Nigerian Journal of Technology (NIJOTECH)
Vol. 44, No. 2, June, 2025, pp.311 - 337
www.nijotech.com

Print ISSN: 0331-8443
Electronic ISSN: 2467-8821
https://doi.org/10.4314/njt.v44i2.15

A SURVEY OF AUTOMATED TESTING TECHNIQUES FOR ANDROID-BASED MOBILE APPLICATIONS

AUTHORS:
N. O. Eke[1], I. A. Salihu[2,*], A. Usman[3], R. Ibrahim[4], and Y. Mshelia[5]

AFFILIATIONS:
[1]Wigwe University, Isiokpo, Rivers State, Nigeria.
[2,4,5]Department of Software Engineering, Nile University of Nigeria, Nigeria.
[3]Department of Computer Science, Nile University of Nigeria, Nigeria.

***CORRESPONDING AUTHOR:**
Email: ibrahim.salihu@nileuniversity.edu.ng

ARTICLE HISTORY:
Received: 19 October, 2024.
Revised: 22 May, 2025.
Accepted: 05 June, 2025.
Published: 07 July, 2025.

KEYWORDS:
Mobile Application Testing, Systematic mapping, Software Testing, Test Automation, Model-Based Testing.

ARTICLE INCLUDES:
Peer review

DATA AVAILABILITY:
On request from author(s)

EDITORS:
Chidozie Charles Nnaji

FUNDING:
None

Abstract

To conduct a survey of Mobile Applications Testing (MAT) to determine the research contributions in mobile applications testing such as the test approach, test strategy, testing techniques and evaluation methods used in mobile applications testing, as well as to determine the publication frequency and the publication region. This study adopted the guidelines provided by Kitchenham and Charters, and Petersen et al. for conducting this systematic mapping study. A total of 242 studies were selected using predefined inclusion/exclusion criteria. Studies were retrieved from five major academic databases (IEEE Xplore, ACM Digital Library, ScienceDirect, Web of Science, and EBSCOhost) using validated search strings. Findings show that MAT publications increased steadily between 2009 and 2022, with 2018 recording the highest number (n = 33). China and United States were the most active contributors. Model-based testing emerged as the most commonly used testing technique, while fault detection and code coverage were the most widely adopted evaluation methods. Dynodroid, with 952 citations and an NCII score of 79.3, was identified as the most influential MAT-related study. This study presents a structured overview of MAT research trends, methods, and influential works, offering a valuable reference for researchers and practitioners in the software testing community.

.

1.0 INTRODUCTION

Computer technology is developing quickly. In recent years, computers have evolved from desktops to laptops and, more recently, a range of portable mobile devices. This movement has profoundly altered the way we govern, run our businesses, and live our lives. It has also had a big impact on how testers and software engineers perform their work [1].

Over 371 million smartphones were bought in the fourth quarter of 2021, and approximately 1.54 billion smartphones were sold to consumers globally in that year [2], [3]. In order to meet the computational demands of its users, mobile applications (both web and native apps) for smartphones and tablets have increased in development in tandem with the rising demand for these devices [4], [5]. A recent study shows that 230 billion mobile applications (Mobile Apps) were downloaded worldwide in 2021 alone [6]. Mobile applications otherwise referred to as Mobile Apps are software programs that run on mobile devices such as smartphones, tablet PCs, and other mobile devices with the capability to run 48 network-

HOW TO CITE:
Eke, N. O., Salihu, I. A., Usman, A., Ibrahim, R., and Mshelia, Y. "A Survey of Automated Testing Techniques for Android-Based Mobile Applications", *Nigerian Journal of Technology*, 2025; 44(2), pp. 311 – 337; https://doi.org/10.4314/njt.v44i2.15

based applications over a cellular or satellite data link [1], [7]. Determining the quality of mobile applications—such as their functionality, behaviour, and performance under specific conditions—has become extremely difficult for software engineering experts, especially software testers, due to the rise in the number of mobile apps and their usage [5], [8].

Software testing is a technique used to guarantee that software programs have high-quality in terms of its features, functionality, behaviors, and performance [9]. Testing according to Morgado et al. [10] is essential in improving the product's quality, hence, a central part of the mobile development. In software testing, test-approach refers to the general guideline, perspective or philosophy adopted by a software tester when designing the test cases. The main test approaches include white-box, black-box, and grey-box. A test strategy outlines the approach and objectives of testing; providing a framework for the testing process and serves as a reference for the entire testing effort. It is more about the "what", "why", and "when" of testing. Test technique, on the other hand, provides specific methods or procedures used to design and execute test cases. It guides testers in creating efficient and effective tests that can uncover defects in software. Despite its significance in any apps development, testing mobile apps is challenging and expensive as a result of numerous factors, such as the short time period to market, and diverse underlining technologies [11], [12]. Several researchers have explored numerous methods, approaches, and techniques of testing Mobile Apps recently and many testing tools were proposed. However, another systematic literature review is needed to address new research questions, update existing knowledge, or to provide fresh insights about new challenges in the area. To ascertain the frequency of publications and the publication region, the study aims to investigate the research contributions in mobile app testing with a particular focus on test approaches, strategies, and techniques used in testing mobile apps as well as the method used to evaluate the performance of the existing tools.

This is how the remainder of the paper is structured. The associated works are discussed in Section 2. The methodology is discussed in Section 3. In Section 4, the survey's findings were examined. Lastly, section 5 offers a recommendation for additional study to wrap up the work.

2.0 RELATED WORKS

There is number of surveys [255], systematic mapping [12][19] and systematic review [20][24] studies on mobile testing conducted in recent times. Though systematic mapping may seem like a systematic literature review in terms of the methodology used in performing the literature search and study selection [17], they are different in area of data analysis. The former focuses on structuring research areas, while the latter is aimed at synthesizing evidence. Table 1 shows the summary of existing systematic reviews.

Zein et al. [12] conducted a comprehensive systematic mapping study (SMS) for mobile and Smartphone apps testing. Their study argued that most related studies on mobile app testing as at the time of their research had limited their focus on the test automation area without real inclusion/exclusion selection criteria, hence, making those studies a subject of biased selection. Consequently, the authors elaborated their study to include other areas of interest in mobile app testing such as context-awareness, usability, test automation, and security testing. Furthermore, the study investigated other issues of mobile application testing like mobile services testing, testing of life cycle conformance, and test metrics.

In another study, "A systematic mapping study on cloud-based mobile application testing" by Ya'u et al. [13], the authors conducted a systematic mapping of 23 research studies to investigate the testing of cloud-based mobile apps and the impact of Testing-as-a-Service (TaaS). The study identified a lack of general testing approaches to accommodate different OS or platforms. According to the authors, most of the studies focus on Android mobile apps neglecting other popular mobile platforms such as Symbian, Blackberry, iOS and Web OS. The systematic mapping reveals insufficient empirical evaluation employed in evaluating the proposed approaches.

Tramontana et al [14] conducted a SMS to investigate the techniques and tools for the functional testing automation of mobile apps. The study reveals that there are inadequate studies in areas such as tools and techniques for testing iOS applications, testing tools on Android targeting C++ based components of Android mobile Apps. Other research gaps identified by the study include fault detection testing, context-aware testing and concurrency testing.

Wimalasooriya et al. [15] conducted a SMS of Mobile applications reliability testing. The study observed lack of research in understanding reliability testing regarding context-awareness; handling of runtime event, self-healing, aging and rejuvenation, then suggested further research on these areas. Another mapping study by Sahinoglu et al. [16] on the existing

mobile application testing observed low research interest on performance testing of mobile applications, and prioritization of system testing.

Petersen et al. [17] proposed an updated guideline for conducting systematic mapping studies in software engineering, while Banerjee et al. [18] conducted a mapping study of 230 articles on Graphical User Interface (GUI) testing. The most recent systematic mapping study by Nie et al. [19] examines 114 research studies published from 2011 to 2022. The authors performed bibliometric and qualitative analysis of the studies to ascertain the most researched topics, author's community, evaluation metrics and the approaches employed in the primary studies.

In terms of literature reviews, Kong et al. [20] provided a thorough analysis of 103 pertinent research articles that were published in prestigious conferences and journals up until 2016. The authors also offered new research directions to guarantee the quality of apps, concentrating on compatibility problems and updates that cause vulnerabilities.

A set of guidelines for performing systematic literature reviews in software engineering was put forth by Kitchenham et al. [21]. In 2022, Kaur and Kaur [23] carried out an SLR with the goal of determining and contrasting the various test estimation methods for traditional software. To determine the trends of static analysis methodologies, their current state, and potential research areas, Li et al. [24] conducted a systematic literature review of 124 studies on static analysis of Android apps.

In this study, an in-dept survey of 242 research studies focusing on automated testing of mobile apps was carried out with the goal of determining the research contributions in the area automated testing of mobile apps, the test approach, the test strategy and testing techniques as well as the evaluation methods employed by the various techniques.

Table 1: Summary of existing systematic mapping studies

Authors	Year	Study Objectives	No. of Articles	Limitation
Zein et al. [12]	2016	Organize and classify published research evidence covering techniques and challenges in the field of mobile apps testing.	79	There aren't many studies that base their research on actual mobile application development environments.
Ya'u et al. [13]	2019	Carried out a SMS on the testing of cloud-based mobile apps.	23	There is insufficient research on gathering testing requirements during the requirement engineering stage.
Tramontana et al. [14]	2019	Offers a classification of scientific material pertaining to the functional testing automation for mobile apps. Solutions were offered for the problems that have been found.	131	Insufficient universal and scalable methods to accommodate the different kinds of mobile app testing for apps running on different platforms, including Web OS, Symbian, Blackberry, and iOS.
Wimalasooriya et al. [15]	2022	Examine the most recent developments in the field of mobile app dependability.	87	Insufficient case studies or other evaluation techniques to support the suggested strategies.
Sahinoglu et al. [16]	2015	Presents a SMS of software verification in the area of mobile apps	123	Strong prevalence of Android-based approaches.
Petersen et al. [17]	2015	Determined the method used to carry out the systematic mapping procedure.	52	Lack of contributions from the industry.
Banerjee et al. [18]	2013	Identify improvement potential in conducting some practices of systematic review guidelines.	136	Lack of specialized journals and venues for automating mobile testing.

Among the studies reviewed in the literature, the study conducted by Zein et al. [12] seems close but different from our study. While the work presented by Zein et al. considered 79 research articles published from 2005 to 2015. Though the study was comprehensive as at the time of the study, a number of new techniques have emanated recently, hence the need to identify and incorporate these newly proposed techniques. The study also did not consider the relationships among the study authors. This study covers relevant research studies published from 2005 to 2025, making our study more elaborate while capturing the concentration of research and the current trends in the different areas of mobile apps testing. This study also analyzes the selected studies to identify the most influential study relating to mobile application testing (MAT).

3.0 METHODOLOGY

This study adopted the guideline provided by Kitchenham and Charters [21] and Petersen et al. [17] in conducting a systematic mapping study to systematically analyze Mobile Apps Testing (MAT) studies to ascertain the publication frequency, region, the testing techniques and strategies for mobile apps, as well as the test automation tools proposed by research community as well as the authors the co-relationship among author. The three primary stages

of the systematic mapping procedure used in this work include (1) planning review, (2) search, and (3) reporting. As seen in Figure 1, each step of the methodical mapping process is further broken down into stages.

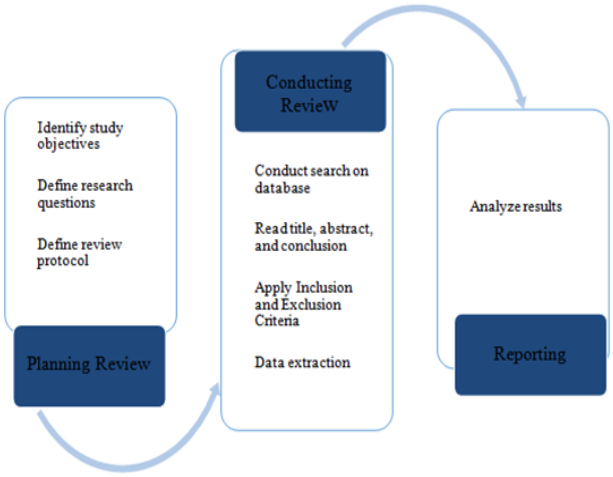


Figure 1: The systematic mapping process

Planning review phase as shown in Figure 1 involves (i) identifying study objectives, (ii) defining research questions to guide the process, and (iii) defining review protocol. The second phase is conducting reviews such as (i) conducting a search on the database, (ii) applying inclusion and exclusion criteria, (iii) keyword abstraction, and (iv) data extraction. Finally, the reporting phase analyzes the results.

3.1 Research Questions

This study aims to survey the research contributions in the area of Mobile Apps Testing (MAT). Kitchenham and Charters guidelines were adopted in the study [21]. The authors conclude that the most crucial step in carrying out a systematic mapping study is identifying the research question or questions that the investigation is meant to answer [21]. The purpose of this is to answer the following research questions (RQs): -

1. RQ1. How frequently are MAT research studies published, and what are the prevalent themes in MAT research?
2. RQ2. Which testing approach and test strategy do these studies apply?
3. RQ3. Which testing technique do the studies use?
4. RQ4. How were these testing techniques evaluated /validated?
5. RQ5. Which study is the most influential MAT-related study?

3.2 Literature Search

The literature search was conducted on five popular academic databases such as Web of Science (WoS), IEEE Xplore, ACM Digital Library, ScienceDirect, and EBSCOhost databases. These databases were chosen due to their broad and reputable coverage of peer-reviewed computing and software engineering literature, particularly in the domain of software testing. Other databases were excluded to overlap and ensure quality control, as the selected sources already index many of the top-ranked journals in the field. Mobile applications testing-focused scholarly articles published between 2005 and 2025 were considered in the study. This timeline was chosen following the earliest MAT related study obtained during the search justifying the study conducted by [12] which revealed that the earliest MAT-focused paper was published in 2005 and further confirmed in the cause of carrying out this research.

3.3 Search Strategy

The search strings were mainly formulated according to the research questions. The primary terms found were testing and mobile applications, which were grouped into sets. Their synonyms were considered while creating the search string using the approach recommended by Zein et al. [12], and Kitchenham and Charters [21] as illustrated below:

a. Search for alternative keywords and synonyms.
b. Use Boolean “OR” to include alternative spellings and synonyms.
c. Use Boolean “AND” to link and include other keywords.

Additional terms were added as synonyms such as “verification”, “technique”, context-aware, and “approach” to expand the search to ensure wider coverage. After arriving at several search strings, the search results were validated by examining their ability to retrieve a set of 16 known Mobile Application testing studies in Table 2, then, used to form the search strings. A pilot search was further conducted on ACM Digital Library on the 15 selected MAT studies.

Table 2: List of the preliminary paper used for search string construction

S/N	Paper Title	Year
1	A comprehensive empirical evaluation of generating test suites for mobile applications with diversity [33]	2020
2	Droidbot: A lightweight ui-guided test input generator for Android [129]	2017
3	Static window transition graphs for Android [34]	2018
4	GUI-Guided test script repair for mobile Apps [42]	2022
5	MobiGUITAR automated model-based testing of mobile Apps [161]	2015
6	Dynodroid: An input generation system for Android Apps [128]	2013

7	Automated test input generation for Android: Are We There Yet? [11]	2015
8	A systematic mapping study on cloud-based mobile application testing [13]	2019
9	Systematic literature review of mobile application development and testing effort estimation [23]	2022
10	Software testing of mobile applications: Challenges and future research directions [216]	2012
11	EvoDroid: Segmented evolutionary testing of Android Apps [126]	2014
12	TEGDroid: Test case generation approach for Android Apps considering context and GUI events [41]	2020
13	Guided, stochastic model-based GUI testing of Android Apps [119]	2017
14	A Whitebox approach for automated security testing of Android applications on the cloud [29]	2012
15	A systematic mapping study addressing the reliability of mobile applications: The need to move beyond testing reliability [15]	2021
16	AMOGA: A static-dynamic model generation strategy for mobile Apps testing [5]	2019

After the pilot evaluation, it was observed that the search string no.2 in Table 3 retrieved all the 16 selected studies, hence was chosen as the final search string. The study references were further screened and included studies that were not captured in the search results.

Table 3: Search String Piloted on ACM Digital Library

Search String	Missed	Returned Results
((“Mobile applications” OR “Mobile apps”) AND (“Testing” OR “Validation”) AND (Approach OR Technique))	6	9,054
((("Mobile applications" OR “Mobile apps” OR "Context aware Mobile app") AND (Verification OR Fault OR Testing) AND (Approach OR Technique OR Method)))	0	15,231
((“Mobile applications” OR “Mobile apps”) AND (“Testing” OR “Test”) AND (approach OR technique))	3	11,453
((“Mobile applications” OR “Mobile apps”) AND (“Testing” OR “Test”))	3	11,724

The distribution of the search results obtained using the search string is presented in Table 4, using the five selected databases.

Table 4: Selected search string and corresponding database results

Search String	Search Results				
	ACM	Science Direct	Web of Science	IEEE Explore	EBSCOhost
((("Mobile applications" OR “Mobile apps” OR "Context aware Mobile app") AND (Verification OR Fault OR Testing) AND (Approach OR Technique OR Method)))	15,231	12,476	2,097	5,404	3,068

3.4 Inclusion and Exclusion Criteria

Applying the inclusion/exclusion criteria listed below serves the primary goal of ensuring that this study only includes related research publications that provide proof of mobile application testing. Therefore, the following inclusion criteria were considered:

a. Studies that focused on Mobile application testing.
b. Studies must be peer-reviewed.
c. Studies must be written in the English Language and have full text.

The following criteria were considered for the exclusion:

i. Studies that do not contain key terms as identified in the search string in their title, abstract and keywords.
ii. Studies written in languages other than the English language.
iii. Duplicates, MAT-focused Books, and non-full text.

The above criteria were applied on the selected databases at three stages: keyword search, database search and whole article scan – the results obtained are presented in Table 5. After the selection a total of 242 papers were selected for the SMS as presented in Table 6.

Table 5: Distribution of Search Results Per Database

Research Database	Search Results	Distribution		
		Conference	Journal	Others
ACM Digital Library	15,231	12,703	1,912	616
ScienceDirect	12,476	86	11,506	884
Web of Science (WoS)	2,097	94	1,687	316
IEEE Xplore	5,404	4,796	526	82
EBSCOhost	3,068	102	2,789	177

Table 6: Application of Inclusion and Exclusion Criteria

Search Stage	Inclusion Criteria	Exclusion Criteria	Search Results (Conference & Journal)
Keyword search	The keywords "mobile application testing," or "mobile app testing,” must be available either in the paper title, keywords or abstract	Publication that does not contain any of the key words in its title, abstract or keywords	15,231
Database search	Full text published in English-Language. Published Articles, dissertations, book chapters, and reviews	Languages other than English MAT focused Books	516

Article scan and selection	Focus on the MAT Peer-review articles Listed in an academic database	Repeated and/or duplicate papers. Publication unrelated to MAT	242

3.5 Data Extraction

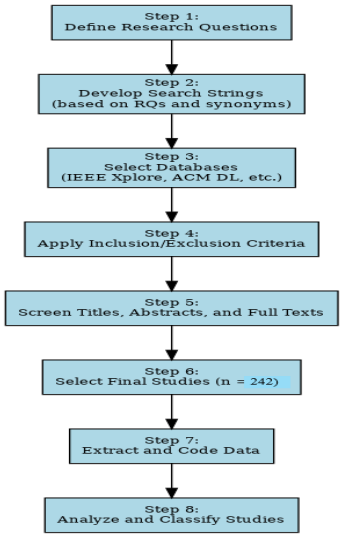
In this study, data extraction – otherwise known as "coding" is the process of obtaining useful knowledge from primary studies for the aim of addressing research questions. A systematic and careful extraction of information was performed on the selected study articles where relevant information such as the year of publication, region of publication, article title, author's name and study contribution, testing technique, testing approach, and test evaluation technique were manually extracted into a Microsoft Excel sheet for further analysis.

3.6 Validity Evaluation (Inter-Rater Reliability)

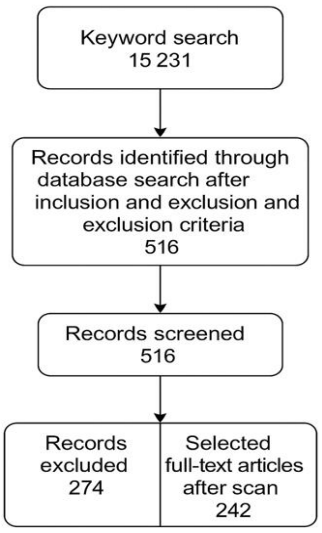
Several evaluations were conducted to guarantee the validity of the study. To address the research questions included in the search strategy section. First, the search string was verified to yield accurate results. Second, a quality evaluation was carried out to make sure the included studies fulfilled specific requirements for relevance, validity, and reliability. This entails assessing each study's methodological soundness and bias risk to ascertain the general level of confidence in the findings. Thirdly, the first and second researchers worked independently to extract (code) the data. When unsure, the researchers double-check the article until a common ground is reached. The inter-rater reliability agreement was calculated for the 242 qualified articles under study using the Cohen's Kappa formula (Equation 1):

$$k = \frac{po-pe}{1-pe} = 1 - \frac{1-po}{1-pe} \quad (1)$$

The results indicate that the average between the researchers is K = (0.97). This is a near-perfect agreement according to Glen [25].



2a: Methodology flow diagram for the systematic mapping study



2b: PRISMA flow diagram for the systematic mapping study

Figure 2: Methodology and PRISMA flow diagram for the systematic mapping study

4.0 RESULTS

In this section, the findings of the study are reported based on the research questions presented in Section 3.1. As shown in Figure 3, the graph reveals that the MAT-focused papers were published in 43 countries with the China recording the highest number of publications (23.08 %, n=54), followed by the USA with (21.79%, n=51). The remaining countries received publication distribution as follows: Italy (10.68%, n=25); Germany (5.56%, n=13); Brazil (4.27%, n=10); Portugal (3.85%, n=9); Malaysia, Taiwan, and Singapore (2.56%, n=6) respectively; Korea and Japan (2.14%, n=5) respectively; Pakistan and India (1.71%, n=4) respectively; Australia, UK, and France (1.28%, n=3) respectively; Vietnam, Indonesia, Israel, Spain, Argentina, Switzerland, and Denmark (0.85%, n=2) respectively; then followed by Colombia, Palestine, Sri Lanka, Iran, Sweden, Bahrain, Netherlands, Ireland, Mauritius, Luxembourg, Jordan, Canada, Austria, Egypt, Bangladesh, South Korea, South Africa, Norway, and Algeria with (0.43%, n=1) respectively.

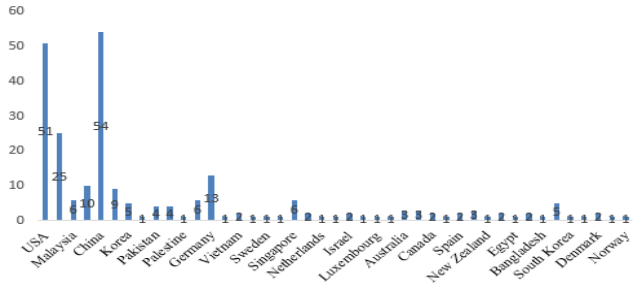


Figure 3: Contribution by countries

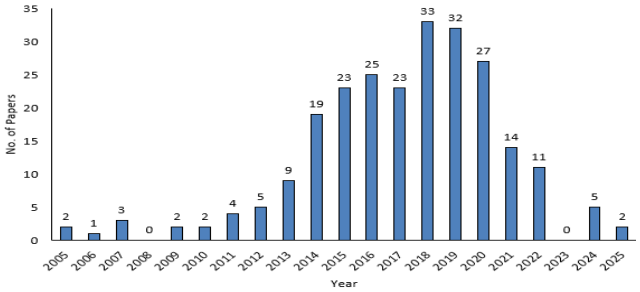


Figure 4: Publication by year

4.1 How Frequently are MAT Research Studies Published, and which Study has the most Citations?

The findings reveal that 2.6% (6 of 242) of the entire MAT-focused research studies were conducted between the years 2005 and 2008, the reason for the low record in the number of publications is not far-fetched as testing of mobile apps within this period was relatively new. Between 2009 and 2013, 9.4% (22 of 242) of MAT-focused papers were published. While the years between 2014 and 2022 as shown in Figure 4, report consistent publications recording 85% (207 of 242) with 2018 recording the highest number 14.1% (33 of 242) of the entire MAT related publications under study. Table 11 reveals that out of the 11 most cited studies, *"DynoDroid: An Input Generation System for Android Apps"*, published in 2013, is the most cited study among all MAT-related studies.

4.2 Which Testing Approach Do These Studies Apply?

Generally, three types of test approaches are used: white-box, black-box and grey-box testing approaches. The white-box (also referred to static) approach is the type in which information for the test generation is extracted either directly from the source code or via reverse engineering, the black-box (also referred to dynamic) does not require the source code. Instead, it aimed at executing the app to detect faults at run-time, and grey-box (also referred to hybrid) which combines the two approaches. The tester extracts information from source code to support the dynamic exploration of the app. Table 7 classifies the studies according to the different testing approaches they implement.

Table 7: Classification of studies according to testing approach

Approach	Reference
White box	[26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [272], [273].
Black box	[42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [28], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [100], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [120], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], [146], [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [266], [267], [268], [269], [270], [271], [272].
Grey box	[5], [160], [161], [33], [162], [163], [164], [165], [166], [167], [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178], [179], [180], [181].

The results obtained show that 75% of testing tools applied the black-box approach, 14% applied grey-box while tools applied white-box are just 11%, as shown in Figure 5.

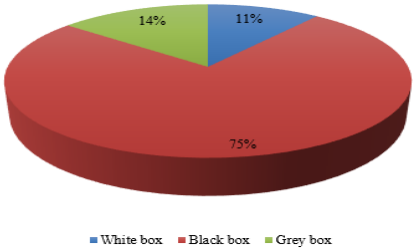


Figure 5: Percentage of testing approach

4.3 Which Test Strategy Do These Studies Apply?

A test strategy covers multiple testing levels, including unit testing, integration testing, system testing, and user acceptability testing [157]. Unit testing is the first phase of testing in software development. It involves testing the individual components of the software. This kind of testing is usually carried out during app development by the developer to identify software faults. Integration testing is the second phase of testing process that involves testing the interface between two software units to determine accurate integration [157]. The main reason for integration testing is to expose faults

and detect any irregularity between the integrated units. The third phase is system testing, which is performed on the entire system. The purpose of system testing is to find defects/errors within both integrated modules and the system as a whole [18]. User acceptance testing (UAT) is the final stage of testing. Just as the name suggests, UAT is a software test that verifies that a given application works as expected by the user [182].

This study identifies that most of the research studies in mobile application testing focus on system-level testing with a few of them focusing on other test levels. The classification of research studies based on testing level is presented in Table 8. Testing at other levels remains a potential research area for further research. The results obtained show that 97% of tools are tested at the system level while 1% each for the tools text other testing levels.

Table 8: Test strategy/level

Strategy/Level	Reference of articles
Unit Testing	[63], [36]
Integration Testing	[32], [120]
System Testing	[42], [43], [44], [45], [46], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [28], [59], [60], [61], [62], [64], [65], [66], [42], [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], [77], [78], [79], [80], [81], [82], [31], [83], [84], [85], [86], [87], [88], [33], [89], [90], [91], [92], [93], [94], [95], [96], [97], [98], [99], [101], [102], [103], [104], [105], [106], [107], [108], [109], [110], [111], [112], [113], [114], [115], [116], [117], [118], [119], [121], [122], [123], [124], [125], [126], [127], [128], [129], [130], [131], [132], [133], [134], [135], [136], [137], [138], [139], [140], [141], [142], [143], [144], [145], [146], [147], [148], [149], [150], [151], [152], [153], [154], [155], [156], [157], [158], [159], [5], [160], [161], [33], [162], [163], [164], [165], [166], [167], [168], [169], [170], [171], [172], [173], [174], [175], [176], [177], [178], [179], [180], [181], [26], [27], [28], [29], [30], [31], [33], [34], [35], [37], [38], [39], [40], [41], [266], [267], [268], [269], [270], [271], [272], [273],
User Acceptance Testing	[185], [100]

4.4 Which Testing Technique Does the Studies Apply?

Generally, test cases are important software testing input that must be created from a kind of information, which is some type of software artifact. Several forms of artefacts are used as a reference input to test case development, including software specifications or design models, program source code, and information gained from dynamic program execution. The popular techniques employed to generate test cases are script-based, capture/replay, random testing and its modified adaptive random testing, symbolic execution, search-based testing, combinatorial testing, systematic exploration, and model-based testing [183], [184]. This study identified that a large majority of the research applied model-based testing (MBT) as shown in Table 9. This indicates the increasing popularity of MBT among the software engineering community as it can improve the construction of test cases and test coverage, though the model generation is a challenging process.

Table 9: Testing technique

Technique	Testing tool
Script-based	METER [42], GUI Ripper [44], FIT4Apps [185], MobileTest [87], GUIDER [118], PREFEST [170, 171], Segen [168], JPF-ANDROID [259], TESTINTENTION [273]
Capture/replay	AppCheck [234], ParaAim [83], MOKA [155]
Random walk & Adaptive Random testing	Monkey [153], T+ [127], Monkey [215, 95, 151], Dynalog [160], Smart-Monkey [228], Fax [173], Dynodroid [128], AutoClicker [133], VanarDena [178], VTest [140], EvoMASTER [272], ROBOTEST [271]
Model-based	Humanoid [49], APPTESTMIGRATOR [48], AMOGA [5], MobiGUITAR [161], Androidetect [249], AIMDROID [84], CRASHSCOPE [82], BBOXTESTER [227], Moneky & Acvtool [52], Android Black-Box Coverage Analyzer (ABCA) [53], SCANCIF [56], FSMdroid [57], Snowdrop [26], MISTA [64], MODISCO [29], AutoDroid [71], VTE [72], APE [74], DroidDev [77], MobTAF [226], DROIDMATE [229], AndroidRipper [243], Word2Vec [65], Textout [75], ACAT [80], Monkey++ [88], Androidetect [294], SENTINEL [225], GATOR [34, 180], A3E [163], PBGT [86], IMPAcT [230], LAND [79], Sketch-guided [70], Android Ripper & Extended Ripper [51], CAT [237], GAT [69], DSML [136], EMMA [154], Calabash [139, 150], JaBUTi/ME [40], AndroTotal [146], AOBAKO [145], Multi-Level GUICC [142], ATUA [179], YAKUSU [177], ENVIAR [137], CrawlDroid [136], DroidMate [175], ODIN [132], DetReduce [131], DroidBot [129], EvoDroid [126], FlakeScanner [123], RacerDroid [122], NAMBA [38], NaviDroid [172, 121], TRIANGLE [120], Stoat [119], ATG [117], AndroidRipper [116], ALARic [115], Latte [114], Mimic [113], Dune [112], MobicoMonkey [111], MobiGolog [110], Appium [108], Swisscom [106], ICCMATT [105], MTEst [104], MoSSOT [102], MT4A [101], ADDET [37], Paladin [99], PLATOOL [98], MBTS4MA [95], CRAFTDROID [165], iMPAct [164], PBGT Tool [174], UIAutomator & EyeAutomator [89], Robotium [158], DeepREST [266], DinoDroid [267]
Search-based	ADAPTDROID [47], Cadage [78], COBWEB [239], SAPIENZdiv [33], A2T2 [60], DroidCrawler [247], ORBIT [162], EHBDroid [81], PJUnit [63], TimeMachine [61], FLAG [66], AGRippin [147], SAPIENZ [94], TEGDroid [41].
Symbolic/Concolic Execution	CRAXDroid [31], SynthesiSE [244], Collider [141], SIERRA [35]
Combinatorial based	CTGen [62], ComboDroid [30], JUnit and Money [176]
Graph transformation	AToM3 [152]

Cloud-based	Tool-AM-TaaS [138]
Machine Learning	ARES [135], Q-testing [97], AutoBlackTest [96], AppsPred [159], Qtool [250], ACETON [144], [264], Code5T [268], ATHENATEST [269], MACdroid [270]
Systematic Exploration	THOR [92], AMOGA [167], A3E [163]

4.5 How Were These Testing Techniques Evaluated/Validated?

It is a fundamental element of the process of scholarly endeavor to validate a new methodology/technique and its results. Validation is defined as a method that ensures a system works as intended as well as satisfying its objectives [263]. The goal here is to determine what validation methods exist and what ways others use to validate the effectiveness of their proposed techniques. Some of the popular validation methods used in software testing include code coverage, a metric that can assist one determine the amount of the source code is tested, fault detection which is the process of discovering the presence of a fault in a system, bug detection, energy leak detection, privacy detection, malware detection, crash detection/report, GUI coverage, screen, activity and method coverage etc.

The methods were grouped according to their relation. Methods that have common aims – such as fault detection, error detection or bug detection – were grouped together. Other techniques were not commonly used in evaluating test methods. The study identified that most of the research employed code coverage and fault/defect detection methods to validate their proposed tools. Table 10 presents the results of the study.

Table 10: Evaluation method

Evaluation method	Testing tool
Code coverage	AMOGA [5, 167], GUI Ripper [44], Moneky & Acvtool [52], Android Black-Box Coverage Analyzer (ABCA) [53], FSMdroid [57], MODISCO [29], ComboDroid [30], Monkey [153], GAT [69], Dynalog [160], AutoDroid [71], DroidDev [77], Cadage [78], LAND [79], BBOXTESTER [227], EHBDroid [81], ParaAim [83], AIMDROID [84], Monkey++ [88], MobiGUITAR [161], SAPIENZ^{div} [33], METER [42], ATOM [55], Appium [108], CTGen [62], MISTA [64], Humanoid [49], APE [74], SynthesiSE [244], METER [42], Sketch-guided [70], TimeMachine [61], MOKA [155], Qtool [250], JaBUTi/ME [40], AGRippin [147], Multi-Level GUICC [142], Collider [141], ATUA [179], CrawlDroid [136], ARES [135], ODIN [132], DetReduce [131], Dynodroid [128], EvoDroid [126], NAMBA [38], Stoat [119], ATG [117], AndroidRipper [243], Mimic [113], ICCMATT [105], Monkey [103], Fax [173], NaviDroid [37, 121], Paladin [99], PLATOOL [98], Q-testing [97], AutoBlackTest [96], SAPIENZ [94], Segen [168], TEGDroid [41], Code5T [268], ATHENATEST [269], MACdroid [270], EvoMASTER [272]
Crash detection	APE [74], CRASHSCOPE [82], TimeMachine [61], ALARic [115], Swisscom [106]
Data leakage detection, Sensor leaks, Memory leak detection	CRAXDroid [37], SENTINEL [225], FunesDroid [67]
Fault/defect detection Bug/Error detection	iMPAcT [164, 45], PJUnit [63], EHBDroid [81], SAPIENZ^{div} [33], COBWEB [239], Smart-Monkey [228], GUI Ripper [44], QUANTUM [50], FSMdroid [57], Textout [75], Cadage [78], ACAT [80], ImMut [224], CAT [237], AndroidRipper [116], PBGT [86], AMOGA [5], SCANCIF [56], DROIDMATE [229], MobTAF [226], Android Ripper & Extended Ripper [51], MobileTest [87], Qtool [250], EMMA [154], AToM3 [152], VanarDena [178], ENVIAR [137], Stoat [119], ADDET [37], PLATOOL [98], Q-testing [97], MBTS4MA [95], PBGT Tool [174], Robotium [158], Latte [114], Dune [112], TEGDroid [41], Monkey [151], YAKUSU [177], JUnit and Money [176], Dynodroid [128], RacerDroid [122], ATG [117], MobicoMonkey [111], Fax [173], NaviDroid [172], PREFEST [171, 170], THOR [92], Monkey [153], Multi-Level GUICC [142], JPF-ANDROID [259], SIERRA [35], Calabash [139], VTE [72], DeepREST [266], DinoDroid [267], ROBOTEST [271], EvoMASTER [272]
GUI coverage	DroidCrawler [247], ADAPTDROID [47]
Screen, Activity and Method coverage, Test coverage, line coverage	A3E [163], GAT [69], Word2Vec [65], FragDroid [241], VTest [140], DroidMate [175], MTEst [104], DeepREST [266]
Test accuracy and migration	FLAG [66], APPTESTMIGRATOR [48]
Precision analysis, Test execution time	ORBIT [162], GATOR [34], CRAFTDROID [165], PJUnit [63], AToM3 [152], Calabash [150], MT4A [101]
Vulnerability/Malware detection	AndroTotal [146], DroidBot [129], MoSSOT [102], GATOR [180],
Energy defect detection/ Flaky test detection	ACETON [144], FlakeScanner [123]
Code Summarization	TESTINTENTION [273]

The distinction between the two is that code coverage quantifies how many lines of code, branches, or functions are executed during tests, ensuring comprehensive test cases and improving the reliability of the software. While fault detection is the ability of testing to identify defects or bugs in the software. It assesses whether the system behaves as expected under various conditions with the aim of finding and reporting defects so they can be fixed before the software is released. While code coverage provides insights into the testing thoroughness, fault detection evaluates the quality and correctness of the software itself. Both are essential for effective software testing.

4.6 Which of the MAT related studies is the Most Influential Study?

To answer this research question, the technique used by Nie et al. [19] was adopted, measuring the number of citations received by the studies. The greater the number of citations, the higher the impact, which determines how influential a study is. The Normalized Citation Impact Index (NCII), was used, applying the formula below to measure the impact factor of a study based on its publication duration. Additionally, 11 articles with the highest NCII scores were considered the most influential studies. Table 11 presents the studies and their corresponding NCII scores. The results show that Dynodroid has the highest impact factor with 952 citations and an NCII value of 79.3 NCII; therefore, Dynodroid is the most influential MAT-related study.

$$NCII = \frac{Citation\ per\ publication}{Publication\ duration(years)} \quad (2)$$

Table 11: Most Influential Studies

S/N	Paper title	Year of Pub.	Citation	NCII
1	Dynodroid: An Input Generation System for Android Apps [128]	2013	952	79.3
2	Using GUI ripping for testing of Android applications [243]	2012	707	54.5
3	Targeted and depth-first exploration for systematic testing of Android apps [163]	2013	643	53.5
4	Automating GUI Testing for Android Applications [176]	2011	450	33.1
5	A grey-box approach for automated GUI model generation of mobile [162]	2013	453	37.7
6	MobiGUITAR Automated Model-Based Testing of Mobile Apps [161]	2015	453	45.3
7	EvoDroid: Segmented Evolutionary Testing of Android Apps [126]	2014	441	40
8	Software testing of mobile applications: Challenges and future research directions [216]	2012	359	27.6
9	Guided, Stochastic Model-Based GUI Testing of Android Apps [119]	2017	433	54.1
10	droidBot: ALightweight UI-Guided Test Input Generator for Android [129]	2017	343	42.8
11	A Guided GUI Crawling Based Technique for Android Mobile Applications Testing [60]	2011	315	22.5

5.0 DISCUSSION

The results of the survey have shown that the evolution of testing approaches in software development is experiencing several key stages, reflecting advancement in technology, methodologies, and the growing complexity of software. In most cases the nature of software determines the type of approach to be used by testers. While the black-box approach is used by most testing tools there are still situations where grey-box can do better as it enables testers to have partial knowledge of the internal structure, which allows them to design tests that leverage some internal insights while still simulating real user behavior. This can lead to more effective testing, especially for identifying vulnerabilities or ensuring quality in applications.

For the testing techniques, the results have shown that most testing techniques applied model-based techniques. Despite the increasing popularity of model-based techniques, developing models is still a challenging task. There is a need for more techniques that can ease the efforts in model generation.

With regards to method of evaluation, the results have shown that most testing tools are utilizing code coverage and fault detection as popular means of validation of their effectiveness. However, many studies have revealed that high code coverage does not guarantee the absence of bugs; it only indicates that the code was executed during tests. As a result of this, a combination of methods can be applied to evaluate the testing tools' effectiveness.

In summary, while the academia aims to deepen theoretical understanding and advance knowledge in software testing that includes software testing principles, model-based testing techniques, testing automation, fault localization, and the development of new testing frameworks, the industry focuses on practical implementation and delivering high-quality software products. Collaboration between the two is highly encouraged for both domains to contribute to the overall development of software testing field.

6.0 CONCLUSION

This paper presented an in-depth survey of MAT focused articles, identifying the research contributions such as the techniques, strategies and the test tools proposed by researchers and practitioners for testing mobile applications. The study has further identified publication frequency as well as the participating countries. This systematic mapping study will serve as a guide for the software testing research community in identifying open areas for further research in mobile Apps Testing.

REFERENCES

[1] Myers, G. J., Sandler, C., and Badgett, T., "Art of Software Testing 3rd ed.," Hoboken, New Jesey: John Wiley & Sons. Inc, 2012.

[2] Statista, "Smartphone sales worldwide 2007-2023", Accessed: May 16, 2025. [Online].

Available: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/

[3] Counterpoint, "Global smartphone market share: Quarterly", Accessed: May 16, 2025. [Online]. Available: https://www.counterpointresearch.com/insights/global-smartphone-share/

[4] Gao, J., Bai, X., Tsai, W. T., and Uehara, T. "Mobile application testing: A tutorial", *Computer*, Feb. 2014, vol. 47, no. 2, pp. 46–55, doi: 10.1109/MC.2013.445.

[5] Salihu, I. A., Ibrahim, R., Ahmed, B. S., Zamli, K. Z., and Usman, A. "AMOGA: A Static-Dynamic Model Generation Strategy for Mobile Apps Testing", *IEEE Access*, vol. 7, pp. 17158–17173, 2019, doi: 10.1109/ACCESS.2019.2895504.

[6] Statista, "Annual number of global mobile app downloads 2016 - 2021", Accessed: May 16, 2025. [Online]. Available: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/

[7] Eke, N. O., and Salihu, I. A. "Design and Implementation of a Mobile Library Management System for Improving Service Delivery", *Path of Science*, vol. 7, no. 4, p. 3001, 2021, doi: 10.22178/pos.69-7.

[8] Yang, S., Yan, D., Wu, H., Wang, Y., and Rountev, A. "Static control-flow analysis of user-driven callbacks in android applications", *in Proceedings of the 37th International Conference on Software Engineering*, vol. 1, pp. 89–99, 2015, doi: 10.1109/ICSE.2015.31.

[9] Tao, C., and Gao, J. "Building a Model-Based GUI Test Automation System for Mobile Applications", *International Journal of Software Engineering and Knowledge Engineering*, vol. 26, no. 9–10, pp. 1605–1615, 2016, doi: 10.1142/S0218194016710042.

[10] Morgado, I. C., Paiva, A. C. R., and Faria, J. P. "Automated pattern-based testing of mobile applications", in *Proceedings of the 9th International Conference on the Quality of Information and Communications Technology (QUATIC)*, pp. 294–299, 2014, doi: 10.1109/QUATIC.2014.47.

[11] Choudhary, S. R., Gorla, A., and Orso, A. "Automated test input generation for Android: Are we there yet?", in *Proceedings of the 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2015, pp. 429–440, doi: 10.1109/ASE.2015.89.

[12] Zein, S., Salleh, N., and Grundy, J. "A systematic mapping study of mobile application testing techniques", *Journal of Systems and Software*, vol. 117, pp. 334–356, Jul. 2016, doi: 10.1016/j.jss.2016.03.065.

[13] Ya'u, B. I., Salleh, N., Nordin, A., Alwan, A. A., Idris, N. B., and Abas, H. "A Systematic Mapping Study on Cloud-Based Mobile Application Testing", *Journal of Information and Communication Technology*, vol. 18, no. 4, pp. 485–527, Oct. 2019, [Online]. Available: http://10.0.128.122/jict2019.18.4.5

[14] Tramontana, P., Amalfitano, D., Amatucci, N., and Fasolino, A. R. "Automated functional testing of mobile applications: A systematic mapping study", *Software Quality Journal*, vol. 27, no. 1, pp. 149–201, Mar. 2019. [Online]. Available: http://10.0.3.239/s11219-018-9418-6

[15] Wimalasooriya, C., Licorish, S. A., da Costa, D. A., and MacDonell, S. G. "A systematic mapping study addressing the reliability of mobile applications: The need to move beyond testing reliability", *Journal of Systems and Software*, vol. 186, Apr. 2021, Art. no. 111166, doi: 10.1016/j.jss.2021.111166.

[16] Sahinoglu, M., Incki, K., and Aktas, M. S. "Mobile application verification: A systematic mapping study", in *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 9159, pp. 147–163, 2015, doi: 10.1007/978-3-319-21413-9_11.

[17] Petersen, K., Vakkalanka, S., and Kuzniarz, L. "Guidelines for conducting systematic mapping studies in software engineering: An update", *Information and Software Technology*, vol. 64, pp. 1–18, 2015, doi: 10.1016/j.infsof.2015.03.007.

[18] Banerjee, I., Nguyen, B., Garousi, V., and Memon, A. "Graphical user interface (GUI) testing: Systematic mapping and repository", *Information and Software Technology*, vol. 55, no. 10, pp. 1679–1694, 2013, doi: https://doi.org/10.1016/j.infsof.2013.03.004.

[19] Nie, L., Said, K. B., Ma, L., S., Zheng, Y., and Zhao, Y. "A systematic mapping study for graphical user interface testing on mobile apps", *IET Software*, Jun. 2023, vol. 17, no. 3, pp. 249-267, doi: 10.1049/sfw2.12123.

[20] Kong, P., Li, L., Gao, J., Liu, K., Bissyandé, T. F., and Klein, J. "Automated testing of Android apps: A systematic literature review", *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 45–66, 2019, doi: 10.1109/TR.2018.2865733.

[21] Keele, S., "*Guideline for performing Systematic Literature Reviews in Software Engineering*",

(vol. 5). Technical report, ver. 2.3 ebse technical report. Ebse, Jul. 2007.

[22] Musthafa, F. N., Mansur, S., and Wibawanto, A. "Automated Software Testing on Mobile Applications: A Review with Special Focus on Android Platform", *ACM SIGSOFT Software Engineering Notes*, vol. 36, no. 3, pp. 1–4, 2020, doi: 10.1145/1968587.1968601.

[23] Kaur, A., and Kaur, K. "Systematic literature review of mobile application development and testing effort estimation," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 2, pp. 1–15, 2022, doi: 10.1016/j.jksuci.2018.11.002.

[24] Li, L., Bissyandé, T. F., Papadakis, M., Rasthofer, S., Bartel, A., Octeau, D., Klein, J., and Traon, L., "Static analysis of Android apps: A systematic literature review", *Information and Software Technology*, vol. 88, pp. 67–95, Aug. 2017, doi: 10.1016/j.infsof.2017.04.001

[25] Glen, S. "Cohen's Kappa Statistic. Statistics How To", Accessed: Jan. 16, 2025. [Online]. Available: https://www.statisticshowto.com/cohens-kappa-statistic/

[26] Zhang, L. L., Liang, C. J. M., Liu, Y., and Chen, E. "Systematically testing background services of mobile apps", in *Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2017, pp. 4–15, doi: 10.1109/ASE.2017.8115613.

[27] Deng, L., Mirzaei, N., Ammann, P., and Offutt, J. "Towards mutation analysis of Android apps", in *Proceedings of the 2015 IEEE 8th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Apr. 2015, pp. 1-10, doi: 10.1109/ICSTW.2015.7107450

[28] Hamza, Z. A., and Hammad, M. "Web and mobile applications' testing using black and white box approaches", *IET Conference Publications*, vol. 2019, no. CP758, pp. 20–23, 2019, doi: 10.1049/cp.2019.0210.

[29] Mahmood, R., Esfahani, N., Kacem, T., Mirzaei, N., Malek, S., and Stavrou, A. "A whitebox approach for automated security testing of Android applications on the cloud", in *Proceedings of the 2012 IEEE 7th International Workshop on Automation of Software Test (AST)*, Jun. 2012, pp. 22–28, doi: 10.1109/AST20855.2012

[30] Wang, J., Jiang, Y., Xu, C., Cao, C., Ma, X., & Lu, J. "ComboDroid: generating high-quality test inputs for Android apps via use case combinations", in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE)*, Jun. 2020, pp. 469–480, doi: 10.1145/3377811.3380382

[31] Yeh, C. C., Lu, H. L., Chen, C. Y., Khor, K. K., and Huang, S. K. "CRAXDroid: Automatic Android system testing by selective symbolic execution", in *Proceedings of the 2014 IEEE 8th International Conference on Software Security and Reliability - Companion*, Jun. 2014, pp. 140–148, doi: 10.1109/SERE-C34493.2014.

[32] Chan, W. K., Chen, T. Y., Lu, H., Tse, T. H., and Yau, S. S. "A metamorphic approach to integration testing of context-sensitive middleware-based applications", in *Proceedings of the IEEE 5th International Conference on Quality Software (QSIC'05)*, Sep. 2005, pp. 241–249, doi: 10.1109/QSIC.2005.3.

[33] Vogel, T., Tran, C., and Grunske, L. "A comprehensive empirical evaluation of generating test suites for mobile applications with diversity", *Information and Software Technology*, vol. 130, p. 106436, 2021, doi: 10.1016/j.infsof.2020.106436.

[34] Yang, S., Liu, Y., Ma, X., Sun, J., Zhou, Y., and Zhang, X. "Static window transition graphs for Android", *Automated Software Engineering*, vol. 25, no. 4, pp. 833–873, 2018, doi: 10.1007/s10515-018-0237-6.

[35] Hu, Y., and Neamtiu, I. "Static detection of event-based races in Android apps", *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 257–270, Mar. 2018, doi: 10.1145/3173162.3173173.

[36] Kim, H., Choi, B., and Yoon, S. "Performance testing based on test-driven development for mobile applications", in *Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication*, Feb. 2009, pp. 612–617, doi: 10.1145/1516241.1516349

[37] Bessghaier, N., Soui, M., Kolski, C., and Chouchane, M. "On the detection of structural aesthetic defects of Android mobile user interfaces with a metrics-based tool", *ACM Transactions on Interactive Intelligent Systems*, vol. 11, no. 1, Apr. 2021, doi: 10.1145/3410468.

[38] Keng, J. C. J., Jiang, L., Wee, T. K., and Balan, R. K. "Graph-aided directed testing of Android applications for checking runtime privacy behaviours", in *Proceedings of the 11th International Workshop on Automation of Software Test (AST)*, May 2016, pp. 57–63, doi: 10.1145/2896921.2896930.

[39] San Miguel, J. L., and Takada, S. "Generating test cases for Android applications through GUI modeling, usage modeling, and Change analysis", in *ACM International Conference Proceeding Series*, Jul. 2015, vol. 13-17-July-2015, pp. 146–147. doi: 10.1145/2790798.2790823.

[40] Delamaro, M. E., Vincenzi, A. M. R., and Maldonado, J. C. "A strategy to perform coverage testing of mobile applications", in *Proceedings of the 2006 IEEE International Workshop on Automation of software test*, May 2006, pp. 118–124, doi: 10.1145/1138929.1138952.

[41] Usman, A., Ibrahim, N., and Salihu, I. A. "TEGDroid: Test case generation approach for Android apps considering context and GUI events", *International Journal on Advanced Science, Engineering and Information Technology*, vol. 10, no. 1, pp. 16–23, 2020, doi: 10.18517/ijaseit.10.1.10194.

[42] Pan, M., Xu, T., Pei, Y., Li, Z., Zhang, T., and Li, X. "GUI-guided test script repair for mobile apps", *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 910–929, 2022, doi: 10.1109/TSE.2020.3007664.

[43] Behrang, F., and Orso, A. "AppTestMigrator: a tool for automated test migration for android apps", in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, Jun. 2020, pp. 17–20, doi: 10.1145/3377812.3382149

[44] Imparato, G. "A combined technique of GUI ripping and input perturbation testing for Android apps", in *Proceedings of the 37th International Conference on Software Engineering (ICSE)*, vol. 2, pp. 760–762, 2015, doi: 10.1109/ICSE.2015.241.

[45] Paiva, A. C. R., Gouveia, J. M. E., Elizabeth, J. D., and Delamaro, M. E. "Testing when mobile apps go to background and come back to foreground", in *Proceedings of the 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Apr. 2019, pp. 102–111, doi: 10.1109/ICSTW.2019.00038

[46] Wang, Y. "An automated virtual security testing platform for Android mobile apps", in *Proceedings of the 2015 IEEE 1st Conference on Mobile Security Services (MOBISECSERV)*, Feb. 2015, pp. 1-2, doi: 10.1109/MOBISECSERV.2015.7072877

[47] Mariani, L., Pezzè, M., Terragni, V., and Zuddas, D. "An evolutionary approach to adapt tests across mobile apps", in *Proceedings of the 2021 IEEE/ACM International Conference on Automation of Software Testing (AST)*, May 2021, pp. 70–79, doi: 10.1109/AST52587.2021.00016.

[48] Behrang, F., and Orso, A. "Test migration between mobile apps with similar functionality", in *Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2019, pp. 54–65, doi: 10.1109/ASE.2019.00016.

[49] Li, Y., Yang, Z., Guo, Y., and Chen, X. "Humanoid: A deep learning-based approach to automated black-box Android app testing", in *Proceedings of the 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2019, pp. 1070–1073, doi: 10.1109/ASE.2019.00104.

[50] Zaeem, R. N., Prasad, M. R., and Khurshid, S. "Automated generation of oracles for testing user-interaction features of mobile apps", in *Proceedings of the 2014 IEEE 7th International Conference on Software Testing, Verification and Validation (ICST)*, Mar. 2014, pp. 183–192, doi: 10.1109/ICST.2014.31.

[51] Amalfitano, D., Fasolino, A. R., Tramontana, P., and Amatucci, N. "Considering context events in event-based testing of mobile applications", in *Proceedings of the IEEE 2013 6th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Mar. 2013, pp. 126–133, doi: 10.1109/ ICSTW.2013.22.

[52] Huang, S., Lin, H., Liu, Y., Zhang, J., and Li, D. "Runtime-Environment Testing Method for Android Applications", in *Proceedings of the 2019 IEEE 19th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, Jul. 2019, pp. 534–535, doi: 10.1109/QRS-C.2019.00111

[53] Huang, S. Y., Yeh, C. H., Wang, F., and Huang, C. H. "ABCA: Android Black-box Coverage Analyzer of mobile app without source code", in *Proceedings of the 2015 IEEE International Conference on Progress in Informatics and Computing (PIC)*, Dec. 2015, pp. 399–403, doi: 10.1109/PIC.2015.7489877.

[54] Chen, S., Fan, L., Su, T., Ma, L., Liu, Y., and Xu, L. "Automated Cross-Platform GUI code generation for mobile apps", in *Proceedings of the 2019 IEEE 1st International Workshop on Artificial Intelligence for Mobile (AI4Mobile)*, Feb. 2019, pp. 13–16, doi: 10.1109/AI4Mobile.2019.8672718

[55] Li, X., Wu, L., Yang, Z., Xu, J., Chen, Z., and Liu, Y. "ATOM: Automatic Maintenance of GUI Test Scripts for Evolving Mobile Applications", in *Proceedings of the 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, Mar. 2017, pp. 161–171, doi: 10.1109/ICST.2017.22

[56] Lau, P. T. "Scan code injection flaws in HTML5-based mobile applications", in *Proceedings of the 2018 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2018, pp. 81–88, doi: 10.1109/ICSTW.2018.00032.

[57] Su, T. "FSMdroid: Guided GUI testing of android apps", in *Proceedings of the 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 689–691, doi: 10.1145/2889160.2891043.

[58] Subramanian, S., Singleton, T., and El Ariss, O. "Class Coverage GUI Testing for Android Applications", in *Proceedings of the 2016 International Conference on System Reliability and Science (ICSRS)*, Nov. 2016, pp. 84–89, doi: 10.1109/ICSRS.2016.7815843.

[59] Morgado, I. C., and Paiva, A. C. R. "Testing Approach for Mobile Applications through Reverse Engineering of UI Patterns", in *Proceedings of the 2015 30th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, Nov. 2015, pp. 42–49, doi: 10.1109/ASEW.2015.11..

[60] Amalfitano, D., Fasolino, A. R., and Tramontana, P. "A GUI Crawling-Based Technique for Android Mobile Application Testing", in *Proceedings of the 2011 IEEE 4th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Mar. 2011, pp. 252–261, doi: 10.1109/ICS TW.2011.77.

[61] Dong, Z., Böhme, M., Cojocaru, L., and Roychoudhury, A. "Time-Travel Testing of Android Apps", in *Proceedings of the 2020 42nd ACM/IEEE International Conference on Software Engineering (ICSE)*, Jun. 2020, pp. 481–492, doi: 10.1145/3377811.3380402.

[62] Huynh, Q. T., Pham, T. K., Nguyen, D. M., Nguyen, P. T., Ha, N. H., and Tran, V. D. "A combinatorial technique for mobile applications software testing", in *Proceedings of the 2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, Oct. 2019, pp. 1–6, 2019, doi: 10.1109/KSE.2019.8919329

[63] Kim, H., Choi, B., and Wong, W. E. "Performance testing of mobile applications at the unit test level", in *Proceedings of the 2009 3rd IEEE International Conference on Secure Software Integration and Reliability Improvement*, Jul. 2009, pp. 171–180, doi: 10.1109/SSIRI.2009.28.

[64] Mirza, A. M., and Khan, M. N. A. "An automated functional testing framework for context-aware applications", *IEEE Access*, vol. 6, pp. 46568–46583, 2018, doi: 10.1109/ACCESS.2018.2865213.

[65] Liu, P., Zhang, X., Pistoia, M., Zheng, Y., Marques, M., and Zeng, L. "Automatic text input generation for mobile testing", in *Proceedings of the 2017 39th IEEE/ACM International Conference on Software Engineering (ICSE)*, May 2017, pp. 643–653, doi: 10.1109/ICSE.2017.65.

[66] Chu, E. T. H., and Lin, J. Y. "Automated GUI testing for Android news applications", in *Proceedings of the 2018 International Symposium on Computer, Consumer and Control (IS3C)*, Dec. 2018, pp. 14–17, 2019, doi: 10.1109/IS3C. 2018.00013

[67] Amalfitano, D., Riccio, V., Tramontana, P., and Fasolino, A. R. "Do memories haunt you? An automated black box testing approach for detecting memory leaks in Android apps", *IEEE Access*, vol. 8, pp. 12217–12231, 2020, doi: 10.1109/ACCESS.2020.2966522.

[68] Wang, J., and 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)*, Sep. 2019, pp. 247–250, doi: 10.1109/ICVRIS.2019.00068

[69] Wu, X., Jiang, Y., Xu, C., Cao, C., Ma, X., and Lu, J. "Testing Android apps via guided gesture event generation", in *Proceedings of the 2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2016, pp. 201–208, doi: 10.1109/APSEC.2016.037

[70] Zhang, C., Cheng, H., Tang, E., Chen, X., Bu, L., and Li, X. "Sketch-guided GUI test generation for mobile applications", in *Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Oct. 2017, pp. 38–43, doi: 10.1109/ASE.2017.8115616

[71] Piparia, S., Adamo, D., Bryce, R., Do, H., and Bryant, B. "Combinatorial testing of context aware Android applications", in *Proceedings of the 16th Conference on Computer Science and*

Intelligence Systems (FedCSIS), vol. 25, 2021, pp. 17–26, doi: 10.15439/2021F003.

[72] Anbunathan, R., and Basu, A. "Automation framework for test script generation for android mobile", in *Advances in Intelligent Systems and Computing*, vol. 731, pp. 571–584, 2017, doi: 10.1007/978-981-10-8848-3_55.

[73] Mateen, A., and Abbas, K. "Optimization of model based functional test case generation for android applications", in *Proceedings of the 2017 International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, Jun. 2017, pp. 90-95, doi: 10.1109/ICPCSI.2017.8391869

[74] Gu, T., Sun, C., Ma, X., Cao, C., Xu, C., Yao, Y., Zhang Q., Lu, J., Su, Z. "Practical GUI testing of Android applications via model abstraction and refinement", in *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, 2019, pp. 269–280, doi: 10.1109/ICSE.2019.00042.

[75] Wang, Y., Xu, H., Zhou, Y., Lyu, M. R., and Wang, X. "Textout: Detecting text-layout bugs in mobile apps via visualization-oriented learning", in *Proceedings of the 2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*, 2019, pp. 239–249, doi: 10.1109/ISSRE.2019.00032.

[76] Wang, Z., Elbaum, S., and Rosenblum, D. S. "Automated generation of context-aware tests", in *Proceedings of the 29th International Conference on Software Engineering (ICSE'07)*, May 2007, pp. 406-415, doi: 10.1109/ICSE.2007.18

[77] Arnatovich, Y. L., Ngo, M. N., Kuan, T. H. B., and Soh, C. "Achieving high code coverage in android UI testing via automated widget exercising", in *Proceedings of the 2016 23rd Asia-Pacific Software Engineering Conference (APSEC)*, Dec. 2016, pp. 193–200, doi: 10.1109/APSEC.2016. 036.

[78] Zhu, H., Ye, X., Zhang, X., and Shen, K. "A context-aware approach for dynamic GUI testing of Android applications", in *Proceedings of the 39th International Computer Software and Applications Conference (COMPSAC)*, vol. 2, 2015, pp. 248–253, doi: 10.1109/COMPSAC.2015.77.

[79] Yan, J., Wu, T., Yan, J., and Zhang, J. "Widget-sensitive and back-stack-aware GUI exploration for testing Android apps", in *Proceedings of the IEEE International Conference on Software Quality, Reliability and Security (QRS)*, Aug. 2017, pp. 42–53, doi: 10.1109/QRS.2017.14.

[80] Rosenfeld, A., Kardashov, O., and Zang, O. "Automation of android applications functional testing using machine learning activities classification", in *Proceedings of the International Conference on Software Engineering (ICSE)*, May 2018, pp. 122–132, doi: 10.1145/3197231.3197241.

[81] Song, W., Qian, X., and Huang, J. "EHBDroid: Beyond GUI testing for Android applications", in *Proceedings of the 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Oct. 2017, pp. 27–37, doi: 10.1109/ASE.2017.8115615.

[82] Moran, K., Linares-Vasquez, M., Bernal-Cardenas, C., Vendome, C., and Poshyvanyk, D. "Crashscope: A practical tool for automated testing of android applications", in *Proceedings of the IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, May 2017, pp. 15–18, doi: 10.1109/ICSE-C.2017.16.

[83] Cao, C., Deng, J., Yu, P., Duan, Z., and Ma, X. "ParaAim: Testing android applications parallel at activity granularity", in *Proceedings of the International Computer Software and Applications Conference (COMPSAC)*, vol. 1, 2019, pp. 81–90, doi: 10.1109/COMPSAC.2019.00021.

[84] Gu, T., Cao. C., Liu, T., Sun, C., Deng, J., Ma, X., and Lü, J. "Aimdroid: Activity-insulated multi-level automated testing for android applications", in *Proceedings of the IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, pp. 103–114, doi: 10.1109/ICSME.2017.72.

[85] Frister, D., Oberweis, A., and Goranov, A. "Automated testing of mobile applications using a robotic arm," in *Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, Dec. 2020, pp. 1729–1735, doi:10.1109/CSCI51800.2020.00321

[86] Costa, P., Paiva, A. C. R., and Nabuco, M. "Pattern based GUI testing for mobile applications", in *Proceedings of the 2014 9th International Conference on the Quality of Information and Communications Technology*, Sep. 2014, pp. 66–74, doi**:** 10.1109/QUATIC.2014.16

[87] Bo, J., Xiang, L., and Xiaopeng, G. "MobileTest: A tool supporting automatic black box test for software on smart mobile devices", in *Proceedings of the 2nd International Workshop on Automation of*

Software Test (AST'07), May 2007, pp. 8–8, doi**:** 10.1109/AST.2007.9

[88] Banafshe Daragh, F. Y., and Malek, S. "Deep GUI: Black-box GUI input generation with deep learning", in *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2021, pp. 905–916, doi**:** 10.1109/ASE51524. 2021.9678778

[89] Ardito, L., Coppola, R., and Torino, P. "towards automated translation between generations of gui-based tests for mobile devices", in *Proceedings of the ISSTA/ECOOP 2018 Workshops*, Jul. 2018, pp. 46-53, doi: 10.1145/3236454.3236488

[90] Moreira, R. M. L. M., and Paiva, A. C. R. "Towards a pattern language for model-based GUI testing", in *ACM Int. Conf. Proceeding Series*, vol. 09–13-July-2014, Jul. 2014, doi: 10.1145/2721956.2721972.

[91] Morgado, I. C., and Paiva, A. C. R. "Test patterns for android mobile applications", in *Proceedings of the 20th European Conference on Pattern Language of Programs*, Jul. 2015, pp. 1-7, doi: 10.1145/2855 321.2855354.

[92] Adamsen, C. Q., Mezzetti, G., and Møller, A. "Systematic execution of android test suites in adverse conditions", in *Proceedings of the 2015 International Symposium on Software Testing and Analysis*, Jul. 2015, pp. 83–93, doi: 10.1145/2771783.277 1786.

[93] Amano, T., Kajita, S., Yamaguchi, H., Higashino, T., and Takai, M. "Smartphone applications testbed using virtual reality", in *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, Nov. 2018, pp. 422–431, doi: 10.1145/3286978.32 87028.

[94] Moreno, I. A. "Search-Based Test Generation for Android Apps", in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*, Jun. 2020, pp. 230–233, doi: 10.1145/ 3377812.3381389.

[95] De Cleva Farto, G., and Endo, A. T. "Reuse of model-based tests in mobile apps", in *Proceedings of the 31st Brazilian Symposium on Software Engineering*, Sep. 2017, pp. 184–193, doi.org/10.1145/3131151.3131160

[96] Adamo, D., Khan, K., Koppula, S., and Bryce, R. "Reinforcement learning for android gui testing", in *Proceedings of the 9th ACM SGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, Nov. 2018, pp. 2-8, doi.org/10. 1145/3278186.3278187

[97] Pan, M., Huang, A., Wang, G., Zhang, T., and Li, X. "Reinforcement learning based curiosity-driven testing of android applications", in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Jul. 2020, pp. 153–164, doi.org/10.1145/3395363.3397354

[98] Marinho, E. H., and Figueiredo, E. "PLATOOL: a functional test generation tool for mobile applications", in *Proceedings of the 34th Brazilian Symposium on Software Engineering*, Oct. 2020, pp. 548–553, doi: 10.1145/3422392.3422508.

[99] Ma, Y., Huang, Y., Hu, Z., Xiao, X., and Liu, X. "Paladin: Automated generation of reproducible test cases for android apps", in *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*, Feb. 2019, pp. 99–104, doi: 10.1145/3301293.3302363.

[100] Holzmann, C., and Hutflesz, P. "Multivariate testing of native mobile applications", in *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, Dec. 2014, pp. 85–94, doi: 10.1145/2684103.2684119.

[101] Coelho, T., Lima, B., and Faria, J. P. "MT4A: a no-programming test automation framework for android applications", in *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation*, Nov. 2016, pp. 59–65, doi: 10.1145/2994291.2994 300.

[102] Shi, S., Wang, X., and Lau, W. C. "MoSSOT: An automated blackbox tester for single sign-on vulnerabilities in mobile applications", in *Proceedings of the 2019 ACM Asia Conference on Computer and Communications Security*, Jul. 2019, pp. 269–282, doi: 10.1145/332170 5.3329801.

[103] Ermuth, M., and Pradel, M. "Monkey see, monkey do: Effective generation of GUI tests with inferred macro events", in *Proceedings of the 25th International Symposium on Software Testing and Analysis*, Jul. 2016, pp. 82–93, doi: 10.1145/2931037.2931053.

[104] Tao, C., and Gao, J. "Modeling mobile application test platform and environment: testing criteria and complexity analysis", in *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industrial Contributions to Test Automation and Model-Based Testing*, 2014, pp. 28–33, doi: 10.1145/2631890.2631896.

[105] Jha, A. K., Lee, S., and Lee, W. J. "Modeling and test case generation of Inter-component communication in android", in *Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems*, Sep. 2015, pp. 113–116, doi: 10.1109/Mobile Soft.2015.24.

[106] Turner, J., Bowen, J., and Reeves, S. "Model-based Testing of Interactive Systems using Interaction Sequences", *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. EICS, Jun. 2020, doi: 10.1145/3397873.

[107] Haller, K. "Mobile Testing", *ACM SIGSOFT Software Engineering Notes*, vol. 38, no. 6, pp. 1–8, Nov. 2013, doi: 10.1145/2532780.2532813.

[108] Mozgovoy, M., and Pyshkin, E. "Mobile farm for software testing", in *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services* Adjunct, Sep. 2018, pp. 31–38, doi: 10.1145/3236112.3236117.

[109] Yu, S., and Takada, S. "Mobile application test case generation focusing on external events", in *Proceedings of the 1st International Workshop on Mobile Development*, Oct. 2016, pp. 41–42, doi: 10.1145/3001854.3001864.

[110] Humayoun, S. R., and Dubinsky, Y. "MobiGolog: Formal task modelling for testing user gestures interaction in mobile applications", in *Proceedings of the 1st International Conference on Mobile Software Engineering and Systems*, Jun. 2014, pp. 46–49, doi: 10.1145/2593902.2593914.

[111] Ami, A. S., Hasan, M. M., Rahman, M. R., and Sakib, K. "Mobicomonkey: Context testing of Android apps", in *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems*, May 2018, pp. 76–79, doi: 10.1145/3197231.3197234.

[112] Gómez, M., Rouvoy, R., Adams, B., and Seinturier, L. "Mining test repositories for automatic detection of UI performance regressions in Android apps", in *Proceedings of the 13th International Conference on Mining Software Repositories (MSR)*, May 2016, pp. 13–24, doi: 10.1145/2901739.2901747.

[113] Ki, T., Park, C. M., Dantu, K., Ko, S. Y., and Ziarek, L. "Mimic: UI Compatibility Testing System for Android Apps", in *Proceedings of the 41st International Conference on Software Engineering (ICSE)*, May 2019, pp. 246–256, doi: 10.1109/ICSE.2019.00040.

[114] Salehnamadi, N., Alshayban, A., Lin, J.W., Ahmed, I., Branham, S. and Malek, S. "Latte: Use-case and assistive-service driven automated accessibility testing framework for android", in *Proceedings of the 43rd International Conference on Software Engineering (ICSE)*, May 2021, doi: 10.1145/3411764.3445455.

[115] Riccio, V., Amalfitano, D., and Fasolino, A. R. "Is This the Lifecycle We Really Want? An Automated Black-Box Testing Approach for Android Activities", in *Companion Proceedings for the ISSTA/ECOOP 2018 Workshops*, Jul. 2018, pp. 68-77. 2018.

[116] Amalfitano, D., Amatucci, N., Fasolino, A. R., Gentile, U., Mele, G., Nardone, R., Vittorini, V. and Marrone, S. "Improving code coverage in android apps testing by exploiting patterns and automatic test case generation", in *Proceedings of the ACM International Workshop on Long-Term Industrial Collabor-ation on Software Engineering (WISE)*, 2014, pp. 29–34, doi: 10.1145/2647648.2656426.

[117] Paulovsky, F., Pavese, E., and Garbervetsky, D. "High-coverage testing of navigation models in android applications", in *Proceedings of the 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST)*, Jun. 2017, pp. 52–58, doi: 10.1109/AST.2017.6.

[118] Xu, T., Pan, M., Pei, Y., Li, G., Zeng, X., Zhang, T., Yuetang, D., and Li, X. "GUIDER: GUI structure and vision co-guided test script repair for Android apps", in *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, Jul. 2021, pp. 191–203, doi: 10.1145/3460319.3464830.

[119] Su, T., Guozhu, M., Yuting, C., Ke, W. Weiming, Y., Yao, Y., Geguang, P., Yang, L., and Zhendong, S. "Guided, stochastic model-based GUI testing of android apps," in *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE)*, Aug. 2017, vol. Part F130154, pp. 245–256, doi: 10.1145/3106237.3106298.

[120] Panizo, L., Salmerón, A., Gallardo, M. D. M., and Merino, P. "Guided test case generation for mobile apps in the TRIANGLE project: work in progress", in *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software (SPIN)*, Jul. 2017, pp. 192–195, doi: 10.1145/3092282.3092298.

[121] Liu, Z., Chen, C., Wang, J., Huang, Y., Hu, J., and Wang, Q. "Guided Bug Crush: Assist manual gui testing of android apps via hint moves", in *Proceedings of the 44th International Conference on Software*

Engineering (ICSE), May 2022, doi: 10.1145/3491102.3501903.

[122] Tang, H., Wu, G., Wei, J., and Zhong, H. "Generating test cases to expose concurrency bugs in android applications", in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*, Aug. 2016, pp. 648–653, doi: 10.1145/2970276.2970320

[123] Dong, Z., Tiwari, A., Yu, X. L., and Roychoudhury, A. "Flaky test detection in Android via event order exploration", in *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Aug. 2021, pp. 367–378, doi: 10.1145/3468264.3468584

[124] Moran, K., and Poshyvanyk, D. "Fixing bug reporting for mobile and GUI-based applications", in *Proceedings of the 38th International Conference on Software Engineering Companion*, May 2016, pp. 831–834, doi: 10.1145/2889160.2889269

[125] Yu, S., and Takada, S. "External event-based test cases for mobile application", in *Proceedings of the 8th International Conference on Computer Science & Software Engineering*, Jul. 2015, pp. 148–149, doi: 10.1145/2790798.2790822

[126] Mahmood, R., Mirzaei, N., and Malek, S. "EvoDroid: Segmented evolutionary testing of android apps", in *Proceedings of the 22nd ACM SIGSOFT International Symposium on the Foundations of Software Engineering*, Nov. 2014, pp. 599–609, doi: 10.1145/2635868.2635896

[127] Linares-V2, pp. M. "Enabling Testing of Android Apps", in *Proceedings of the 2015 IEEE/ACM 37th International Conference on Software Engineering*, Aug. 2015, vol.2, pp. 763–765, doi: 10.1109/ICSE.2015.242

[128] Machiry, A., Tahiliani, R., and 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*, Aug. 2013, pp. 224–234, doi: 10.1145/2491411.2491450

[129] Li, Y., Yang, Z., Guo, Y., and Chen, X. "DroidBot: a lightweight UI-guided test input generator for Android", in *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Jun. 2017, pp. 23–26, doi: 10.1109/ICSE-C.2017.8

[130] Tramontana, P., Amalfitano, D., Amatucci, N., Memon, A., and Fasolino, A. R. "Developing and evaluating objective termination criteria for random testing", *ACM Transactions on Software Engineering and Methodology*, vol. 28, no. 3, Jun. 2019, doi: 10.1145/3339836.

[131] Choi, W., Sen, K., Necula, G., and Wang, W. "DetReduce: minimizing Android GUI test suites for regression testing", in *Proceedings of the 40th International Conference on Software Engineering*, May 2018, doi: 10.1145/3180155.3180173

[132] Wang, J., and Lu, J. "Detecting Non-crashing Functional Bugs in Android Apps via Deep-State Differential Analysis", *Association for Computing Machinery*, vol. 1, no. 1, 2022, doi: 10.1145/3540250.3549170.

[133] Li, X., Yang, Y., Liu, Y., Gallagher, J. P., and Wu, K. "Detecting and diagnosing energy issues for mobile applications", in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Jul. 2020, pp. 115–127, doi: 10.1145/3395363.3397350

[134] Ki, T., Simeonov, A., Park, C. M., Dantu, K., Ko, S. Y., and 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*, Jun. 2017, p. 185, doi: 10.1145/3081333.3089330

[135] Romdhana, A., Merlo, A., Ceccato, M., and Tonella, P. "Deep Reinforcement Learning for Black-box Testing of Android Apps", *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 4, Jul. 2022, doi: 10.1145/3502868.

[136] Cao, Y., Wu, G., Chen, W., and Wei, J. "CrawlDroid: Effective model-based GUI testing of Android apps", in *Proceedings of the 10th Asia-Pacific Symposium on Internetware*, Sep. 2018, pp. 1–6, doi: 10.1145/3275219.3275238.

[137] de Almeida, D. R., Machado, P. D. L., Andrade, W. L., and An, W. L. "Context-aware Android applications testing", in *Proceedings of the 34th Brazilian Symposium on Software Engineering*, Oct. 2020, pp. 283-292, doi: 10.1145/3422392.3422405

[138] Rojas, I. K. V., Meireles, S., and Dias-Neto, A. C. "Cloud-based mobile app testing framework: Architecture, implementation and execution",

in *Proceedings of the 1st Brazilian Symposium on Systematic and Automated Software Testing*, Sep. 2016, pp. 1-10, doi: 10.1145/2993288.2993301

[139] Hesenius, M., Griebe, T., Gries, S., and Gruhn, V. "Automating UI tests for mobile applications with formal gesture descriptions", in *Proceedings of the 16th ACM International Conference on Human-Computer Interaction with Mobile Devices and Services*, Sep. 2014, pp. 213–222, doi: 10.1145/2628363.2628391

[140] Ran, D. "Automated visual testing for mobile apps in an industrial setting", *Association for Computing Machinery*, vol. 1, no. 1, doi: 10.1145/3510457.3513027.

[141] Jensen, C. S., Prasad, M. R., and Møller, A. "Automated testing with targeted event sequence generation", in *Proceedings of the 2013 International Symposium on Software Testing and Analysis*, Jul. 2013, pp. 67–77, doi: 10.1145/2483760.2483777

[142] Baek, Y. M. and Bae, D. H. "Automated model-based Android gui testing using multi-level gui comparison criteria", in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Aug. 2016, pp. 238–249, doi: 10.1145/2970276.2970313

[143] Xue, F. "Automated mobile apps testing from visual perspective", in *Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis*, Jul. 2020, pp. 577–581, doi: 10.1145/3395363.3402644

[144] Jabbarvand, R., Mehralian, F., and Malek, S. "Automated construction of energy test oracles for Android", in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Nov. 2020, pp. 927–938, doi: 10.1145/3368089.3409677

[145] Yumura, T., Enomoto, M., Akashi, K., Hirose, F., Inoue, T., Uda, S., Miyachi, T., Tan, Y., and Shinoda, Y. "AOBAKO: A testbed for context-aware applications with physicalizing virtual beacons", in *Proceedings of the 2018 ACM International Joint Conference and 2018 International Symposium on Pervasive and Ubiquitous Computing and Wearable Computers*, Oct. 2018, pp. 476–479, doi: 10.1145/3267305.3267568

[146] Maggi, F., Valdi, A., and Zanero, S. "AndroTotal: A flexible, scalable toolbox and service for testing mobile malware detectors", in *Proceedings of the 3rd ACM Workshop on Security and privacy in smartphones & mobile devices*, Nov. 2013, pp. 49-54, doi: 10.1145/2516760.2516768

[147] Amalfitano, D., Amatucci, N., Fasolino, A. R., and Tramontana, P. "AGRippin: a novel search based testing technique for Android applications", in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, Aug. 2015, pp. 5–12, doi: 10.1145/2804345.2804348

[148] Jabbarvand, R., and Malek, S. "Advancing energy testing of mobile applications", in *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, Jun. 2017, pp. 491–492, doi: 10.1109/ICSE-C.2017.45

[149] Santiago, D., Clarke, P. J., Alt, P., and King, T. M. "Abstract flow learning for web application test generation", in *Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation*, Nov. 2018, pp. 49–55, doi: 10.1145/3278186.3278194.

[150] Griebe, T. T., and Gruhn, V. "A model-based approach to test automation for context-aware mobile applications", in *Proceedings of the ACM Symposium on Applied Computing (SAC)*, 2014, pp. 420–427, doi: 10.1145/2554850.2554942.

[151] Cuixiong H, and Lulian N. "A GUI bug finding framework for android applications". in *Proceedings of the 36th International Conference on Software Engineering (ICSE Companion)*, 2012, pp. 1490–1491. doi.org/10.1145/1982185.1982504.

[152] Hettab, A., Kerkouche, E., and Chaoui, A. "A graph transformation approach for automatic test cases generation from UML activity diagrams", in *Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE)*, Jul. 2015, vol. 13–17-July, pp. 88–97, doi: 10.1145/2790798.2790801.

[153] Bernaschina, C., Fedorov, R., Frajberg, D., and Fraternali, P. "A framework for regression testing of outdoor mobile applications", in *Proceedings of the 2017 IEEE/ACM 4th International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, Jul. 2017, pp. 179–181, doi: 10.1109/MOBILESoft.

[154] Baluda, M., Pistoia, M., Castro, P., and Tripp, O. "A framework for automatic anomaly detection in mobile applications", in *Proceedings of the International Conference on Mobile Software Engineering and Systems*

(MOBILESoft), May 2016, pp. 297–298, doi: 10.1145/2897073.2897718.

[155] Fazzini, M., Gorla, A., and Orso, A. "A framework for automated test mocking of mobile apps", in *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Sep. 2020, pp. 1204–1208, doi: 10.1145/3324884.3418927.

[156] Ridene Y, Belloir N, Barbier F, Couture N. A DSML for mobile phone applications testing. In Proceedings of the 10th Workshop on Domain-Specific Modeling 2010 Oct 17 (pp. 1-6).

[157] Amalfitano, D., Fasolino, A. R., Tramontana, P., and Robbins, B. "Testing Android mobile applications: Challenges, strategies, and approaches", in *Advances in Computers*, vol. 89, Academic Press Inc., 2013, pp. 1–52, doi: 10.1016/B978-0-12-408094-2.00001-1

[158] De Cleva Farto, G., and Endo, A. T. "Evaluating the model-based testing approach in the context of mobile applications", *Electronic Notes in Theoretical Computer Science*, vol. 314, pp. 3–21, 2015, doi: 10.1016/j.entcs.2015.05.002.

[159] Sarker, I. H., and Salah, K. "AppsPred: Predicting context-aware smartphone apps using random forest learning", *Internet of Things*, vol. 8, Dec. 2019, doi: 10.1016/j.iot.2019.100106.

[160] Alzaylaee, M. K., Yerima, S. Y., and Sezer, S. "Improving dynamic analysis of Android apps using hybrid test input generation", in *Proceedings of the 2017 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, 2017, (pp. 1-8) doi: 10.1109/CyberSecPODS.2017.8074845.

[161] Amalfitano, D., Fasolino, A. R., Tramontana, P., Ta, B. D., and Memon, A. M. "MobiGUITAR – A tool for automated model-based testing of mobile apps", *IEEE Software*, vol. 32, no. 5, pp. 53–59, 2014, doi: 10.1109/MS.2014.55.

[162] Yang W, Prasad MR, Xie T. A grey-box approach for automated GUI-model generation of mobile applications. InInternational Conference on Fundamental Approaches to Software Engineering 2013 (pp. 250-265). Berlin, Heidelberg: Springer Berlin Heidelberg.

[163] Azim, T., and Neamtiu, I. "Targeted and depth-first exploration for systematic testing of Android apps", in *Proceedings of the ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, 2013, pp. 641–660, doi: 10.1145/2509136.2509549.

[164] Morgado, I. C., and Paiva, A. C. R. "The iMPAcT tool for Android testing", *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. EICS, Jun. 2019, doi: 10.1145/3300963.

[165] Lin, J. W., Jabbarvand, R., and Malek, S. "Test transfer across mobile apps through semantic mapping", in *Proceedings of the 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Nov. 2019, pp. 42–53, doi: 10.1109/ASE.2019.00015.

[166] Usman, A., Ibrahim, N., Salihu I. A. "Test case generation from android mobile applications focusing on context events", *In Proceedings of the 2018 7th international conference on software and computer applications* 2018; pp. 25-30.

[167] Salihu, I. A., and Ibrahim, R. "Systematic exploration of Android apps' events for automated testing", in *ACM International Conference Proceeding Series*, Nov. 2016, pp. 50–54, doi: 10.1145/3007120.3011072.

[168] Neto NM, Vilain P, Mello RD. "Segen: Generation of test cases for selenium and selendroid", *In Proceedings of the 18th International Conference on Information Integration and Web-based Applications and Services* 2016 Nov., pp. 433-442, doi: 10.1145/3011141 .3011154.

[169] Do, Q., Yang, G., Che, M., Hui, D., and Ridgeway, J. "Regression test selection for android applications", In *Proceedings of the International Conference on Mobile Software Engineering and Systems*, May 2016, pp. 27–28, doi: 10.1145/2897073.2897 127.

[170] Pan, M., Lu, Y., Pei, Y., Zhang, T., and Li, X. "Preference-wise testing of android apps via test amplification", *ACM Transactions on Software Engineering and Methodology*. 2023 Feb 13;32(1):1-37, doi: 10.1145/3511804.

[171] Lu, Y., Pan, M., Zhai, J., Zhang, T., and Li, X. "Preference-wise testing for Android applications," in *Proceedings of the 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE)*, Aug. 2019, pp. 268–278, doi: 10.1145/3338906.3338980.

[172] Liu, Z., Chen, C., Wang, J., Su, Y., and Wang, Q. "NaviDroid: a tool for guiding manual Android testing via hint moves", *In Proceedings of the ACM/IEEE 44th*

international conference on software engineering: companion proceedings 2022 May (pp. 154-158).

[173] Yan, J., Liu, H., Pan, L., Yan, J., Zhang, J., and Liang, B. "Multiple-entry testing of Android applications by constructing activity launching contexts", in *Proceedings of the International Conference on Software Engineering (ICSE)*, Jun. 2020, pp. 457–468, doi: 10.1145/3377811.3380347.

[174] Naith, Q., and Ciravegna, F. "Hybrid crowd-powered approach for compatibility testing of mobile devices and applications", in *Proceedings of the 3rd International Workshop on CrowdSourcing in Software Engineering (CSI-SE)*, Jul. 2018, doi: 10.1145/3265689.3265690.

[175] Borges, N. P. "Data flow-oriented UI testing: Exploiting data flows and UI elements to test Android applications", in *Proceedings of the 26th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, Jul. 2017, pp. 432–435, doi: 10.1145/3092703.3098234.

[176] Hu C, Neamtiu I. "Automating GUI testing for Android applications", in *Proceedings of the 6th International Workshop on Automation of Software Test (AST)*, 2011, pp. 77–83, doi: 10.1145/1982595.1982612.

[177] Fazzini, M., Prammer, M., D'Amorim, M., and Orso, A. "Automatically translating bug reports into test cases for mobile apps", in *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, Jul. 2018, pp. 141–152, doi: 10.1145/3213846.3213869.

[178] Ravindranath, L., Nath, S., Padhye, J., and Balakrishnan, H. "Automatic and scalable fault detection for mobile applications", in *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2014, pp. 190–203, doi: 10.1145/2594368.2594377.

[179] Ngo, C. D., Pastore, F., and Briand, L. "Automated, cost-effective, and update-driven app testing", *ACM Transactions on Software Engineering and Methodology*, vol. 31, no. 4, Jul. 2022, doi: 10.1145/3502297.

[180] Keng, J. C. J. "Automated testing and notification of mobile app privacy leak-cause behaviours", in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, Aug. 2016, pp. 880–883, doi: 10.1145/2970276.2975935.

[181] Kumar, D., Peimankar, A., Sharma, K., Domínguez, H., Puthusserypady, S., and Bardram, J. E. "Deepaware: A hybrid deep learning and context-aware heuristics-based model for atrial fibrillation detection", *Computer Methods and Programs in Biomedicine*, vol. 221, Jun. 2022, Art. no. 106899, doi: 10.1016/j.cmpb.2022.106899.

[182] Li, Y.-F., Das, P. K., and Dowe, D. L. "Two decades of Web application testing—A survey of recent advances", *Information Systems*, vol. 43, pp. 20–54, 2014, doi: 10.1016/j.is.2014.02.001.

[183] Salihu, I. A., Ibrahim, R., and Mustapha, A. "A hybrid approach for reverse engineering GUI model from Android apps for automated testing", *International Journal of Engineering Research and Technology (IJERT)*, vol. 9, no. 3, pp. 45–49, 2017.

[184] Anand, S., Burke, E., Chen, T., Clark, J., Harman, M., Hierons, M., Jia, Y., McMinn, P., and Shahbaz, M. "An orchestrated survey of methodologies for automated software test case generation", *Journal of Systems and Software*, vol. 86, pp. 1978–2001, 2013.

[185] Holl, K., Scherr, S. A., and Elberzhager, F. "Using scenario-based reading for testing mobile applications with FIT4Apps", in *Proceedings of the 2018 International Conference on the Quality of Information and Communications Technology (QUATIC)*, 2018, pp. 175–183, doi: 10.1109/QUATIC.2018.00035.

[186] Amalfitano, D., Fasolino, A. R., Tramontana, P., and Robbins, B. "*Testing Android Mobile Applications: Challenges, Strategies, and Approaches*", *Elsevier*. 2013 vol. 89, pp. 1-52, doi: 10.1016/B978-0-12-408094-2.00001-1.

[187] Tonjes R, Reetz ES, Fischer M, Kuemper D. Automated testing of context-aware applications. In 2015 IEEE 82nd Vehicular Technology Conference (VTC2015-Fall) 2015 Sep, pp. 1-5. doi: 10.1109/VTCFall.2015.7390847.

[188] Siqueira, B. R., Ferrari, F. C., Souza, K. E., Santibanez, D. S. M., and Camargo, V. V. "Fault types of adaptive and context-aware systems and their relationship with fault-based testing approaches", in *Proceedings of the 2020 IEEE 13th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, 2020, pp. 284–293, doi: 10.1109/ICSTW50294.2020.00054.

[189] Coppola, R., Morisio, M., and Torchiano, M. "Mobile GUI testing fragility: A study on open-

source Android applications", *IEEE Transactions on Reliability*, vol. 68, no. 1, pp. 67–90, 2019, doi: 10.1109/TR.2018.2869227.

[190] Salihu, I. A., Ibrahim, R., and Usman, A. "A static-dynamic approach for UI model generation for mobile applications", in *Proceedings of the 2018 7th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)*, 2018, pp. 96–100, doi: 10.1109/ICRITO.2018.8748410.

[191] Junior, M. C. "Automated verification of compliance of non-functional requirements on mobile applications through metamorphic testing", in *Proc. 2020 IEEE 13th Int. Conf. Software Testing, Verification and Validation (ICST)*, pp. 421–423, 2020, doi: 10.1109/ICST46399.2020.00053.

[192] Pan, M., Xu, T., Pei, Y., Li, Z., Zhang, T., and Li, X. "GUI-guided repair of mobile test scripts", in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering Companion (ICSE-Companion)*, 2019, pp. 326–327, doi: 10.1109/ICSE-Companion.2019.00137.

[193] Behrang, F., and Orso, A. "Poster: Automated test migration for mobile apps", in *Proceedings of the 2018 International Conference on Software Engineering (ICSE)*, 2018, pp. 384–385, doi: 10.1145/3183440.3195019.

[194] Yoo, H., and Lee, Y. "An automatic mobile app testing method with user event scenario", in *Proceedings of the 18th IEEE International Conference on Mobile Data Management (MDM)*, 2017, pp. 394–396, doi: 10.1109/MDM.2017.71.

[195] Moran, K., Linares-Vásquez, M., and Poshyvanyk, D. "Automated GUI testing of Android apps: From research to practice", in *Proceedings of the 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 2017, p. 648, doi: 10.1109/ICSME.2016.79.

[196] Escobar-Velasquez, C. "Source-codeless testing for Android apps", in *Proceedings of the 2020 IEEE 13th International Conference on Software Testing, Verification and Validation (ICST)*, 2020, pp. 433–435, doi: 10.1109/ICST46399.2020.00057.

[197] Yang, S., Huang, S., and Hui, Z. "Theoretical analysis and empirical evaluation of coverage indicators for closed source app testing", *IEEE Access*, vol. 7, pp. 162323–162332, 2019, doi: 10.1109/ACCESS.2019.2951941.

[198] Sundara Rajan, V. S., Malini, A., and Sundarakantham, K. "Performance evaluation of online mobile application using Test My App", in *Proceedings of the 2014 IEEE International Conference on Advanced Communications, Control, and Computing Technologies (ICACCCT)*, 2015, pp. 1148–1152, doi: 10.1109/ICACCCT.2014.7019277.

[199] Zhang, T., Gao, J., Cheng, J., and Uehara, T. "Compatibility testing service for mobile applications", in *Proceedings of the 9th IEEE International Symposium on Service-Oriented System Engineering (SOSE)*, 2015, pp. 179–186, doi: 10.1109/SOSE.2015.35.

[200] Motan, M., and Zein, S. "Android App Testing: A Model for Generating Automated Lifecycle Tests", in *Proceedings of the 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, 2020, doi: 10.1109/ISMSIT50672.2020.9254285.

[201] Kirubakaran, B., and 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 (PRIME)*, pp. 79–84, 2013, doi: 10.1109/ICPRIME.2013.6496451.

[202] Jha AK, Kim DY, Lee WJ. A framework for testing Android apps by reusing test cases. In2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft) 2019 May 25 (pp. 20-24), doi: 10.1109/MOBILESoft.2019.00012

[203] Li, X., Jiang, Y., Liu, Y., Xu, C., Ma, X., and Lu, J. "User guided automation for testing mobile apps", in *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC)*, vol. 1, pp. 27–34, 2014, doi: 10.1109/APSEC.2014.13.

[204] Ardito, L., Coppola, R., Leonardi, S., Morisio, M., and Buy, U. "Automated test selection for Android apps based on APK and activity classification", *IEEE Access*, vol. 8, pp. 187648–187670, 2020, doi: 10.1109/ACCESS.2020.3029735.

[205] Franke, D., and Weise, C. "Providing a software quality framework for testing of mobile applications", in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pp. 431–434, 2011, doi: 10.1109/ICST.2011.18.

[206] Vilkomir, S., and Amstutz, B. "Using combinatorial approaches for testing mobile applications", in *Proceedings of the IEEE 7th International Conference on Software Testing,*

Verification and Validation Workshops (ICSTW), pp. 78–83, 2014, doi: 10.1109/ICSTW.2014.9.

[207] Shafiei, Z., and Rafsanjani, A. J. "A test case design method for context aware Android applications", in *Proceedings of the International Computer Conference on Computer Society (ICCCS Iran)*, 2020 Jan (pp. 1-8), doi: 10.1109/CSICC49403.2020.9050065.

[208] Karlsson, S., Causevic, A., Sundmark, D., and Larsson, M. "Model-based automated testing of mobile applications: An industrial case study", in *Proceedings of the 2021 IEEE 14th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 130–137, 2021, doi: 10.1109/ICSTW52544.2021.00033.

[209] Arevalo, F., Oestanto, E., and Schwung, A. "Development of a mobile app for fault detection assessment based on information fusion", in *Proceedings of the IEEE 16th International Conference on Industrial Informatics (INDIN)*, pp. 635–640, 2018, doi: 10.1109/INDIN.2018.8471933.

[210] Ahmed, M., Ibrahim, R., and Ibrahim, N. "Adaptation model for testing android application", in *Proceedings of the 2015 2nd International Conference on Computer Technology and Information Management (ICCTIM)*, pp. 130–133, 2015, doi: 10.1109/ICCTIM.2015.7224606.

[211] Coppola, R., Morisio, M., and Torchiano, M. "Maintenance of Android widget-based GUI testing: A taxonomy of test case modification causes", in *Proceedings of the 2018 IEEE 11th International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, pp. 151–158, 2018, doi: 10.1109/ICSTW.2018.00044.

[212] Mirza AM, Khan MN, Wagan RA, Laghari MB, Ashraf M, Akram M, Bilal M. ContextDrive: Towards a functional scenario-based testing framework for context-aware applications. IEEE Access. 2021 May; 9:80478-90, doi: 10.1109/ACCESS.2021.3084887.

[213] Vilkomir, S., Marszalkowski, K., Perry, C., and Mahendrakar, S. "Effectiveness of multi-device testing mobile applications", in *Proceedings of the 2nd ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft)*, vol. 400, pp. 44–47, 2015, doi: 10.1109/MobileSoft.2015.12.

[214] Merina, C., Anggraini, N., and Hakiem, N. "A comparative analysis of test automation frameworks performance for functional testing in android-based applications using the distance to the ideal alternative method", in *Proceedings of the 3rd International Conference on Informatics and Computing (ICIC)*, pp. 1–6, 2018, doi: 10.1109/IAC.2018.8780548.

[215] Patel, P., Srinivasan, G., Rahaman, S., and Neamtiu, I. "On the effectiveness of random testing for Android: Or how I learned to stop worrying and love the monkey", in *Proceedings of the International Conference on Software Engineering*, pp. 34–37, 2018, doi: 10.1145/3194733.3194742.

[216] Muccini, H., Di Francesco, A., and Esposito, P. "Software testing of mobile applications: Challenges and future research directions", in *Proceedings of the 2012 7th International Workshop on Automation of Software Test (AST)*, pp. 29–35, 2012, doi: 10.1109/IWAST.2012.6228987.

[217] Van Der Lee, W., and Verwer, S. "Vulnerability detection on mobile applications using state machine inference", in *Proceedings of the 3rd IEEE European Symposium on Security and Privacy Workshops (EuroSPW)*, pp. 1–10, 2018, doi: 10.1109/EuroSPW.2018.00008.

[218] Ricky, M. Y., Purnomo, F., and Yulianto, B. "Mobile application software defect prediction", in *Proceedings of the 2016 IEEE Symposium on Service Systems Engineering (SOSE)*, pp. 307–313, 2016, doi: 10.1109/SOSE.2016.25.

[219] Wang W, Li D, Yang W, Cao Y, Zhang Z, Deng Y, Xie T. An empirical study of android test generation tools in industrial cases. In Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering 2018 Sep (pp. 738-748), doi: 10.1145/3238147.3240465.

[220] Tu, J., Xie, X., Zhou, Y., Xu, B., and Chen, L. "A search based context-aware approach for understanding and localizing the fault via weighted call graph", in *Proceedings of the 2016 3rd International Conference on Trusted Systems and Their Applications (TSA 2016)*, pp. 64–72, 2016, doi: 10.1109/TSA.2016.20.

[221] Cho, H. T., Huang, P. C., Luo, R. H., and Kuo, Y. H. "A new context-aware application validation method based on quality-driven Petri net models", in *Proceedings of the Second International Conference on Innovative Computing and Information Control (ICICIC 2007)*, pp. 200–203, 2007, doi: 10.1109/ICICIC.2007.48.

[222] Hsu, C. W., Lee, S. H., and Shieh, S. W. "Adaptive virtual gestures for GUI testing on smartphones", *IEEE Software*, 2017, doi: 10.1109/MS.2017.265095033.

[223] Aggarwal, P. K., Grover, P. S., and Ahuja, L. "A performance evaluation model for mobile applications", in *Proceedings of the 2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU 2019)*, pp. 1–3, 2019, doi: 10.1109/IoT-SIU.2019.8777497.

[224] Li, W., Jiang, Y., Ma, J., and Xu, C. "Automatic performance testing for image displaying in Android apps", in *Proceedings of the Asia-Pacific Software Engineering Conference (APSEC 2021)*, vol. 2021-Decem, no. 1, pp. 317–326, 2021, doi: 10.1109/APSEC53868.2021.00039.

[225] Wu, H., Wang, Y., and Rountev, A. "SENTINEL: Generating GUI tests for Android sensor leaks", in *Proceedings of the International Conference on Software Engineering*, May 2018, pp. 27–33, doi: 10.1145/3194733.3194734.

[226] Nagowah, L., and Sowamber, G. "A novel approach of automation testing on mobile devices", in *Proceedings of the 2012 International Conference on Computer and Information Sciences (ICCIS 2012)*, vol. 2, pp. 924–930, 2012, doi: 10.1109/ICCISci.2012.6297158.

[227] Zhauniarovich, Y., Philippov, A., Gadyatskaya, O., Crispo, B., and Massacci, F. "Towards black box testing of Android apps", in *Proceedings of the 10th International Conference on Availability, Reliability and Security (ARES 2015)*, pp. 501–510, 2015, doi: 10.1109/ARES.2015.70.

[228] Liu, Z., Gao, X., and Long, X. "Adaptive random testing of mobile application", in *Proceedings of the 2010 International Conference on Computer Engineering and Technology (ICCET 2010)*, vol. 2, pp. 297–301, 2010, doi: 10.1109/ICCET.2010.5485442.

[229] Jamrozik, K., and Zeller, A. "Droid mate: A robust and extensible test generator for Android", in *Proceedings of the International Conference on Mobile Software Engineering and Systems (MOBILESoft 2016)*, pp. 293–294, 2016, doi: 10.1145/2897073.2897716.

[230] Morgado, I. C., and Paiva, A. C. R. "The iMPAcT tool: Testing UI patterns on mobile applications", In 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE) 2015 Nov (pp. 876-881), doi: 10.1109/ASE.2015.96.

[231] Negara, S., Esfahani, N., and Buse, R. "Practical Android test recording with Espresso Test Recorder", in *Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP 2019)*, pp. 193–202, 2019, doi: 10.1109/ICSE-SEIP.2019.00029.

[232] Lafi, M., Osman, M. S., and Wasmi, H. A. "Improved Monkey tool for random testing in mobile applications", in *Proceedings of the 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT 2019)*, pp. 658–662, 2019, doi: 10.1109/JEEIT.2019.8717506.

[233] Arif, K. S., and Ali, U. "Mobile application testing tools and their challenges: A comparative study", in *Proceedings of the 2019 2nd International Conference on Computer, Mathematics and Engineering Technologies (iCoMET 2019)*, pp. 1–6, 2019, doi: 10.1109/ICOMET.2019.8673505.

[234] Wu, G., Cao, Y., Chen, W., Wei, J., Zhong, H. and Huang, T. "AppCheck: A crowdsourced testing service for Android applications", in *Proceedings of the 2017 IEEE 24th International Conference on Web Services (ICWS 2017)*, pp. 253–260, 2017, doi: 10.1109/ICWS.2017.40.

[235] Ryan, C., and Rossi, P. "Software, performance and resource utilisation metrics for context-aware mobile applications", in *Proceedings of the International Software Metrics Symposium*, vol. 2005, no. Metrics, pp. 95–104, 2005, doi: 10.1109/METRICS.2005.44.

[236] Liu, D., Feng, Y., Zhang, X., Jones, J. A., and Chen, Z. "Clustering crowdsourced test reports of mobile applications using image understanding", *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1290–1308, 2022, doi: 10.1109/TSE.2020.3017514.

[237] Peng, C., Rajan, A., and Cai, T. "CAT: Change-focused Android GUI testing", in *Proceedings of the 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME 2021)*, pp. 460–470, 2021, doi: 10.1109/ICSME52107.2021.00047.

[238] Chen, C. M., Lin, J. M., and Lai, G. H. "Detecting mobile application malicious behaviors based on data flow of source code", in *Proceedings of the 1st International Conference on Trusted Systems and Their*

Applications (TSA 2014), pp. 1–6, 2014, doi: 10.1109/TSA.2014.10.

[239] Jabbarvand, R., Lin, J. W., and Malek, S. "Search-based energy testing of Android", in *Proceedings of the International Conference on Software Engineering (ICSE)*, vol. 2019-May, pp. 1119–1130, 2019, doi: 10.1109/ICSE.2019.00115.

[240] Qin, J., Zhang, H., Guo, J., Wang, S., Wen, Q., and Shi, Y. "Vulnerability detection on Android apps—inspired by case study on vulnerability related with web functions", *IEEE Access*, vol. 8, pp. 106437–106451, 2020, doi: 10.1109/ACCESS.2020.2998043.

[241] Chen, J., Han, G., Guo, S., and Diao, W. "FragDroid: Automated user interface interaction with activity and fragment analysis in Android applications", in *Proceedings of the 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2018)*, pp. 398–409, 2018, doi: 10.1109/DSN.2018.00049.

[242] Liu, Z., Hu, Y., and Cai, L. "Research on software security and compatibility test for mobile application", in *Proceedings of the 4th International Conference on Innovative Computing Technology (INTECH 2014) and 3rd International Conference on Future Generation Communication Technologies (FGCT 2014)*, pp. 140–145, 2014, doi: 10.1109/INTECH.2014.6927764.

[243] Amalfitano, D., Fasolino, A. R., Tramontana, P., De Carmine, S., and Memon, A. M. "Using GUI ripping for automated testing of Android applications", in *Proceedings of the 2012 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012)*, pp. 258–261, 2012, doi: 10.1145/2351676.2351717.

[244] Gao, X., Dong, Z., Tan, S. H., and Roychoudhury, A. "Android testing via synthetic symbolic execution", in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE 2018)*, pp. 419–429, 2018, doi: 10.1145/3238147.3238225.

[245] Liu, J., Xiao, X., Xu, L., Dou, L., and Podgurski, A. "DroidMutator: An effective mutation analysis tool for Android applications", in *Proceedings of the 2020 ACM/IEEE 42nd International Conference on Software Engineering: Companion (ICSE-Companion 2020)*, pp. 77–80, 2020, doi: 10.1145/3377812.3382134.

[246] Li, C., Mills, K., Niu, D., Zhu, R., Zhang, H., and Kinawi, H. "Android malware detection based on factorization machine," *IEEE Access*, vol. 7, pp. 184008–184019, 2019, doi: 10.1109/ACCESS.2019.2958927.

[247] Wang, P., Liang, B., You, W., Li, J., and Shi, W. "Automatic Android GUI traversal with high coverage", in *Proceedings of the 2014 4th International Conference on Communication Systems and Network Technologies (CSNT 2014)*, pp. 1161–1166, 2014, doi: 10.1109/CSNT.2014.236.

[248] Cruz Quental, N., de Albuquerque Siebra, C., Peixoto Quintino, J., Florentin, F., da Silva, F. Q. B., and de Medeiros Santos, A. L. "Automating GUI response time measurements in mobile and web applications", in *Proceedings of the IEEE International Conference on Automated Software Engineering (AST)*, pp. 35–41, 2019, doi: 10.1109/AST.2019.00011.

[249] Wei, L., Luo, W., Weng, J., Zhong, Y., Zhang, X., and Yan, Z. "Machine learning-based malicious application detection of Android", *IEEE Access*, vol. 5, pp. 25591–25601, 2017. doi: 10.1109/ACCESS.2019.2958927

[250] Vuong, T. A. T., and Takada, S. "A reinforcement learning based approach to automated testing of android applications", in *A-TEST 2018 - Proceedings of the 9th ACM SIGSOFT International Workshop on Automating TEST Case Design, Selection, and Evaluation, Co-located with FSE 2018*, pp. 31–37, Nov. 2018, doi: 10.1145/3278186.3278191.

[251] Modesti, P., "A script-based approach for teaching and assessing Android application development", *ACM Transactions on Computing Education*, vol. 21, no. 1, Mar. 2021, doi: 10.1145/3427593.

[252] Coppola, R., Raffero, E., and Torchiano, M. "Automated mobile UI test fragility: An exploratory assessment study on android", in *INTUITEST 2016 - Proceedings of the 2nd International Workshop on User Interface Test Automation, Co-located with ISSTA 2016*, pp. 11–20, Jul. 2016, doi: 10.1145/2945404.2945406.

[253] Ran D, Li Z, Liu C, Wang W, Meng W, Wu X, Jin H, Cui J, Tang X, Xie T. Automated visual testing for mobile apps in an industrial setting. In Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice 2022 May, pp. 55–64, doi: 10.1145/3510457.35 13027.

[254] Ahmed, S., Taj-Eddin, I. A. T. F., and Ismail, M. A. "MuHyb: A proposed mutation testing tool for hybrid mobile applications", *ACM International Conference Proceeding Series*, pp. 67–72, Nov. 2020, doi: 10.1145/3436829.3436848.

[255] Souza, M., Dias-Neto, A. C., Villanes, I. K., and Endo, A. T. "On the exploratory testing of mobile apps", *ACM International Conference Proceeding Series*, pp. 42–51, Sep. 2019, doi: 10.1145/3356317.3356322.

[256] Coppola, R., Morisio, M., and Torchiano, M. "Scripted GUI testing of android apps: A study on diffusion, evolution and fragility", *ACM International Conference Proceeding Series*, pp. 22–32, Nov. 2017, doi: 10.1145/3127005.3127008.

[257] Usman, A., Ibrahim, N., and Salihu, I. A. "Test case generation from android mobile applications focusing on context events", *ACM International Conference Proceeding Series*, pp. 25–30, Feb. 2018, doi: 10.1145/3185089.3185099.

[258] Pecorelli F, Catolino G, Ferrucci F, De Lucia A, Palomba F. Testing of mobile applications in the wild: A large-scale empirical study on android apps. In Proceedings of the 28th international conference on program comprehension 2020 Jul (pp. 296-307). doi.org/10.1145/3387904.3389256.

[259] van der Merwe, H., van der Merwe, B., and Visser, W. "Verifying Android applications using Java PathFinder", *ACM SIGSOFT Softw. Eng. Notes*, vol. 37, no. 6, pp. 1–5, Nov. 2012, doi: 10.1145/2382756.2382797.

[260] Amalfitano, D., Amatucci, N., Memon, A. M., Tramontana, P., and Fasolino, A. R. "A general framework for comparing automatic testing techniques of Android mobile apps". Journal of Systems and Software. 2017 Mar, vol. 125, pp. 322–343. doi: 10.1016/j.jss.2016.12.017.

[261] Holl, K., Vieira, V., and Faria, I. "An approach for evaluating and improving the test processes of mobile application developments", *Procedia Computer. Science 2016*, vol. 94, pp. 33–40, 2016, doi: 10.1016/j.procs.2016.08.009.

[262] Zhang, X., Breitinger, F., Luechinger, E., and O'Shaughnessy, S. "Android application forensics: A survey of obfuscation, obfuscation detection and deobfuscation techniques and their impact on investigations", *Forensic Science International: Digital Investigation*, vol. 39. Elsevier Ltd, Dec. 01, 2021. doi: 10.1016/j.fsidi.2021.301285.

[263] The Institute of Electrical and Electronics Engineers (IEEE), *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*, New York, NY, USA: IEEE, 1991.

[264] Usman, A., Boukar, M. M., Suleiman, M. A., and Salihu, I. A. "Test case generation approach for Android applications using reinforcement learning", *Engineering Technology Applied Science Research*, vol. 14, no. 4, pp. 15127–15132, 2024. doi.org/10.48084/etasr.7422.

[265] Usman, A., Ibrahim, R., Sulaiman, M. A., and Salihu, I. A. "An in-depth analysis of machine learning based techniques for automated testing of Android applications", *International Journal of Communication Networks and Information Security*, vol. 16, no. 3, pp. 663–683, 2024.

[266] Corradini, D., Montolli, Z., Pasqua, M., and Ceccato, M. "DeepREST: Automated test case generation for REST APIs exploiting deep reinforcement learning", In Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering 2024 Oct pp. 1383–1394. doi.org/10.1145/3691620.3695511

[267] Zhao, Y., Harrison, B., and Yu, T. "Dinodroid: Testing android apps using deep q-networks", *ACM Transactions on Software Engineering and Methodology. 2024 Jun 4;33(5):1-24.*, vol. 33, no. 5, pp. 1–24, Jun. 2024. doi.org/10.1145/3652150

[268] Shin, J., Hashtroudi, S., Hemmati, H., and Wang, S. "Domain adaptation for code model-based unit test case generation", InProceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), 2024, pp. 1211–1222. doi.org/10.1145/3650212.368035.

[269] Hoffmann, J., and Frister, D. "Generating software tests for mobile applications using fine-tuned large language models", In Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test *(AST)*, 2024, pp. 76–77. doi.org/10.1145/3644032.3644454.

[270] Zhang, Y., Liu, C., Xie, X., Lin, Y., Dong, J. S., Hao, D., and Zhang, L. "GUI test migration via abstraction and concretization," ACM Transactions on Software Engineering and Methodology. doi.org/10.1145/3726525

[271] Yu, C. Fang, M. Du, Y. Ling, Z. Chen, and Z. Su, "Practical non-intrusive GUI exploration testing with visual-based robotic arms," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng. (ICSE)*, 2024, pp. 1–13.

[272] Belhadi, A., Zhang, M., and Arcuri, A. "Random testing and evolutionary testing for fuzzing GraphQL APIs", ACM Transactions on the Web. vol. 18, no. 1, pp. 1–41, Jan. 2024. doi.org/10.1145/3609427.

[273] Yu, S., Fang, C., Liu, J., and Chen, Z. "Test script intention generation for mobile application via GUI image and code understanding". ACM Transactions on Software Engineering and Methodology. vol. 34, no. 1, pp. 1–29, 2025. doi.org/10.1145/372210