Comprehensive Analysis of Software Testing for Intrinsically Challenging Systems

A Technical Paper submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Jassiel Mendoza

Spring, 2024

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Advisor

Rosanne Vrugtman, Department of Computer Science

Comprehensive Analysis of Software Testing for Intrinsically Challenging Systems

CS4991 Capstone Report, 2024

Jassiel Mendoza
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
jjm5mh@virginia.edu

ABSTRACT

A piece of software can never be fully tested, which already fogs the line of what is considered "enough testing." This problem is exacerbated when considering software with intrinsic properties that directly oppose the feasibility of testing software across varying contexts, such as mobile applications. An industry-wide adoption of set standards for these intrinsically challenging systems is needed to address the issue, especially as this type of software is increasingly being applied to security- and safety-critical applications. To thoroughly evaluate existing testing methods, I conducted a meta-analysis to gain a comprehensive understanding of the shortcomings of current techniques and the necessary improvements needed to create effective and adaptable approaches that bring us closer to establishing an industry standard. The findings indicate that the testing techniques that are currently in use lack the necessary functionality and scalability that could make them useful for companies. To overcome these deficiencies, current research is focused on leveraging new technologies to develop cost-effective and practical solutions. As software continues to evolve, testing must also evolve to keep pace with the increasingly sophisticated software being developed.

1. INTRODUCTION

Every product made available to the public undergoes testing to ensure it adheres to technical standards and functions within specified parameters. Software differs in that, unlike physical products which are tested before reaching consumers and are often accompanied by warranties, software products are subject to continuous testing through development and even after release, known as maintenance, and can never be entirely free of flaws.

This maintenance phase holds immense importance with a significant portion of a software product or service budget allocated to post-release maintenance. This is due to the cost of maintaining software increasing exponentially the further into development it is, particularly if critical defects are identified in the later stages. Given the resource-intensive demands of testing and maintaining software, coupled with the fact that software cannot and will not ever be "perfect," meaning it exhibits no flaws, vulnerabilities, or errors, presents software companies with the dilemma of determining what is "enough testing." This question is of crucial importance in industries that employ software for security- and safety-critical applications, such as finance and aviation, as they must adhere to strict regulatory guidelines.

In recent years, not only has software itself become increasingly sophisticated, but it continues to be applied in complex contexts as well. The rise of mobile devices brought with it the need for mobile applications which presents new challenges in testing for software developers. The intrinsic properties of mobile applications add another layer of

complexity to the already difficult dilemma of software testing. For instance, the inherent non-deterministic nature of these applications makes it difficult to define expected behaviors and outputs during testing, inhibiting the design of deterministic test cases necessary for accurately evaluating the application's behavior; the primary objective of software testing to begin with.

The biggest problem, however, is the issue of device fragmentation. The wide variety of mobile devices, each with their own screen sizes, resolutions, operating systems, and hardware capabilities, presents a daunting testing landscape. Adapting and modeling a single mobile application to work correctly on all combinations of these in an ideal, controlled environment is already a difficult task, but software must also be tested against the real-world conditions it will face, including fluctuating network conditions like poor connectivity or transitions between different cell tower networks, or unpredictable user interruptions like incoming calls, notifications, and background processes.

All of these consume device resources, affect the performance of subsequent tasks, and are handled differently by each device and its operating system. The extensive input space presented makes it impractical, if not nearly impossible, to model and account for every possible scenario, both technically and financially, thereby limiting the ability to identify inputs that reveal critical failures. This highlights the importance of finding cost-effective solutions to the software testing problem, as software will only become more intricate.

2. RELATED WORKS

An article by Linares-Vasquez, et al. (2017), was the primary catalyst that influenced this research. The article outlined the problem mobile application testing faces and shed light on the issue of device fragmentation, particularly within the Android ecosystem. Furthermore, it delves into the challenge of non-deterministic outcomes impacting assertions, a phenomenon known as test flakiness, as well as the issue of varying runtime device states influenced by available resources during test execution. The article underscores the necessity for a mobile-specific fault model to proficiently test mobile applications. The authors advocate for a continuous, evolutionary, and large-scale testing methodology integrating crowd-based and cloud-based testing, along with real user feedback, to address the complexities inherent in mobile application testing. Although the paper is dated, it highlights the slow progress made in the field and reinforces the need for development.

Additionally, a frequently cited article by Choudhary, et al. (2015), provided foundational information on the topic of automating mobile application software testing, specifically on the Android platform. It provides a comprehensive analysis of the merits and limitations of the state of the art, assessing the effectiveness of each tool and technique according to four metrics: ease of use, ability to work on multiple platforms, code coverage, and ability to detect faults. Comparing and using these tools and techniques in combination, properly leveraging their strengths, provided insight into potential ways to make each more effective and efficient overall.

3. META STUDY

There are various, ever-evolving industry standards in place today with the primary objective of outlining an agreed set of standards for software testing, the most widely accepted being the ISO/IEC/IEEE International Standard. In sensitive industries such as finance and aviation, there also exist governing bodies whose sole purpose is to regulate certification and ensure software

involving their respective industries meets defined standards. However, most mobile applications do not fall under this umbrella and do not adhere to these strict software testing standards, as they are not mandatory unless legal or contractual obligations are involved.

This is concerning as mobile applications continue to be integrated into daily life affecting users' privacy and security of personal information. According to a publication by NowSecure (n.d.), recognized experts in mobile security, "95% of ~6500 popular mobile applications fail to meet the world's most recognized [minimum] industry standard for mobile app security." The areas with the highest rates of failure include the exposure and theft of critical user information, and improper coding practices such as failure to properly validate information and the use of insecure libraries. Although companies are not bound to follow a set of defined guidelines, the high rate of failure to even meet the minimum industry standard is unacceptable and highlights the importance of providing a cost-effective and flexible way of testing mobile applications.

The current state-of-the-art software testing script automation currently remains a flawed process, as emphasis remains on significant manual testing. Whether due to the financial barrier of testing on a large and agile scale or because of the existing inefficiencies of testing automation that fail in "accurately capturing and reproducing test scripts across diverse devices" (Yu, et al., 2023). This defeats the primary objective of automating the testing process as "extensive modifications are often necessary to ensure the test scripts accurately reflect the interactions and behaviors" (Yu, et al., 2023).

Consequently, the industry continues to rely on manual testing as mobile applications are dynamic pieces of software, that constantly change due to feature additions and updates. It is not only impractical but expensive to automate a testing script that ultimately requires manual modification to port to differing device configurations. Additionally, this process is not scalable as by the time it is completed, the cycle restarts as a new update or feature is developed and released, trapping software test designers in a perpetual uphill battle and forcing companies to utilize other tools, each with their limitations.

By far the next most popular technique used in industry today is graphical user interface (GUI) tests. This approach simulates and captures user interactions on a target app, compiling it into a test script that can be reproduced and tested at scale. It leverages the fact that mobile applications typically share "identical or nearly identical functionalities and consistent appearances across platforms" (Ji, et al., 2023).

This similarity across devices ideally facilitates the migration of GUI tests. However, this testing methodology abstracts the fact that mobile applications across varying platforms are implemented in different languages, causing the underlying codebase to differ significantly. Therefore, while porting GUI tests across platforms may prove to be successful in testing the graphical interface and front-end functionality across platforms, "the recorded test scripts, which capture interactions specific to one app's architecture, may not align with the design patterns and structure" of an application on a different platform (Yu, et al., 2023). The difference in the underlying codebase can cause the same GUI elements and sequence of events to behave and operate differently. This limitation again calls for the need for manual intervention as the testing scripts need to be adjusted to reproduce the user interactions. This once again hinders the scalability of such an approach as it suffers from the same pitfalls as the use of automated test scripts.

It is common practice in software engineering to distribute a process among a large network to ease the load of having to perform the same task independently. In software testing, this is seen in the approach of crowd-sourced testing. This largely eliminates the device fragmentation problem as mobile applications are tested with "in the wild" conditions across varying devices. This said, though, the current state-of-the-art crowd-sourcing techniques all share the limitation of involving "human intervention, which is time-consuming and error-prone" (Sun, et al., 2023).

This is problematic as crowd-sourcing testing, as the name suggests, is outsourced, meaning that "crowd-workers tend to submit low-quality bug reports" as they have limited to no knowledge of the underlying functionality of the mobile application (Sun, et al., 2023). Additionally, software testing is a resource-intensive process already, but crowd-sourcing testing puts an additional financial burden on the company. To larger companies, the additional cost might be manageable and ultimately a worthy investment, but smaller companies with limited resources would be unable to leverage this type of testing at a scale that would make it advantageous.

The current state-of-the-art techniques and approaches to software testing all exhibit unwanted limitations. It is no surprise that most mobile applications fail to pass even the minimum standards, as there is simply no efficient and effective way for companies to comprehensibly test their application across platforms and devices. Fortunately, software testing is a hot topic in research, and developers are constantly looking for new approaches and techniques to mitigate the issue.

4. ANTICIPATED SOLUTIONS

With extensive, ongoing research on new approaches and ways to leverage technologies, the limitations outlined previously, may be improved. Yu, et al. (2023), define the promising potential of using large language models (LLM) in the test script generation and migration. Their research demonstrates an advantage of utilizing LLMs as they "can delve deeper into the intricacies of the app's functionality…generating meaningful input strings that facilitate comprehensive testing" (Yu, et al., 2023). Additionally, research conducted by Ji, et al. (2023), focuses on improving "vision-based widget mapping" to facilitate GUI test migration and although their research did not outright solve the GUI tests limitation, it presents a step in the right direction as the first empirical study on the topic. Sun, et. al. (2023) also investigated ways of improving crowd-sourcing techniques in hopes of providing a "lightweight" solution that eliminated the need for human intervention in the testing process. As each approach continues to be refined, new avenues are also being explored. With new developments and by leveraging each approach's strengths, a cost-effective and practical solution will exist.

5. CONCLUSION

Testing is a vital stage that assures the integrity of any product, and software must not be treated any differently. Not only does it provide reassurance to developers that the functionality of their software is correct, but it also protects customers' privacy and security by ensuring that the software does not contain harmful bugs or defects. Testing software is especially difficult due to the impossibility of comprehensive testing, which is only compounded by the complexities of mobile. Therefore, there is a strong need for improvements in the way testing in general is conducted, much less for mobile application testing.

The first step towards this goal is understanding the current state of the art's

weaknesses and strengths to then knowing and improving on the shortcomings, while simultaneously reinforcing the capabilities. Ultimately, working towards incrementally providing effective and efficient tools to developers and companies; is crucial for the objective of making software more reliable.

6. FUTURE WORK

Research must continue to assess the efficacy of current tools and techniques used for software testing. As new technologies continue to emerge and evolve, along with our understanding of them, finding ways to leverage these technologies will result in the improvement of the field. Much like the way software will never be perfect, software testing techniques will always have room for improvement and will need to adapt to the increasing complexities of the digital age.

REFERENCES

Choudhary, S. R., Gorla, A., & Orso, A. (2015, November). Automated test input generation for android: Are we there yet? (E). *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. Presented at the 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE), Lincoln, NE, USA. doi:10.1109/ase.2015.89

ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing --Part 1:General concepts. (2022). *ISO/IEC/IEEE 29119-1:2022(E)*, 1–60. doi:10.1109/IEEESTD.2022.9698145

Ji, R., Zhu, T., Zhu, X., Chen, C., Pan, M., & Zhang, T. (2023). Vision-Based Widget Mapping for Test Migration Across Mobile Platforms: Are We There Yet? *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 1416–1428. doi:10.1109/ASE56229.2023.00068

Linares-Vásquez, M., Moran, K., & Poshyvanyk, D. (2017). Continuous, Evolutionary and Large-Scale: A New Perspective for Automated Mobile App Testing. *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 399–410. doi:10.1109/ICSME.2017.27

NowSecure. (n.d.). 95% of Mobile Apps Fail the OWASP MASVS Industry Standard for Mobile Security, Finds NowSecure Industry Benchmark. Retrieved from https://www.nowsecure.com/press-releases/95-of-mobile-apps-fail-the-owasp-masvs-industry-standard-for-mobile-security-finds-nowsecure-industry-benchmark/#:~:text=Established%20by%20industry%20experts%20and,standard%20for%20mobile%20application%20security.

Sun, X., Chen, X., Liu, Y., Grundy, J., & Li, L. (2023). Taming android fragmentation through lightweight crowdsourced testing. *IEEE Transactions on Software Engineering*, 1–17. doi:10.1109/tse.2023.3266324

Yu, S., Fang, C., Ling, Y., Wu, C., & Chen, Z. (2023, October 22). LLM for test script generation and migration: Challenges, capabilities, and opportunities. *2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS)*. Presented at the 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security (QRS), Chiang Mai, Thailand. doi:10.1109/qrs60937.2023.00029