.1.1. Test Objectives

In this section, we present the various test concerns in the mobile automation testing process, which includes satisfying the functional and non-functional requirements. We discuss some of the concerns from the investigated literature.

Reusability: Reusability is the third priority test concern according to our investigation [29]. A total of 41% of the literature discusses the significance of reusability in mobile application testing. Reusability in this context can be for the test scripts, functions/keywords, and test automation frameworks. Reusable test scripts help testers not to write test cases for every application or module. Many mobile applications are produced that while being different in their objective, they will have generic features. So, instead of writing test cases for each application, test cases are written for unique features or rather for functions and objects. The selected publication discussed several insights on how to improve reusability. Zhongqian et al. [16] proposed an android-based keyword-driven automated testing framework (AKDT) that improves the reusability of test scripts. Pereira et al. [30] confirmed that a keyword-driven testing framework has high reusability. Recent studies [28,31] also showed that the reusability of keywords with good design practice of keyword-driven testing has a latent effect on reducing maintenance costs. The work of Rwemalika et al. [28,31] confirms the previous findings by [2,32], in that a keyword-driven testing framework is a better approach for achieving reusability in mobile automation testing.

Efficiency: Efficiency is concerned with the utilization of test resources appropriately [33]. It is related to testing time and cost. According to our review, 54% of the articles discussed about the importance of efficiency, surpassing reusability by 13%. Designing and implementing efficient test automation approaches is the second priority in the selected publications. An efficient test framework supports test engineers to produce test scripts with a fast execution time and reduced cost. Fazzini [34] reported that mobile companies do not have enough time and the right framework to test their application. This study also raises the issue of fragmentation in the android environment.

To mitigate the efficiency issue, a preliminary study was performed by Fazzini [34] to propose three techniques to test apps more efficiently. The first technique enables developers to generate independent test cases so that developers can be more effective and efficient in testing applications. The second technique identifies screen compatibility issues caused by the fragmentation of the ecosystem. Thirdly, they designed a bug report analysis technique that changes error reports into test cases, so that developers can promptly start repairing defects.

Another study by Machiry et al. [35] proposed a Dynodroid system that aids in input generation for android apps. The system follows the “observe–select–expect” principle to efficiently generate a sequence of such inputs for android apps. Dynodroid is more efficient in finding bugs, compared to the popular Monkey tool.

When we come to the applicability of the existing frameworks in achieving efficiency, from the examined publications, our analysis showed that only 40% of publications use test automation frameworks such as keyword-driven and data-driven testing approaches for achieving test efficiency [27,36].

Functionality: According to the review, functionality testing is the number one priority. A total of 66% of the literature discussed the importance of functionality testing in mobile application testing. Functional testing is a process of testing the functional aspect of the application such as meeting the user’s expectations, reducing errors, and ensuring the quality of the app [11]. Mobile apps are suffering from functionality defects that degrade the quality and acceptance by end-users [37,38]. Mohammad et al. [11] conducted a comparative analysis of quality assurance automated testing tools for windows mobile applications. The study set out major quality factors such as functionality, usability, etc., and indicates which testing tools are suitable for achieving the quality factors. The results showed that there is no single automated testing tool that meets all the quality factors. Instead, a combination of mobile testing tools should be used to achieve the desired objective. Ranorex and HockeyApp are the recommended testing tools in the study [11]

The work of Boushehrinejadmoradi et al. [39] also discussed functionality as one of the concerns in mobile application testing.

Reliability: Reliability testing is concerned with checking whether the application can operate without failure for a specific time interval in a particular context [8]. In our review, 21% of the selected publications discussed the reliability test objective. Several authors have reported the importance of reliability testing [5,40,41]. Lovreto et al. [40] stated that using automated testing is essential for creating a reliable mobile app. Hu et al. [42] presents an approach for automating the testing process for android applications. The author claimed that the reliability of android apps is an important issue, and it is affected by the frequency of bugs identified in the apps. The proposed automation testing framework provides effective solutions for activity, event, and type-related bugs. The technique will facilitate the creation of reliable applications and improve the quality of android apps.

Compatibility: Compatibility is another test objective that checks whether the application can run on different hardware devices and operating system platforms [43,44]. Mobile applications are suffering from compatibility issues that are related to the fragmentation of the android ecosystem [34,45]. However, despite the significance of compatibility testing, only 16% of the selected articles provide a review on it. To mitigate compatibility problems, we need to develop efficient and scalable automation frameworks that perform compatibility testing for mobile applications [45]. For instance, Choudhary et al. [45] performed a comparative study among test input generation tools. The authors showed Monkey, GUIRipper [46], and A3E [47] testing tools are compatible with every android framework version. We did not find a single article that uses test automation frameworks for achieving compatibility. We believe compatibility testing is overlooked in the current studies, but it is crucial to the successful performance of mobile applications.

Scalability: Scalability testing is a non-functional requirement that measures an application's ability to meet the growing needs of the user such as increased user traffic, data size, transaction frequency, etc. [11]. Only 13% of the selected articles introduce the concept of scalability in their automating testing research [5,48,49]. We also observed from the review that scalability is given less emphasis compared to compatibility. As discussed in functionality testing, the work of Mohammad et al. [11] also noted that a few automated testing tools, namely Perfecto mobile and Keynote, are available for assurance of scalability. When we come to the applicability of the existing frameworks in achieving scalability, just like that of compatibility, none of the examined publications apply the existing test automation frameworks for achieving scalability in mobile test automation.

Performance: Performance testing evaluates the speed, response time, and overall stability of the mobile application [11]. Mobile applications are sensitive to performance issues such as slow user interaction, poor responsiveness, and unstable application [50,51]. Kulkarni et al. [52] developed a performance analysis module using the Calabash automation tool which determines the launch time of the mobile application. The authors also reported that calabash is an open-source tool that supports all API levels. Kannan et al. [53] also reported performance-related bottlenecks that happened when migrating a very huge database of a client using a migration tool. The tool was subjected to performance testing and obtained unsatisfactory results.

The selected publications for our SLR in terms of the test objectives are considered in Table 6. Through our in-depth review, the most considered testing objective is functionality, accounting for 66% of the selected publications. Efficiency and reusability test objectives ranked second and third, respectively, which account for 54% and 41% of the primary selected publications.

6.1.2. Test Techniques

Test techniques discuss the kind of test approaches and test types used or discussed in the examined literature. Test approaches discuss the different automation methodologies applied in mobile application testing (e.g., data-driven or keyword-driven testing) as well as the test types (e.g., white-box or black-box testing).

I. Test Approaches

Modular and Library automation testing techniques were not discussed in the selected articles, so they are not discussed in this section. However, linear, data-driven, keyword-driven, model-driven techniques, and a combination of one or more techniques (i.e., hybrid techniques) were discussed in selected publications. Table 7 lists all the testing approaches addressed in the literature review.

Linear testing is a testing approach that applies a record/playback method, i.e., testers set the testing tool to the record mode while performing actions on the application under test [2,52]. Song et al. [54] proposed an integrated test automation framework to test multiple heterogeneous platforms efficiently.

Data-driven testing is a testing methodology that uses tables to store test data [2]. It allows testers to input a single test script that can perform tests for all test data from a table, and the test environment settings and control are not hardcoded, i.e., the test criteria, the input values, and output values are stored on a specified data source, such as Excel, CSV files, etc. In [55], the authors developed a novel framework for testing mobile applications based on a data-driven approach. The framework uses the Appium test library and server to automatically test mobile applications. Hanna et al. [2] conducted a review of scripting techniques for automation testing and recommended a data-driven scripting technique since it satisfies the following characteristics (i.e., reusable functions, low maintenance cost, separation of data from test scripts, using scripts in regression testing, etc.).

Keyword-driven testing is an extension of the data-driven testing approach, which is developed to mitigate the limitation of data-driven testing [2]. The keyword-driven approach is an improvement to its predecessor where all tests are alike and producing new tests requires new code in the framework. In keyword-driven testing, the test data and the keywords are kept in external files. Keywords are the core components in the framework that are used to store the testing process. Even if most of the investigated literature did not use existing automation testing framework in their research work, our review has shown that from the existing automation test approaches under investigation, keyword-driven testing is the most common approach used in mobile automation testing; a total of 55% of the publications have involved some keyword-driven technology. Zun et al. [56] used a keyword-driven testing approach to create a multi-platform automatic testing framework (MATF). The framework can automatically generate and execute the test script on android and iOS platforms.

The benefit of the keyword-driven technique is also discussed by Zhongqian et al. [16]. The advantages mentioned include an easy and efficient way of creating test scripts, introducing business logic into one's automated test batches, the early building of automated tests to save time, and easy maintenance of test scripts.

Hybrid testing is the combination of one or more testing frameworks, such as linear, data-driven, keyword-driven, or model-driven testing. The reason for combining is to gain the benefit from each framework and to eliminate possible disadvantages. For example, Pajunen et al. [57] designed and implemented a general-purpose hybrid test automation framework. The framework is constructed from a keyword-driven and model-based testing approach. The paper describes the integration of the online model-based testing tool (TEMA toolset) and keyword-driven test automation framework (Robot Framework). The integration helps testers to take advantage of the wide variety of library support that the Robot framework provides and that can be used in the online TEMA toolset. In [48], the authors showed that a combination of data-driven and keyword-driven technology with the help of XML can separate test scripts, data, and business logic, which not only enhances the reusability of test scripts but also divides the human labor into framework developers and testers.

Model-driven testing is a new paradigm in software testing that helps testers in automatic code generation from a given model. Similarly, model-based testing automates the generation and execution of test cases from a given model [58,59]. In [60], the authors proposed the MATeL model (mobile application testing language) that applies the concepts

of both the model driven-approach and domain-specific modeling language (DSML). The model helps in improving the quality of mobile applications. Furthermore, adding DSML behavior facilitates the test scenario in mobile applications because the testing is based on specific domain elements that are close to the area of mobile device concepts. The authors are trying to improve the model by adding new benchmarks such as smooth integration of the DSML with industrial testing platforms to provide a full-fledged testing tool for test engineers.

II. Test Types

In general, there are different types of testing based on the box approach or levels of testing. Based on the box approach, there are three testing methods, namely white-box, black-box, and grey-box testing [53,61]. Based on the level of testing, we have unit testing, integration testing, system testing, regression testing, acceptance testing, and so forth. We discussed the box approach and regression testing. The review results show that priority is given to black-box testing, followed by white-box and grey-box testing. On the other hand, the box approach has achieved more emphasis than regression testing. The statistics showed that 52% of the publications discussed the box testing approach, whereas 39% of the examined articles discuss the regression testing method.

Table 6. Test objectives from the examined literature.

Paper Id	Tool/Technique/ Framework	Reusability	Efficiency	Performance	Reliability	Reliability	Scalability	Compatibility	Functionality
SA1	Mustafa Abdul et al. [62]		✓	✓					✓
SA2	KDT [63]	✓							✓
SA3	KDT/DSL [30]	✓	✓						✓
SA4	EarlGrey [49]		✓	✓					✓
SA5	KDT [31]	✓							✓
SA6	Ukwikora [28]	✓							
SA7	Mohammad et al. [11]	✓	✓	✓	✓	✓	✓	✓	✓
SA8	Appium [37]		✓						✓
SA9	Sinaga et al. [50]			✓					✓
SA10	Appium/OpenCV [40]		✓	✓	✓	✓	✓	✓	✓
SA11	Mattia Fazzini [34]		✓					✓	✓
SA13	DDT [55]	✓	✓	✓					
SA15	Appium [64]		✓		✓	✓			✓
SA16	MDT [58]		✓						✓
SA17	Hussain et al. [32]	✓	✓				✓	✓	✓
SA18	Divya Kumar et al. [8]	✓	✓		✓	✓		✓	✓
SA19	Vahid Garousi et al. [65]		✓						
SA20	Jamil et al. [5]	✓	✓		✓	✓			✓
SA23	Calabash [52]		✓	✓					✓
SA24	ISO/IEC/IEEE [27]	✓	✓		✓	✓			✓

Table 6. Cont.

Paper Id	Tool/Technique/ Framework	Reusability	Efficiency	Performance	Reliability	Reliability	Scalability	Compatibility	Functionality
SA25	Shauvik Roy et al. [45]	✓	✓	✓			✓	✓	✓
SA26	Nader et al. [39]			✓			✓	✓	✓
SA27	Gunasekaran et al. [38]	✓	✓	✓	✓	✓			✓
SA28	MAT [44]			✓			✓	✓	✓
SA29	Bansal et al. [61]				✓	✓			✓
SA30	Rasneet et al. [41]		✓	✓					✓
SA31	Hanna et al. [2]	✓			✓	✓			✓
SA32	Kannan et al. [53]			✓					✓
SA34	Pallavi et al. [66]								✓
SA35	PLC Open XML/KDT [67]	✓							✓
SA36	Shiwangi et al. [3]	✓	✓	✓					✓
SA37	Dynodroid [35]		✓	✓					✓
SA38	KDTFA [29]	✓	✓						
SA39	Android KDATF [16]	✓	✓						✓
SA40	Testdroid [68]	✓		✓					
SA41	AndroidRipper [46]	✓							
SA43	MobTAF [69]		✓		✓	✓			✓
SA44	Song et al. [54]		✓						
SA45	Hu et al. [42]				✓	✓			
SA47	Crawler [70]	✓							✓
SA48	Dominik et al. [71]								✓
SA49	MDT/KDT [57]								✓
SA50	Quadri et al. [10]	✓	✓						
SA51	MobileTest [33]		✓			✓			
SA53	TDD [51]		✓	✓	✓				✓
SA54	KDT/DDT [48]	✓	✓				✓		
SA55	Adaptive KDT [72]								✓
SA56	LKDT [36]	✓	✓					✓	✓
	Count	23	30	17		12	7	9	37

White-box testing is a type of testing that mainly focuses on the internal details and structure of the system. It needs the tester's full knowledge of the program structure [61]. It is applied in the early phase of system development such as in unit testing. It is considered exhaustive and time-consuming. Software developers carried out white-box testing [41].

Table 7. Test approaches employed in the literature.

Paper Id	Tool/Technique/ Framework	Linear	Data-driven	Keyword-driven	Reliability	Model-driven	Test-driven	Hybrid
SA2	KDT [63]			✓				
SA3	KDT/DSL [30]			✓				
SA5	KDT [31]			✓				
SA6	Ukwikora [28]			✓				
SA13	DDT [55]		✓					
SA16	MDT [58]					✓		
SA22	MATF/Appium [56]			✓	✓			
SA23	Calabash [52]	✓						
SA24	ISO/IEC/IEEE [27]			✓				
SA31	Hanna et al. [2]	✓	✓	✓	✓			✓
SA35	PLC Open XML/KDT [67]			✓				
SA38	KDTFA [29]			✓				
SA39	Android KDATF [16]			✓	✓			
SA43	MobTAF [69]		✓					
SA44	Song et al. [54]	✓						
SA46	DSML [60]				✓	✓		
SA49	MBT/KDT [57]			✓		✓		✓
SA52	OSGi [73]			✓	✓			
SA53	TDD [51]						✓	
SA54	KDT/DDT/XML [48]		✓	✓	✓			✓
SA55	Adaptive KDT [72]			✓				
SA56	LKDT [36]			✓				
	Count	3	4	15		3	1	3

Black-box testing involves testing a system functionality with no prior knowledge of its internal code structure. A tester provides input and observes the output generated by the system under test [49]. It is carried out by independent testers. It is a powerful testing technique that reveals how a system reacts to expected and unexpected situations [37].

Grey-box testing is also called translucent testing. It combines the benefits of black-box and white-box testing [45]. It is conducted with limited information about the internal functionality of the system [5]. It has access to detailed design documents together with information about requirements.

Regression testing is categorized under functional-level testing. According to Kannan et al. [53], regression testing is a test technique that finds negative side effects (regression tests) and re-executes those tests that are impacted by the code changes. The examined literature indicates that automation testing is suitable to perform repetitive and labor-intensive tasks. This makes automated testing a suitable approach to conduct regression testing [2,54,65]. In [38], the authors conducted a comparative study on mobile automation testing tools. Their study indicated that Ranorex and Maveryx mobile testing tools help to conduct robust regression testing.

6.2. What Are the Challenges of Automation Testing Frameworks in Mobile Testing?

Test Challenges

Mobile test automation has its challenges in using the current test automation frameworks. This is the major reason researchers developed a novel test approach by utilizing the existing test automation framework. In this section, we have identified and discussed the bottlenecks in mobile automation testing frameworks. The challenges include complexity, maintenance cost, time, and fragmentation. Table 8 summarizes the test challenges discussed in the literature work.

Complexity: Mobile applications are becoming larger and more complex. This raises the issue of application quality [16,64]. The need for better-quality apps puts pressure on test engineers and the mobile automation testing process. In [27], it was stated that the need for high-quality apps also adds to the complexity of designing and testing applications. To reduce the complexity issue, a proper keyword-driven testing technology that applies different levels of keyword abstraction is crucial. On the contrary, there are research works from the literature that states that keyword-driven testing increases complexity [2,28]. For instance, early research indicates that the keyword-driven scripting technique is complex as it needs a special framework and effort in creating test scripts [2]. Recent studies by Rwemalika et al. [31] also suggest that keyword-driven testing test design is complex with several levels of abstraction. The same authors [28] extend the previous research work to address the complexity issue [28]. The authors introduce an automated tool called Ukwikora—a continuous monitoring tool for keyword-driven testing (KDT). The tool supports testers in automating and analyzing acceptance test suites.

Maintenance cost: Most mobile app testers agree that mobile app maintenance costs 15–20% of the total app development cost [74]. This figure can be reduced with the support of test automation frameworks. Mobile app maintenance is not only about fixing defects, but is also an inclusive process of several practices such as performance enhancement, upgrading the app, maintaining robustness, etc. In their review of the impact of automation on software cost, Kumar et al. [8] mentioned that timely maintenance of automated testing suits is necessary to reduce the maintenance cost. In a review of scripting techniques [2], the authors indicate that data-driven and keyword-driven scripting have low maintenance costs as compared with modular, library, and linear scripting techniques. On other hand, ISO/IEC/IEEE [27] provides a guideline that aids in creating an efficient and consistent solution for keyword-driven testing. During the discussion on the benefits of KDT, ISO/IEC/IEEE [27] reported that maintenance of the keyword scripts does not affect the execution of the test cases. On the contrary, continuous maintenance and support of the keyword library will require time, budget, and professional experts.

Time consumption: Despite the available test automation tools and approaches, mobile testing has always been a time-consuming task [55]. To alleviate this issue, the mobile development industry practices test automation [62]. However, test automation by itself requires development effort and significant time. In the research by Kumar et al. [8], the experimental analysis showed that test automation has a positive impact on time and cost as compared with manual testing. In addition, with the appropriate use of automation approaches and tools, we can decrease testing time to execute test scripts. Jamil et al. [5] performed a literature review of software testing techniques and stated that using automation testing techniques is time effective, as it saves a tremendous amount of labor time. In its discussion on types of test automation frameworks, Jamil et al. [5] pointed out that keyword-driven testing has encompassed all the benefits of data-driven testing. In addition to that the keywords reusability feature benefits quality assurance testers to test their test script efficiently and cut down the testing time and ensure on-time application release.

Fragmentation: Mobile device fragmentation is a trending test challenge that occurs when some mobile users are running an older version of an operating system, while other mobile users are using updated versions [65]. The issue is normally associated with android devices but not with iOS devices. Even though that is the ground fact, our systematic review faces a lack of research effort on the topic (refer to Table 8). Fazzini [34], reported

the fragmentation of the android ecosystem as one of the bottlenecks mobile companies face in testing mobile applications. The study proposed a new technique that identified compatibility issues caused by android fragmentation. In [45,60], the authors agree with Fazzini [34] that ecosystem fragmentation is a serious challenge for conducting tests that can reveal all issues that an end-user might face on a specific device runtime environment. Together, these studies provide important insights into mobile device fragmentation as the issue is related to other test concerns also such as compatibility and complexity [34,45,60].

Table 8. Test challenges reviewed in the literature.

Paper Id	Tool/Technique/ Framework	Time	Maintenance cost	Complexity	Fragmentation
SA1	Mustafa Abdul et al. [62]	✓		✓	
SA2	KDT [63]	✓			
SA3	KDT/DSL [30]		✓	✓	
SA4	EarlGrey [49]	✓			
SA5	KDT [31]		✓	✓	
SA6	Ukwikora [28]		✓	✓	
SA7	Mohammad et al. [11]	✓	✓	✓	
SA8	Appium [37]			✓	
SA9	Sinaga et al. [50]	✓			
SA10	Appium/OpenCV [40]	✓		✓	
SA11	Mattia Fazzini [34]	✓			✓
SA12	Anusha et al. [75]	✓			
SA13	DDT [55]	✓	✓		
SA14	AutoClicker [76]	✓			
SA15	Appium [64]	✓		✓	
SA16	MDT [58]	✓	✓	✓	
SA17	Hussain et al. [32]	✓		✓	
SA18	Divya Kumar [8]	✓	✓		✓
SA19	Vahid Garousi et al. [65]		✓		
SA20	Muhammad Jamil et al. [5]		✓	✓	
SA24	ISO/IEC/IEEE [27]	✓	✓	✓	
SA25	Shauvik Roy et al. [45]				✓
SA26	Nader et al. [39]			✓	
SA27	Gunasekaran et al. [38]	✓		✓	
SA28	MAT [44]	✓			
SA29	Bansal et al. [61]	✓			
SA30	Rasneet et al. [41]	✓			
SA31	Hanna et al. [2]	✓	✓		
SA35	PLC Open XML/KDT [67]		✓		

Table 8. Cont.

Paper Id	Tool/Technique/ Framework	Time	Maintenance cost	Complexity	Fragmentation
SA36	Shiwangi et al. [3]	✓			
SA39	Android KDATF [16]	✓	✓	✓	
SA40	Testdroid [68]				✓
SA41	AndroidRipper [46]		✓		
SA42	MobiTest [77]	✓			
SA43	MobTAF [69]	✓			
SA46	DSML [60]	✓	✓		✓
SA47	Crawler [70]		✓	✓	
SA50	Quadri et al. [10]	✓	✓		
SA51	MobileTest [33]		✓	✓	
SA53	TDD [51]	✓	✓		
SA56	LKDT [36]	✓	✓	✓	
	Count	28	20	18	5

7. Discussion

There have been research works on mobile application testing for the past few years. However, most of the work did not indicate the relation between automated testing frameworks and mobile application testing. Our discussion focuses on the trends that we observed while conducting the SLR as well as future research directions and challenges that need to be addressed by the research community.

7.1. Trend Analysis

Figure 7 shows the trend in test approaches over the years. Keyword-driven testing is dominating the literature on automation testing frameworks for testing mobile applications. Recent publications [28,30,31,63] confirm this statement.

Trend analysis of testing types in Figure 8 shows that most of the literature focuses on regression testing. This is not a surprising result since automated testing is very effective in performing regression testing. Together, white-box, black-box, and grey-box testing cover 52% of the research work, while 39% is regression testing and the rest 9% did not focus on any of the testing types.

Regarding testing objectives, Figure 9 depicts that functionality and efficiency as the most trending objectives in the examined literature, whereas time and maintenance cost are the most trending test challenges.

From 2016 onwards, efficiency surpasses the reusability test concern, and it is widely covered in the literature just like that as functionality test objective. Regarding test challenges, Figure 10 shows that time test concern has attracted automation testing research. On the other hand, maintenance cost and complexity have had similar results over the years.

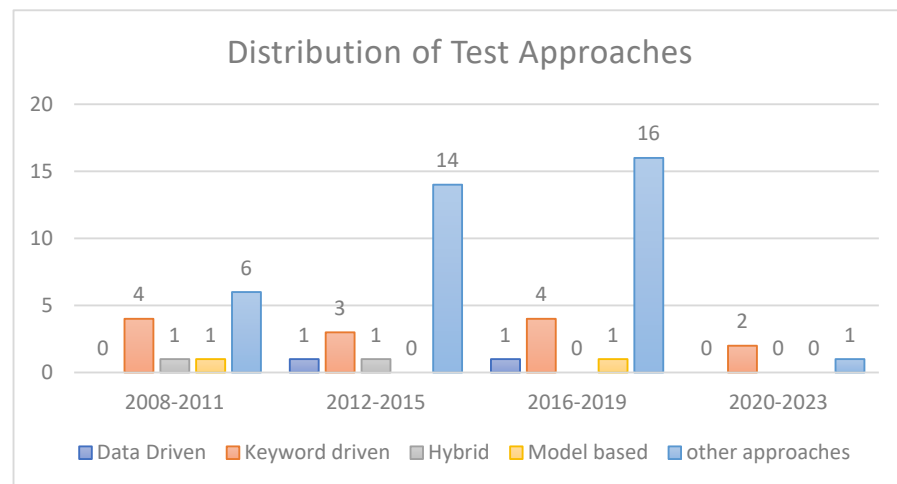


Figure 7. Testing approaches' trends.

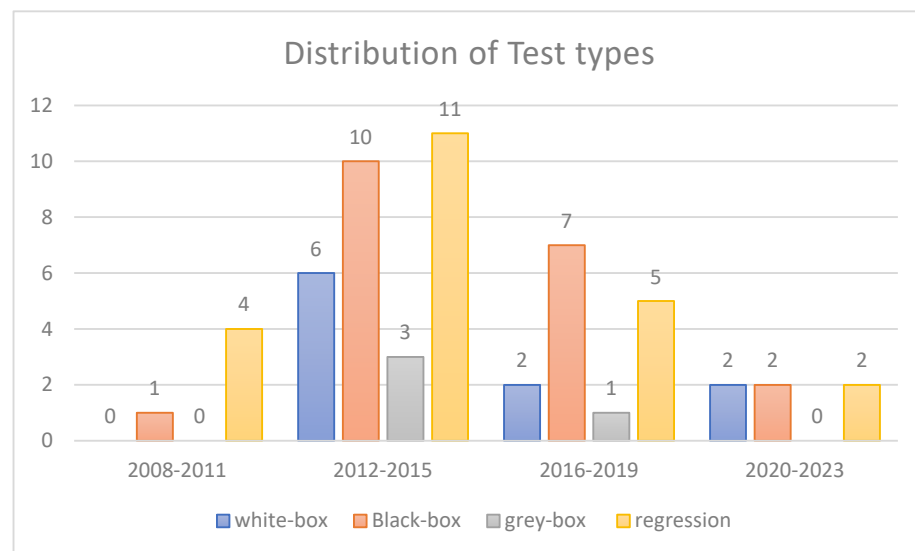


Figure 8. Testing types' trends.

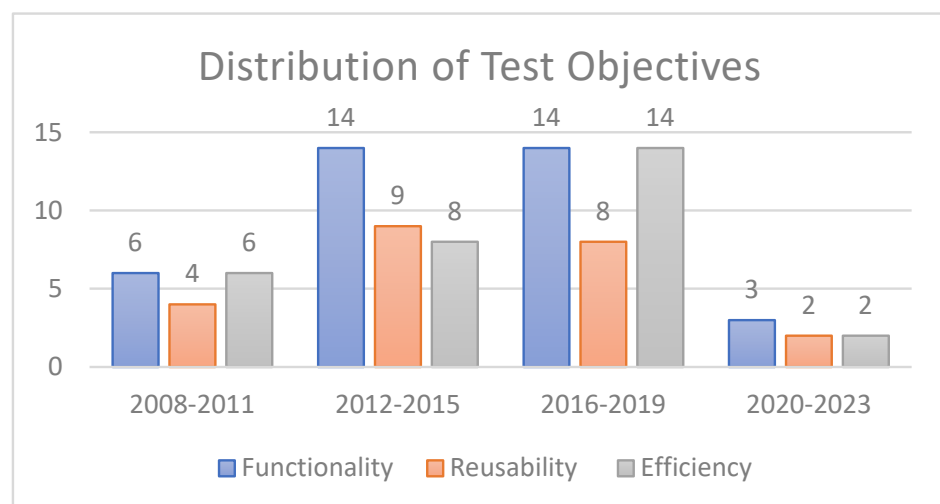


Figure 9. Test objectives' trends.

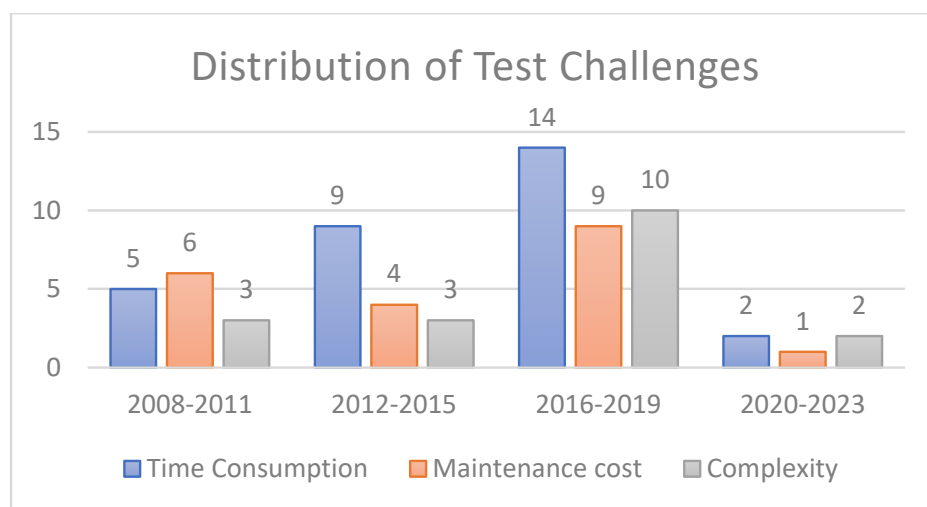


Figure 10. Test challenges' trends.

7.2. Research Question Insights

RQ1: What is the applicability of automation testing frameworks in mobile automation testing?

First, with this research question, we wanted to explore the usage of the existing test automation framework for mobile application testing based on the satisfaction of the test objectives.

“**Functionality**” and “**Efficiency**” have been the most trending concerns in functional and non-functional test objectives, respectively.

“**Reusability**” is another testing objective that needs attention in mobile automation testing since designing reusable test scripts for the automated process will increase the tester’s productivity and lower the maintenance burden.

“**Compatibility**” testing is specific testing of mobile applications’ suitability with mobile devices. The cause of the compatibility issue comes from android fragmentation, which is given less emphasis in this work.

Choosing the right mobile testing tools (e.g., Appium, Calabash, Ranorex, etc.) can assist in conducting the test types such as white-box, black-box, and regression testing.

Secondly, we observed that from 2016 onwards, the keyword-driven testing (KDT) framework appears to be the most widely applicable test approach, as depicted in Figure 7. In addition, it provides a solution for most of the test objectives and challenges such as reusability, efficiency, functionality, time, complexity, etc. Based on this evidence, we proposed a hybrid architecture called MATF that utilizes keyword-driven testing.

Keyword-driven testing approach is composed of the following components: object repository, keyword library, execution engine, data storage, and an excel sheet for editing purposes. Researchers have shown that the architecture has its benefits including a high degree of reusing test scripts, test abstraction, and ease of the maintenance of the testing process. However, we believe the components of the architecture can be improved by adding domain-specific and high-level keywords in the keyword library. This will help in building a robust keyword library that will have a positive impact on creating reusable test scripts.

The architecture of the improved mobile automation testing framework (MATF) is depicted in Figure 11.

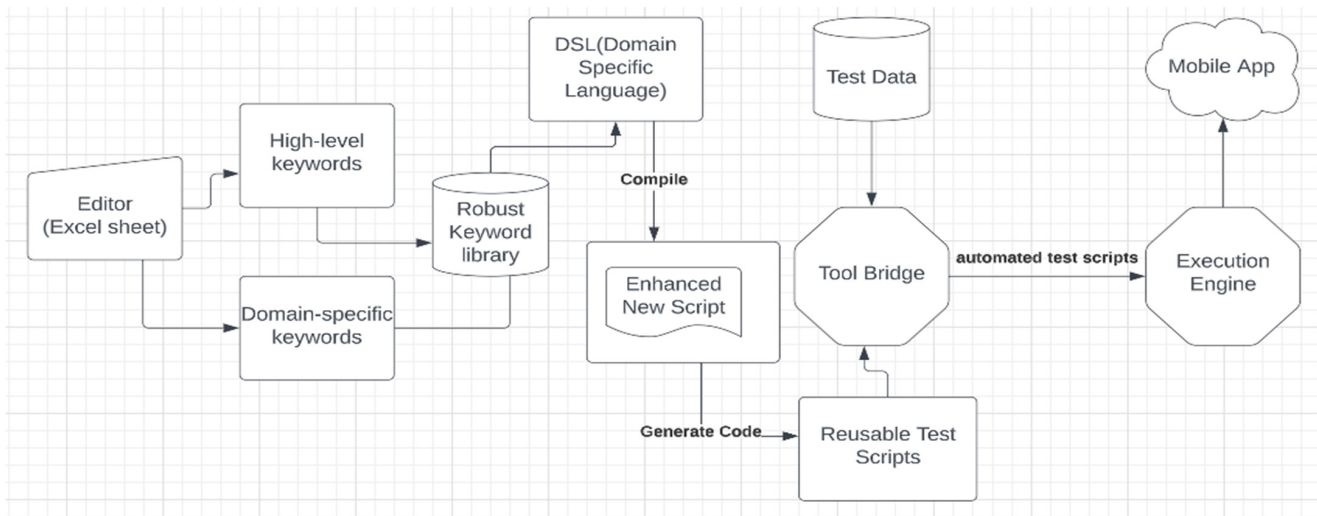


Figure 11. Architecture of MATF framework based on DSL and KDT technology.

The proposed MATF architecture includes the following components:

- **Editor:** An MATF editor is a tool that allows testers to create and edit keyword test scripts. It provides a user-friendly interface for defining high-level and domain-specific keywords. Practically, the MATF editor can be implemented using spreadsheet applications such as Excel;
- **High-level keywords:** These are generic keywords that are used across multiple test cases and are not specific to any mobile application;
- **Domain-specific keywords:** These are keywords that are specific to a particular mobile application. They are used to describe the behavior of the mobile application under test and are typically defined by the test engineers who are familiar with the application;
- **Robust keyword library:** This is another component that acts as a repository of the high-level and domain-specific keywords. The library should be well-organized and easy to navigate, with clear descriptions of each keyword and its purpose;
- **Domain specific language (DSL):** This is a language that is used to define the high-level and domain-specific keywords in the MATF architecture. The DSL should be easy to read and use, with a clear syntax and structure;
- **Reusable test scripts:** These are test scripts that can be used across multiple test cases and applications. They are typically created using a combination of high-level and domain-specific keywords and are designed to be modular and easy to maintain;
- **Tool bridge:** The tool bridge connects the MATF editor to the execution engine, allowing test scripts and test data to be passed between the two components;
- **Execution engine:** This is the component that runs the test scripts and interacts with the mobile app, using the keywords defined in the test scripts to perform actions and verify behavior;
- **Test data:** These are the input data that are required to execute the test cases. The test data are usually stored in a separate file or database;
- **Mobile app:** This is the application under test and interacts with the execution engine using the keywords defined in the test scripts.

The proposed MATF architecture can help address some of the test challenges associated with mobile application testing such as fragmentation, maintenance cost, etc. Table 9 shows what the architecture provides to address the test challenges.

Table 9. MATF architecture to address the test challenges.

	Test Challenges	Test Challenges Description	What MATF Offers
MATF Architecture	Fragmentation	Mobile apps and operating systems (OSs) are highly fragmented due to compatibility issues	Allows testers to create modular and reusable test scripts that can be easily adapted to different device configurations and OSs.
	Complexity	Mobile apps have multiple screens, features, and interactions	Provides a clear and structured way to define and organize test scripts to manage the test suites easily.
	Maintenance cost	Mobile apps are evolving, and new features and updates are released on a regular basis	It provides a modular and reusable framework that can easily be changed and maintained.
	Time consumption	Testing mobile apps across multiple devices can be time-consuming	Allows testers to create test scripts that can be executed across multiple devices.

RQ2: What are the challenges of the existing automation testing frameworks for mobile automation testing?

Given the limitations of earlier test automation approaches (i.e., linear, modular, and library architecture) and the complexity of mobile applications, it is not a surprise that most approaches divert to recent techniques (i.e., data-driven, keyword-driven, model-based, hybrid). Even when we compare the recent approaches, keyword-driven testing was the dominant technology, since it has several benefits that facilitate the testing activities (e.g., easy creation and maintenance of test cases, independent of the application under test and test tools, achieves abstraction with the help of keywords, non-technical personnel can automate the scripts, etc.).

The analysis of the selected articles revealed that keyword-driven testing is a suitable approach for mobile application testing. Keyword-driven testing for mobile applications is advantageous, as it facilitates the creation of automated tests by utilizing and enhancing the existing keywords in the framework.

Keyword-driven testing has its challenges that need to be improved upon. For instance, the keyword reusability feature has room for improvement; it also has a complicated framework, and test cases tend to get complex as compared with data-driven testing. This indicates that the keyword-driven approach needs to pair up with trending automation testing approaches and techniques such as model-driven testing and domain-specific modeling languages to create an efficient hybrid framework that improves the issue of reusability and complexity in mobile applications [57,60].

Finally, to ease the development and maintenance of mobile application testing, efficient and reliable mobile-based keyword-driven testing needs to be developed. This will improve the readability, reusability, and maintainability of the test cases.

7.3. Future Research Directions and Challenges

Even though the articles we investigated have their contributions, some authors posed open challenges and future research directions to call more research attention to the domain. We have summarized the concerns as follows:

- **Addressing mobile fragmentation issue:** Most of the research studies focus on functional and usability defects of mobile apps. Mobile ecosystem fragmentation is not given enough emphasis in the literature. Out of the examined literature, only five research articles discuss the fragmentation of the mobile ecosystem [34,45,60,65,68];
- **Enhancing mobile automation testing frameworks:** The examined literature indicates that there is still room for improvement in mobile automation testing frameworks. According to the investigated literature, a keyword-driven testing framework is a potential candidate for mobile application testing. The framework should be utilized and enhanced to make it suitable for testing mobile apps;

- **Making the existing automation frameworks scalable and compatible with different mobile platforms:** Scalability and compatibility are important test concerns that need to be fulfilled in mobile automation testing. However, these factors were overlooked in the investigated literature. Few research articles mentioned the limitation of the existing mobile testing tools/or frameworks in performing scalable and compatible testing [11,40,45];
- **Developing automation testing techniques and guidelines:** Appropriate testing techniques and guidelines need to be developed for mobile automation testing [5]. These techniques dictate developers and test engineers in creating quality mobile applications for the market. For example, the keyword-driven testing approach and other related guidelines were published by International Organization for Standardization(ISO/EC/IEEE) [27];
- **Evaluation of test tools and framework:** We believe proper validation of mobile testing frameworks should be conducted to check the applicability and capability of test tools. Evaluation of testing tools supports test experts to choose the right tool for their app [32,56];
- **Finally,** in the next part of the research study, we are presenting a novel mobile automation testing framework called MATF that conducts automation testing for mobile apps based on the improved keyword-driven testing technology.

8. Threats to Validity

This section discusses threats to the validity of this study and the measures taken to mitigate them.

On potential misses of related work: We have not considered for our systematic review master's or doctoral theses related to automation testing framework. The threat can be solved by incorporating thesis work in peer-reviewed journals and conferences.

On quality assessment: A researcher conducted the quality assessment checklist. The researcher tried to mitigate this issue by following the SLR process according to the review protocol. The protocol is reviewed to ensure the clarity of the quality criteria.

On data extraction errors: A researcher conducted the data extraction using the data extraction excel form to make the results as formal and error-free as possible. However, we cannot conclude all the investigated papers are precise. Also, the extracted data may not have been reliable for all approaches, and the data aggregation process can still contain errors. Nevertheless, we attempted a cross-checking mechanism on the extracted results to mitigate the issue.

The aspects and the measurements used in this review may not represent characteristics of mobile automated testing. Yet, the test concerns are collected from the examined literature to build the taxonomy and are useful in comparing the existing test automation frameworks.

9. Conclusions

We report in this paper a systematic literature review performed in mobile automation testing frameworks. Our review explored 56 publications that appeared in major conferences, symposiums, and journals. We then proposed a taxonomy of the related research by looking into three dimensions. The first one is the test objectives that focuses on functional and non-functional concerns addressed by the automation testing frameworks. The second one is the test techniques, which involve the test approaches (i.e., data-driven, keyword-driven testing, hybrid, etc.) and test types (i.e., white-box, black-box testing, etc.). We have further investigated the methods used in the literature indicating the strengths and weaknesses of the research papers. Lastly, we have provided open challenges and new research directions for mobile automation testing research such as how to design and build an improved MATF for assuring the quality of mobile apps regarding compatibility and fragmentation issues, etc.

Author Contributions: Conceptualization, N.G.B., C.D., and J.A.V.d.P.; methodology, N.G.B. and C.D.; validation, C.D. and J.A.V.d.P.; writing—original draft preparation, N.G.B.; writing—review and editing, C.D. and J.A.V.d.P.; visualization, N.G.B., C.D. and J.A.V.d.P.; supervision, C.D. and J.A.V.d.P.; project administration, C.D. and J.A.V.d.P.; funding acquisition, J.A.V.d.P. and C.D. All authors have read and agreed to the published version of the manuscript.

Funding: The APC of the paper is funded by page fees from the University of South Africa (Unisa) as well as the Unisa research professor fund of the third author.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Table A1. Examined publications.

Paper Id	Year	Venue	Publication Title
SA1	2022	NILES	Advanced framework for automated testing of mobile applications
SA2	2021	ICAECT	Test automation framework using soft computing techniques
SA3	2020	ICTSS	Architecture based on keyword driven testing with domain specific language for a testing system
SA4	2019	MOBILESoft	EarlGrey: iOS UI automation testing framework
SA5	2019	ICST	On the evolution of keyword-driven test suites
SA6	2019	ISSTA	Ukwikora: continuous inspection for keyword-driven testing
SA7	2019	CCWC	A comparative analysis of quality assurance automated testing tools for windows mobile applications
SA8	2018	ICSST	Environment for automated functional testing of mobile applications
SA9	2018	ICITEE	Performance of automation testing tools for android applications
SA10	2018	SBGAMES	Automated tests for mobile games: an experience report
SA11	2018	ESEC/FSE	Automated support for mobile application testing and maintenance
SA12	2017	IRJET	Comparative study on different mobile application frameworks
SA13	2017	IJMECS	Novel framework for automation testing of mobile applications using Appium
SA14	2017	MOBISYS	Fully automated UI testing system for large-scale android apps using multiple devices
SA15	2017	ICICCS	Uberisation of mobile automation testing
SA16	2017	Software Qual J	Testing of mobile-driven development applications
SA17	2017	JEST	The perceived usability of automated testing tools for mobile applications
SA18	2016	J.Procs	The impacts of test automation on software's Cost, quality and time to market
SA19	2016	IEEE Software	Test automation not just for test execution
SA20	2016	ICT4M	Software testing techniques: a literature review
SA21	2016	ICCSNT	The design and implement of the cross-platform mobile automated testing framework
SA22	2016	CCIS	Research on automated testing framework for multi-platform mobile applications
SA23	2016	J4R	Deployment of Calabash automation framework to analyze the performance of an android application

Table A1. Cont.

Paper Id	Year	Venue	Publication Title
SA24	2016	ISO/IEC/IEEE	Software and systems engineering- software testing- keyword-driven testing
SA25	2015	ASE	Automated test input generation for android: are we their yet?
SA26	2015	ASE	Testing cross-platform mobile app development frameworks
SA27	2015	IJAERS	Survey on automation testing tools for mobile applications
SA28	2014	ICACCCT	An automated testing framework for testing android mobile applications in the cloud
SA29	2014	IJCSMC	A comparative study of software testing techniques
SA30	2014	IJCET	Latest research and development on software testing techniques and tools
SA31	2014	IJACSA	A review of scripting techniques used in automated software testing
SA32	2014	IJCSE	A study on variations of bottlenecks in software testing
SA33	2014	IJICT	A strategic approach to software testing
SA34	2014	IJMAS	Android mobile automation framework
SA35	2014	ETFA	Adapting keyword driven test automation framework to IEC 61131-3 industrial control applications using PLCopen XML
SA36	2014	IJCET	Automated testing of mobile applications using scripting technique: a study on Appium
SA37	2013	ESEC/FSE	Dynodroid: An input generation system for android apps
SA38	2013	AMM	Keyword-driven automation test
SA39	2013	ICCSEE	Keyword-driven testing framework for android applications
SA40	2012	MUM	Testdroid: automated remote UI testing on android
SA41	2012	ASE	Using GUI ripping for automated testing of android applications
SA42	2012	ICSEA	MobiTest: a cross-platform tool for testing mobile applications
SA43	2012	ICCIS	A novel approach of automation testing on mobile devices
SA44	2011	ACIS	An integrated test automation framework for testing on heterogeneous mobile platforms
SA45	2011	AST	Automating GUI testing for android applications
SA46	2011	ECSA	A model-driven approach for automating mobile applications testing
SA47	2011	ICSTW	A GUI crawling-based technique for android mobile application testing
SA48	2011	ICST	Providing a software quality framework for testing of mobile applications
SA49	2011	ICSTW	Model-based testing with a general purpose keyword-driven test automation framework
SA50	2010	IJCA	Software testing-goals, principles, and limitations
SA51	2010	ICSE	Test automation on mobile device
SA52	2009	QSIC	An adapter framework for keyword-driven testing
SA53	2009	ICUIMC	Performance testing based on test-driven development for mobile applications
SA54	2009	WCSE	Design and implementation of GUI Automated testing framework based on XML
SA55	2008	ICAL	Towards adaptive framework of keyword-driven automation testing
SA56	2008	CSSE	LKDT: A keyword-driven based distributed test framework

References

- Rafi, D.M.; Moses, K.R.K.; Petersen, K.; Mäntylä, M.V. Benefits and limitations of automated software testing: Systematic literature review and practitioner survey. In Proceedings of the 2012 7th International Workshop on Automation of Software Test, Zurich, Switzerland, 2–3 June 2012; pp. 36–42. [[CrossRef](#)]
- Hanna, M.; El-Haggag, N.; Sami, M. A Review of Scripting Techniques Used in Automated Software Testing. *Int. J. Adv. Comput. Sci. Appl.* **2014**, *5*, 194–202. [[CrossRef](#)]

3. Singh, A.S.; Gadgil, A.R.; Chudgor, A.A. Automated Testing of Mobile Applications using Scripting Technique: A Study on Appium. *Int. J. Curr. Eng. Technol. India Accept.* **2014**, *362744*, 3627–3630.
4. Aebersold, K. Test Automation Framework. Available online: <https://smartbear.com/learn/automated-testing/test-automation-frameworks/> (accessed on 1 June 2021).
5. Jamil, M.A.; Arif, M.; Abubakar, N.S.A.; Ahmad, A. Software testing techniques: A literature review. In Proceedings of the Proceedings—6th International Conference on Information and Communication Technology for the Muslim World, ICT4M, Jakarta, Indonesia, 22–24 November 2016. [CrossRef]
6. Muccini, H.; Informatica, D.; Di Francesco, A.; Informatica, D.; Esposito, P.; Informatica, D. Software Testing of Mobile Applications: Challenges and Future Research Directions. In Proceedings of the 7th International Workshop on Automation of Software Test (AST), Zurich, Switzerland, 2–3 June 2012; pp. 29–35. [CrossRef]
7. Tramontana, P.; Amalfitano, D.; Amatucci, N. Automated functional testing of mobile applications: A systematic mapping study. *Softw. Qual. J.* **2019**, *149*–201. [CrossRef]
8. Kumar, D.; Mishra, K.K. The Impacts of Test Automation on Software’s Cost, Quality and Time to Market. In Proceedings of the Procedia Computer Science, Mumbai, India, 26–27 February 2016; Volume 79. [CrossRef]
9. Idri, A.; Moumane, K.; Abran, A. On the use of software quality standard ISO/IEC9126 in mobile environments. In Proceedings of the 2013 20th Asia-Pacific Software Engineering Conference (APSEC), Bangkok, Thailand, 2–5 December 2013; Volume 1, pp. 1–8. [CrossRef]
10. Quadri, S.M.; Farooq, S.U. Software Testing—Goals, Principles, and Limitations. *Int. J. Comput. Appl.* **2010**, *6*, 7–10. [CrossRef]
11. Mohammad, D.R.; Al-Momani, S.; Tashtoush, Y.M.; Alsmirat, M. A comparative analysis of quality assurance automated testing tools for windows mobile applications. In Proceedings of the 2019 IEEE 9th Annual Computing and Communication Workshop and Conference, CCWC 2019, Las Vegas, NV, USA, 7–9 January 2019; pp. 414–419. [CrossRef]
12. Kirubakaran, B.; Karthikeyani, V. Mobile application testing—Challenges and solution approach through automation. In Proceedings of the 2013 International Conference on Pattern Recognition, Informatics and Mobile Engineering, Salem, India, 21–22 February 2013; pp. 79–84. [CrossRef]
13. Usability. 2021. Available online: <https://www.interaction-design.org/literature/topics/usability> (accessed on 1 August 2021).
14. Sheetal Sharma, A.J. An efficient Keyword Driven Test Automation Framework for Web Applications. *Int. J. Eng. Sci. Adv. Technol.* **2012**, *2*, 600–604.
15. Hayes, L.G. *The Automated Testing Handbook*, 2nd ed.; Software Testing Institute, 1 March 2004. Available online: https://books.google.com.hk/books/about/The_Automated_Testing_Handbook.html?id=-jangThcGikC&redir_esc=y (accessed on 1 August 2021).
16. Wu, Z.; Liu, S.; Li, J.; Liao, Z. Keyword-Driven Testing Framework For Android Applications. In *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*; Atlantis Press: Paris, France, 2013; pp. 1096–1102. [CrossRef]
17. Corral, L.; Sillitti, A.; Succi, G. Software assurance practices for mobile applications. *Computing* **2015**, *97*, 1001–1022. [CrossRef]
18. Sahinoglu, M.; Incki, K.; Aktas, M.S. Mobile Application Verification: A Systematic Mapping Study. In *Proceedings of the Computational Science and Its Applications—ICCSA*; Springer International Publishing: Cham, Switzerland; Banff, AB, Canada, 22–25 June 2015; Volume 9159, pp. 147–163. [CrossRef]
19. Zein, S.; Salleh, N.; Grundy, J. A systematic mapping study of mobile application testing techniques. *J. Syst. Softw.* **2016**, *117*, 334–356. [CrossRef]
20. Kong, P.; Li, L.; Gao, J.; Liu, K.; Bissyande, T.F.; Klein, J. Automated Testing of Android Apps: A Systematic Literature Review. *IEEE Trans. Reliab.* **2019**, *68*, 45–66. [CrossRef]
21. Singh, J.; Sahu, S.K.; Singh, A.P. Implementing Test Automation Framework Using Model-Based Testing Approach. In *Intelligent Computing and Information and Communication; Advances in Intelligent Systems and Computing*; Springer: Singapore, 2018; pp. 695–704. [CrossRef]
22. Linares-Vasquez, M.; Moran, K.; Poshyvanyk, D. Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing. In Proceedings of the 2017 IEEE International Conference on Software Maintenance and Evolution (ICSME), Shanghai, China, 17–22 September 2017; pp. 399–410. [CrossRef]
23. Ahmad, A.; Li, K.; Feng, C.; Asim, S.M.; Yousif, A.; Ge, S. An Empirical Study of Investigating Mobile Applications Development Challenges. *IEEE Access* **2018**, *6*, 17711–17728. [CrossRef]
24. Wang, J.; Wu, J. Research on Mobile Application Automation Testing Technology Based on Appium. In Proceedings of the 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), 14–15 September 2019; IEEE: Jishou, China; pp. 247–250. [CrossRef]
25. Luo, C.; Goncalves, J.; Velloso, E.; Kostakos, V. A Survey of Context Simulation for Testing Mobile Context-Aware Applications. *ACM Comput. Surv.* **2020**, *53*, 1–39. [CrossRef]
26. Keele, S. *Guidelines for performing Systematic Literature Reviews in Software Engineering*; ACM: New York, NY, USA, 2007. Available online: <https://dl.acm.org/doi/10.1145/1134285.1134500> (accessed on 1 March 2020).
27. ISO/IEC/IEEE 29119-5:2016; Software and Systems Engineering—Software Testing—Part 5: Keyword-Driven Testing. International Organization for Standardization; International Electrotechnical Commission; Institute of Electrical and Electronics Engineers: Geneva, Switzerland, 2016. Available online: <https://standards.ieee.org/ieee/29119-5/5563/> (accessed on 16 April 2020).

28. Rwemalika, R.; Kintis, M.; Papadakis, M.; Le Traon, Y.; Lorrach, P. Ukwikora: Continuous inspection for keyword-driven testing. In Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis, Beijing, China, 15–19 July 2019; pp. 402–405. [[CrossRef](#)]
29. Wu, Z.Q.; Li, J.Z.; Liao, Z.Z. Keyword Driven Automation Test. *Appl. Mech. Mater.* **2013**, *427–429*, 652–655. [[CrossRef](#)]
30. Pereira, R.B.; Brito, M.A.; Machado, R.J. Architecture Based on Keyword Driven Testing with Domain Specific Language for a Testing System. In Proceedings of the International Conference on Testing Software and Systems (ICTSS), Naples, Italy, 9–11 December 2020; pp. 310–316. [[CrossRef](#)]
31. Rwemalika, R.; Kintis, M.; Papadakis, M.; Le Traon, Y.; Lorrach, P. On the Evolution of Keyword-Driven Test Suites. In Proceedings of the 2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST), Xi'an, China, 22–27 April 2019; pp. 335–345. [[CrossRef](#)]
32. Hussain, A.; Razak, H.A.; Mkpjoiguo, E.O.C. The perceived usability of automated testing tools for mobile applications. *J. Eng. Sci. Technol.* **2017**, *12*, 86–93.
33. Zhifang, L.; Bin, L.; Xiaopeng, G. Test automation on mobile device. In Proceedings of the 5th Workshop on Automation of Software Test, 3–4 May 2010; ACM: Cape Town, South Africa; pp. 1–7. [[CrossRef](#)]
34. Fazzini, M. Automated support for mobile application testing and maintenance. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Lake Buena Vista, FL, USA, 4–9 November 2018; pp. 932–935. [[CrossRef](#)]
35. Machiry, A.; Tahiliani, R.; Naik, M. Dynodroid: An input generation system for android apps. In Proceedings of the 2013 9th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE 2013—Proceedings, Saint Petersburg, Russia, 18–26 August 2013; pp. 224–234. [[CrossRef](#)]
36. Jie, H.; Lan, Y.; Luo, P.; Guo, S.; Gao, J. LKDT: A Keyword—Driven based distributed test framework. In Proceedings of the Proceedings—International Conference on Computer Science and Software Engineering, CSSE 2008, Wuhan, China, 12–14 December 2008; Volume 2, pp. 719–722. [[CrossRef](#)]
37. Vajak, D.; Grbic, R.; Vranjes, M.; Stefanovic, D. Environment for Automated Functional Testing of Mobile Applications. In Proceedings of the 2018 International Conference on Smart Systems and Technologies (SST), Osijek, Croatia, 10–12 October 2018; pp. 125–130. [[CrossRef](#)]
38. Gunasekaran, S.; Bargavi, V. Survey on Automation Testing Tools for Mobile Applications. *Int. J. Adv. Eng. Res. Sci.* **2015**, *2*, 2349–6495. Available online: www.ijaers.com (accessed on 10 July 2021).
39. Boushehrinejadmoradi, N.; Ganapathy, V.; Nagarakatte, S.; Iftode, L. Testing Cross-Platform Mobile App Development Frameworks (T). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–13 November 2015; pp. 441–451. [[CrossRef](#)]
40. Lovreto, G.; Endo, A.T.; Nardi, P.; Durelli, V.H.S. Automated Tests for Mobile Games: An Experience Report. In Proceedings of the 2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Foz do Iguacu, Brazil, 29 October–1 November 2018; Volume 2018-Novem, pp. 48–488. [[CrossRef](#)]
41. Kaur Chauhan, A.R.; Singh, B.Ä.I. Latest Research and Development on Software Testing Techniques and Tools. *Int. J. Curr. Eng. Technol.* **2014**, *4*, 2368–2372. Available online: <http://inpressco.com/category/ijcet> (accessed on 10 September 2021).
42. Hu, C.; Neamtui, I. Automating GUI testing for Android applications. In Proceedings of the 6th International Workshop on Automation of Software Test, 23–24 May 2011; ACM: Honolulu, HI, USA; pp. 77–83. [[CrossRef](#)]
43. Singh, K.; Mishra, S.K. A Strategic Approach to Software Testing. *Int. J. Inf. Comput. Technol.* **2014**, *4*, 1387–1394.
44. Prathibhan, C.M.; Malini, A.; Venkatesh, N.; Sundarakantham, K. An automated testing framework for testing Android mobile applications in the cloud. In Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control and Computing Technologies, Ramanathapuram, India, 8–10 May 2014; pp. 1216–1219. [[CrossRef](#)]
45. Choudhary, S.R.; Gorla, A.; Orso, A. Automated Test Input Generation for Android: Are We There Yet? (E). In Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA, 9–13 November 2015; pp. 429–440. [[CrossRef](#)]
46. Amalfitano, D.; Fasolino, A.R.; Tramontana, P.; De Carmine, S.; Memon, A.M. Using GUI ripping for automated testing of Android applications. In Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering, Essen, Germany, 3–5 September 2012; pp. 258–261. [[CrossRef](#)]
47. Azim, T.; Neamtui, I. Targeted and depth-first exploration for systematic testing of android apps. In Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, Indianapolis, IN, USA, 29–31 October 2013; pp. 641–660. [[CrossRef](#)]
48. Mu, B.; Zhan, M.; Hu, L. Design and Implementation of GUI Automated Testing Framework Based on XML. In Proceedings of the 2009 WRI World Congress on Software Engineering, Xiamen, China, 19–21 May 2009; pp. 194–199. [[CrossRef](#)]
49. Tirodkar, A.A.; Khandpur, S.S. EarlGrey: iOS UI Automation Testing Framework. In Proceedings of the 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft), Montreal, QC, Canada, 25–25 May 2019; pp. 12–15. [[CrossRef](#)]
50. Sinaga, A.M.; Wibowo, P.A.; Silalahi, A.; Yolanda, N. Performance of Automation Testing Tools for Android Applications. In Proceedings of the 2018 10th International Conference on Information Technology and Electrical Engineering (ICITEE), Bali, Indonesia, 24–26 July 2018; pp. 534–539. [[CrossRef](#)]

51. Kim, H.; Choi, B.; Yoon, S. Performance testing based on test-driven development for mobile applications. In *Proceedings of the International Conference on Ubiquitous Information Management and Communication*; ACM: Suwon, Republic of Korea, 2009; pp. 612–617. [CrossRef]
52. Kishan Kulkarni Soumya, M.A. Deployment of Calabash Automation Framework to Analyze the Performance of an Android Application. *J. Res.* **2016**, *02*, 70–75. Available online: www.journalforresearch.org (accessed on 5 July 2022).
53. Kannan, S.; Pushparaj, T. A Study on Variations of Bottlenecks in Software Testing. *Int. J. Comput. Sci. Eng.* **2014**, *2*, 8–14. Available online: https://www.ijcsonline.org/pdf_paper_view.php?paper_id=150&IJCSE-00256.pdf (accessed on 5 October 2021).
54. Song, H.; Ryoo, S.; Kim, J.H. An Integrated Test Automation Framework for Testing on Heterogeneous Mobile Platforms. In *Proceedings of the 2011 First ACIS International Symposium on Software and Network Engineering*, Seoul, Republic of Korea, 19–20 December 2011; pp. 141–145. [CrossRef]
55. Alotaibi, A.A.; Qureshi, R.J. Novel Framework for Automation Testing of Mobile Applications using Appium. *Int. J. Mod. Educ. Comput. Sci.* **2017**, *9*, 34–40. [CrossRef]
56. Zun, D.; Qi, T.; Chen, L. Research on automated testing framework for multi-platform mobile applications. In *Proceedings of the 2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Beijing, China, 17–19 August 2016; pp. 82–87. [CrossRef]
57. Pajunen, T.; Takala, T.; Katara, M. Model-Based Testing with a General Purpose Keyword-Driven Test Automation Framework. In *Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, Berlin, Germany, 21–25 March 2011; pp. 242–251. [CrossRef]
58. Marín, B.; Gallardo, C.; Quiroga, D.; Giachetti, G.; Serral, E. Testing of model-driven development applications. *Softw. Qual. J.* **2017**, *25*, 407–435. [CrossRef]
59. Kolawole, G. Model Based Testing Mobile Applications: A Case Study of Moodle Mobile Application. Master’s Thesis, Tallinn University of Technology, Tallinn, Estonia, 2017.
60. Ridene, Y.; Barbier, F. A model-driven approach for automating mobile applications testing. In *Proceedings of the 5th European Conference on Software Architecture: Companion Volume*, 13–16 September 2011; ACM: Essen, Germany; pp. 1–7. [CrossRef]
61. Bansal, A. A Comparative Study of Software Testing Techniques. *Int. J. Comput. Sci. Mob. Comput.* **2014**, *3*, 579–584. Available online: http://link.springer.com/10.1007/978-3-319-59647-1_27 (accessed on 1 March 2020).
62. Salam, M.A.; Taha, S.; Hamed, M.G. Advanced Framework for Automated Testing of Mobile Applications. In *Proceedings of the 2022 4th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, Giza, Egypt, 22–24 October 2022; pp. 233–238. [CrossRef]
63. Swathi, B.; Tiwari, H. Test Automation Framework using Soft Computing Techniques. In *Proceedings of the 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, Bhilai, India, 19–20 February 2021; pp. 1–4. [CrossRef]
64. Seth, P.; Rane, N.; Wagh, A.; Katade, A.; Sahu, S.; Malhotra, N. Uberisation of mobile automation testing. In *Proceedings of the 2017 International Conference on Intelligent Computing and Control Systems (ICICCS)*, Madurai, India, 15–16 June 2017; pp. 181–184. [CrossRef]
65. Garousi, V.; Elberzhager, F. Test Automation: Not Just for Test Execution. *IEEE Softw.* **2017**, *34*, 90–96. [CrossRef]
66. Raut, P.; Tomar, S. Android Mobile Automation Framework. *Int. J. Multidiscip. Approach Stud. (IJMAS)* **2014**, *1*, 1–12. Available online: <http://ijmas.com/upcomingissue/1.06.2014.pdf> (accessed on 1 March 2020).
67. Peltola, J.; Sierla, S.; Vyatkin, V. Adapting Keyword driven test automation framework to IEC 61131-3 industrial control applications using PLCopen XML. In *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, Barcelona, Spain, 16–19 September 2014; pp. 1–8. [CrossRef]
68. Kaasila, J.; Ferreira, D.; Kostakos, V.; Ojala, T. Testdroid:automated remote UI testing on Android. In *Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia*, Ulm, Germany, 4–6 December 2012; pp. 1–4. [CrossRef]
69. Nagowah, L.; Sowamber, G. A novel approach of automation testing on mobile devices. In *Proceedings of the 2012 International Conference on Computer & Information Science (ICCIS)*, Kuala Lumpur, Malaysia, 12–14 June 2012; Volume 2, pp. 924–930. [CrossRef]
70. Amalfitano, D.; Fasolino, A.R.; Tramontana, P. A GUI Crawling-Based Technique for Android Mobile Application Testing. In *Proceedings of the 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops*, Berlin, Germany, 21–25 March 2011; pp. 252–261.
71. Franke, D.; Weise, C. Providing a Software Quality Framework for Testing of Mobile Applications. In *Proceedings of the 2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, Berlin, Germany, 21–25 March 2011; pp. 431–434.
72. Tang, J.; Cao, X.; Ma, A. Towards adaptive framework of keyword driven automation testing. In *Proceedings of the 2008 IEEE International Conference on Automation and Logistics, Qingdao, China, 1–3 September 2008*; IEEE: Piscataway, NJ, USA, 2008; pp. 1631–1636. [CrossRef]
73. Takala, T.; Maunumaa, M.; Katara, M. An Adapter Framework for Keyword-Driven Testing. In *Proceedings of the 2009 Ninth International Conference on Quality Software*, Jeju, Republic of Korea, 24–25 August 2009; pp. 201–210.
74. Cherednichenko, S. What’s the Cost to Maintain and Support an App in 2021. 2021. Available online: <https://www.mobindustry.net/blog/whats-the-cost-to-maintain-and-support-an-app-in-2020/> (accessed on 20 December 2021).

75. Anusha, M.; Kn, S. Comparative Study on Different Mobile Application Frameworks. *Int. Res. J. Eng. Technol.* **2017**, *4*, 1299–1300. Available online: <https://irjet.net/archives/V4/i3/IRJET-V4I3306.pdf> (accessed on 1 March 2020).
76. Ki, T.; Simeonov, A.; Park, C.M.; Dantu, K.; Ko, S.Y.; Ziarek, L. Demo: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, 19–23 June 2017; ACM: Niagara Falls, ON, Canada; New York, NY, USA; p. 185. [[CrossRef](#)]
77. Bayley, I.; Flood, D.; Harrison, R.; Martin, C. MobiTest: A Cross-Platform Tool for Testing Mobile Applications. In Proceedings of the ICSEA 2012: The Seventh International Conference on Software Engineering Advances, Lisbon, Portugal, 18–23 November 2012; pp. 619–622. Available online: http://www.thinkmind.org/index.php?view=article&articleid=icsea_2012_22_20_10114 (accessed on 3 July 2022).

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.