THE 21ST ANNUAL INTERNATIONAL WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS
(ACM HOTMOBILE 2020)

How to beat the best Android UI testing tools? An infrastructure approach

Wenyu Wang[1], Tao Xie[1,2], Tianyin Xu[1]

[1]University of Illinois at Urbana-Champaign
[2]Peking University

1. Extended Abstract

Automated User Interface (UI) testing has been widely used for quality assurance during Android app development. Numerous Android UI testing tools have emerged from both the research community and the industry after years of development. *Monkey*, a randomized Android UI input generator developed by Google, is one of the earliest efforts in this direction. Authors of various tools developed after Monkey have all claimed that their tools can perform superiorly to Monkey based on more sophisticated app modeling and analysis; however, recent work [1, 2] shows that many of these tools barely outperform Monkey, despite their more advanced design. The arising question is *how to improve these testing tools to achieve better testing effectiveness in practice?*

As a step toward this goal, we propose improving an important part of Android UI testing tool implementation that has been constantly overlooked by researchers and practitioners: *the infrastructure support*, which provides interfaces to enable tool developers to obtain necessary runtime information and take actions accordingly on the app under test. While such support is supposed to be entirely provided by the Android framework, it can make huge impacts on UI testing tools' overall efficiency, contributing to their surprisingly disappointing testing effectiveness. For example, in our experiments, the widely-used official Android UI testing framework *UIAutomator* spends 2.7 seconds on average in taking a snapshot of the current UI hierarchy on some apps. Such interface can be invoked thousands of times during testing. While the app under testing is mostly idle, the tool has to wait until the interface finishes execution before moving to the next step. As a result of inefficiency, much of testing time budget is wasted.

We focus on improving three main interfaces used as primitives of existing UI testing tools: (1) *UI hierarchy capturing*, (2) *UI event execution*, and (3) *UI event monitoring*. While UI hierarchy capturing and UI event execution are fundamentally required by most Android UI testing tools to perform their duties, UI event monitoring provides additional benefits for debugging and reusing the generated test input traces. Specifically, we design and build a new Android UI testing infrastructure named TOLLER that performs the tasks underlying these three interfaces directly within the same DalvikVM as the app code. TOLLER avoids the overhead brought by complicated Android framework internal logistics as well as Inter-Process Communications (IPC) among the app, the Android framework processes, and the on-device agent employed by UIAutomator.

To measure the end-to-end benefit of TOLLER, we integrate TOLLER with a production-quality Android UI testing tool *WCTester* [3] and conduct experiments with six widely-used industrial apps. TOLLER reduces the average time cost of obtaining UI hierarchy snapshots from 997ms to merely 33ms—a more than 30× speedup. The average number of UI snapshots taken could increase from 6307 to 25099 in three hours of testing. Integrating with TOLLER, WCTester achieves a much higher average method coverage, from 22.30% to 27.61%, in three hours of testing. According to our previous study [2], such degree of improvements on some apps reaches or even outperforms some of the best-performing UI testing tools equipped with sophisticated modeling and analysis.

References

[1] CHOUDHARY, S. R., GORLA, A., AND ORSO, A. Automated Test Input Generation for Android: Are We There Yet? In *ASE* (2015).

[2] WANG, W., LI, D., YANG, W., CAO, Y., ZHANG, Z., DENG, Y., AND XIE, T. An Empirical Study of Android Test Generation Tools in Industrial Cases. In *ASE'18* (2018).

[3] ZENG, X., LI, D., ZHENG, W., XIA, F., DENG, Y., LAM, W., YANG, W., AND XIE, T. Automated Test Input Generation for Android: Are We Really There Yet in an Industrial Case? In *FSE'16* (2016).