Record and Replay for Android: Are We There Yet in Industrial Cases?

Wing Lam
Zhengkai Wu
Dengfeng Li
Wenyu Wang
{winglam2,zw3,dli46,wenyu2}@illinois.edu
University of Illinois at Urbana-Champaign, USA

Haibing Zheng
Hui Luo
Peng Yan
Yuetang Deng
{mattzheng,huiluo,peteryan,yuetangdeng}@tencent.com
Tencent, Inc., China

Tao Xie
taoxie@illinois.edu
University of Illinois at Urbana-Champaign, USA

ABSTRACT

Mobile applications, or apps for short, are gaining popularity. The input sources (e.g., touchscreen, sensors, transmitters) of the smart devices that host these apps enable the apps to offer a rich experience to the users, but these input sources pose testing complications to the developers (e.g., writing tests to accurately utilize multiple input sources together and be able to replay such tests at a later time). To alleviate these complications, researchers and practitioners in recent years have developed a variety of record-and-replay tools to support the testing expressiveness of smart devices. These tools allow developers to easily record and automate the replay of complicated usage scenarios of their app. Due to Android's large share of the smart-device market, numerous record-and-replay tools have been developed using a variety of techniques to test Android apps. To better understand the strengths and weaknesses of these tools, we present a comparison of popular record-and-replay tools from researchers and practitioners, by applying these tools to test three popular industrial apps downloaded from the Google Play store. Our comparison is based on three main metrics: (1) ability to reproduce common usage scenarios, (2) space overhead of traces created by the tools, and (3) robustness of traces created by the tools (when being replayed on devices with different resolutions). The results from our comparison show which record-and-replay tools may be the best for developers and identify future directions for improving these tools to better address testing complications of smart devices.

CCS CONCEPTS

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

KEYWORDS

Android, GUI testing, Record-and-replay

ESEC/FSE'17, September 4–8, 2017, Paderborn, Germany

ACM ISBN 978-1-4503-5105-8/17/09…$15.00
https://doi.org/10.1145/3106237.3117769

ACM Reference format:
Wing Lam, Zhengkai Wu, Dengfeng Li, Wenyu Wang, Haibing Zheng, Hui Luo, Peng Yan, Yuetang Deng, and Tao Xie. 2017. Record and Replay for Android: Are We There Yet in Industrial Cases?. In *Proceedings of 2017 11th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, Paderborn, Germany, September 4–8, 2017 (ESEC/FSE'17),* 6 pages.
https://doi.org/10.1145/3106237.3117769

1 INTRODUCTION

Smart devices such as mobile phones are becoming increasingly widespread and these devices rely on mobile applications, or apps for short, in order to perform a variety of activities. These apps make use of the many input sources included in a smart device, such as the touchscreen, sensors (e.g., GPS, accelerometer, compass, gyroscope), and transmitters (e.g., WiFi, Bluetooth) to perform their activities. The numerous input sources on a smart device pose challenges for app developers to create tests that accurately utilize these input sources and to replay an input generated from them at a later time. Record-and-replay tools developed by both practitioners and researchers have aimed to address such problems. In particular, these tools allow developers to easily record and replay complicated usage scenarios of their app. By automating the replay of the complicated usage scenarios of their app, developers no longer have to manually control each input source every time they want to test their app. The automation also allows developers to test their apps at scale (e.g., testing their app on multiple devices at once).

Various record-and-replay tools have been developed by practitioners and researchers for the Android platform for three main reasons. First, majority of the smart devices in the world are using the Android platform and consequently, tools compatible with the platform are thereby compatible with most apps and devices out there. Second, the Android platform is used in the majority of smart devices partly because the Android platform is compatible with a variety of devices. These devices (often varying in screen sizes and available input sources) make app developers' testing process expensive. Record-and-replay tools can automate what otherwise would be a manual process for developers to confirm that their app works as expected on a variety of devices. Third, with the Android platform being open-sourced, record-and-replay tools can make use of the Android platform to gain complete access to record and

replay the interactions between the app under test and the Android platform or other apps.

To investigate the practicality of these different record-and-replay tools for industrial practices, we present the first comparison of popular record-and-replay tools from practitioners and researchers by applying these tools on three popular industrial apps from the Google Play store. In particular, we study the strengths and weaknesses of these tools, how effective they are compared to each other, and how they could be improved. Our comparison is based on three main metrics: (1) ability to reproduce common usage scenarios, (2) space overhead of traces created by the tools, and (3) robustness of traces created by the tools (when being replayed on devices with different resolutions). We evaluate the tools' ability to reproduce common usage scenario and their traces' robustness to different devices, since prior to a new version's release, developers commonly have to test their app on a variety of devices to ensure that their changes do not unexpectedly affect the common usage scenarios of their app on such devices. In addition, we evaluate the space overhead of the traces generated by the tools to show which tools utilize space efficiently.

The results from our comparison show which record-and-replay tools may be the best for developers and identify future directions for improving these tools to better address testing complications of smart devices. In our experiment, we find that the tools under study have different problems and they do not meet the expectation of developers. On our project website [13], readers can see and replay the recorded traces of common usage scenarios for each of the apps under test.

This paper makes the following main contributions:

- A survey of popular record-and-replay tools for Android apps.
- A comparative study of applying such tools on three popular industrial apps.
- An analysis of the results for highlighting the strengths and weaknesses of the different tools under study and possible future directions for record-and-replay tools.

2 RECORD-AND-REPLAY TOOLS: DESIRABLE CHARACTERISTICS

The primary goal of record-and-replay tools is to accurately record an app's execution in order to support automatic replay. By allowing developers to easily record and automate the replay of complicated usage scenarios of their app, record-and-replay tools can be valuable for various software engineering tasks, such as software debugging and testing. Although the goals of these tools are mostly the same, they often vary in functionality. For example, some record-and-replay tools are inadequate for smart devices because these tools can record the user's activity only in terms of discrete high-level actions. These high-level actions cannot easily express complex gestures such as swipes, and they do not support record and replay for events from sensors.

Researchers and practitioners have created record-and-replay tools through two main avenues. The tools are either desktop tools or mobile app tools. Desktop tools, as the name suggests, are tools that are meant to be executed on a desktop machine. These tools commonly use the Android Debug Bridge to send commands to the target device in order to record and replay events. Mobile app tools, on the other hand, are tools that ought to be installed directly onto a mobile device, and the recording and replaying of events are controlled through the installed app. These tools commonly require the user's device to be rooted and often lack functionality compared to desktop tools. The tools used in our study are found by examining the research literature in record-and-replay tools or are recommended to us by the WeChat developers [18], a highly popular messenger app with over 900 million monthly active users.

To understand desirable characteristics of record-and-replay tools from industry practitioners, we consult with over 30 developers from WeChat's Android Development and Engineering Tools team. These WeChat developers have over 100 years of experience combined in testing mobile apps. The WeChat developers inform us of five highly desirable characteristics that a record-and-replay tool should have and four secondary characteristics that are also of interest. The five highly desirable characteristics that a record-and-replay tool should have are whether the tool is coordinate sensitive (recording based on coordinates), widget sensitive (recording based on a GUI widget's unique identifier, e.g., R.id), state sensitive (the tool can restore the state of the app when the trace was recorded), timing sensitive (whether the tool can record the time difference between the events when they were recorded) and lastly, whether the tool requires the developers to install a custom operation system (OS) onto their device. The four secondary characteristics that are also of interest for a record-and-replay tool are access to the tool's source code, and not having to require access to the source code of the app under test, instrumenting the app, and rooting the device in order to use the tool.

Table 1 provides an overview of the record-and-replay tools for Android that researchers and practitioners commonly use. The table reports all of the tools and classifies them according to what type of tool it is (desktop or app). Additionally, the table presents other useful information for each of the tools, such as whether the tool is sensitive to coordinates, widgets, state or timing changes, and whether the tool is publicly available, instruments the app, requires a custom OS, root access, or the source code of the app under test. More details about each of these characteristics are described in the remainder of this section.

Open source. "✓" if source code of the tool is available online. "✗" otherwise.

The WeChat developers do not prioritize this characteristic as top interest in a record-and-replay tool but they state that it is preferable when a tool has its source code online so that they can improve the tool to suit their needs and debug any issues of the tool when needed.

Sensitivity - Coordinate. "✓" if there exist any instances where the events recorded by the tool are based on coordinates of the events on the GUI screen. "✗" otherwise.

Sensitivity - Widget. "✓" if there exist any instances where the events recorded by the tool are based on the widgets (e.g., buttons, text fields) that the events interacted with. "✗" otherwise.

The WeChat developers often prefer a tool to be both widget and coordinate sensitive. When this preference is not an option, they prefer a tool to be widget sensitive instead of being coordinate sensitive. They credit this preference to the fact that coordinate-sensitive tools are often prone to failing due to the slightest GUI

Table 1: Overview of record-and-replay tools for Android. "?" indicates unable to verify this characteristic for the tool.

Name	Open source	Sensitivity				Instruments app	Custom OS	Root access	Needs source
		Coordinate	Widget	State	Timing				
Desktop tools									
appetizer [1]	✗	✓	✗	✓	✓	✗	✗	✗	✗
Bot-bot [2]	✓	✗	✓	✗	✗	✓	✗	✗	✗
Culebra [3]	✓	✓	✓	✓	✗	✗	✗	✗	✗
Espresso [4]	✓	✗	✓	✗	✗	✓	✗	✗	✓
MobiPlay [14]	✗	✓	✗	?	✓	✗	✓	✗	✗
monkeyrunner [12]	✗	✓	✗	✓	✗	✗	✗	✗	✗
Mosaic [8]	✓	✓	✗	✓	✓	✗	✗	✓	✗
Ranorex [15]	✗	✓	✓	✓	✓	✓	✗	✗	✗
RERAN [7]	✓	✓	✗	✓	✓	✗	✗	✓	✗
Robotium [17]	✗	✗	✓	✗	✓	✓	✗	✗	✗
VALERA [10]	✓	✓	✗	✗	✓	✓	✓	✓	✗
App tools									
HiroMacro [9]	✗	✓	✗	✓	✓	✗	✗	✓	✗
RepetiTouch [16]	✗	✓	✗	✓	✓	✗	✗	✓	✗

changes or they are not robust enough for their traces to be replayed on multiple devices.

Sensitivity - State. "✓" if the tool can be replayed successfully only if the device is currently in the state that it was in when the trace was recorded. "✗" if the tool can be replayed even if the phone is in any state. Tools not being state sensitive ("✗") generally require new recordings to start in a new instance of the app, and restart the app before each replay.

The WeChat developers feel that this characteristic is the most important characteristic of record-and-replay tools. They emphasize that one of the major complications with testing an app is the implicit dependences of the app (e.g., the money balance of the account, the friendship status of two accounts) and tools that can help reduce the burden of developers to manually set up such dependences are considered highly valuable to them.

Sensitivity - Timing. "✓" if the tool automatically sets the timing between recorded events, or "✗" if it requires the user to manually specify the timing between events. Tools that can automatically set the timing between recorded events generally set the timing to be the time delay between the events when they were recorded.

The WeChat developers feel that this characteristic would be an important one largely because apps such as WeChat make frequent calls to the Internet, and tools that require developers to manually set the time delay of events are troublesome and error prone. Additionally, tools that set arbitrary time delays between events are also undesirable because they are likely to fail due to insufficient loading time between events.

Instruments app. "✓" if the tool requires that the app under test be first instrumented by the tool. "✗" otherwise. Tools that have this requirement typically can record and replay events only within the app under test and cannot support record-and-replay of any interactions that involve the app under test and the OS or another app (e.g., sharing a photo from another app through the app under test).

The WeChat developers do not prioritize this characteristic as of top interest but are concerned about instrumentations that may be incompatible with WeChat.

Custom OS. "✓" if the tool requires that the host OS be a customized version of Android. "✗" otherwise. Tools that have this requirement can generally record any events within the app under test, and between the app under test and the OS or another app. However, the overhead of installing a custom OS often makes these tools unappealing to developers and troublesome for them to test their app with a variety of OS versions.

The WeChat developers are highly interested in this characteristic. Their interest is largely due to two main factors: (1) they are afraid that results from scenarios reproduced on custom OSs will not translate to the same results on the original Android OS; (2) they are afraid that the use of custom OSs either will not work on many devices or may even damage the phone due to incompatibilities of the OS and the device.

Root access. "✓" if the tool requires the Android device be rooted. "✗" otherwise. Similar to the Custom OS characteristic, tools that have this requirement can generally record any event within the app under test, and between the app under test and the OS or another app. However, unlike the Custom OS characteristic, root access is easily granted to most Android OS versions [11].

The WeChat developers feel that tools that do not require a device to be rooted would be preferable since rooting a device would often break its warranty and make the device more vulnerable to security issues. However, they would be willing to compromise on this characteristic if a tool satisfies the characteristics that they are highly interested in.

Access to app source code. "✓" if the tool requires the source code of the app under test. "✗" otherwise. Generally, record-and-replay tools do not need the app's source code for recording and replaying the events of an app. In our study, we find only one tool, Espresso [4], that requires the source code of an app in order to record and replay. Upon further investigation, we find a third-party tool that can enable Espresso to be used for record and replay without the source code; however, the third-party tool is no longer available and appears to be quite outdated. We suspect that Espresso requires the source code mainly because its other functionalities such as writing Android UI tests requires access to the source code.

The WeChat developers sometimes outsource the testing of the app to other companies so that these companies can verify that apps such as WeChat work well with their products (e.g., a new mobile device). When the WeChat developers outsource the testing of the app, they prefer that these other companies use the same record-and-replay tool as they do, so that if a problem arises, these other companies can share the replay with the WeChat developers and allow them to debug the problem quickly. Moreover, to protect the intellectual property of WeChat, the WeChat developers refrain from exposing the source code of the app to other companies. Therefore, the WeChat developers do prefer a tool that does not require the source code of the app, because this preference helps keep WeChat's source code confidential while allowing the other companies to test WeChat in a manner that is easy for the WeChat developers to debug if necessary.

3 EMPIRICAL STUDY

3.1 Selected Tools

Our study cannot consider all the tools listed in Table 1. For the tools that we omit, we gather the results in Table 1 by emailing the tool authors or going through the documentations of the tools. First, we have to omit Bot-bot and VALERA because we cannot get the tools to work. Second, we have to omit Mosaic and MobiPlay from our study because Mosaic has incomplete documentation to use the tool and MobiPlay's tool is unavailable to get. We have contacted the developers of Bot-bot, VALERA, Mosaic, and MobiPlay with the preceding respective problems, but at the time of our writing, we have yet to hear back or receive a working solution from them.

We are able to get Culebra to record and replay; however, the tool is extremely slow to use. For example, even on a real device, it could take half a minute to type in a single text field, or about ten seconds to perform a single click. Due to Culebra's slowness to use, it seems impractical for developers to use the tool in an industrial setting. Therefore, we decide to omit the tool from our study.

Since our empirical study focuses on only apps from industry, obtaining the source code for our subject apps is infeasible. Therefore, we are unable to include Espresso in our study. Espresso does contain a third-party tool that could enable it to bypass the requirement for source code. However, when we attempt to download the third-party tool, the tool is no longer accessible. Other tools that require the user to pay include Robotium and Ranorex. We decide to omit Ranorex from our study because Ranorex fails to instrument two of the three apps that we study due to the large size of the two apps. Robotium can record events to the GUI only if such GUI elements are controlled by the main process of the app. In other words, when there are other processes that create GUI elements, Robotium would be unable to record actions on such elements. One example for how WeChat uses other processes to create GUI elements is through WeChat's mini programs, which are embedded apps in the WeChat app. Due to Robotium's lack of support for GUI elements created by other processes, we decide to omit Robotium from our study as well.

Lastly, we decide to ignore app tools, since developers must control these tools through the app on the phone. This requirement makes batch testing with these tools much more complicated.

Table 2: Overview of the apps selected to evaluate the record-and-replay tools in our study.

Name	Version	Category	Size (MB)
Facebook [5]	122.0.0.17.71	Social	72.2
File Explorer [6]	5.1.3.0	Business	9.2
WeChat [18]	6.5.7	Communication	40.5

Additionally, the requirement of root permission from these record-and-replay tools can also be troublesome for developers. In the end, we select three tools for our empirical study, namely appetizer, monkeyrunner, and RERAN.

appetizer-toolkit [1], or appetizer for short, is a desktop tool that relies solely on coordinates to record events. Appetizer is closed source but from what we can tell it relies heavily on the Android Debug Bridge (ADB) in order to record and replay events. With appetizer, a developer does not need to worry about manually setting the time between events for recordings but for replays they do need to ensure that the state of the device is the same as it was when the trace was recorded. Assuming that two devices have the same aspect ratio, one of appetizer's most prominent features is that it supports recording events on one device and then replaying the events on a different device. Section 3.5 presents our evaluation of the selected tools on this feature.

monkeyrunner [12] is a desktop tool developed by Google Inc. that provides an API for writing Python programs to control an Android device or emulator. With monkeyrunner, a developer can write Python programs to install an Android app, record and replay keystrokes to it, take screenshots of its user interface, and store the screenshots onto the desktop. The monkeyrunner tool is coordinate sensitive, state sensitive, and not time sensitive. These characteristics make monkeyrunner quite difficult to use as a record-and-replay tool but as we show in later sections of the paper, monkeyrunner does have its advantages compared to the other tools.

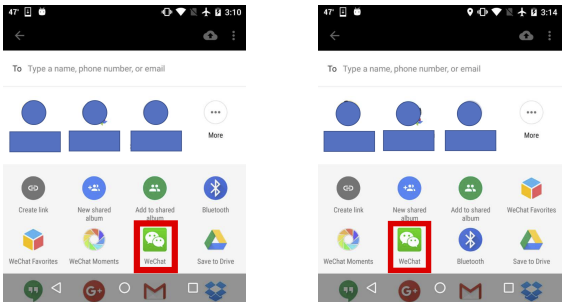
RERAN [7] is a desktop tool similar to the preceding tools in that it is also coordinate sensitive and state sensitive. What sets RERAN apart is that RERAN requires the device to be rooted in order to replay recorded events. RERAN relies solely on ADB commands such as getevent in order to record and replay events. RERAN consists of three steps. First, RERAN uses ADB to record. Second, the recorded trace is then translated by RERAN. Third, the translated trace is replayed by RERAN onto the device.

3.2 Selected Apps

To evaluate the tools that we select, we download three popular apps that belong to distinct categories from the Google Play store. Since the record stage requires human effort to properly configure and use a tool to record popular usage scenarios of an app, and the replay stage requires human effort to verify whether the specified goals of the scenarios are accomplished or not, these factors limit the number of apps that we evaluate. Table 2 shows the apps that we collect and the apps' version, category, and size in megabytes.

3.3 Reproducing Common Usage Scenarios

In this section, we evaluate how the selected record-and-replay tools perform on the selected apps. More specifically, we identify common scenarios in which the selected apps are used for and evaluate whether the record-and-replay tools can accurately record and



(a) Position of WeChat button during record. **(b)** Position of WeChat button during replay.

Figure 1: When recording and replaying one of the common usage scenarios of WeChat (Gallery/Photo), the Android OS may reposition buttons used in the scenario.

replay such scenarios. For WeChat, the WeChat developers provide us the common scenarios for evaluation. For the remaining apps, we identify common usage scenarios from reading the apps' description on the Google Play store. Some of the scenarios identified by us require only one device while some scenarios require two. In either case, we identify one device to be the primary device while the other device to be the secondary device. The primary device is a Nexus 5X running Android 6.0 while the secondary device is a Nexus 5 running Android 6.0. All of the scenarios in which we evaluate the tools on are described in greater detail and shown through videos on our project website [13].

The results of our evaluation can be found in Table 3. We find that in the Facebook scenarios, all three tools successfully replay Facebook's scenarios except for RERAN, which fails on the "Play games" scenario. The reason why RERAN fails for that scenario is that the exit button on the top-left corner of the Facebook app does not appear to be responsive every time we replay the trace. Without the source code of Facebook, we are unable to verify the actual cause of this failure but it is likely that the failure is due to Facebook's handling of the event while other processes of the app are running. For the File Explorer app, appetizer fails for the first two scenarios. The tool works well in the beginning, but it is unable to replay the last clicking event of the scenarios. This inability of appetizer seems to be due to a limitation of the tool.

For the WeChat scenarios, the three tools behave quite differently. Appetizer fails on the "Moments/Photo&Link" scenario due to the unexpected long loading time of Moments, a feature similar to Facebook's wall for allowing users to share and get access to accepted WeChat friends' information. After increasing the time delay between events to 30 seconds, appetizer ceases to fail anymore. For this scenario, appetizer's failure seems to be due to the nondeterministic loading time of Moments in WeChat and not due to the tool itself. We find monkeyrunner to be quite difficult to use especially when recording and replaying on two devices simultaneously. The tool may replay the traces of the primary phone onto the secondary phone if we start the second replay script less than two seconds after the first script starts. Therefore, in our experiment, we get the replay to succeed only 1 out of 10 times for the "Group chat/Link" scenario due to the difficulty of running monkeyrunner on two devices. Furthermore, the tool does not have the functionality of recording long presses; therefore, the tool is unable to record

Table 3: Reproducibility of the top usage scenarios. "✓" indicates that the tool is able to accomplish the scenario specified goals every time. "-" indicates that the tool is able to accomplish the scenario specified goals some of the times. "✗" indicates that the tool is unable to accomplish the scenario specified goals every time.

Scenarios	appetizer	monkey runner	RERAN
Facebook			
Create post	✓	✓	✓
Watch video	✓	✓	✓
Share photo	✓	✓	✓
Play games	✓	✓	-
File Explorer			
Tutorial	✗	✓	✓
Explore menu	✗	✓	✓
Browse directories	✓	✓	✓
New windows	✓	✓	✓
WeChat			
Group chat/Link	✓	-	✓
Group chat/Photo	✓	✗	✓
Gallery/Photo	✓	✗	-
Moments/Photo&Link	-	✗	✓
Moments/Post	✓	✗	✓

the other four scenarios. RERAN is able to successfully replay all of WeChat's scenarios except for the "Gallery/Photo" scenario. Our investigation into the failure finds that the failure is due to the state of the Android OS. The OS may change the order of the options in the sharing menu as shown in Figure 1. This change causes RERAN to share the photo to another app. This cause of failure is not specific to the RERAN tool, and after we replay the scenario a couple of times, the location of the WeChat button stops changing and all subsequent replays of this scenario are no longer affected by this cause of failure.

3.4 Space Overhead

Table 4 shows the space overhead of the traces recorded for the scenarios of each app as described in Section 3.3. In general, the traces recorded by RERAN tend to be 67 times greater in size than the size of the traces recorded by monkeyrunner. On the other hand, appetizer's traces tend to be 2-3 times greater in size than those of monkeyrunner.

Our investigation of the trace files generated by RERAN shows that RERAN produces much more trace information when the trace involves dragging events. Appetizer and monkeyrunner do not appear to suffer from this caveat. For monkeyrunner, the trace file is in plain text and we can see that monkeyrunner simply generates one "DRAG" event for each dragging event. For appetizer, the trace file is a binary file, which we are unable to decipher exactly how dragging events are stored.

3.5 Replaying Robustness

One primary usage of record-and-replay tools is to ease the manual efforts of developers during regression testing. Prior to a new version's release, developers commonly have to test their app on a variety of devices to ensure that their changes do not negatively

Table 4: Space overhead of the record-and-replay tools in kilobytes.

Name	appetizer	monkeyrunner	RERAN
Facebook	10	3	215
File Explorer	4	3	94
WeChat	24	8	625
Total	38	14	934

affect the app on such devices. To replicate the developers' intended usage of record-and-replay tools, we evaluate how recorded traces from Section 3.3 perform when they are replayed on a device that contains a resolution different than the device that recorded the traces. More specifically, all of WeChat's traces described in Section 3.3 are recorded in either a Nexus 5X or a Nexus 5, which both have a 1920 by 1080 resolution. To evaluate the robustness of these tools on different resolutions, traces recorded on the Nexus 5X are replayed on a Samsung Galaxy SIII that has a resolution of 1280 by 720, while traces recorded on the Nexus 5 are replayed on a Nexus 6P that has a resolution of 2560 by 1440.

For all of the WeChat scenarios, none of the three tools are able to successfully replay the scenarios. However, these tools do behave differently. RERAN would generate only the first one or two clicking events. For example, it can successfully open the menu to create a new group chat, but then no additional events are replayed. Monkeyrunner would click into wrong coordinates for all of the scenarios. Appetizer can successfully create a group chat and even finish typing the message. However, when it comes to dragging events, appetizer is unable to replay the events at the right coordinates.

Since all of the tools are unsuccessful in replaying the scenarios of WeChat, we also replay the traces of Facebook on two different devices. Instead of a Nexus 5X for the primary phone, we use a Nexus 5 and instead of a Nexus 5 for the secondary phone, we use a Nexus 6p. We find that RERAN fails for all of Facebook's scenarios on both the Nexus 5 and Nexus 6p phones. Monkeyrunner successfully replays the steps of the first two scenarios on only the Nexus 5, but it fails on the steps of the Nexus 6p. For the other scenarios, both the Nexus 5 and Nexus 6p fail to replay successfully. Appetizer successfully replays the first three scenarios on both phones with a small exception for the "Create post" scenario. For this scenario, the typed string on the Nexus 6p changes from "Test" to "Gest". Since the "Play games" scenario contains two dragging events, appetizer fails this scenario for Facebook similar to what it did for WeChat's scenarios.

4 DISCUSSION

Our results from Sections 2 and 3 show that although there exist many record-and-replay tools from practitioners and researchers, very few of these tools are readily available for developers to actually use in practice. The three tools that we end up using for our study also do not produce stable results (indicated by "-" in Table 3) in four of the scenarios that we evaluate them on. RERAN, which is one of the very first record-and-replay tools proposed by researchers, is the only tool that is able to record and replay all scenarios and able to achieve the goal of the scenarios at least some of the times. However, the traces produced by RERAN do occupy about 67 times the space that traces from monkeyrunner would occupy. For those scenarios that we evaluate the record-and-replay tools with, the traces of each tool occupy less than 1 megabyte. However, if industrial app developers are to heavily rely on such tools, the space overhead induced by RERAN may start to become an issue. Monkeyrunner is provided by Google. However, the tool is difficult to use and quite problematic, especially when replaying traces on two devices simultaneously. Appetizer is easier to use but also contains some unexpected bugs such as the imprecise replay of dragging events on different devices. Lastly, we find that when replaying traces on a device with a different screen resolution, RERAN could not replay any scenarios. Monkeyrunner can succeed in some simple scenarios and appetizer can do slightly better than monkeyrunner. However, none of these tools are able to replay complicated scenarios on different screen sizes successfully.

Our interactions with the WeChat developers help derive desirable characteristics of a record-and-replay tool (Section 2). Unfortunately, none of the tools that we find exhibit all of these characteristics. Researchers and practitioners working on record-and-replay tools may want to develop a more desirable tool to exhibit this set of characteristics.

5 CONCLUSION

In this paper, we have presented a comparison of the popular record-and-replay tools from researchers and practitioners, by applying these tools to testing three popular industrial apps downloaded from the Google Play store. The results from our comparison show that none of these tools are desirable for developers to use in practice, calling for the need of developing more desirable tools in future work.

ACKNOWLEDGMENTS

The work of UIUC authors is supported in part by NSF under grants no. CCF-1409423, CNS-1434582, CNS-1513939, CNS-1564274.

REFERENCES

[1] appetizer-toolkit, 2017. https://github.com/appetizerio/appetizer-toolkit.
[2] Bot-bot, 2017. http://imaginea.github.io/bot-bot/index.html.
[3] Culebra, 2017. https://github.com/dtmilano/AndroidViewClient/wiki/culebra.
[4] Espresso Test Recorder, 2017. https://developer.android.com/studio/test/espresso-test-recorder.html.
[5] Facebook, 2017. https://play.google.com/store/apps/details?id=com.facebook.katana.
[6] File Explorer, 2017. https://play.google.com/store/apps/details?id=nextapp.fx.
[7] L. Gomez, I. Neamtiu, T. Azim, and T. Millstein. RERAN: Timing- and touch-sensitive record and replay for Android. In *ICSE '13*, pages 72–81, 2017.
[8] M. Halpern, Y. Zhu, R. Peri, and V. J. Reddi. Mosaic: cross-platform user-interaction record and replay for the fragmented Android ecosystem. In *ISPASS'15*, pages 215–224, 2017.
[9] HiroMacro Auto-Touch Macro, 2017. https://play.google.com/store/apps/details?id=com.prohiro.macro.
[10] Y. Hu, T. Azim, and I. Neamtiu. Versatile yet lightweight record-and-replay for Android. In *OOPSLA'15*, pages 349–366, 2017.
[11] KingoRoot, 2017. https://www.kingoapp.com.
[12] monkeyrunner, 2017. https://developer.android.com/studio/test/monkeyrunner/index.html.
[13] Record-and-Replay tool study project website, 2017. https://sites.google.com/view/record-and-replay.
[14] Z. Qin, Y. Tang, E. Novak, and Q. Li. MobiPlay: A remote execution based record-and-replay tool for mobile applications. In *ICSE '16*, pages 571–582, 2017.
[15] Ranorex, 2017. http://www.ranorex.com/mobile-automation-testing.html.
[16] RepetiTouch Free (root) (ads), 2017. https://play.google.com/store/apps/details?id=com.cygery.repetitouch.free.
[17] Robotium Recorder, 2017. https://robotium.com/products/robotium-recorder.
[18] WeChat, 2017. https://play.google.com/store/apps/details?id=com.tencent.mm.