IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 April 2025, accepted 24 May 2025, date of publication 5 June 2025, date of current version 13 June 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3576823

APPLIED RESEARCH

ODACE-RMS: A Remote Web-Based Platform for Automated Multi-Device Android Testing and Certification

SUNDOS MOJAHED[1], (Student Member, IEEE), RÉJEAN DROUIN[2], AND LOKMAN SBOUI[3], (Senior Member, IEEE)
[1]Software and IT Engineering Department, École de Technologie Supérieure (ÉTS), University of Quebec in Montreal, Montreal, QC H2L 2C4, Canada
[2]Vidéotron, Montreal, QC H3B 1E3, Canada
[3]Systems Engineering Department, École de Technologie Supérieure (ÉTS), University of Quebec in Montreal, Montreal, QC H2L 2C4, Canada
Corresponding author: Sundos Mojahed (sundos.mojahed.1@ens.etsmtl.ca)

ABSTRACT The evolving nature of the software industry has increased the complexity and cost of software testing. This paper highlights the critical need for automation in software testing, specifically for mobile Android device certification. We introduce ODACE-RMS, a platform designed to streamline the certification process by enabling the automated execution of comprehensive telecommunication test scenarios. ODACE-RMS runs as an application on a tester's PC, featuring a browser-based interface powered by Appium. The paper also outlines ODACE-RMS's modular architecture that combines Appium, ADB, and USB-over-IP to support remote and parallel testing. With a Spring Boot backend and web-based frontend, the platform enables flexible multi-device test sessions, whether connected locally via USB or remotely through a USB-over-IP hub. These features significantly reduce certification time and allow engineers to execute tests without physically handling devices. Our study compares ODACE-RMS with traditional systems, which reduced engagement time in certification testing by 89%, significantly decreasing the need for human intervention and enhancing the overall efficiency of the certification process. Additionally, the proposed ODACE-RMS architecture results show that testing remotely is not much slower than testing locally through physical ports, even when multiple devices are tested in parallel, with an average 7% delay.

INDEX TERMS Android mobile devices, Appium, automation, certification, multi-device testing, remote testing, software testing, testing framework, USB-over-IP.

I. INTRODUCTION

Telecommunication operators are required to certify each new mobile device and software update to ensure reliable service delivery to network users while meeting marketing, operational, and legal requirements. Traditionally, the certification process was entirely manual, which demanded significant time and effort for routine tasks. This manual procedure underlined the need for automation in the certification. According to an earlier study [2], testing costs can account for nearly 50% of the total development cost; this proves the importance of efficient and streamlined testing methods [3]. Especially considering that nearly 5 million mobile devices are sold every day [4], the demand for mobile technology continues to grow. Additionally, the total number of mobile users has reached 5.75 billion in recent years, with expectations for significant growth in the near future [5], [6].

Parallel to this development, automated testing in the mobile devices industry became more important and widespread. In fact, many tools have been built to automate parts of the testing process, such as Appium, an open-source tool that uses the WebDriver protocol to perform testing of iOS and Android applications [7]. For instance, Calabash is an automation framework that aims to perform automated UI acceptance tests on the Android and iOS platforms [3]. Also, Frank is an automated acceptance testing framework for testing iOS applications [8]. On the contrary, Robotium

The associate editor coordinating the review of this manuscript and approving it for publication was Tai-Hoon Kim.

is specifically designed to automate Android application UI test cases [9].

Automation of certification is expected to increase productivity while simultaneously reducing costs by saving the certification engineers (CE) time and effort and reducing the risks of human errors [10], [11]. In addition, automated testing is a highly efficient alternative to manual testing that offers several other advantages, as follows: Providing greater flexibility for testing, handling repetitive tasks [12], making debugging easier, providing more accurate results, offering reusable processes, automatically recording test results (execution logs, counter results, summaries, etc.) and providing standardization for testing executions. These points motivate us to address new ideas that were not implemented previously, such as using the traditional automation tool to automate the testing of mobile applications to enhance the mobile phone certification process. This involves ADB (Android Debug Bridge) and Appium for automated control and testing of the device functions and features. Another idea is to integrate dedicated mobile phones acting as bots to fully automate telephony scenarios.

To adapt to the ongoing developments, we develop ODACE [1], which in French refers to: "Outil D'Automatisation des tests de CErtification" (Certification Test Automation Tool). This platform extends Appium beyond traditional application testing to full mobile device certification, integrates novel solution bots to reduce human intervention and delivers efficiency improvements.

The remainder of this paper is structured as follows: Section II describes the related study. Section III provides an overview of the certification process within the telecommunications industry. Section IV introduces the proposed automated certification solution. Section V details the remote and multi-session capabilities. Section VI discusses experimental results. Section VII discusses the platform and outlines potential future research directions. Finally, Section VIII presents the conclusions.

II. RELATED WORK

The development of an automation framework to test software with less effort and cost is a crucial goal of multiple previous works. Numerous automation tools for application testing have recently emerged. However, we identified a gap in the availability of automation tools specifically designed for the certification process in the telecom sector. Furthermore, we found a lack of platforms or tools dedicated to testing mobile phone devices' functionalities beyond just the applications. For instance, Vajak et al. in [8] presented a novel idea for automation tests, which is a simple environment that uses efficient libraries to reduce code lines and execution time. Their work can be a base upon which to build for future work. However, researchers need to find a way to test iOS and Android under the same conditions and the same script file.

Some earlier studies by Kim et al. in [13] investigated testing and automation and looked at the performance of these tools and the vulnerabilities of testing software tools. Furthermore, Salam et al. in [14] discussed an advanced automated functional testing framework for mobile applications designed to be easy to use with a testing process throughout the software development life cycle (SDLC). Also, this design proposed new ideas and solutions for issues in previous tools and frameworks. Nevertheless, this solution used many dependency libraries, so maintenance will be more complex in the future. In addition, Verma in [15] discussed using Appium to test mobile applications, but this work did not implement a touch action instance for each test case. Also, there are no log files for each test executed in this solution. Moreover, Singh et al. in [16] explained the importance of the Appium testing tool in efficient software and bug-free and quality-rich applications. Motwani et al. in [17] developed a framework for browser compatibility testing using Selenium WebDriver. However, none of the previous articles addressed the scenarios where Appium might not be the optimal choice.

Tran et al. [18] published an article about a framework based on Appuim to automate the testing of IP multimedia subsystems, which focuses on automating service validation such as voice calls and messaging within telecommunications systems. This solution could include some other tasks like taking videos of the process and rebooting the device. However, the solution does not cover testing other device functions or technologies beyond the IP Multimedia Subsystem.

Another mobile automation framework was designed and implemented to be compatible with both iOS and Android platforms by Cui et al. [19]. This framework was built based on Appium, and the user could use it to customize it and automate the testing of his own application. This solution does not cover the automation testing for the device and network functions.

Alotaibi and Qureshi [20] proposed the SerME database framework for efficient data storage and management but lacks detailed technical explanations and may be less accessible to non-technical users due to programming knowledge requirements. Additionally, Rao et al. in [21] proposed a method to develop the testing process. Also, Jain and Sharma in [22] created a test-driven automation framework by creating keyword-based test cases.

AirMochi, introduced by Lukić et al. [23], is a versatile and platform-independent tool designed for automated testing and vulnerability analysis of iOS apps. It enables remote control, video stream recording of app interfaces, and timestamps UI events. Additionally, its Model Extractor component extracts behavioural models of apps for further analysis. The tool produces accurate and valuable behavioural models. However, the solution is complex and uses many technologies and languages, which could be challenging to debug. It is limited to iOS and requires a certain level of technical expertise and knowledge to use it effectively.

Segron's Automated Testing Framework [24] is a solution designed for telecommunications services and device testing. It supports different technologies, including 5G, VoLTE, and it offers automated test execution on various devices and networks. However, it is limited to testing the network functions without the other device's essential function.

While the previous works offer robust frameworks for UI testing, they fall short in automating the comprehensive telecommunication testing required for certification [25]. Most of the platforms focused on application testing, ignoring mobile device functionality such as voice calls, SMS, and network interaction, which is required for the certification process [26], [27]. In addition, the load times of libraries before each test run were a big challenge, which increased execution time and testing the user interface [14]. Additionally, there are no log files to monitor and track the test execution process [15].

To address these limitations, we introduce ODACE-RMS (ODACE Remote Multi-Sessions), a platform to test all mobile telephony functions and both native and non-native applications with enhanced execution efficiency.

The key contributions:

- Specialized Device Certification: While existing tools focus on application-level testing, we introduce a new framework which focuses explicitly on device certification requirements to cover not only application-level testing but also the device functionalities and features, ensuring comprehensive validation for every new device software update. This specialization fills a gap not covered by general mobile testing solutions and reduces the manual testing, and engagement time for the certification engineers.
- Parallel Testing: The platform supports simultaneous testing sessions across multiple devices, enabling users to execute parallel tests on different devices without waiting for prior sessions, which improves productivity.
- Remote Testing: Incorporates a remote access feature that enables certification engineers to perform certification tasks and do their tests without being physically present in the lab. This is especially relevant in hybrid work environments, which provide more flexibility.
- Optimized Test Execution Architecture: The solution introduces a session-based initialization process that pre-loads required libraries and Appium configurations on both mobile devices and testing PCs. This significantly reduces redundant setup time across multiple tests and reduces execution time.
- Accessible Interface: Offers a user interface specifically engineered to accommodate users with different technical skill levels, lowering the difficulty of certification workflows.

III. PRELIMINARY: PROCEDURE OF MOBILES DEVICES CERTIFICATION

A. INTRODUCTION TO MOBILES DEVICES CERTIFICATION

To ensure quality services and adhere to legal regulations, telecommunications operators mandate certification for each new mobile device and software version update. This certification encompasses tests to validate device-network compatibility and the device's ability to perform basic and advanced functions. The tests range from assessing signal strength, call quality, and data transmission rates to battery life across various network conditions. Basic tests, for example, involve making calls, sending texts, and accessing emergency services (e.g., 911), whereas advanced tests include performing a conference call with multiple devices and verifying if the mobile device is transmitting in the correct frequency band combinations.

These certification tests are grouped into specific test plans, each targeting different technologies (Device, LTE, Voice over LTE (VoLTE), Voice over Wi-Fi (VoWi-Fi), 5G, etc). The four categories of test plans are:

1) **Basic Test Plan**: Encompasses essential telephony functions (voice calls, messaging, emergency calls) and mobile data (internet browsing, email) across various mobile technologies.
2) **High-level Test Plan**: Extends the Basic category to include advanced telephony functions like call forwarding and conference calls, along with non-telephony functions such as Wi-Fi connectivity and internet browsing.
3) **Regression Test Plan**: Builds upon High-level testing with thorough coverage of most functions. Primarily utilized following significant software updates (e.g., Operating System upgrades).
4) **Complete Test Plan**: This exhaustive testing phase evaluates all device features and functionalities, including the user interface, connectivity, battery performance, audio and video quality, roaming behaviour, Wi-Fi APN settings, and more. Conducting complete test plans ensures that both the device and its initial software iterations comply with company standards and governmental regulations.

Figure 1 shows the structure for the test categories and how each category contains the previous one plus other tests.

The aim of having different test plans is to cover each certification scenario with different technologies like LTE, 5G, VoLTE, and VoWi-fi, as well as device testing scenarios. For instance, the launch of a new device requires the Complete test category for each technology, while software updates are certified with Regression, High-level or Basic.

B. CERTIFICATION PROCESS AND PREREQUISITES

The certification process starts with the reception of a new device or software from the device manufacturers. The certification engineers prepare the devices before creating a ticket in a test management tool (e.g., Jira) with the relevant information about the task, adding the test list based on technology types and levels of certifications, and then start conducting manual certification tests. Certain functions, such as network connectivity and telephony functions, must pass testing with success to obtain certification approval. The process involves the following steps:

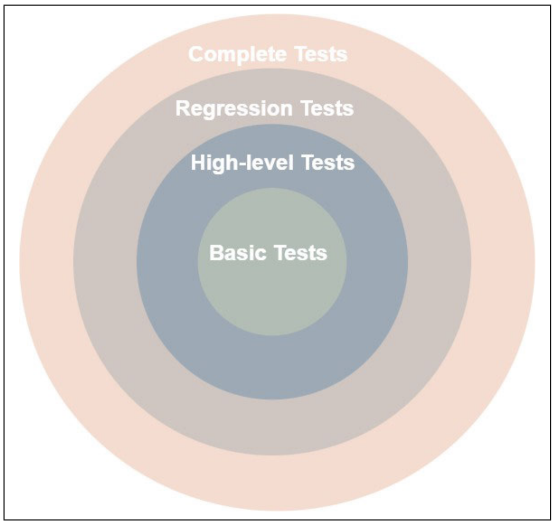


FIGURE 1. Tests categories.

1) Receive new units of device models, label them and document their details in inventory.
2) Review the handset requirements form previously filled out by the manufacturer.
3) Evaluate the level of testing required based on the software Release Notes.
4) Update the software version of the devices.
5) Create the test campaign in the test management tool.
6) Retest and update previously reported anomalies.
7) Execute the tests and analyze the results. In case of new urgent anomalies or issues discovered, manufacturers are contacted immediately.
8) Prepare the certification report to determine whether the device has passed the certification test process. If accepted, the new device and software are documented in the certification database.

This process takes a few days. Certification engineers perform all these steps manually, which costs time and effort. This explains the need for a tool or platform that can save time, cost and effort while maintaining the efficiency and accuracy of the certification process.

C. CERTIFICATION TESTS EXAMPLES

Before automating the certification tests, it is important to note that they follow the same sequence as manual executions. Verifying that each step is completed as expected before going to the next. For instance, to test the Call Function (Figure 2), the corresponding steps are given by:

- Preparation:
  - Update the DUT (Device Under Test) with the correct software version.
  - Insert a valid SIM card into DUT.
- Execution:
  - Initiate an Outgoing Voice call.
  - Verify that the call can be connected successfully.
  - Verify that the radio technology corresponds to SIM and DUT capabilities.
  - Verify that the call stays connected for 60 seconds.
  - Verify that the call termination completes correctly.
  - Repeat with an Incoming Voice Call.
- End of the Test

Another exemple is testing the Email Application, (Figure 2), the corresponding steps are given by:

- Preparation:
  - Update the DUT with the correct software version.
  - Insert a valid SIM card into DUT.
  - Activate Mobile data.
  - Turn Wi-Fi off.
- Execution:
  - Open the email application.
  - Create a new email.
  - Set destination address and subject.
  - Attach a file to the new email.
  - Send the email.
  - Wait for a reply email.
  - Open reply email.
  - Verify the attached file.
- End of the Test

If a test step fails, the test is declared Failed, and the event is documented as an anomaly reported in the test management tool and communicated to the mobile device manufacturer.

IV. PROPOSED AUTOMATED CERTIFICATION SOLUTION

The previous section mentioned that network operators must certify mobile devices to meet various requirements and specifications. The certification process can include more than 400 tests, leading to long execution times for each certification. ODACE's solution discussed reducing the time and effort spent on mobile device certification.

We design ODACE to enable certification engineers (CEs) to quickly configure and start automation processes for each DUT. The solution streamlines test execution, allowing CEs to focus on other tasks during the process. ODACE also simplifies the repetition of tests for performance testing. Moreover, we design the tool to be user-friendly, and accessible to all users, requiring no programming or technical skills.

A. AUTOMATION DESIGN

The automated certification testing process aims to enhance the efficiency of mobile device testing. This process involves the following key components:

1) Automated Test Script Development and Software Environment: Scripts are developed to automate repetitive and standardizable tests. These scripts simulate user interactions with the device, such as navigating menus, initiating calls, or sending messages. The automation framework requires a stable software environment (like Appium).

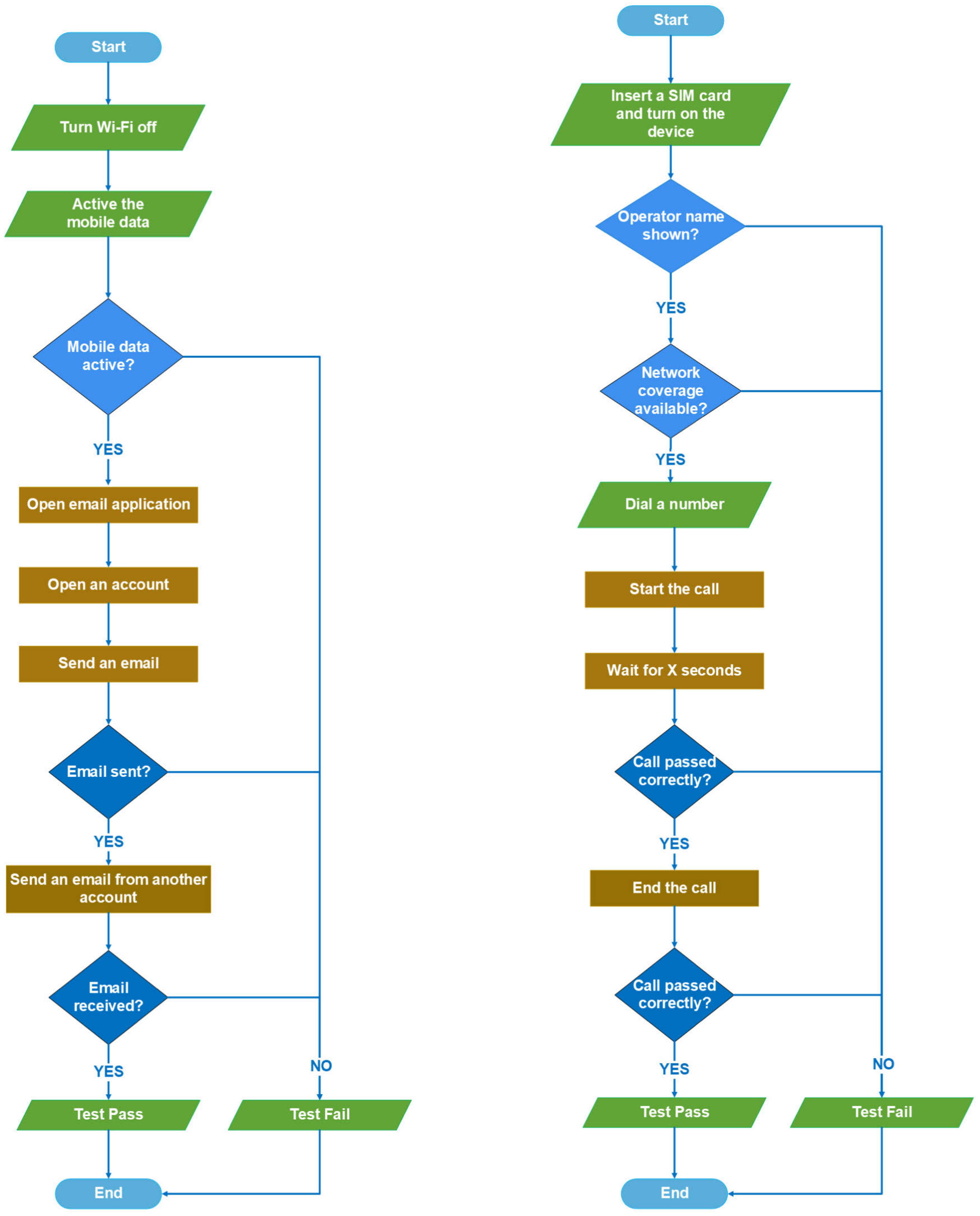


FIGURE 2. Flow chart of testing the "Call" function and the "Email" function.

2) Hardware Setup for Automated Testing: This includes setting up devices in a controlled environment where they can be remotely accessed and tested.
3) User Interface for Test Execution: A user-friendly interface developed for testers to easily initiate, monitor, and analyze test runs. This interface also allows for manual intervention when necessary.
4) Reporting and Analytics: Post-test execution, the system automatically generates detailed reports and analytics, providing insights into test coverage, defect trends, and other key metrics.

ODACE is based on Android Debug Bridge (ADB) and Appium, an HTTP server that interacts with Android clients using frameworks like Google's UiAutomator or could be run across iOS devices [28]. Additionally, ODACE employs Scrcpy, a screen mirroring tool that enables test monitoring and recording on a PC.

B. AUTOMATABILITY OF TESTS

The testing process for mobile devices encompasses a variety of tests, some of which can be automated, while others require manual/human intervention. For instance, the Emergency Call test necessitates human interaction with Public Safety Answering Point (PSAP) agents, making automation infeasible. Likewise, tests involving Google Maps and Voice Messaging systems resist automation due to the need for dynamic responses to voice instructions.

Additionally, the Factory Reset test requires manual reactivation of the USB debugger mode. Furthermore, both video call and audio tone tests require human supervision to verify audio and image quality. Most tests are performed under live commercial cellular coverage, requiring control only over the tested device. However, a subset of tests must be executed under controlled radio environments, such as lab Radio Frequency (RF) boxes, preventing automation. Additionally, certain physical actions, like swapping SIM cards or connecting headphones, are manual by nature and thus resistant to automation. There are also tests that remain unautomated due to their infrequent use and the lack of substantial benefits that automation would bring. In addition, some tests can be partially automated to reduce manual interactions. Therefore, we propose an automatability percentage that indicates if the test can be automated or not.

C. HARDWARE ARCHITECTURE

The ODACE platform has developed significantly to improve efficiency and scalability. The original ODACE hardware architecture comprised three main components:

1) Personal Computer (PC): Serving as the system's server, the PC ran the required software and supported any operating system to execute the certification processes.
2) Device Under Test (DUT): The DUT is the primary component. The certified engineer (CE) connects it to the PC via a USB cable to initiate the certification process. It requires an active internet connection through both a Wi-Fi access point and mobile data over a cellular network to perform the tests.
3) Bots: We propose the concept of Bots, which are Android mobile phones dedicated to executing the tests and interacting with the DUT autonomously, eliminating the need for manual intervention.
   In the exemple of call testing, an additional device other than the DUT is required. ODACE-RMS implements a novel solution to test telephony functions based on the 'bot' phones instead of relying on other DUTs.
   The bots are programmed to execute various tasks using the Automate Android application. Custom-made automated "flows" handle tasks like receiving and sending SMS and MMS, initiating and answering calls, etc. For example, when a bot receives a call, it searches for the correct Automate flow and follows the designed steps automatically. As shown in Figure 3, the Bot will wait for 2s and then answer the call.
   Bots are available in a pool and are listed with their MSISDN (phone number) and capabilities - radio technology, voice codec and services - in a shared document. ODACE selects bots randomly and updates their success/fail counters to automatically eliminate those that are not working correctly from the pool.
4) Server: The server hosts the database, which stores the necessary data for user profiles and other required data and manages the DUT information. This enables ODACE to efficiently handle multiple sessions both locally when running on a single PC and remotely across multiple users.
5) USB-Over-IP Server: This component is the core for the remote testing feature in our solution. It enables devices connected to physical USB ports to be accessed over the network as if locally connected.

Figure 4 shows the hardware architecture components.

D. SOFTWARE ARCHITECTURE

The ODACE-RMS platform is composed of several software components, each of which plays a critical role in automating the certification process for Android devices. This Architecture introduces additional capabilities to support scalability, multi-session management, and remote accessibility. Following, we outline the key components:

1) Device Under Test (DUT) Software: Device manufacturers regularly release new software or new updates for their software, which could be a major update, a fault fixing, or a security edit. These updates sometimes necessitate adaptations within the ODACE-RMS software architecture to maintain compatibility with new versions. Consequently, DUT software is a crucial component of the platform.
   For a PC to control DUT via USB, Android security mandates that USB access permission is allowed from

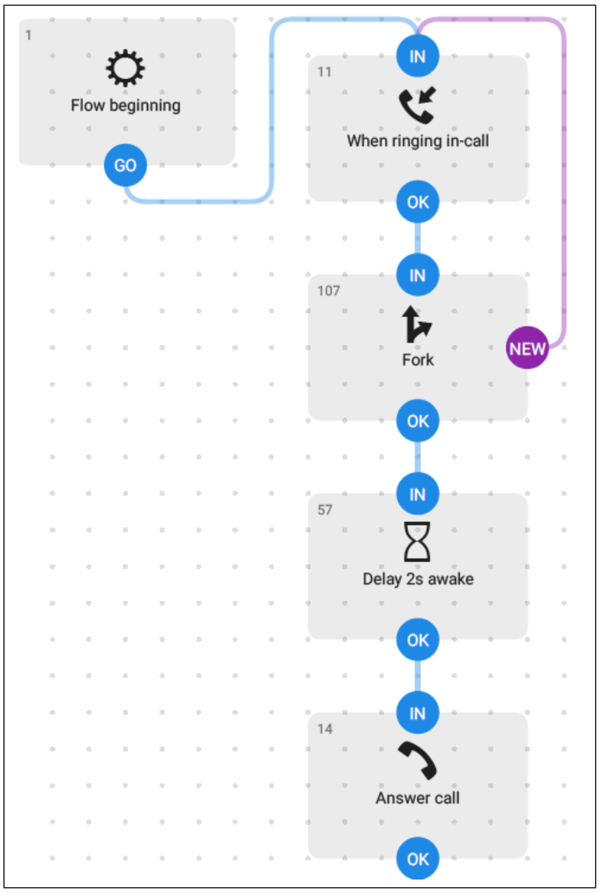


FIGURE 3. Incoming call flow chart - Bot.

the device via an on-screen pop-up, as shown in Figure 5.
Remote access to the device is blocked until that is granted, which was a big challenge for us when implementing the remote feature. This solution is introduced to solve that challenge and complete that step without manual intervention. An Automate application program which automatically clicks "Allow" upon the appearance of the "Allow USB debugging?" message. The automated workflow is shown in Figure 6.

2) ODACE-RMS Frontend and Backend: It includes the following parts:
   a) Backend: The backend is implemented using the Spring Boot Java framework, offering a scalable and robust architecture. This design enables the management of multiple devices and processes simultaneously. Additionally, we use ADB commands to execute various device actions.
   b) Frontend: The frontend is developed using jQuery and HTMX, enabling dynamic web interfaces for enhanced user experience. These technologies simplify the creation of responsive and interactive interfaces for engineers managing the certification process.

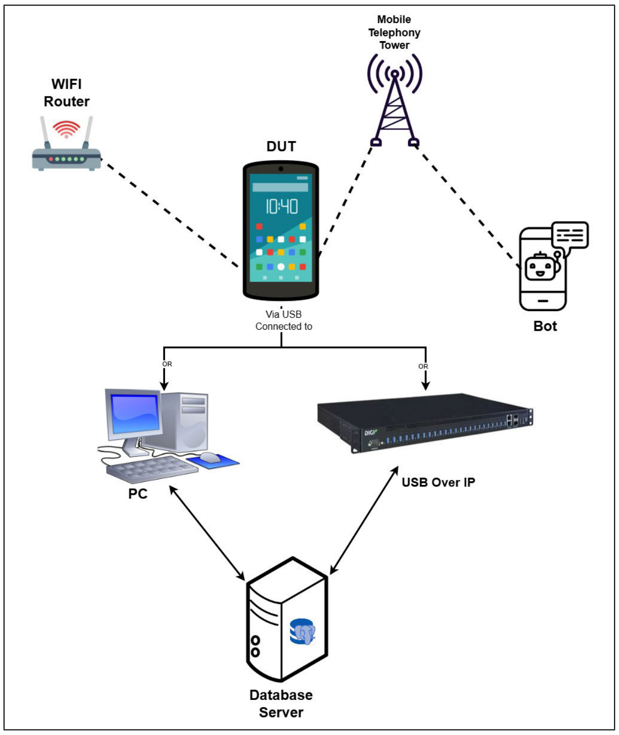
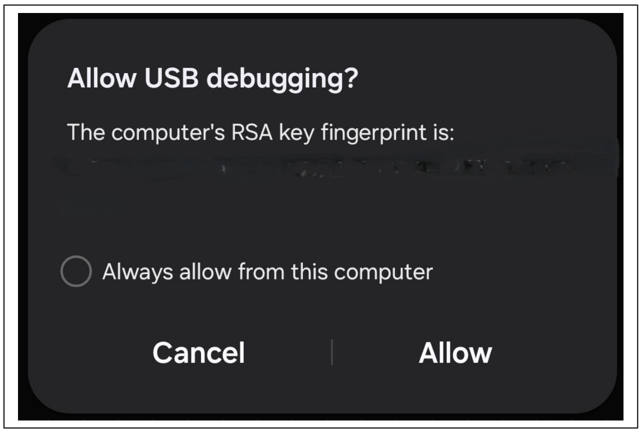


FIGURE 4. ODACE-RMS hardware architecture.


Allow USB debugging?

The computer's RSA key fingerprint is:

[redacted]

Always allow from this computer

Cancel Allow

FIGURE 5. USB debugging notification.

3) Appium-based Testing Tool: Appium is the core testing tool within ODACE-RMS, chosen for its superior features compared to alternative tools, as we explained previously. Appium supports a wide range of programming languages, excels in black-box testing, and offers extensive compatibility with different devices, as highlighted by da Silva and de Souza Santos [29]. This tool allows ODACE-RMS to execute test scripts that simulate user interactions, validating both the functionality and performance of DUT.
4) Bots Automation Application: ODACE-RMS employs bot devices to perform some tests that require

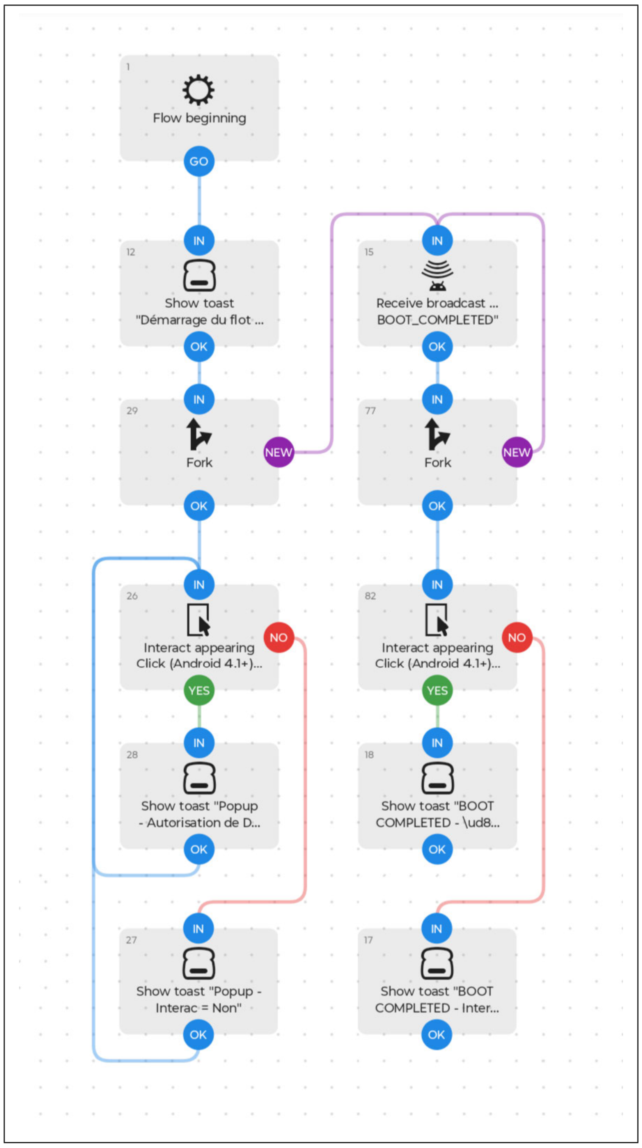
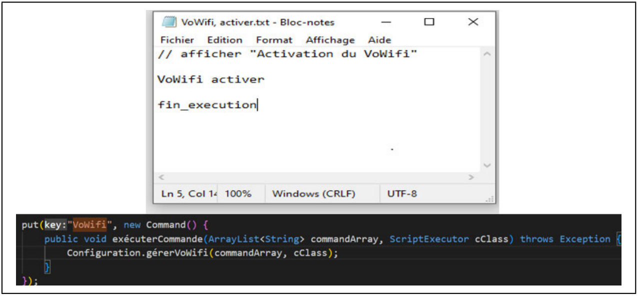

1
Flow beginning
GO
12
IN
Show toast "Démarrage du flot ...
OK
29
IN
Fork
NEW
OK
26
IN
Interact appearing Click (Android 4.1+)...
NO
YES
28
IN
Show toast "Popup - Autorisation de D...
OK
27
IN
Show toast "Popup - Interac = Non"
OK
15
IN
Receive broadcast ... BOOT_COMPLETED"
OK
77
IN
Fork
NEW
OK
82
IN
Interact appearing Click (Android 4.1+)...
NO
YES
18
IN
Show toast "BOOT COMPLETED - \ud8...
OK
17
IN
Show toast "BOOT COMPLETED - Inter...
OK

FIGURE 6. ODACE-RMS USB connection workflow.



```
// afficher "Activation du VoWifi"

VoWifi activer

fin_execution
```

```
put(key:"VoWifi", new Command() {
    public void exécuterCommande(ArrayList<String> commandArray, ScriptExecutor cClass) throws Exception {
        Configuration.gérerVoWifi(commandArray, cClass);
    }
});
```

FIGURE 7. Example implementations of ASCII text handling in ODACE code.

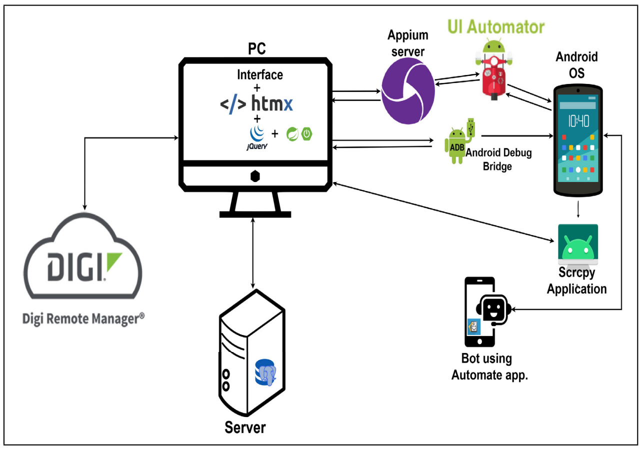


FIGURE 8. ODACE-RMS software architecture.

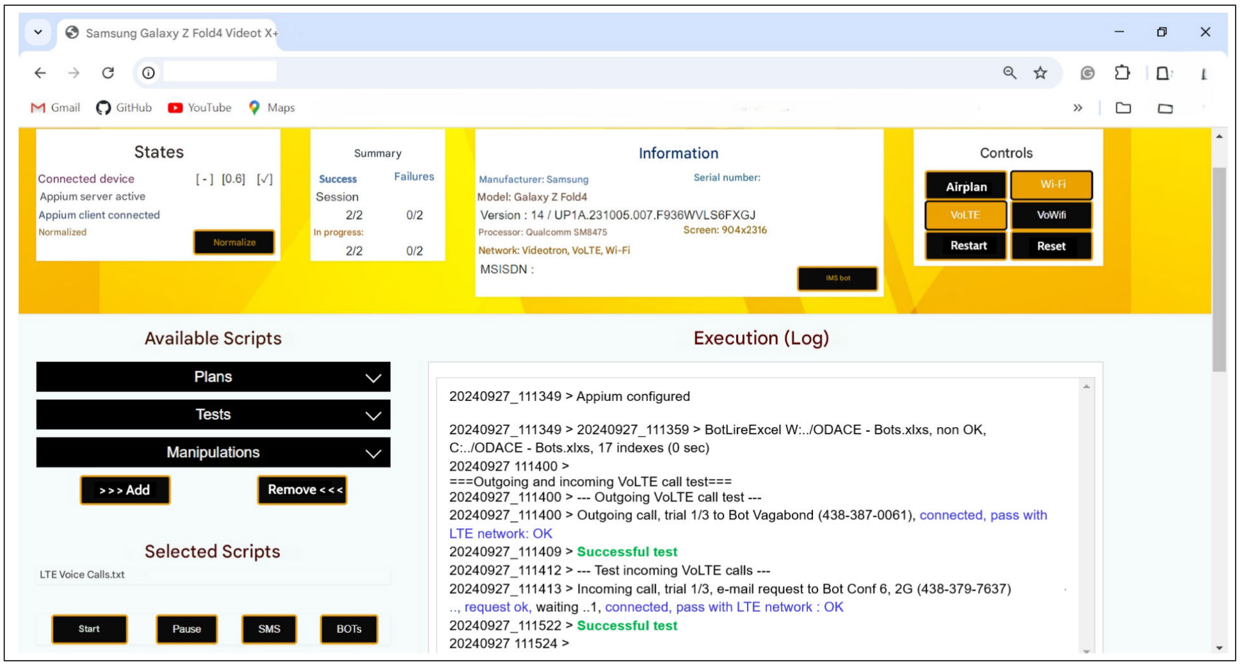
interaction with DUT. At each Bot, we develop a flow of the "Automate" Android application (developed by LlamaLab) that controls these devices, which uses a customized automation program (flowchart) for specific tasks and functions such as answering incoming calls or initiating calls to particular numbers without human interaction.

5) Database Integration: ODACE-RMS integrated PostgreSQL for managing and storing data related to bots, devices, test results, and user profiles. Integrated with the Spring Boot backend, this database mainly ensures efficient management of session data and supports multi-session operations.

6) Digi Remote Manager: ODACE-RMS use Digi Remote Manager to ensure remote device management and connectivity. This component provides functionalities such as streamlined deployment, asset management, automated updates, and real-time alerts [30]. It connects with the DIGI hub that provides USB over IP technology, facilitating remote device management and connectivity.

7) ODACE-RMS Scripts: ODACE-RMS execution is driven by script files that define the actions to be performed. Script files are provided with ODACE-RMS to cover defined test sets (Plans), others are used to execute each specific test (Tests), and a set of command files provide manipulations (Manipulations) like activating Wi-Fi, managing Plane mode, restarting the DUT, etc.
Script files are in plain ASCII text format. Users can view, edit, and create them, providing flexibility and modularity. These script files are parsed by the dedicated LineScript class. Each line in these files is a combination of a command and predefined parameters in LineScript, which calls the required text execution methods as shown in Figure 7.
Script files make the interaction between the user and the tool more user-friendly and easier to understand and allow for test and manipulation customization without the need to edit the source code ODACE-RMS.

Figure 8 explains the software architecture and the connection between them.



States
Connected device [-] [0.6] [√]
Appium server active
Appium client connected
Normalized

Summary

	Success	Failures
Session	2/2	0/2
In progress:	2/2	0/2

Information
Manufacturer: Samsung
Serial number:
Model: Galaxy Z Fold4
Version : 14 / UP1A.231005.007.F936WVLS6FXGJ
Processor: Qualcomm SM8475
Screen: 904x2316
Network: Videotron, VoLTE, Wi-Fi
MSISDN :

Controls

Selected Scripts
LTE Voice Calls.txt

Execution (Log)

```
20240927_111349 > Appium configured

20240927_111349 > 20240927_111359 > BotLireExcel W:../ODACE - Bots.xlxs, non OK,
C:../ODACE - Bots.xlxs, 17 indexes (0 sec)
20240927 111400 >
===Outgoing and incoming VoLTE call test===
20240927_111400 > --- Outgoing VoLTE call test ---
20240927_111400 > Outgoing call, trial 1/3 to Bot Vagabond (438-387-0061), connected, pass with
LTE network: OK
20240927_111409 > Successful test
20240927_111412 > --- Test incoming VoLTE calls ---
20240927_111413 > Incoming call, trial 1/3, e-mail request to Bot Conf 6, 2G (438-379-7637)
.., request ok, waiting ..1, connected, pass with LTE network : OK
20240927_111522 > Successful test
20240927 111524 >
```

FIGURE 9. **ODACE-RMS user interface (translated from French).**

V. REMOTE AND MULTI-SESSION CAPABILITIES

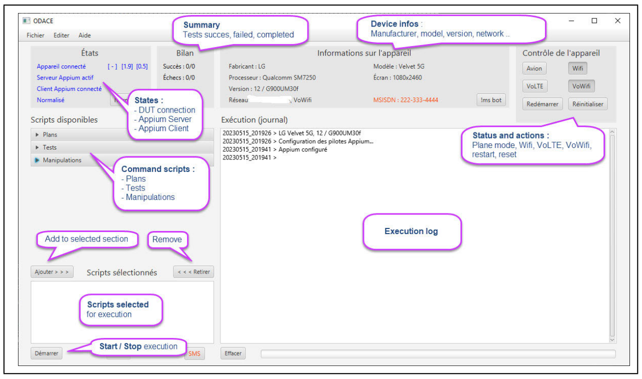
ODACE-RMS is a solution built upon the ODACE platform. The old platform faces challenges like being limited to automation only and single-device testing, which restricts engineer productivity, and the need for physical presence to manage devices that can't leave the operator's premises for confidentiality. This limitation is significant under remote working conditions post-COVID-19, motivating us to a transformative approach to mobile Android device certification, leveraging automation to enhance efficiency, flexibility, and scalability in certification testing by improving the solution to be a remotely multi-device platform.

A. MULTI-SESSION USER INTERFACE

The user interface (UI) is designed to reflect the certification context and be user-friendly for certification engineers without the need for special training or prior programming expertise.

In ODACE-RMS UI, the main sections and layout have been preserved to ensure an easy transition for certification engineers who are used to using the previous ODACE interface. This approach minimizes the need for additional training and ensures that users can easily adopt the new UI.

We redesigned using Spring Boot's capabilities, transitioning from the previous FXML interface, as shown in Figure 10, to an HTML-based web interface accessible through a browser. This redesign enhances the user experience by making the system more user-friendly and flexible. The web-based interface allows users to open multiple tabs simultaneously when they run multiple ODACE-RMS sessions.



États
Appareil connecté [-] [1.9] [0.5]
Serveur Appium actif
Client Appium connecté
Normalisé

Bilan
Succès : 0/0
Échecs : 0/0

Informations sur l'appareil
Fabricant : LG
Modèle : Velvet 5G
Processeur : Qualcomm SM7250
Écran : 1080x2460
Version : 12 / G900UM30f
Réseau , VoWifi
MSISDN : 222-333-4444

Exécution (journal)

```
20230515_201926 > LG Velvet 5G, 12 / G900UM30f
20230515_201926 > Configuration des pilotes Appium...
20230515_201941 > Appium configuré
20230515_201941 >
```

FIGURE 10. **ODACE user interface.**

As shown in Figure 9, the redesigned UI is organized into several key top and middle parts. The top part includes:

- **Tools and Device Status:** Located in the top left corner, this section provides real-time updates on critical system components, including the ADB connection, the status of the Appium server (PC), the Appium client (device), and the device normalization status (e.g., language and other settings required for automation).
- **Test Results Counters (Summary):** The second part of the top row displays counters summarizing the results of the executed tests, which provide users with a quick overview of progress.
- **Device Information:** The third section on the top row presents detailed information about the connected device, such as brand, model, software version, and network connection status.

- **Status and Actions:** The rightmost section of the top row includes buttons for executing basic device functions, enabling quick interaction and control.

The central section includes:

- **Script Management:** The middle row includes an accordion menu, which gives access to the different script categories, including Plans, Tests, and Manipulations. Users can select scripts for execution. The selected scripts are displayed in the Scripts Selected section below.
- **Execution Log:** Occupying the rest part of the interface, this section displays real-time progress and detailed logs of ongoing tests.

This structured layout improves the overall efficiency of device certification tasks and ensures that it is easy to use for both novice and experienced users.

However, we face challenges with multi-session testing. Specifically, CPU usage becomes 100% when running two or more sessions simultaneously, which causes significant delays. We addressed this challenge by optimizing the code. The significant improvement was minimizing the number of ADB shell commands opened and closed during the tests. This optimization significantly reduced CPU usage and resolved the delays in multi-session scenarios.

B. REMOTE DEVICE TESTING USING USB-OVER-IP

The need for remote features has become more important, especially with the growing trend of hybrid work in companies. This need encouraged us to find a solution that allows certification engineers to interact with devices remotely, improving flexibility and efficiency.

To implement this remote feature, we explored technologies that can share USB connections over the network. After evaluating different options, we decided to use USB-over-IP technology, which allows remote access to physical USB devices as if they were connected locally.

In our ODACE-RMS solution, we chose the DIGI AnywhereUSB Plus server,[1] which supports USB-over-IP and is reliable for multi-device environments. This server acts as a bridge between the DUTs and remote engineers. We connect the devices to the DIGI server using USB cables. Then, from the engineer's PC, the corresponding ports can be mapped through the DIGI software.

Once connected, ODACE-RMS interacts with the devices as if they are physically attached to the engineer's PC. For example, running the ‘adb devices‘ command will list both local and remotely connected Android devices, allowing seamless integration into the testing workflow. The number of users who can test at the same time depends on the number of server ports. Each port can also support multiple devices using USB hubs, which gives the platform more scalability and allows parallel testing across many devices and users. Figure 11 shows the design of the remote feature components and the connection between them.

[1] More details available at https://www.digi.com

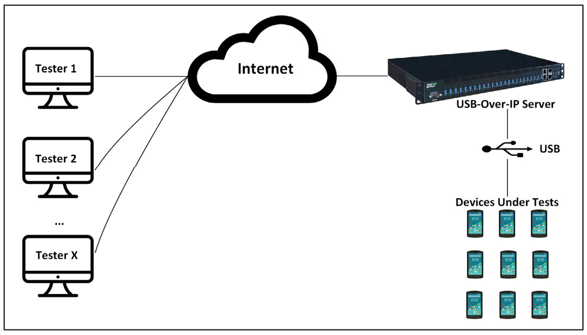


FIGURE 11. ODACE-RMS remote components management design.

C. ODACE-RMS WORKFLOW

1) **Device Connection:** The ODACE-RMS testing process begins when a user connects a DUT to the PC and starts the application. For the local version, the DUT is connected directly to the PC via USB. In the remote scenario, the DUT is already connected to a USB-over-IP server in the lab.
2) **Device Detection and Info Collection:** ODACE-RMS monitors the PC's USB ports and uses ADB commands to detect connected devices automatically. Once detected, the platform collects the main details (model, OS version, software version, etc.) and then stores them in a database.
3) **Device Selection and Reservation:** If only one device is detected, ODACE-RMS automatically assigns it to the active UI tab. If multiple devices are connected, the user manually selects one. The selected device is then reserved in the database and disabled in other tabs to avoid conflicts.
4) **Appium Configuration:** To minimize delays, ODACE-RMS collects necessary information from shared files, such as bot data, to prepare for the tests efficiently.
5) **Test Preparation:**
    a) To save time, ODACE-RMS retrieves supporting information (e.g., bot data, Clippy Phone) from shared files before starting tests.
    b) Whether using USB or USB-over-IP, ODACE-RMS behaves the same. All data collection and actions are performed using ADB, ensuring consistency in both local and remote setups.
    c) When started, ODACE-RMS displays all available devices (local and remote). If a user selects a remote DUT, the platform reserves the device until the user releases it. By default, the system supports up to eight parallel devices per user. This limit can be changed by modifying the `ADB_LOCAL_TRANSPORT_MAX_PORT`. However, the default limit is practical for our solution since users typically test three to four devices simultaneously. For remote testing, the maximum

number of users depends on the port numbers of the hub.

6) **Test Plan Execution:** ODACE-RMS offers users various test execution options. Users can select predefined test Plan scripts, which consist of a set of tests and manipulations corresponding to specific test types (e.g., VoLTE, VoWi-Fi, 5G) and levels (e.g., Basic, High-level, Regression, or Complete) as detailed in Section III-A. For example, users can choose plans to test LTE Basic, VoLTE Complete, or Wi-Fi Calling Basic. Alternatively, users can execute specific actions by selecting individual Test and Manipulation scripts.
7) **Real-Time Monitoring:** Every three seconds, ODACE-RMS checks for new devices, disconnections, or status changes. It also verifies radio tech, call status, mobile data, and Wi-Fi. Additionally, the proposed solution verifies the status of the Appium server, ensuring that it remains active and that the DUT's Appium client is connected during the test execution.
8) **Script Execution and Reporting:** Upon script selection, the user clicks "Execute." ODACE-RMS performs required tasks by using the appropriate methods. Upon completion, it exports a report summarizing the outcome. The system then returns to wait for new commands and tests to execute.

Figure 12 presents the process of the platform.

D. CASE STUDY-MOBILE ORIGINATING CALL

One of the most important functions that need to be verified is The Mobile Originating (MO) call test. This test verifies a device's ability to initiate and complete outgoing calls. This fundamental test ensures proper network registration, call setup, voice transmission, and call termination functionality from the device under test.

To validate the MO call function on a new software version, the CE performs the following steps:

1) **Preparation of DUT:** Update the DUT with the new software version, insert an operator SIM card, activate USB debugging, and connect the device to the PC.
2) **Launching ODACE-RMS:** Start ODACE-RMS on the PC to initiate the Appium server and verify the presence of the DUT.
3) **Device Profile and Status Collection:** ODACE-RMS continually collects and displays the DUT's profile and status on its user interface, logging this information simultaneously.
4) **Activation of Appium Client:** ODACE-RMS prepares the DUT for testing by activating the Appium client.
5) **Test File Selection and Execution:** The user selects the necessary test files, adds them to the script list and initiates the testing process.
6) **Call Test Procedure:** ODACE-RMS conducts the call test by randomly selecting a BOT (Breakout Tester) and controlling the DUT to initiate a call. It monitors the time and waits for the call to be established.

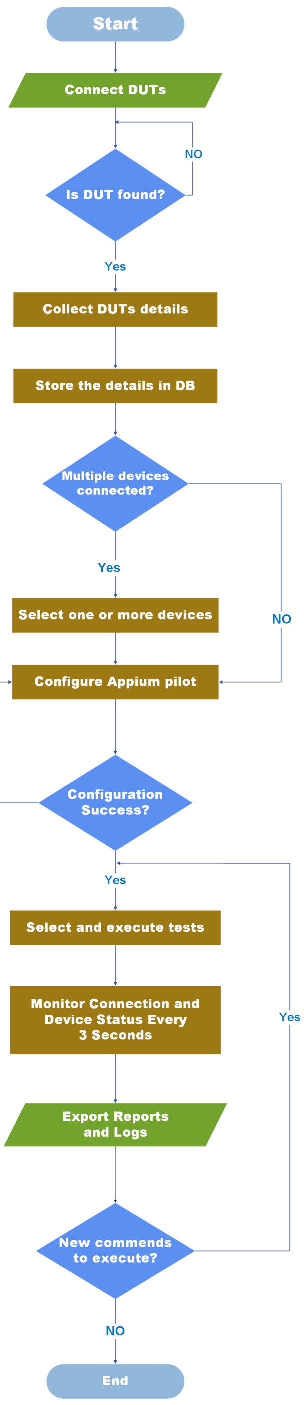


FIGURE 12. ODACE-RMS workflow.

7) **Handling Failed Call Attempts:** If a call setup fails due to BOT unavailability or malfunction, ODACE-RMS tries with another BOT. Three unsuccessful attempts with different BOTs resulted in a test failure.
8) **Successful Call Handling:** Upon successful call connection, ODACE-RMS checks the Voice technology on the DUT and then terminates the call.

9) **Test Outcome Determination:** The test is declared a PASS if the call setup, voice technology verification, and call termination are successful; otherwise, it is marked as FAIL.

This process ensures that the call function is thoroughly tested and malfunctions are detected for investigation by engineers who will create anomaly documents and report to the manufacturer when necessary. The test cases in the remote testing feature remain the same as those in local testing. The only difference between local and remote testing is the method of connection. Local devices are connected directly via USB, while remote devices are connected using USB-over-IP technology from the DIGI hub. Despite this difference in connection methods, the execution process is exactly the same for both scenarios.

Figure 13 describes the testing process.

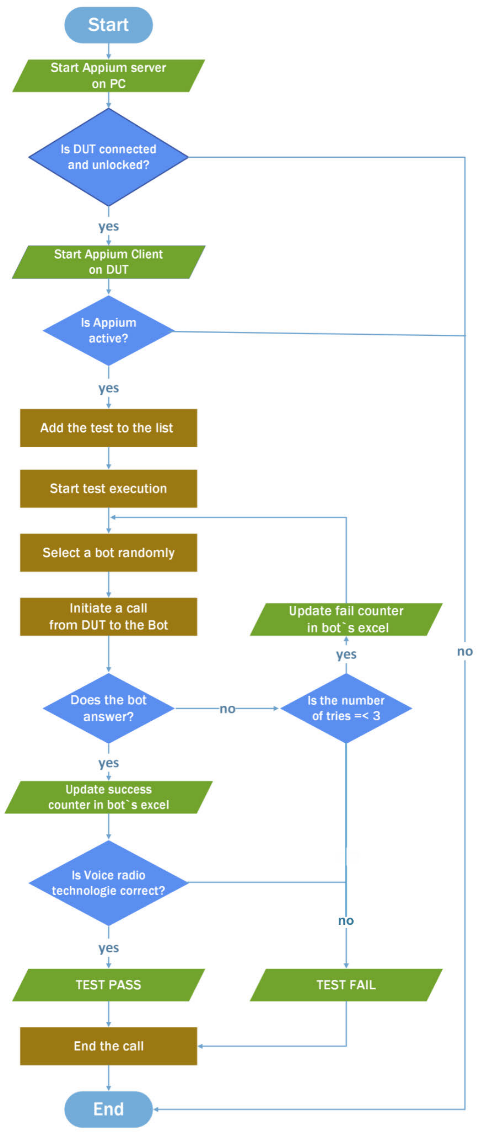

Start
Start Appium server on PC
Is DUT connected and unlocked?
yes
Start Appium Client on DUT
Is Appium active?
yes
Add the test to the list
Start test execution
Select a bot randomly
Initiate a call from DUT to the Bot
Update fail counter in bot`s excel
yes
no
Does the bot answer?
no
Is the number of tries =< 3
yes
Update success counter in bot`s excel
Is Voice radio technologie correct?
no
yes
TEST PASS
TEST FAIL
End the call
End

FIGURE 13. Case study: Mobile originating call test flow chart - ODACE-RMS.

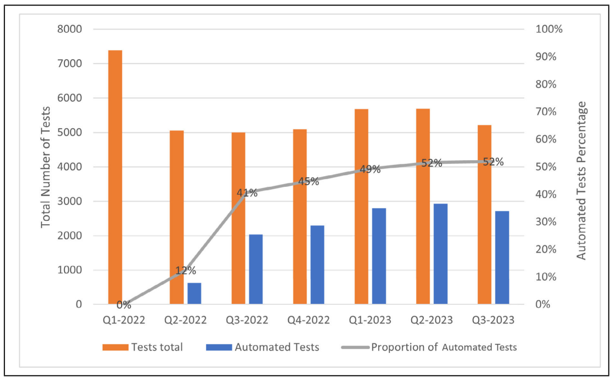


FIGURE 14. Quarterly numbers of certification tests at Vidéotron.

VI. EXPERIMENTAL RESULTS

A. AUTOMATION RESULTS

To assess the proposed automation solution in the certification process, we evaluate its performance through a series of benchmarks, comparing its execution times, accuracy, and reliability against manual testing processes in previous certification scenarios. This evaluation focuses on the automated solution through the ODACE platform. We, therefore, tracked the number of tests executed by the Certification team in Vidéotron during each quarter from Q1 2022, before ODACE introduction, until Q3 2023. Automated tests were gradually introduced from Q2-2022 to Q1-2023. During the total period, 39135 tests were performed, of which 13406 were with ODACE. On average, 5528 tests are performed every quarter, of which, since the complete introduction of ODACE, 2815 are automated. Figure 14 shows the number of tests performed quarterly; during the last quarters of 2023, ODACE succeeded in automating 52% of all tests executed.

To evaluate ODACE, we compare the test execution durations of manual testing with automated testing. The time required for preparation steps has been excluded since it is always the same. Table 1 shows the manual and automated time of ODACE quarterly. We clearly show that the time required for automated tests is less than for manual testing execution by 19% to 23%. The average test execution time is 16.7h/month for each certification engineer and is reduced to 12.9h/month when the same tests are done by ODACE. This reflects the time-saving by the tool even if we ignore the engagement time (defined as the time when the CE is engaged in the testing process) and assume that the user should monitor the devices, whether manually or through automation.

TABLE 1. Manual time vs. Automated time.

Quarter/year	Manual Time (h)	Automated Time (h)
Q1-2023	56.1	45.2
Q2-2023	53.9	41.5
Q3-2023	48.2	36.9
Sum	158.2	123.7
Average	52.7	41.2

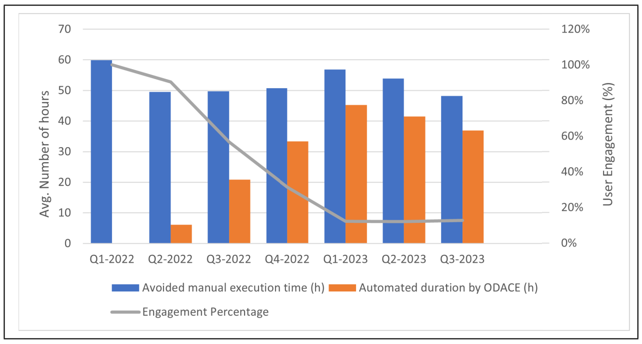


FIGURE 15. Quarterly time of certification tests with user engagement.

Figure 15 shows the times elapsed to execute predefined 'automatable' tests when performed manually or using ODACE during the period covered.

Another evaluation presented by Figure 15 compares the user engagement time, which is one of the most important aspects of evaluating the solution. As discussed previously in Table 1, ODACE cuts test execution time by 21.8%, but it also allows the certification engineer to do other tasks while ODACE executes the tests. In fact, the certification engineer engagement time for manual testing is 100%, while automated execution cuts that to 11%, meaning a reduction of 89% in terms of engagement time.

B. REMOTE AND MULTI-SESSION RESULTS

In this part of the evaluation, we focus on determining how the remote multi-session feature impacts time efficiency, operational flexibility, and CPU usage among various browsers.

To evaluate ODACE-RMS's performance, we designed different scenarios to highlight improvements in process times, remote capabilities, and overall efficacy. Figure 16 presents these scenarios, each based on more than 50 test cases. We then collected the average execution time in seconds for comparison. All tests were conducted in the same environment while varying the number of devices to test the multi-session capability. The results for the classic ODACE curve were obtained by multiplying the execution time of a single device by the number of devices, as the classic version does not support parallel testing (test sequentially) For example, the test locally takes approximately 81.7 sec. to complete. Without multi-session support, testing four devices sequentially would require around 326.8 seconds in total. In contrast, with ODACE-RMS's multi-session feature, the same tests can be executed in parallel, reducing the total execution time to 86–91.7 second. Which is significant time-saving. To further evaluate the remote feature, we test ODACE-RMS under different connectivity conditions. First, direct the USB connection to the PC. In the second case, remote access through a DIGI hub on the same network (geographically close). Finally, remote access from outside Montréal while the devices remained in the lab (over 20 km away). These scenarios covered various aspects, including testing Appium, ADB commands, calling features, SMS, network performance, and other test cases relevant to the certification process.

The results showed consistency in the execution time as more devices are added with ranging between 81.0 and 84.0 seconds for up to four devices, because of parallel execution. In remote testing scenarios from within Montreal, execution times increased slightly due to network overhead with ranging from 85.6 to 90.0 seconds. Moreover, when devices are tested from outside Montreal (over 20 km), execution times increase further to between 86.1 and 95.0 seconds, primarily due to added network latency. Despite these increases, the delay introduced by remote testing remains within an acceptable range of 2% to 7% compared to local execution.

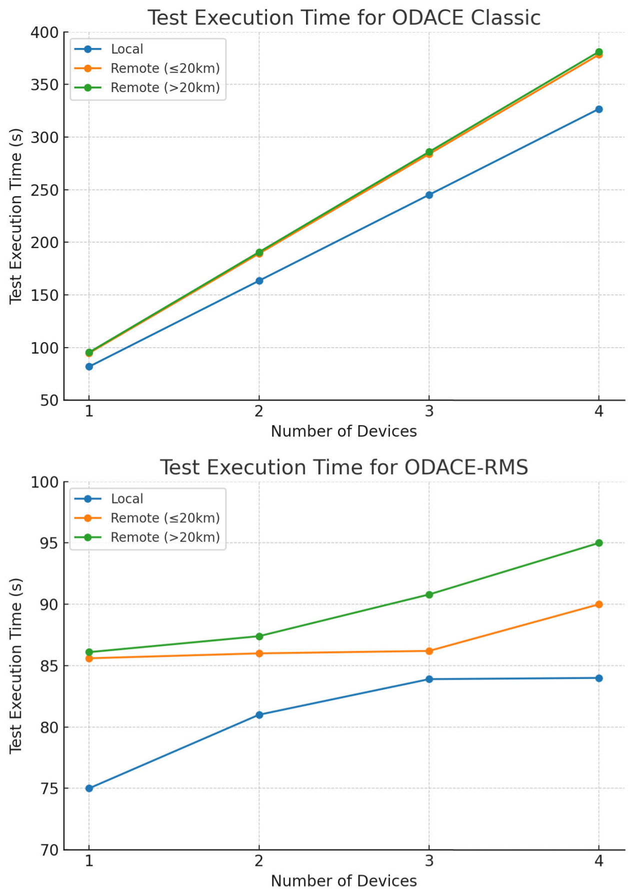


FIGURE 16. Test execution time for ODACE vs. ODACE-RMS across different scenarios.

C. QUALITATIVE RESULTS OF ODACE-RMS

In this part of the evaluation, we focus on its impact on usage. We assess its effectiveness in terms of time efficiency, resource usage, and overall utility based on engineer feedback and the survey outcomes.

- **Time Efficiency:** We assess test completion times in specific quarters using ODACE-RMS compared to the original ODACE and Manual testing. The results show reductions due to simultaneous multi-device testing. Figure 17 illustrates the time savings ODACE-RMS (2 devices) achieved compared to manual and ODACE for the same number of tests.

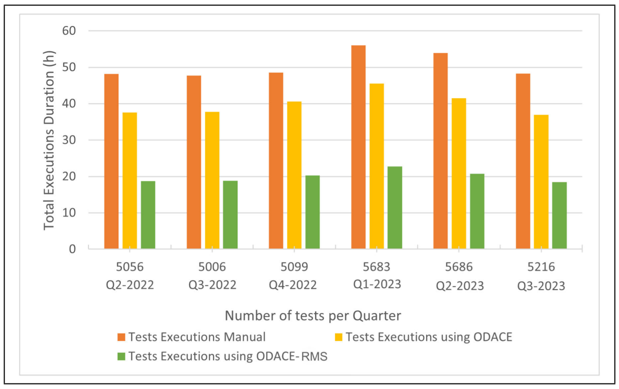


FIGURE 17. ODACE-RMS improvement time.

- **Resources Efficiency:** We execute a series of tests to compare resource usage across three major browsers (Chrome, Edge, and Firefox). We evaluate that for single-device and multi-device usage. All other applications and browser tabs were closed to isolate the resource consumption of the ODACE-RMS system. A custom Java script collected system resource data every 20 seconds, measuring CPU (system and process) and memory usage (reflects overall system consumption). This evaluation was necessary to suggest the best browser to use. All three browsers show high CPU usage during the first 50 seconds of startup. After that, usage drops and stabilizes below 30% for system CPU and below 5% for process CPU. ODACE-RMS scales efficiently with minimal overhead, with memory usage increasing by only around 1.6% for Edge, 3.4% for Firefox, and 2.5% for Chrome, which consumes slightly more system memory than other browsers.
  Figure 19 shows the results for single-device usage, where all browsers have good performance with little distinction for Edge, as it is more stable than the other browsers. Firefox is the most lightweight in terms of memory and CPU use, while Chrome is more resource-intensive initially. All browsers stabilize quickly.
  However, in Figure 20, we can see that Edge proved better results in the multi-device scenario, consuming the least memory and maintaining a balanced CPU utilization compared to Chrome and Firefox. Also, when we add a device, CPU load is negligible after initialization, regardless of DUT count. Memory usage grows by 1.6% to 8.8% with the additional DUT.
- **Efficacy:** ODACE-RMS allows users to complete certifications without physical device access. We expect the multi-device feature to boost the number of tests conducted daily or weekly, enhancing performance efficiency.
  To evaluate the efficacy of ODACE-RMS, we surveyed certification engineers at Vidéotron. Participation was voluntary and anonymous, with responses collected from all engineers in the department.
  The questionnaire comprised 10 questions. Results indicated that engineers rated the difficulty of lab-only tests (without remote capability) at an average of 3.5 out of 5, which is around 70%. This difficulty rating explains the unanimous support (100%) for hybrid certification testing (local and remote). Additionally, 67% of engineers reported a commute time of 30-60 minutes each way, while 33% exceeded one hour to arrive at the office, highlighting potential time savings weekly with remote testing. Furthermore, 90% agreed that remote capabilities reduce the complexity of cables/devices.
  These results indicate strong preference and flexibility gains with ODACE-RMS's remote features.

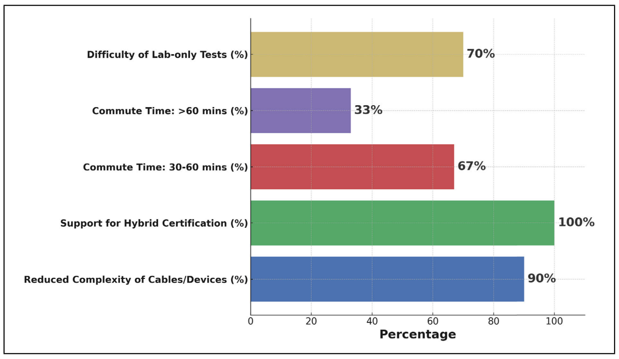


FIGURE 18. Survey results: Preferences for remote certification with ODACE-RMS.

VII. DISCUSSION AND FUTURE DIRECTIONS

A. ODACE-RMS AUTOMATION DISCUSSION

While the ODACE-RMS platform has shown clear benefits by reducing user effort, enhancing time efficiency, and decreasing execution times to save certification time, certain challenges remain unresolved. For example, achieving consistency across various device vendors, models, operating systems, and software versions is particularly challenging, making it difficult to maintain a universal code base. As a result, supporting new Android releases oftentimes requires manual updates to the ODACE-RMS code. Another challenge is that specific tests, such as emergency call functionality and audio quality verification, cannot be fully automated and still require human intervention. Additionally, the increased complexity introduced by new features makes platform maintenance more challenging. Although remote testing offers flexibility, some tests still should be done in person, and device issues during remote sessions may necessitate a technician's presence in the lab. To manage this,

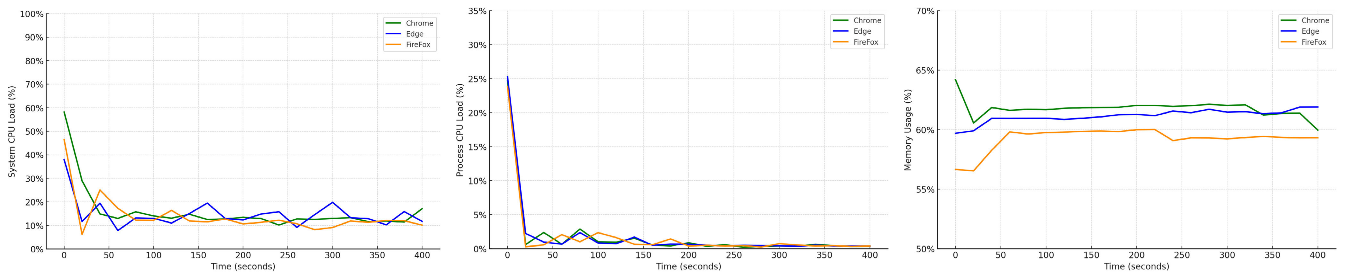


FIGURE 19. Resource usage across different browsers - single DUT.

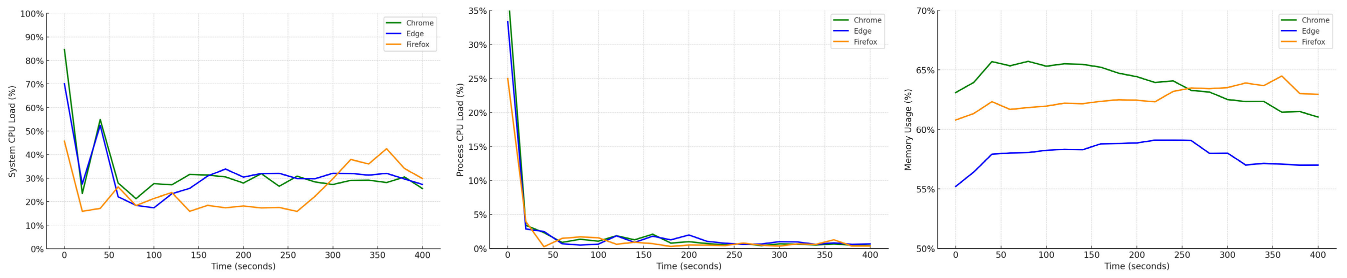


FIGURE 20. Resource usage across different browsers - multi DUT (Two devices).

the Vidéotron lab rotates one CE daily to handle such cases. Remote testing via USB-over-IP may also cause some delays from 7.5% to 11% in some cases when we compare the local execution time with remote execution time(distention of more than 20km) due to the network latency, which could impact the overall testing time.

The testing environment and framework infrastructure can be costly and challenging. Maintaining automation tools and keeping them up to date can also be complex.

B. FUTURE DIRECTIONS

Some potential options for improving research directions to enhance ODACE-RMS's capabilities further include integrating the operator's network tracing application into ODACE-RMS, which would expand testing capabilities. Furthermore, using AI-powered image recognition presents a valuable enhancement for managing OS software version updates, effectively addressing challenges related to device compatibility and ensuring consistency across various vendors and operating system versions. In addition, developing support to include iOS devices would significantly extend the platform's applicability across multiple testing environments.

VIII. CONCLUSION

In this paper, we introduce ODACE-RMS, a platform designed to automate part of the manual mobile phone certification process. ODACE-RMS further advances the certification testing process for Android devices by introducing simultaneous multi-device testing and remote testing capabilities. These new features simplify the testing workflow and reduce certification process time. ODACE-RMS aligns with the remote work trend by allowing tests to be performed without physical device interaction. Our evaluation shows that these features increase user satisfaction and enhance the certification process, making ODACE-RMS a timely and flexible solution.

One of ODACE-RMS's significant strengths is its user-friendly design, which makes it accessible to new employees and interns who may lack technical expertise. By reducing the time needed for routine tasks, ODACE-RMS improves the learning experience of interns during their brief training periods while allowing companies to reallocate resources toward more critical tasks. This approach improves overall efficiency. The results showed that ODACE-RMS significantly reduces time and effort. Additionally, ODACE addresses several challenges highlighted in related studies, including reducing execution time, providing a friendly interface, and supporting comprehensive history and log files for retrospective analysis and evaluation.

ACKNOWLEDGMENT

The authors would like to acknowledge the contributions of the certification laboratory team and interns at Vidéotron, a Canadian telecom operator. Their collaboration and support were instrumental in the development of ODACE, encompassing both hardware and software components.

An earlier version of this paper was presented in part at the 7th IEEE Workshop on NEXt Level of Test Automation (NEXTA 2024) and in part at the IEEE International Conference on Software Testing, Verification, and Validation (ICST 2024), Toronto, ON, Canada, in May 2024 [DOI: 10.1109/ICSTW60967.2024.00060].

REFERENCES

[1] S. Mojahed, R. Drouin, and L. Sboui, "ODACE: An appium-based testing automation platform for Android mobile devices certification," in *Proc. IEEE Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, May 2024, pp. 301–308.

[2] A. Bertolino, "Software testing research: Achievements, challenges, dreams," in *Proc. Future Softw. Eng. (FOSE)*, May 2007, pp. 85–103.

[3] N. G. Berihun, C. Dongmo, and J. A. Van der Poll, "The applicability of automated testing frameworks for mobile application testing: A systematic literature review," *Computers*, vol. 12, no. 5, p. 97, May 2023.

[4] S. Godboley, D. Dalei, R. Sadam, and D. P. Mohapatra, "Agile GUI testing by computing novel mobile app coverage using appium tool," in *Proc. 38th ACM/SIGAPP Symp. Appl. Comput.*, Mar. 2023, pp. 1026–1029.

[5] We Are Social Hootsuite. (2024). *DIGITAL 2024: October Global Statshot*. Accessed: Oct. 30, 2024. [Online]. Available: https://datareportal.com/reports/digital-2024-october-global-statshot

[6] S. W. G. AbuSalim, R. Ibrahim, and J. A. Wahab, "Comparative analysis of software testing techniques for mobile applications," *J. Phys., Conf. Ser.*, vol. 1793, no. 1, Feb. 2021, Art. no. 012036.

[7] A. Li and C. Li, "Research on the automated testing framework for Android applications," in *Proc. Int. Conf. Comput. Eng. Netw.*, Jan. 2022, pp. 1056–1064.

[8] D. Vajak, R. Grbic, M. Vranješ, and D. Stefanović, "Environment for automated functional testing of mobile applications," in *Proc. Int. Conf. Smart Syst. Technol. (SST)*, Oct. 2018, pp. 125–130.

[9] A. M. Sinaga, Y. Pratama, and F. O. Siburian, "Comparison of graphical user interface testing tools," *J. Comput. Netw., Archit. High Perform. Comput.*, vol. 3, no. 2, pp. 123–134, Jul. 2021.

[10] L. C. Chaves, F. C. M. Oliveira, L. A. Tiago, and R. G. V. Castro, "Robert: An automated tool to perform mobile application test," in *Proc. 10th Int. Conf. Comput. Technol. Appl.*, May 2024, pp. 33–36.

[11] K. R. Halani, Kavita, and R. Saxena, "Critical analysis of manual versus automation testing," in *Proc. Int. Conf. Comput. Perform. Eval. (ComPE)*, Dec. 2021, pp. 132–135.

[12] K. Thant and H. Tin, "The impact of manual and automatic testing on software testing efficiency and effectiveness," *Indian J. Sci. Res.*, vol. 3, no. 3, pp. 88–93, 2023.

[13] H. Kim, B. Choi, and S. Yoon, "Performance testing based on test-driven development for mobile applications," in *Proc. 3rd Int. Conf. Ubiquitous Inf. Manage. Commun.*, Feb. 2009, pp. 612–617.

[14] M. A. Salam, S. Taha, and M. G. Hamed, "Advanced framework for automated testing of mobile applications," in *Proc. 4th Novel Intell. Lead. Emerg. Sci. Conf. (NILES)*, Oct. 2022, pp. 233–238.

[15] N. Verma, *Mobile Test Automation With Appium*. Birmingham, U.K.: Packt Publishing Ltd, 2017.

[16] S. Singh, R. Gadgil, and A. Chudgor, "Automated testing of mobile applications using scripting technique: A study on appium," *Int. J. Current Eng. Technol. (IJCET)*, vol. 4, no. 5, pp. 3627–3630, Jan. 2014.

[17] A. Motwani, A. Agrawal, N. Singh, and A. Shrivastava, "Novel framework for browser compatibility testing of a Web application using selenium," *Int. J. Comput. Sci. Inf. Technol.*, vol. 6, no. 6, pp. 5159–5162, 2015.

[18] H. Minh Tran, T. Duc Ninh, T. Duc Tran, V. Van Ngo, and L. Duc Nguyen, "Automation testing with appium framework in IP multimedia subsystem," in *Proc. 14th Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2023, pp. 579–582.

[19] J. Cui, W. Chen, Q. Wan, Z. Gan, and Z. Ning, "Design and analysis of a mobile automation testing framework: Evidence and AI enhancement from Chinese Internet technological companies: A case study," *Frontiers Bus., Econ. Manage.*, vol. 14, no. 2, pp. 163–170, Apr. 2024.

[20] A. A. Alotaibi and R. J. Qureshi, "Novel framework for automation testing of mobile applications using appium," *Int. J. Modern Educ. Comput. Sci.*, vol. 9, no. 2, pp. 34–40, Feb. 2017.

[21] A. K. Rao, S. Prasad, and E. Rao, "Quality benefit analysis of software automation test protocol," *Int. J. Mod. Eng. Res.*, vol. 2, no. 5, pp. 3930–3933, 2012.

[22] A. Jain and S. Sharma, "An efficient keyword driven test automation framework for Web applications," *Int. J. Eng. Sci. Adv. Technol*, vol. 2, no. 3, pp. 600–604, 2012.

[23] N. Lukić, S. Talebipour, and N. Medvidović, "Remote control of iOS devices via accessibility features," in *Proc. ACM Workshop Forming Ecosystem Around Softw. Transformation*, Nov. 2020, pp. 35–40.

[24] *Automated Testing Framework (ATF)*, SEGRON, Bratislava, Slovakia, 2024.

[25] J. Yoon, R. Feldt, and S. Yoo, "Autonomous large language model agents enabling intent-driven mobile GUI testing," 2023, *arXiv:2311.08649*.

[26] S. Yu, C. Fang, M. Du, Z. Ding, Z. Chen, and Z. Su, "Practical, automated scenario-based mobile app testing," *IEEE Trans. Softw. Eng.*, vol. 50, no. 7, pp. 1949–1966, Jul. 2024.

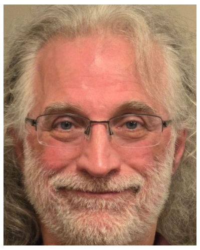
[27] Z. Liu, C. Chen, J. Wang, M. Chen, B. Wu, X. Che, D. Wang, and Q. Wang, "Make LLM a testing expert: Bringing human-like interaction to mobile GUI testing via functionality-aware decisions," in *Proc. IEEE/ACM 46th Int. Conf. Softw. Eng.*, Apr. 2024, pp. 1–13.

[28] J. Li and H. Cao, "Design and implementation of API automation testing system for mobile hybrid mode based on appium technology," in *Proc. 7th Int. Conf. Electron. Inf. Technol. Comput. Eng.*, Oct. 2023, pp. 1478–1484.

[29] G. da Silva and R. de Souza Santos, "Comparing mobile testing tools using documentary analysis," in *Proc. ACM/IEEE Int. Symp. Empirical Softw. Eng. Meas. (ESEM)*, Oct. 2023, pp. 1–6.

[30] *Digi Remote Manager User Guide*, Digi Int., Hopkins, MN, USA, 2023.

SUNDOS MOJAHED (Student Member, IEEE) received the Diplôme degree in computer science from Palestine Polytechnic University (PPU), Hebron, Palestine, in 2017. She is currently pursuing the master's degree in software engineering with École de Technologie Supérieure (ÉTS), Montreal, Canada. From 2017 to 2019, she was a Laboratory Supervisor with the College of IT and Computer Engineering and as a Lecturer in applied professions with PPU. She completed an internship with Vidéotron, Montreal, from 2021 to 2024. Her current research interests include software engineering, mobile and web development, and software automation.



RÉJEAN DROUIN received the Baccalauréat en Génie Électrique (B.Sc.A.es) degree from Université Laval, Canada, in 1993. He practiced as a Telecommunication Engineer with a major cellular telephony equipment provider and operator in Q&A and technical support, providing emergency resolution, network planning, commissioning, optimization, and tool development. Since 2021, he has pioneered and led the development of Automated Device Certification (ODACE) with Vidéotron, Montreal, Canada. His current research interest includes networking and automation.

LOKMAN SBOUI (Senior Member, IEEE) received the Diplôme d'Ingénieur degree (Hons.) from École Polytechnique de Tunisie (EPT), La Marsa, Tunisia, in 2011, the M.Sc. and Ph.D. degrees from the King Abdullah University of Science and Technology (KAUST), in 2013 and 2017, respectively. He was a Certification Analyst for mobile devices with Vidéotron and a Canadian Telecom Operator located in Québec City, from 2020 to 2021. He is currently an Associate Professor with École de Technologie Supérieure (ÉTS), Montreal, Canada, in the Department of System Engineering. His current research interests include energy-efficient wireless communications, LEO satellites for the IoT, automation, urban air mobility (UAM), edge computer vision, cognitive radio, industrial digital twins, and VoIP protocols. He has been with IEEE WIRELESS COMMUNICATIONS LETTERS Editorial Board, since 2021.