ScoutDroid: A Tool For Mobile Augmented Testing with Live Feedback

ScoutDroid: A Tool For Mobile Augmented Testing with Live Feedback*

Lorenzo Laudadio
lorenzo.laudadio@polito.it
Politecnico di Torino
Turin, Italy

Riccardo Coppola
riccardo.coppola@polito.it
Politecnico di Torino
Turin, Italy

Marco Torchiano
marco.torchiano@polito.it
Politecnico di Torino
Turin, Italy

Stevan Tomic
Blekinge Institute of Technology
Karlskrona, Sweden
stevan.tomic@bth.se

Abstract

In the evolving landscape of mobile applications, effective and efficient testing methods are crucial for ensuring high-quality user experiences. This paper introduces a novel end-to-end mobile testing technique designed to enhance exploratory testing by incorporating gamification strategies. We developed a plugin that integrates these innovative techniques, aiming to make the testing process more engaging and effective. With the use of a live feedback system, the plugin drives testers to thoroughly explore the application, leading to the discovery of more defects and improved software quality. Preliminary evaluation suggests that this approach could not only increase tester engagement but also improve the detection rate of critical issues. This research highlights the potential of merging exploratory testing with gamification, setting the stage for more dynamic and productive mobile testing methodologies.

CCS Concepts

• **Software and its engineering → Software testing and debugging**.

Keywords

Software Testing, GUI Testing, Gamification, Software Engineering, Mobile Applications

ACM Reference Format:
Lorenzo Laudadio, Riccardo Coppola, Marco Torchiano, and Stevan Tomic. 2024. ScoutDroid: A Tool For Mobile Augmented Testing with Live Feedback. In *Proceedings of the 3rd ACM International Workshop on Gamification in Software Development, Verification, and Validation (Gamify '24), September 17, 2024, Vienna, Austria.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3678869.3685688

*This study was carried out within the "EndGame - Improving End-to-End Testing of Web and Mobile Apps through Gamification" project (2022PCCMLF) – funded by European Union – Next Generation EU within the PRIN 2022 program (D.D.104 - 02/02/2022 Ministero dell'Università e della Ricerca). This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

Gamify '24, September 17, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1113-8/24/09
https://doi.org/10.1145/3678869.3685688

1 Introduction

End-to-end (E2E) testing plays a crucial role in validating the functionality and performance of software applications by simulating real-world user scenarios from start to finish, ensuring that all integrated components work seamlessly together. This kind of software testing is particularly suitable for those applications which provide a Graphical User Interface (GUI), such as web and mobile applications. Very often, E2E testing requires the tester to manually run test cases from start to end several times, which can be time-consuming. Capture and Replay (CR) testing is a technique which aims to overcome this limitation [23]. CR consists of recording use case scenarios while the tester is interacting with the Application Under Test (AUT) and transforming those scenarios into test cases. The test cases can later be executed several times without the need for major human intervention.

In this paper, we present a novel tool for E2E mobile application testing which aims to produce effective test cases and to increase the tester involvement in the exploratory phase, through the use of gamification. Gamification consists of applying game design mechanics in activities of different nature [5]. The effectiveness of gamification in software engineering has been widely witnessed by several authors [8, 10, 18, 19]. The Octalysis Framework [3] is a comprehensive evaluation framework developed by Yu-kai Chou which can be used to assess the presence of gamification elements in a system. It defines 8 Core Drives (CDs) which push the human motivation towards the accomplishment of activities. We used the Octalysis Framework as inspiration to identify the CDs which could be integrated into our software.

In particular, we focused on CD3: Empowerment of Creativity & Feedback, enforced when users are engaged in creative processes and receive feedback. In our tool, this CD is implemented with a live feedback system, that allows the testers to interact with the AUT and experience the consequences of their actions.

2 Background and Related Work

In this section we present background information about Exploratory (and Augmented) testing and the state of practice of Gamification.

2.1 Exploratory and Augmented Testing

Exploratory Testing (ET) is a manual testing technique which is widely adopted in the assessment of software applications. It consists of generating test cases while the tester interacts with the AUT.

The effectiveness and efficiency of ET compared to the standard Scripted Testing (ST) was proven in an experiment by Afzal et al. [1]. One example in which ET is very effective is regression testing. In the context of software testing, a regression refers to a situation where a previously functioning feature or functionality stops working correctly after changes are made to the software. ET is heavily based on testers' intuition and experience, and emphasizes their personal freedom and responsibility. At the same time it is an highly repetitive and monotonous activity.

Capture and Replay (CR) tools are often used to reduce the repetitiveness of ET. CR tools are capable of recording the actions performed by the tester during the exploration of the AUT so that they can be re-executed later. Several CR tools have been proposed in the context of web application testing, such as TESTAR [24] and WebRR [15]. Within the Android ecosystem, CR techniques have been used widely for testing purposes [7, 11, 14, 17, 20].

The main issue with CR is that the quality of the test cases produced during the exploratory phase is obviously influenced by human factors, like stress, fatigue and boredom. Most studies in the field of Android E2E testing only focused on the implementation of CR techniques, while none of them considered the impact that the aforementioned human factors have on the quality of the produced test cases.

Augmented Testing (AT) was proposed in the context of E2E web application testing as a technique to reduce the impact of these factors [16]. It consists of introducing a visual layer over the AUT's GUI, which can display information that could help testers in making decisions. Such information could include, for example, suggestions on the possible actions to be performed, comments, statistics, etc. A web application AT tool called Scout was developed based on the Selenium driver [12]. Scout was also evaluated in an industrial workshop which led to promising results. In our work, we used Scout as a starting point to implement an Android AT tool.

2.2 Gamification

Gamification is the application of game design elements to other activities. There is a general trend to bring gamification into the world of software engineering as well. For example, gamification has been successfully applied to project management, coding challenges and competitions, continuous integration/continuous deployment, etc. The application of gamification to software testing is an emerging field aimed at enhancing tester engagement, motivation, and performance [9, 21, 22]. Gamification can transform the often monotonous task of testing into a more enjoyable and rewarding activity by incorporating game-like elements such as points, badges, leaderboards, and rewards. In Scout, this is realized through live augmented feedback of actions that can be performed against the GUI. More specifically, when the user hovers the mouse over the widgets it is suggested possible actions to take, and after the actions have been taken the widgets are surrounded by color-coded visual overlays. Such overlays can significantly increase the number of interactions and coverage on a web application [4].

To the best of our knowledge, the existing CR mobile testing tools do not implement any gamification mechanics. Our goal is to overcome this shortcoming by providing a CR plugin for mobile testing which implements a live feedback system, associated with CD3 from the Octalysis Framework.

3 ScoutDroid for Scout

The system which we describe here does not consist of a single application, but rather includes different pieces of software which are interconnected. In particular, we can identify three main parts:

(1) The Android Emulator.
(2) The Appium Server, which instruments the Android Emulator through the UiAutomator2 driver.
(3) The ScoutDroid plugin, which runs inside Scout.

Scout is a software product which provides an environment for augmented testing. The core idea behind augmented testing is to improve End-to-End (E2E) testing by helping the tester with visual artifacts superimposed on the GUI of the Application Under Test (AUT).

Scout adopts a plugin-based architecture: the Scout software itself does not implement specific features, but rather acts as a host for plugins which can be developed and maintained independently. Scout uses a binary format which is used to store the testing session information, such as the interactions performed by the tester, the assertions, the session duration, etc. ScoutDroid, which we describe here, can be hosted in Scout, and implements augmented testing for mobile applications. The ScoutDroid plugin is based upon Appium, an open-source automation tool which provides an HTTP server (Appium Server) that waits for commands and reflects them on the Android Emulator. The Appium Server needs a driver to operate on the emulator. The open-source UiAutomator2 driver was chosen for that purpose, for its extensive documentation and support [2].

The architectural diagram is shown in Figure 1. The ScoutDroid plugin, hosted in Scout, interacts with the Appium Server, which in turn instruments the Android Emulator thanks to the UiAutomator2 driver. This modular approach also allows the different processes to be executed on different machines, communicating through sockets. When the connection is established, ScoutDroid requests and parses the XML page source of the application at regular intervals, to see whether it has changed. When the page changes, ScoutDroid takes a screenshot of the emulator page and displays it on the Scout's frame.

On the other side, user-generated events, like key presses and mouse motions, are translated by ScoutDroid into commands which are sent to the Appium Server, and are reflected in the Android Emulator.

Figure 2 displays a screenshot of a running session of ScoutDroid. As can be seen, the Scout + ScoutDroid GUI mirrors the content of the Android Emulator screen. The Appium Server, which runs in the background, acts as a bridge between ScoutDroid and the Android Emulator, directing the plugin requests to the emulator, and sending back the emulator responses.

ScoutDroid can be used to easily detect regressions which could be introduced when new versions of an application are released, since it allows to make assertions on the values of the application components at some point in time, and later check that those assertions are still true. This process is made easier by the augmented layer, which associates a color with each assertion value (e.g. if the

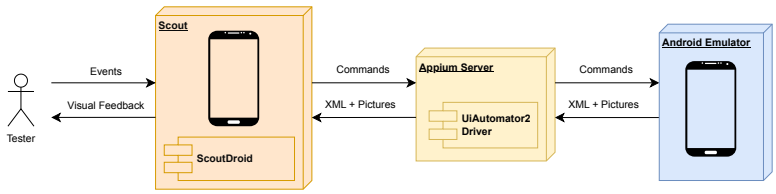

Figure 1: Architectural diagram

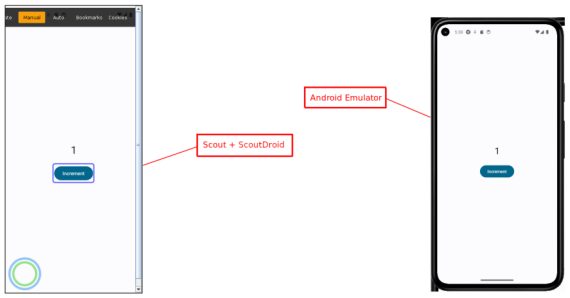

Figure 2: Screenshot of Scout and Android Emulator running

assertion value is false, the component on which the assertion is made gets surrounded by a yellow rectangle).

Figure 3 shows a sample testing session performed with ScoutDroid on a simple mobile application, in which we identify a regression. The application consists of a button and a text widget. When the button is pressed, the text widget should display the string "You clicked!". During the first execution (1) the tester clicks on the button; the string "You clicked!" appears; the tester adds an assertion on the text widget by clicking on it.[1] A green rectangle which surrounds the text widget appears, by highlighting that the assertion is true. The second execution (2) targets a different version of the application, in which we introduced a bug on purpose; at the beginning, the button is surrounded by a blue rectangle, which indicates the tester that in a previous execution the button was clicked; when the tester presses the button, a different text is displayed ("You clacked!"); a yellow rectangle is displayed around the text widget, implying that the assertion is failed; by hovering on the text widget with the mouse, the tester can see the reason why the assertion is failing. This is a classical example of regression testing: we have a working application on which assertions are generated; the working application gets modified and a bug is introduced; the regression testing is successful because it allows the tester to detect the bug (i.e. the regression).

4 Application and Research Directions

ScoutDroid represents a pivotal advancement in the domain of mobile application testing. It provides a tool designed to easily investigate the possible regression of the AUT by making assertions on GUI components. This facilitates the rapid execution of repetitive test cases, which is essential for maintaining an efficient testing workflow, especially where scripted solutions are not viable options. This enhances the overall agility of the development process, which is a key factor in an industrial environment.

Although the addition of an augmented HUD for the tester does not inherently qualify as gamification, the presence of this layer can be considered as a proof of concept of the application of visual elements to the otherwise traditional manual ET activity that can be carried out with the tool. Also, the provision of live visual feedback is - according to the Octalysis framework - one of the pillars of gamification, and with this prototype tool we demonstrate the feasibility of providing such an element for Android ET.

Research directions include the possibility of involving other gamification mechanics in mobile testing, like leaderboards, missions, rewards, etc. A promising key area to be investigated is the introduction of mutation testing into the software, which allows to assess the bug-detection effectiveness of test cases. While similar solutions exist for web application testing ([13]), no studies were made in the mobile application field.

One of the main limitations of our approach lies in the inability to inject live code into the AUT. Differently from the web application context, where a JavaScript execution environment is available, in Android it is not possible to modify the executed code at runtime. This is due to the internal working of the Android RunTime (ART), which requires compiled bytecode [6]. This is an important aspect to take into consideration when implementing, for instance, a mutation testing system. Another primary drawback of the plugin is the limited speed of interaction. While the usage of the Appium API facilitates the development of the Appium Plugin, it constitutes a bottleneck in the live streaming pipeline, due to the creation of an additional socket: the images flow from the emulator to the Appium Server, and from the Appium Server to the client (i.e. ScoutDroid). Some mitigations have been adopted: for instance, whenever a change happens in the Android Emulator GUI, ScoutDroid only requires two screenshots to synchronize with the emulator. This means that we can avoid continuously fetching images from the Android Emulator, thus speeding up the whole streaming process. Our preliminary evaluation of the plugin found a synchronization speed of just over one second at worst, which in some cases may have a negative impact on the usability of the system. However, we are actively exploring several optimization strategies, to further reduce the synchronization time. As an example, future versions of the plugin may fetch the video stream directly from the Android Emulator, without relying on the Appium Server.

[1] By default, the generated assertion checks for the text content to be equal to the current text value, but different types of assertion can be specified.

Figure 3: Sample testing session with ScoutDroid

5 Conclusion and Future Work

In this paper, we presented a novel tool for the end-to-end testing of mobile applications, named ScoutDroid. This tool is capable of helping testers assess mobile applications with an innovative approach that is based on Exploratory Testing and Augmented Testing. It also introduces gamification mechanics to drive the tester's motivation through the testing process.

Further work is currently in progress involving the introduction of other gamification mechanics into the tool. Additionally, we plan to employ mutation testing techniques to assess the quality of test cases generated by our tool.

In conclusion, our software application represents a significant advancement in the field of mobile testing, providing a solution to improve the quality and reliability of mobile applications. By addressing the critical challenges in end-to-end testing, we contribute to the development of more efficient, effective, and user-centric mobile technologies.

References

[1] Wasif Afzal, Ahmad Nauman Ghazi, Juha Itkonen, Richard Torkar, Anneliese Andrews, and Khurram Bhatti. 2015. An experiment on the effectiveness and efficiency of exploratory testing. *Empirical software engineering : an international journal* 20, 3 (2015), 844–878.

[2] Appium. 2024. *Appium - UiAutomator2 Driver.* https://github.com/appium/appium-uiautomator2-driver

[3] Yu-Kai Chou. 2023. The Octalysis Framework for Gamification & Behavioral Design. https://yukaichou.com/gamification-examples/octalysis-complete-gamification-framework/

[4] Riccardo Coppola, Tommaso Fulcini, Luca Ardito, Marco Torchiano, and Emil Alègroth. 2024. On Effectiveness and Efficiency of Gamified Exploratory GUI Testing. *IEEE Transactions on Software Engineering* 50, 2 (2024), 322–337. https://doi.org/10.1109/TSE.2023.3348036

[5] Sebastian Deterding. 2012. Gamification: designing for motivation. *Interactions* 19, 4 (jul 2012), 14–17. https://doi.org/10.1145/2212877.2212883

[6] Android Developers. 2024. *Android runtime and Dalvik.* https://source.android.com/docs/core/runtime

[7] Mattia Fazzini, Eduardo Noronha De A. Freitas, Shauvik Roy Choudhary, and Alessandro Orso. 2017. Barista: A Technique for Recording, Encoding, and Running Platform Independent Android Tests. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST).* 149–160. https://doi.org/10.1109/ICST.2017.21

[8] Gordon Fraser. 2017. Gamification of Software Testing. In *2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST).* 2–7. https://doi.org/10.1109/AST.2017.20

[9] Tommaso Fulcini and Luca Ardito. 2022. Gamified Exploratory GUI Testing of Web Applications: a Preliminary Evaluation. In *2022 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).* 215–222. https://doi.org/10.1109/ICSTW55395.2022.00045

[10] Félix García, Oscar Pedreira, Mario Piattini, Ana Cerdeira-Pena, and Miguel Penabad. 2017. A framework for gamification in software engineering. *Journal of Systems and Software* 132 (2017), 21–40. https://doi.org/10.1016/j.jss.2017.06.021

[11] Jiaqi Guo, Shuyue Li, Jian-Guang Lou, Zijiang Yang, and Ting Liu. 2019. Sara: self-replay augmented record and replay for Android in industrial cases. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA 2019).* Association for Computing Machinery, New York, NY, USA, 90–100. https://doi.org/10.1145/3293882.3330557

[12] Selenium HQ. 2024. *Selenium WebDriver.* https://www.selenium.dev/documentation/webdriver/

[13] Maurizio Leotta, Davide Paparella, and Filippo Ricca. 2024. Mutta: a novel tool for E2E web mutation testing. *Software quality journal* 32, 1 (2024), 5–26.

[14] Chien Hung Liu, Chien Yu Lu, Shan Jen Cheng, Koan Yuh Chang, Yung Chia Hsiao, and Weng Ming Chu. 2014. Capture-Replay Testing for Android Applications. In *2014 International Symposium on Computer, Consumer and Control.* 1129–1132. https://doi.org/10.1109/IS3C.2014.293

[15] Zhenyue Long, Guoquan Wu, Xiaojiang Chen, Wei Chen, and Jun Wei. 2020. WebRR: self-replay enhanced robust record/replay for web application testing. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (Virtual Event, USA) *(ESEC/FSE 2020).* Association for Computing Machinery, New York, NY, USA, 1498–1508. https://doi.org/10.1145/3368089.3417069

[16] Michel Nass, Emil Alégroth, and Robert Feldt. 2019. Augmented Testing: Industry Feedback To Shape a New Testing Technology. In *2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).* 176–183. https://doi.org/10.1109/ICSTW.2019.00048

[17] Stas Negara, Naeem Esfahani, and Raymond Buse. 2019. Practical Android Test Recording with Espresso Test Recorder. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP).* 193–202. https://doi.org/10.1109/ICSE-SEIP.2019.00029

[18] Savas Ozturk. 2022. Gamification of exploratory testing process. In *Proceedings of the 1st International Workshop on Gamification of Software Development, Verification, and Validation* (Singapore, Singapore) *(Gamify 2022).* Association for Computing Machinery, New York, NY, USA, 14–17. https://doi.org/10.1145/3548771.3561411

[19] Wei Ren. 2023. Gamification in Test-Driven Development Practice. In *Proceedings of the 2nd International Workshop on Gamification in Software Development, Verification, and Validation* (San Francisco, CA, USA) *(Gamify 2023).* Association for Computing Machinery, New York, NY, USA, 38–46. https://doi.org/10.1145/3617553.3617889

[20] Onur Sahin, Assel Aliyeva, Hariharan Mathavan, Ayse Coskun, and Manuel Egele. 2019. RANDR: Record and Replay for Android Applications via Targeted Runtime Instrumentation. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE).* 128–138. https://doi.org/10.1109/ASE.2019.00022 ISSN: 2643-1572.

[21] Philipp Straubinger and Gordon Fraser. 2022. Gamekins: gamifying software testing in jenkins. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings* (Pittsburgh, Pennsylvania) *(ICSE '22).* Association for Computing Machinery, New York, NY, USA, 85–89. https://doi.org/10.1145/3510454.3516862

[22] Philipp Straubinger and Gordon Fraser. 2024. Improving Testing Behavior by Gamifying IntelliJ. 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 49, 13 pages. https://doi.org/10.1145/3597503.3623339

[23] Mario Linares Vasquez, Kevin Moran, and Denys Poshyvanyk. 2018. Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing. *arXiv (Cornell University)* (2018).

[24] Tanja E. J. Vos, Pekka Aho, Fernando Pastor Ricos, Olivia Rodriguez-Valdes, and Ad Mulders. 2021. testar – scriptless testing through graphical user interface. *Software Testing, Verification and Reliability* 31, 3 (2021), e1771. https://doi.org/10.1002/stvr.1771 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/stvr.1771 e1771 stvr.1771.