Towards Autonomous Mobile Testing with LLM-Guided Non-Invasive Interaction

Davi Freitas
Federal University of Pernambuco
Recife, Brazil
dsf3@cin.ufpe.br

Breno Miranda
Federal University of Pernambuco
Recife, Brazil
bafm@cin.ufpe.br

Juliano Iyoda
Federal University of Pernambuco
Recife, Brazil
jmi@cin.ufpe.br

Abstract

This work explores the use of Large Language Models (LLMs) as autonomous agents for mobile task execution through a non-invasive visual perception-based approach. To achieve this, we employ the RFNIT robotic framework, which is responsible for identifying graphical interface elements and executing interactions on the device, while the LLMs act as task-oriented decision-making agents.

Unlike traditional approaches based on predefined scripts and internal application instrumentation, our proposal enables high-level tasks to be dynamically translated into actions over real graphical interfaces. To investigate the potential of this approach, we defined a benchmark composed of 30 mobile tasks with different complexity levels and evaluated multiple LLMs in real interaction scenarios.

Initial results indicate that LLMs show promising capabilities for goal-oriented mobile automation, being able to perform navigation, contextual adaptation, and iterative interaction with graphical interfaces. At the same time, the experiments reveal challenges related to context maintenance, failure recovery, and long-term planning. We believe this approach opens new possibilities for exploratory testing, non-invasive automation, and intelligent agents for mobile application interaction.

Keywords

Large Language Models, Mobile Testing, GUI Testing, Non-Invasive Testing, Smartphone Automation, Robotic Arms

1 Introduction

Mobile applications have become the primary interface for accessing digital services, making their reliability and quality increasingly important. However, testing mobile applications remains a challenging task due to device heterogeneity, dynamic graphical interfaces, and the complexity of user interactions [4]. Traditional mobile testing frameworks, such as Appium, Espresso, and UIAutomator, rely on application instrumentation or direct access to the interface hierarchy, limiting their applicability in black-box scenarios involving third-party applications, obfuscated software, or devices without privileged permissions [14].

Even when such frameworks can be employed, test execution is typically driven by predefined scripts that manipulate interface components through programmatic events or system commands. Although effective in controlled environments, these approaches often overlook important aspects of real user interaction, including visual perception, contextual adaptation, partially visible elements, overlays, and custom graphical components [13]. Consequently, maintaining automated test scripts across evolving interfaces remains costly and error-prone.

Recent advances in Large Language Models (LLMs) have enabled the emergence of intelligent GUI agents capable of interpreting interface states, reasoning about high-level goals, and dynamically selecting interaction sequences [5]. Rather than executing predefined scripts, these agents can adapt their decisions according to the current state of the interface, creating new opportunities for autonomous and goal-oriented mobile testing.

In this paper, we propose a non-invasive architecture for autonomous mobile testing that integrates Large Language Models with the Robotic Framework for Non-Invasive Testing (RFNIT). The framework performs visual perception and physical interaction on real smartphones, while the LLM acts as a decision-making agent responsible for translating high-level testing goals into sequences of GUI interactions. From a software testing perspective, RFNIT acts as the test driver, and task completion criteria define the test oracle, enabling autonomous execution without application instrumentation or manually specified interaction scripts.

The proposed approach follows an iterative perception–reasoning–action cycle. At each iteration, the LLM receives (i) the task description, (ii) a structured representation of the current interface state extracted by RFNIT [2], and (iii) the set of supported actions. Based on this context, the model selects the next interaction required to progress toward the testing objective. The execution continues until the task is successfully completed or a predefined stopping criterion is reached.

To assess the feasibility of the proposed architecture, we conducted an empirical evaluation using a benchmark composed of 30 representative mobile tasks. From a software testing perspective, each task represents a high-level test objective, while its success criterion defines the expected outcome (test oracle) used to determine whether the task has been successfully completed. Three state-of-the-art LLMs were integrated into the same execution pipeline to investigate how different decision-making models behave under identical testing conditions.

The main contributions of this work are as follows:

- A non-invasive architecture for autonomous mobile testing based on the integration of RFNIT and Large Language Models;
- A ReAct-based prompting strategy that enables iterative decision making through visual perception of real smartphone interfaces;
- A benchmark composed of 30 representative mobile testing tasks with different complexity levels;
- An empirical evaluation of the proposed architecture using multiple state-of-the-art LLMs;
- An analysis of the current strengths and limitations of LLM-guided non-invasive mobile testing.

Overall, this work investigates the feasibility of employing LLMs as autonomous decision-making agents in robotic mobile testing. We believe that integrating language models with non-invasive robotic interaction opens new opportunities for exploratory testing, goal-oriented automation, and intelligent testing of real mobile applications.

2 Background and Related Work

Automated testing of mobile applications remains a challenging problem in Software Engineering due to device heterogeneity, dynamic graphical interfaces, and the complexity of user interactions on real smartphones. Traditional automation frameworks, such as Appium, Espresso, and UIAutomator, rely on application instrumentation or access to the interface hierarchy, making them invasive and often fragile in the presence of interface changes.

To overcome these limitations, recent research has explored robotic and non-invasive approaches capable of interacting directly with physical devices. Systems such as RoScript [8], ROBOTEST [12], and the robotic-arm-based approach proposed by Mao *et al.* [7] demonstrated that robotic interaction increases test realism by enabling direct interaction with real smartphone interfaces. However, these approaches still rely on manually written scripts or prerecorded interaction sequences, limiting their ability to autonomously adapt to dynamic interfaces and unexpected execution states.

More recently, Large Language Models (LLMs) have emerged as a promising alternative for increasing the autonomy of GUI agents. AutoDroid [10] employs LLMs to execute Android tasks through natural language reasoning over textual interface representations, while Wang *et al.* [9] and DroidBot-GPT [11] investigate conversational interaction and LLM-guided exploration of mobile interfaces. Although these approaches improve autonomous decision making, they rely on software-based interaction mechanisms rather than non-invasive robotic execution on real devices.

Beyond mobile automation, recent studies have explored LLMs in Software Engineering tasks, including *test smell* detection and compilation error identification in configurable systems [1, 6]. These studies demonstrate that different LLMs exhibit distinct reasoning and decision-making capabilities, motivating further investigation of their behavior in software testing scenarios.

Overall, existing studies have focused either on robotic interaction or on LLM-guided GUI agents. To the best of our knowledge, there is still little empirical evidence on how different LLMs behave when integrated into the same non-invasive robotic testing pipeline operating on real smartphones.

Unlike previous work, our goal is not to propose another GUI agent, but to investigate the feasibility of integrating different state-of-the-art LLMs into a non-invasive robotic mobile testing architecture. To this end, we evaluate multiple state-of-the-art LLMs under the same execution pipeline, prompt strategy, and standardized benchmark, providing initial evidence on the strengths and current limitations of LLM-guided robotic mobile testing.

3 Approach

Our approach integrates the RFNIT framework with Large Language Models (LLMs) to enable autonomous task execution on smartphones through a non-invasive interaction strategy. High-level tasks are dynamically translated into actions executed directly on the device interface.

3.1 Overview

The proposed architecture combines RFNIT and Large Language Models (LLMs) to enable autonomous task execution on real smartphones through non-invasive visual interaction. Figure 1 presents an overview of the execution pipeline adopted in this work.

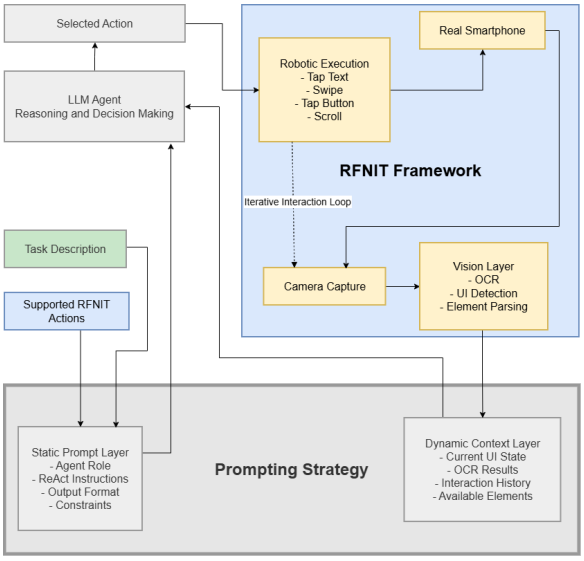


Figure 1: Overview of the RFNIT + LLM execution pipeline.

RFNIT is responsible for capturing the current interface state, identifying graphical elements, and physically executing interactions on the device. The framework operates exclusively on the visual interface of the smartphone, without requiring internal access to the application or the interface component hierarchy.

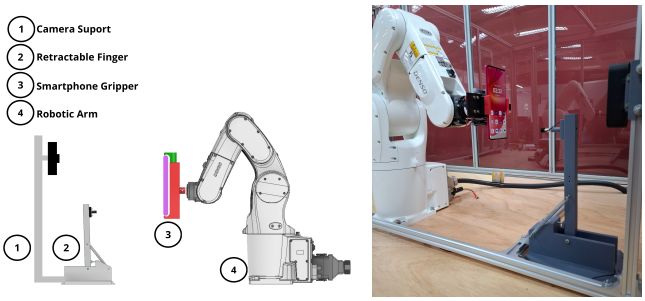
During execution, RFNIT continuously captures screen images using an external camera and applies computer vision techniques to identify interactive components such as buttons, text fields, icons, and menus. This information is converted into a structured textual representation and sent to the LLM together with the task description and the set of actions supported by the system.

Based on this context, the LLM acts as the decision-making mechanism, iteratively selecting the next action required to advance the task. The interactions are then physically executed on the device through an external actuator responsible for reproducing actions similar to those performed by human users, including taps, gestures, and scrolling operations.

Unlike traditional approaches based on predefined scripts, the interaction flow is generated dynamically during execution, enabling adaptation to interface changes, contextual navigation, and partial recovery from incorrect decisions.

Hardware Infrastructure. The infrastructure employed by RFNIT consists of a real smartphone, an external camera responsible for continuously capturing the device interface, and a physical actuator

used to execute interactions on the device. Figure 2a presents a conceptual representation of the system, while Figure 2b shows the physical setup used during the experiments.



(a) RFNIT representation. **(b) RFNIT hardware.**

Figure 2: (a) conceptual representation and (b) physical hardware used for interaction with smartphones.

3.2 Execution Cycle

The execution process follows an iterative cycle that combines perception, reasoning, and interaction. Initially, RFNIT captures the current state of the smartphone screen and extracts the visible interface elements. The detected elements are then transformed into a structured textual representation containing relevant interface information, such as element types, labels, positions, and available interaction options.

Next, the task description, the current interface state, and the list of supported actions are sent to the LLM through a prompt. Based on this context, the model selects the next action to be executed. The selected action is then physically performed by RFNIT on the device.

After each interaction, RFNIT captures the updated interface state and verifies whether the task oracle has been satisfied. Otherwise, the LLM receives the new state and replans the next action. The execution terminates when the task is completed or when predefined stopping criteria, such as timeout, maximum number of interaction steps, or repetitive action loops, are reached.

The cyclic nature of the approach allows the LLM to continuously adapt its decisions according to changes in the interface state. As a result, the system can react to dynamic application behaviors and partially recover from incorrect decisions by analyzing the updated visual context after each interaction.

4 Methodology

Our experimental study investigates the ability of Large Language Models (LLMs) to autonomously execute tasks on real smartphones using a non-invasive robotic approach. To achieve this, we integrated different LLMs with the RFNIT framework, which is responsible for visual interface perception and physical execution of interactions on the device.

4.1 Goal and Research Questions

We structured this study following the Goal-Question-Metric (GQM) approach. The goal of this work is to evaluate the effectiveness of LLMs in orchestrating non-invasive interactions on smartphones from the perspective of task-oriented mobile testing.

To achieve this goal, we defined the following research questions:

- **RQ1:** To what extent can LLMs autonomously complete tasks on smartphones?
- **RQ2:** How do different LLMs behave under the proposed non-invasive mobile testing architecture?
- **RQ3:** What are the most common failures observed during task execution?

4.2 Experimental Protocol

The experiments were conducted using the RFNIT framework on a real Android smartphone. The evaluation covered native Android applications, including Settings, Camera, Gallery, Clock, Contacts, Browser, Maps, Messaging, and Email. RFNIT was responsible for visual perception and physical interaction with the device, while the evaluated LLMs generated the interaction decisions.

For each benchmark task, the corresponding task description and the current interface representation generated by RFNIT were provided to the LLM. The model iteratively selected actions until the task oracle was satisfied or one of the predefined stopping criteria was reached, including timeout, maximum interaction step.

Each task was initially executed once for every evaluated LLM. Whenever an execution resulted in a *Partial* or *Fail* outcome, the task was repeated to verify whether the observed behavior was reproducible. For each execution, we recorded the final outcome (*Success*, *Partial*, or *Fail*) together with qualitative observations regarding recurrent failure patterns.

The benchmark comprises 30 representative mobile tasks covering device configuration, application navigation, communication, authentication, and e-commerce scenarios.

4.3 Evaluated LLMs

To evaluate how different LLMs perform in reasoning, contextual interpretation, and interaction with mobile graphical interfaces, we selected three widely used models: GPT-4o, Gemini, and Grok. These models were chosen because they represent state-of-the-art proprietary architectures with advanced capabilities for textual understanding, iterative reasoning, and multimodal analysis.

GPT-4o was selected due to its consistent performance in tasks related to Software Engineering, planning, and natural language interaction. Gemini was included because of its ability to handle large context windows and its focus on multimodal and contextual reasoning tasks. Grok was considered as a recent alternative designed for real-time interaction and agent-oriented reasoning.

All LLMs were evaluated using the same task set, experimental environment, and prompt template, enabling a more consistent comparison across models. During the experiments, we analyzed task completion together with qualitative observations regarding recurrent failure patterns, contextual adaptation, and recovery from incorrect decisions.

4.4 Prompt Strategy

To guide the decision-making process of the LLMs, we employed a prompting strategy based on the ReAct (Reasoning and Acting) paradigm, which combines explicit reasoning with iterative action

execution. This approach allows the LLM to continuously analyze the current interface state, reason about task progress, and dynamically select the next interaction to be executed.

The use of ReAct is particularly relevant in mobile interaction scenarios, where the interface may change after each executed action. By explicitly exposing the reasoning process before action selection, the model tends to produce more consistent decisions, reduce invalid interactions, and better adapt to unexpected changes during navigation.

For example, during the execution of the task "Enable Wi-Fi", the LLM may produce the following reasoning cycle:

> **THOUGHT:** The settings application is already open and the Wi-Fi option is visible but disabled. I should enable the Wi-Fi toggle.
> **ACTION:** CLICK_THE_TEXT
> **PARAMETER:** Wi-Fi

The prompting strategy adopted in this work was organized into two complementary layers: a static instruction layer and a dynamic contextual interaction layer.

The static layer defines the overall behavior of the agent throughout task execution. This layer includes information related to the LLM role, execution constraints, mandatory response format, the set of actions supported by RFNIT, and reasoning guidelines based on the ReAct paradigm. Since these instructions remain unchanged during execution, they act as a persistent mechanism for controlling agent behavior.

In contrast, the dynamic layer is iteratively updated at each execution cycle. This layer contains information related to the current interface state, including the task description, detected screen elements, recent interaction history, and available actions. Based on this context, the LLM performs contextual reasoning and selects the next action to be executed by the robotic framework.

This separation between global instructions and dynamic context reduces prompt redundancy, improves the organization of information provided to the model, and facilitates reuse of the prompting strategy across different LLMs and tasks.

4.5 Task Complexity Levels

To evaluate different reasoning, navigation, and contextual adaptation capabilities of the LLMs, we organized the benchmark into three complexity levels: low, medium, and high complexity.

Low-complexity tasks involve direct interactions and short navigation flows, typically requiring only a few actions to complete the task. These tasks include simple device configuration activities and basic interface interactions.

Medium-complexity tasks require multiple navigation steps, contextual identification of interface elements, and textual information input. In this group, the LLMs must maintain execution consistency and adapt their decisions according to the current interface state.

High-complexity tasks involve long interaction flows, authentication, multi-step forms, and dynamic context changes. These tasks frequently require sequential planning, error recovery, and adaptation to different application states. Table 1 presents the benchmark tasks.

5 Results

The experiments conducted in this study aimed to investigate the ability of different Large Language Models (LLMs) to autonomously execute mobile tasks using the RFNIT robotic framework. To this end, we evaluated the models using a benchmark composed of 30 tasks with different complexity levels, considering aspects related to task completion rate, contextual adaptation, and failure patterns during execution.

The following subsections present the results organized according to the research questions defined in this study.

RQ1 – Autonomous Task Completion

A task was considered successful when its predefined oracle (Table 1) was satisfied. Otherwise, the execution was classified as Partial or Fail according to the achieved progress. Table 2 summarizes the overall task completion results for each evaluated LLM.

Overall, the evaluated models successfully completed a considerable portion of the benchmark tasks, particularly those with low and medium complexity. GPT-4o achieved the highest success rate, fully completing 50% of the tasks, while Grok obtained comparable performance. Gemini produced the largest number of partially completed executions, suggesting difficulties in concluding specific stages of the interaction flow.

We also observed that many partial executions were associated with tasks involving text input, authentication, or contextual interpretation of interface elements. These results indicate that current LLMs are already capable of performing meaningful goal-oriented interactions on real smartphones, although important limitations still remain in long and context-dependent workflows.

RQ2 – LLM Behavior Across Task Complexity

To answer RQ2, we analyzed how different LLMs behaved under the proposed RFNIT-based testing architecture across tasks with different complexity levels. Overall, all models successfully executed most low-complexity tasks, particularly those involving simple navigation, application launching, and short interaction flows.

More substantial differences emerged in medium- and high-complexity tasks, where execution required longer interaction sequences, contextual adaptation, and maintenance of execution consistency over multiple steps. In these scenarios, we observed a higher occurrence of failures related to context maintenance, form filling, text input, and incorrect selection of interface elements.

GPT-4o achieved the highest overall success rate, fully completing 15 out of the 30 evaluated tasks. The model also demonstrated better adaptation during navigation and greater ability to recover from incorrect decisions. Gemini achieved intermediate performance but produced a larger number of partially completed executions, indicating difficulties in concluding specific stages of the interaction workflow. Grok showed performance comparable to GPT-4o in low-complexity tasks but demonstrated greater sensitivity to dynamic interface changes and a higher occurrence of failures during long interaction flows.

Tasks classified as high complexity represented the most challenging scenarios for all evaluated models. Interactions involving authentication, multi-step forms, contextual navigation, and textual

Table 1: Benchmark tasks used in the experimental evaluation.

ID	Task	Category	Complexity	Oracle (Success Criterion)
1	Open settings application	Navigation	Low	Settings application opened
2	Enable Wi-Fi	Configuration	Low	Wi-Fi enabled
3	Disable Bluetooth	Configuration	Low	Bluetooth disabled
4	Adjust screen brightness	Configuration	Low	Brightness level changed
5	Open camera application	Application	Low	Camera application opened
6	Capture a photo	Application	Low	Photo captured
7	Open gallery application	Application	Low	Gallery application opened
8	Delete a photo	Application	Medium	Photo removed from gallery
9	Open clock application	Application	Low	Clock application opened
10	Create an alarm	Configuration	Medium	New alarm created
11	Open browser	Navigation	Low	Browser opened
12	Search a term on Google	Web Navigation	Medium	Search results displayed
13	Open contacts application	Application	Low	Contact list displayed
14	Create a new contact	Form	Medium	Contact saved
15	Edit an existing contact	Form	Medium	Information updated
16	Send an SMS message	Communication	Medium	Message sent
17	Open email application	Application	Low	Email application opened
18	Compose an email	Communication	Medium	Email content filled
19	Search for a location on maps	Navigation	Medium	Location displayed
20	Navigate between application tabs	Navigation	Medium	Correct tab selected
21	Log into an application	Authentication	High	User authenticated
22	Log out from the application	Authentication	Medium	Session terminated
23	Add a product to the cart	E-commerce	High	Product added to cart
24	Remove a product from the cart	E-commerce	Medium	Product removed from cart
25	Change an internal application setting	Configuration	Medium	Setting modified
26	Fill a multi-step form	Form	High	Form submitted
27	Search for a product in an application	E-commerce	Medium	Search results displayed
28	Apply a filter to a product list	E-commerce	High	Product list correctly filtered
29	Start a call in a messaging application	Communication	High	Call initiated
30	Complete a simulated purchase	E-commerce	High	Workflow completed until confirmation

Table 2: Overall task completion results.

LLM	Success	Partial	Fail
GPT-4o	15 (50.0%)	11 (36.7%)	4 (13.3%)
Gemini	13 (43.3%)	13 (43.3%)	4 (13.3%)
Grok	14 (46.7%)	11 (36.7%)	5 (16.7%)

input frequently resulted in failures or partial executions. In particular, tasks requiring typing operations consistently produced high error rates across all models, suggesting that textual interaction remains one of the main limitations of current LLM-based mobile agents.

Table 3 presents the results organized by complexity level.

Table 3: Task completion by complexity level.

Complexity	GPT-4o	Gemini	Grok
Low	9/11	8/11	8/11
Medium	5/13	3/13	4/13
High	1/6	2/6	2/6

The results indicate that increasing task complexity directly impacts the decision-making capabilities of the LLMs, particularly in scenarios requiring long-term planning, context maintenance, and dynamic adaptation to interface changes. Overall, GPT-4o demonstrated the most stable behavior across different complexity levels, while all evaluated models still exhibited significant limitations in long and context-dependent workflows.

RQ3 – Failure Analysis

To answer RQ3, we analyzed the main failure patterns observed during task execution by the LLMs. Although the models were able to autonomously perform several tasks, important limitations remain regarding contextual perception, long-term planning, and dynamic interface adaptation.

A recurrent failure pattern was related to text input in interface fields. Tasks involving authentication, form filling, or message composition frequently resulted in partial executions or failures. In several scenarios, the models struggled to identify the active input field, maintain typing consistency, or adapt to interface changes after the virtual keyboard was displayed.

We also observed failures related to incorrect selection of graphical elements. In some cases, the LLMs selected visually similar components that were semantically incompatible with the task being executed, particularly in interfaces containing multiple nearby buttons or menus with similar options.

Another recurring pattern was the occurrence of navigation loops. In complex tasks involving multiple steps or transitions across applications, the models frequently repeated action sequences without making progress. For instance, during the "Create New Contact" task, some LLMs correctly navigated to the contact creation form but repeatedly lost focus on the active text field after the virtual keyboard was displayed, producing redundant interactions without completing the form. This behavior indicates limitations related to context maintenance during iterative execution.

Additionally, some tasks exhibited failures associated with understanding the global context of the application. In scenarios involving multiple applications, such as purchase workflows or messaging calls, the LLMs occasionally struggled to determine which application should be used to complete the task.

High-complexity tasks concentrated most of the observed failures, indicating that increasing execution steps and contextual adaptation requirements directly impacts the decision-making capabilities of the LLMs.

Overall, the results suggest that the main limitations of current LLMs in non-invasive mobile testing are related to text input, long-term context maintenance, and adaptation to dynamic graphical interfaces. Table 4 summarizes the main categories of failures observed during the experiments.

Table 4: Main failure categories observed during execution.

Failure Type	Occurrences
Text input errors	12
Navigation loops	9
Incorrect element selection	7
Context loss	6
Wrong application selection	4

5.1 Discussion

The obtained results indicate that LLMs have strong potential to operate as autonomous agents in non-invasive mobile interaction scenarios. Even without internal access to the application or the interface hierarchy, the models were able to execute different categories of tasks using only visual perception and natural language-based reasoning.

At the same time, the experiments revealed important limitations related to context maintenance, error recovery, and long-term planning. As task complexity increases, model performance tends to decrease, particularly in workflows involving multiple steps, text input, and dynamic adaptation to interface changes.

The results suggest that LLM-based approaches can become promising alternatives for exploratory automation, high-level goal-oriented testing, and intelligent interaction with mobile applications. Nevertheless, this work still represents an initial step toward fully autonomous mobile agents, and several aspects can be improved to increase robustness and generalization capabilities.

One important limitation observed during the experiments is the strong dependence on textual interface representations generated through OCR. Consequently, perception errors or incomplete descriptions of graphical components may directly affect the reasoning process of the LLMs.

Future improvements may include more advanced prompting strategies, contextual memory mechanisms, and richer semantic representations of non-textual interface elements, such as icons and visual indicators. Overall, the results indicate that integrating LLMs, computer vision, and robotic interaction represents a promising direction for autonomous and non-invasive mobile testing.

6 Threats to Validity

Our evaluation is based on a benchmark of 30 mobile tasks executed on a limited set of applications, which may not fully represent real-world mobile interaction scenarios. Partial and failed executions were repeated to confirm the observed behavior; however, a larger number of independent runs would provide a more robust assessment of execution variability.

The results may also be influenced by the adopted ReAct-based prompt, while perception errors caused by OCR-based interface descriptions may affect the LLM decision-making process. Finally, future updates to proprietary LLMs may impact the reproducibility of the reported results.

7 Conclusion

This work investigated the use of Large Language Models (LLMs) as autonomous agents for mobile task execution on real smartphones using a non-invasive robotic approach based on the RFNIT framework. The results showed that different LLMs are capable of executing low- and medium-complexity tasks using only visual perception and natural language-based reasoning.

At the same time, the experiments revealed important limitations related to context maintenance, text input, and long-term planning. These findings indicate that the integration of LLMs, computer vision, and robotics represents a promising direction for task-oriented mobile testing.

As future work, we intend to investigate more advanced prompting strategies, contextual memory mechanisms, and more robust semantic descriptions of non-textual graphical interface elements.

Artifact Availability

The replication package, including the task benchmark, prompts, and experimental scripts, is available online [3].

Acknowledgments

This work was partially supported by the Brazilian National Council for Scientific and Technological Development (CNPq) under grants 315765/2023-2, 408651/2023-7, and 408817/2024-0. Additional support was provided by INES.IA (National Institute of Science and Technology for Software Engineering Based on and for Artificial Intelligence) and by CRIAR (Center for Innovation in Robotics and Responsible Artificial Intelligence), a laboratory hosted at CIn-UFPE, coordinated by Softex, and funded through the MCTI Futuro initiative of the Brazilian Ministry of Science, Technology and Innovation (MCTI), with resources provided under Brazilian Law No. 8,248, of October 23, 1991.

References

[1] Lucas Albuquerque, Rohit Gheyi, and Márcio Ribeiro. 2024. Evaluating the Capability of LLMs in Identifying Compilation Errors in Configurable Systems. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 574–580. doi:10.5753/sbes.2024.3560

[2] Davi Freitas, Breno Miranda, and Juliano Iyoda. 2024. RFNIT: Robotic Framework for Non-invasive Testing. In *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. doi:10.1145/3663529.3663871

[3] Davi Freitas, Breno Miranda, and Juliano Manabu Iyoda. 2026. *Replication Package for "Towards Autonomous Mobile Testing with LLM-Guided Non-Invasive Interaction"*. doi:10.5281/zenodo.21442908

[4] Pingfan Kong, Li Li, Jun Gao, Kui Liu, Tegawendé F. Bissyandé, and Jacques Klein. 2019. Automated Testing of Android Apps: A Systematic Literature Review. *IEEE Transactions on Reliability* 68, 1 (2019), 45–66. doi:10.1109/TR.2018.2865733

[5] Guangyi Liu, Pengxiang Zhao, Yaozhen Liang, Liang Liu, Yaxuan Guo, Han Xiao, Weifeng Lin, Yuxiang Chai, Yue Han, Shuai Ren, Hao Wang, Xiaoyu Liang, WenHao Wang, Tianze Wu, Zhengxi Lu, Siheng Chen, LiLinghao, Hao Wang, Guanjing Xiong, Yong Liu, and Hongsheng Li. 2025. LLM-Powered GUI Agents in Phone Automation: Surveying Progress and Prospects. arXiv:2504.19838 [cs.HC] https://arxiv.org/abs/2504.19838

[6] Keila Lucas, Rohit Gheyi, Elvys Soares, Márcio Ribeiro, and Ivan Machado. 2024. Evaluating Large Language Models in Detecting Test Smells. In *Anais do XXXVIII Simpósio Brasileiro de Engenharia de Software* (Curitiba/PR). SBC, Porto Alegre, RS, Brasil, 672–678. doi:10.5753/sbes.2024.3642

[7] Ke Mao, Mark Harman, and Yue Jia. 2017. Robotic Testing of Mobile Apps for Truly Black-Box Automation. *IEEE Software* 34, 2 (2017), 11–16. doi:10.1109/MS.2017.49

[8] Ju Qian, Guizhou Lv, Yiming Jin, Zhengyu Shang, Shuoyan Yan, Yan Wang, and Lin Chen. 2025. Robotic Visual GUI Testing for Truly Non-Intrusive Test Automation of Touch Screen Applications. *IEEE Transactions on Software Engineering* 51, 4 (2025), 1205–1231. doi:10.1109/TSE.2025.3544441

[9] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction with Mobile UI using Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 432, 17 pages. doi:10.1145/3544548.3580895

[10] Hao Wen, Yuanchun Li, Guohong Liu, Shanhui Zhao, Tao Yu, Toby Jia-Jun Li, Shiqi Jiang, Yunhao Liu, Yaqin Zhang, and Yunxin Liu. 2024. AutoDroid: LLM-powered Task Automation in Android. In *Proceedings of the 30th Annual International Conference on Mobile Computing and Networking* (Washington D.C., DC, USA) *(ACM MobiCom '24)*. Association for Computing Machinery, New York, NY, USA, 543–557. doi:10.1145/3636534.3649379

[11] Hao Wen, Hongming Wang, Jiaxuan Liu, and Yuanchun Li. 2024. DroidBot-GPT: GPT-powered UI Automation for Android. arXiv:2304.07061 [cs.SE] https://arxiv.org/abs/2304.07061

[12] Shengcheng Yu, Chunrong Fang, Mingzhe Du, Yuchen Ling, Zhenyu Chen, and Zhendong Su. 2024. Practical Non-Intrusive GUI Exploration Testing with Visual-based Robotic Arms. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering* (Lisbon, Portugal) *(ICSE '24)*. Association for Computing Machinery, New York, NY, USA, Article 130, 13 pages. doi:10.1145/3597503.3639161

[13] Shengcheng Yu, Chunrong Fang, Ziyuan Tuo, Quanjun Zhang, Chunyang Chen, Zhenyu Chen, and Zhendong Su. 2025. Vision-Based Mobile App GUI Testing: A Survey. *ACM Comput. Surv.* 58, 6, Article 142 (Dec. 2025), 46 pages. doi:10.1145/3773027

[14] Yury Zhauniarovich, Anton Philippov, Olga Gadyatskaya, Bruno Crispo, and Fabio Massacci. 2015. Towards Black Box Testing of Android Apps. In *2015 10th International Conference on Availability, Reliability and Security*. 501–510. doi:10.1109/ARES.2015.70