2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)

GUI-Guided Repair of Mobile Test Scripts

Minxue Pan[†‡*], Tongtong Xu[†§], Yu Pei[¶], Zhong Li[†§], Tian Zhang[†§*] and Xuandong Li[†§*]
[†]State Key Laboratory for Novel Software Technology, Nanjing University, China
[‡]Software Institute, Nanjing University, China
[§]Department of Computer Science and Technology, Nanjing University, China
[¶]Department of Computing, The Hong Kong Polytechnic University, Hong Kong, China
mxp@nju.edu.cn, xttluck@gmail.com, csypei@comp.polyu.edu.hk, zzlexp@gmail.com, {ztluck,lxd}@nju.edu.cn

***Abstract*—Graphical User Interface (GUI) testing has been the focus of mobile app testing. Manual test cases, containing valuable human knowledge about the apps under test, are often coded as scripts to enable automated and repeated execution for test cost reduction. Unfortunately, many test scripts may become broken due to changes made during app updates. Broken test scripts are expected to be updated for reuse; however, the maintenance cost can be high if large numbers of test scripts require manual repair. We propose an approach named METER to repairing broken test scripts automatically when mobile apps are updated. METER novelly leverages computer vision techniques to infer GUI changes between two versions from screenshots and uses the GUI changes to guide the repair of test scripts. In experiments conducted on 18 Android apps, METER was able to repair 78.3% broken test scripts.**

***Index Terms*—Mobile apps, GUI testing, Test script repair, Computer vision, OCR.**

I. INTRODUCTION

Mobile apps are frequently updated to remain competitive in the market. Unfortunately, frequent updates may also cause problems. With shorter developing time for each update, the quality of apps becomes more difficult to guarantee.

Testing is still the mainstream quality assurance approach for mobile apps. Due to the event-driven nature and gesture-based interactions, GUI testing is now one of the most widely used methodologies for testing mobile apps [1]. GUI tests used in industry are typically programmed or recorded as scripts to enable automated and repeated execution by test harnesses/tools. For such scripts to comprehensively cover the business logic of apps, human testers often need to invest valuable time to transcribe their domain knowledge and experience. However, test scripts may become broken after the app is updated and its GUI changed. Maintenance cost can be high if it is to be done manually and the number of test scripts that need fixing is large.

Mobile apps typically support a rich set of GUI features for good user experience, and they often have GUI elements in image than in text to make the GUI more attractive, which poses challenges in test script repair. We propose an approach called METER (**MOBILE TEST REPAIR**) to automatically repairing broken test scripts when mobile apps are updated. METER determines if the update affects a GUI element or a screen via computer vision (CV) techniques, and retains or repairs a test action based on if the GUI elements and screens associated with the action's execution are influenced by the update. We implemented our approach in a tool also called METER and carried out experiments on 18 open-source Android apps. METER was able to automatically repair 78.3% broken Android test scripts, which speaks well for its effectiveness.

∗ Corresponding author.

This research was supported by the National Natural Science Foundation of China (No. 61502228), the Hong Kong RGC General Research Fund (GRF) (PolyU 152703/16E) and The Hong Kong Polytechnic University internal fund (1-ZVJ1 and G-YBXU).

II. RELATED WORK

The idea of test script repair for desktop applications is first proposed in [2], which devises a model-based approach called GUI Ripper. Work [3] extends [2] by adding a mechanism to obtain the application model through reverse engineering. GUI code changes, either recorded in an IDE [4] or extract through code analysis [5], are also used to repair test cases.

Studies target mobile domain are just emerging. In two recent studies [6], [7], model-based tools for Android GUI test repair are proposed. Both assume that precise models of the app are already prepared as inputs to guide the test repair, whereas it is known that considerable manual effort is required for constructing or perfecting such models.

Computer vision techniques have been used to help users write GUI test scripts, e.g., Sikuli [8] and JAutomate [9]. These approaches, however, cannot process GUI changes, and therefore, are not capable of repairing broken test scripts.

III. APPROACH OVERVIEW

Fig. 1 illustrates the overview of METER. Given the version N and N+1 of a mobile app and the test scripts for version N as inputs, METER first executes the test scripts on version N, to ensure all the scripts can pass without failures. Meanwhile, for each test action in the script, METER records the screenshots after the test action is executed, as well as the position and size of the widget on which the action is performed. With this knowledge, METER maintains an execution history for each test script, which will serve as the guideline for the subsequent script repair for version N+1. Then METER tries to execute the test scripts on version N+1. This time, METER takes a screenshot before executing a test action, and queries the execution history to obtain the screenshot and widget recorded for version N before executing the same test action in the same test script. By leveraging a set of computer vision techniques,

2574-1934/19/$31.00 ©2019 IEEE
DOI 10.1109/ICSE-Companion.2019.00137

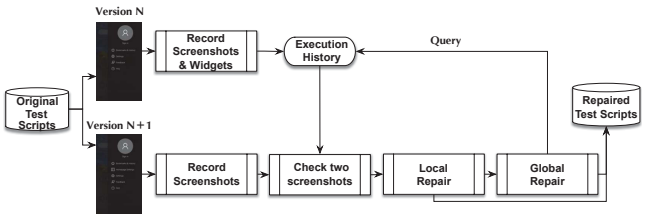

Fig. 1: Overview of METER

including the contour detection, Scale-Invariant Feature Transform and OCR, METER extracts and maps he text elements and non-text GUI elements between the two screenshots, to tell whether the two screenshots can be considered mapped. Note that in METER, two screenshots are mapped not by image pixel comparison but by GUI elements mapping. Therefore, when two screenshots are mapped, METER has already mapped the GUI elements within the same screen page. Repairs that can be performed locally, i.e., the modification, movement or replacement of a widget within the current page, will be tried. To ensure the repair is correct, METER also checks whether the two screenshots after executing the test action are mapped or not.

Having local repair is not sufficient when METER cannot locate the widget for the test action to be executed on version N+1 or map the screenshots either before or after executing the test action. To increase the repairing ability, METER also includes a global repair module, which currently consists of two repair methods. One focuses on repairing the failure caused by moving a widget to another screen in the app. The other focuses on repairing the failure caused by changing redirecting relations between app screens, e.g., the triggering of a menu item opens a different page.

After the repair, the output is the repaired test scripts, as shown in Fig. 1. Meanwhile, these repaired test scripts have already been successfully executed on version N+1, so the testing of version N+1 is also completed provided that all the scripts can be repaired.

IV. EVALUATION

To evaluate the effectiveness and efficiency of METER in repairing GUI test scripts for mobile apps, we conducted experiments that apply METER to 18 Android apps from previous studies, which are shown in Table I.

We downloaded multiple versions of each selected app, chose the latest version as the updated one, and an earlier version as the base one. We asked nine postgraduate students in Computer Science, each with at least two-year experience in mobile application development and testing, to write test scripts to achieve around 50% statement coverage for the base versions of the apps. These test scripts are the ones to be repaired on the updated versions.

The experiments were run on a Samsung Galaxy S6 phone and a MacBook Air laptop running Mac OS X10 with one Quad-core 1.7GHz CPU and 8 GB memory.

For each app, Table I gives its name (APP), the two versions used (VERSIONS), the number of test scripts we prepare for the app (#S), the number of broken test scripts on the updated version (#B), the number of repaired test scripts (#R), and the repair time (TIME) recorded in minutes.

TABLE I: Android Apps used as subjects in the experiments.

APP	VERSIONS	#S	#B	#R	TIME
SuperGenPass	2.2.3→3.0.0	5	3	2	157.27
OI File Manager	2.0.5→2.2.2	29	17	13	213.33
Remote Keyboard	1.6→1.7	9	5	5	17.06
SMS Scheduler	1.37→1.48	19	2	2	32.36
Tasks Astrid To-Do List	4.9.14→5.0.2	22	1	1	74.21
Lighting Web Browser	4.4.2 → 4.5.1	23	9	9	222.29
DumbphoneAssistant	0.4→0.5	6	4	4	35.43
SysLog	2.0.0→2.1.1	5	3	1	88.41
Budget Watch	0.18→0.2	14	3	1	125.60
Pedometer	5.16→5.18	5	1	1	44.65
Who Has My Stuff?	1.0.24→1.0.30	10	2	1	34.10
Notepad	1→1.12	10	6	5	24.41
A2DP Volume	2.12.4 → 2.12.9	8	2	2	37.34
K-9 Mail	5.207→5.4	39	11	9	413.92
AnyMemo	10.8→10.10.1	31	6	6	102.12
Auto answer & callback	1.9→2.3	17	10	9	133.23
AnkiDroid	2.6→2.8.3	29	5	1	502.28
Open Camera	1.40.0→1.42.2	17	13	9	132.12
Total	-	327	120	94	2603.46

Overall, 327 test scripts were written, among which 120 test scripts were broken on the updated version. METER successfully repaired 94 test scripts, demonstrating a 78.3% repair rate. The total time cost of repairing test scripts for all the apps amounts to 2603.46 minutes, with the maximum repairing time 502.3 minutes and the minimum 17 minutes. Given that the application of METER is fully automatic and requires no human intervention, we believe the overall efficiency of METER is acceptable.

V. CONCLUSION

In this paper, we propose METER—a novel approach to automatically repairing GUI test scripts for mobile apps based on computer vision techniques. Experimental evaluation of METER on 18 real-world Android apps shows that METER is both effective and efficient.

REFERENCES

[1] C. Hu and I. Neamtiu, "Automating GUI testing for android applications," in *Proceedings of AST '11*. ACM, 2011, pp. 77–83.
[2] A. M. Memon and M. L. Soffa, "Regression testing of GUIs," in *Proceedings of ESEC/FSE '11*. ACM, 2003, pp. 118–127.
[3] A. M. Memon, "Automatically repairing event sequence-based GUI test suites for regression testing," *ACM Trans. Softw. Eng. Methodol.*, vol. 18, no. 2, pp. 4:1–4:36, 2008.
[4] B. Daniel, Q. Luo, M. Mirzaaghaei, D. Dig, D. Marinov, and M. Pezzè, "Automated GUI refactoring and test script repair," in *Proceedings of ETSE '11*. ACM, 2011, pp. 38–41.
[5] M. Grechanik, Q. Xie, and C. Fu, "Maintaining and evolving GUI-directed test scripts," in *Proceedings of ICSE '09*. IEEE Computer Society, 2009, pp. 408–418.
[6] X. Li, N. Chang, Y. Wang, H. Huang, Y. Pei, L. Wang, and X. Li, "ATOM: automatic maintenance of GUI test scripts for evolving mobile applications," in *Proceedings of ICST '17*. IEEE Computer Society, 2017, pp. 161–171.
[7] N. Chang, L. Wang, Y. Pei, S. K. Mondal, and X. Li, "Change-based test script maintenance for android apps," in *Proceedings of QRS '18*. IEEE Computer Society, 2018, pp. 215–225.
[8] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using GUI screenshots for search and automation," in *Proceedings of UIST '09*. ACM, 2009, pp. 183–192.
[9] E. Alégroth, M. Nass, and H. H. Olsson, "JAutomate: A tool for system- and acceptance-test automation," in *Proceedings of ICST '13*. IEEE Computer Society, 2013, pp. 439–446.