A Framework for Regression Testing of Outdoor Mobile Applications

Carlo Bernaschina, Roman Fedorov, Darian Frajberg, Piero Fraternali
Politecnico di Milano
Milan, Italy
first.last@polimi.it

***Abstract*—Outdoor mobile applications are becoming popular in fields such as gaming, tourism and environment monitoring. They rely on the input of multiple, possibly noisy, sensors, such as the camera, GPS, compass and gyroscope. The regression testing of such applications requires the reproduction of the real conditions in which the application works, which are hard to reproduce without automated support. We present a capture & replay framework that automates regression testing of mobile outdoor applications, by recording data streams in real-time on the field from multiple sensors, replays them in lab and computes quality metrics to trace regression errors.**

***Keywords*-Mobile applications; regression testing; augmented reality;**

I. Introduction

Outdoor mobile applications are becoming more and more popular. Exemplary applications are maps (e.g., Google Maps, `maps.google.com`), touristic guides (e.g., mTrip, `www.mtrip.com`), games (e.g., Pokemon Go, `www.pokemongo.com`), and augmented reality (e.g., PeakLens, `www.peaklens.com`). They are unique due to their need to process, in real-time, data streams coming from multiple, heterogeneous, and often noisy sensors. Augmented reality applications process data streams from GPS, compass, accelerometer, gyroscope, and camera, to identify object on the view. Thus, they are extremely sensible to different devices and external conditions (e.g., GPS by meteorological conditions and compass by electromagnetic fields). Due to the heterogeneity and correlation of input data, testing cannot be done on synthetic data, but should be performed on input captured on the field. Therefore, *regression testing*, i.e., the practice of testing a new version of an application to verify its correctness after a set of changes, assumes a prominent role in the development process. Incremental releases, required to extend functionality and/or improve performance, must not jeopardize already working functions. These characteristics of outdoor, sensor-based mobile applications make testing, and regression testing in particular, challenging for the following reasons: **1)** *Reproducibility.* Field conditions in which the app is used (e.g. location, sensor streams) are difficult or even impossible to reproduce in a lab. **2)** *Non Functional Requirements.* Besides absence of bugs, testing must also identify improper handling of non functional requirements, such as the accuracy perceived by a user. **3)** *Data Gathering.* Testing requires the collection of data series of input signals and the correlated application outputs. We present a framework for black-box regression testing of outdoor mobile apps, specifically aimed towards efficient verification of new releases with large collections of complex test data suites. It supports capture, replay, custom metrics and regression testing.

II. Related work

Capture and replay frameworks for the assessment of software quality have been largely studied in the past due to the importance that they represent for maintenance and testing purposes. In [1] the authors presented a tool to capture and replay classic desktop Java program executions in the field. In their work they described how all the interactions between the main program and the system are stored, including the GUI displayed. Additionally, they replay such executions presenting each thread with exactly the same input sequence it had during the capture. Moreover, in [2] the authors presented their technique and tool for the same purpose and they proposed their utilization for post-mortem dynamic analysis of user executions, debugging of deployed applications and regression testing. They state that the effectiveness of regression testing highly depends on how well it represents the way the program is used in the field. We agree with the authors, but we highlight that in outdoor sensor-based applications require a non trivial capture and replay process, which is the motivation of this work. In [3], [4] studies on capture and replay were presented focusing just at GUI level. Conversely, in [5], the authors present an approach specifically conceived for mobile devices, in which they record and replay Android apps usage traces by replicating GUI gestures and sensor readings. An Android specific framework was presented in [5] focusing also on interactions via gestures and sensor readings. However, it was missing support for some critical signals, like camera and GPS. Our original contribution is the design of a capture and replay framework for outdoor multi-sensor applications. We also show how it has been used to define quality metrics and automate regression testing, for a mobile application[6].

III. Capture and replay framework

In this section we illustrate the architecture of the proposed framework. For the sake of concreteness, we show its use with PeakLens, an outdoor mobile application[6]), which

Version	**A**				**B**				**C**				**D**			
Sequence	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
AAE (deg)	0.62	0.47	0.66	0.50	12.45	7.94	2.30	4.28	3.20	2.85	2.50	3.12	0.92	0.64	0.77	0.49
Precision (%)	98.96	99.56	99.59	98.79	84.70	85.53	96.40	92.68	98.85	99.14	99.73	98.86	99.11	98.66	99.50	98.26
Recall (%)	66.32	94.89	97.73	60.60	75.99	87.45	96.73	90.45	98.49	99.60	98.57	98.82	99.33	99.30	98.68	99.39
PQ (%)	25.30	86.60	93.80	0.00	3.82	5.20	64.80	10.82	22.09	80.60	88.60	33.51	92.57	94.40	96.40	99.48
RE	0.88	0.80	0.91	0.75	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table I: Evaluation of versions A, B, C and D on 4 sequences

identifies mountain peaks and overlays them in real-time on the view. However, the framework is applicable to any app that produces output based on multiple sensor inputs.

Architecture. The proposed capture and replay framework consists of a set of coordinated software modules: **1)** *Capture*. It executes in the mobile device; it acquires the sensor data streams. Readings at each time stamp stored together with the business logic output to compose a *sequence*. **2)** *Replay*. It permits the visual inspection of the app GUI, which reproduces the same type of outputs recorded on the field. Besides, it produces an execution trace, composed by the original sequence and the corresponding new computed outputs. **3)** *Report*. It enables testing new versions of the business logic on previously recorded sequences by assessing the defined quality metrics. The component compares the output produced by the app business logic during replay steps with the reference outputs stored in the original sequence. **4)** *Sequence Editor*. It is a GUI able to replay a sequence and used to mark it as correct, manually fix positioning errors and save it as a *gold* sequence.

Input test data and gold standard. Through a beta testing program we have gathered hundreds of sequences worldwide. We identified a set of *gold* sequences. using a web interface in which the user can mark a sequence as correct (does not contains significant positioning errors) or adjust it manually frame by frame. Even tough this approach may overlook minor deviations, this simple process only requires the visual inspection of the replay, confirming that these deviations would go unnoticed by an average user.

Defects and output quality metrics. Classic regression testing assesses the presence of *bugs* re-introduced in the system by a change. However, in complex, multi-sensor outdoor applications, their success depends primarily on non-functional features such as *accuracy* of the outputs. Therefore, we focus on this aspect during the evaluation of the presented framework. To assess defects, the Report component uses metrics defined at the level of the individual sequence or frames and averaged on the set of sequences of a test suite. In the case study, the following metrics have been defined, to quantify the defects in peak positioning. The **Average Angular Error (AAE)**, formally described in [6], considers the positioning errors of all the peaks w.r.t. to the position in the gold sequence. The **Precision** measures the fraction of peaks included in a frame that were supposed to be shown. The **Recall** measures the fraction of peaks present in a frame of the gold sequence that also appear in the corresponding frame of the tested sequence. The **Perceived Quality (PQ)** measures the percentage of the frames of a sequence that are "good enough". This indicator can be regarded as the fraction of the entire sequence time during which the user experience was satisfactory. The definition of "good" is binary an is computed by properly thresholding the previously presented metrics (3deg, 80% and 80% respectively). Finally, the **Ranking Error (RE)** assesses the difference between the ordering of peaks in the test and gold frames.It is defined as the Normalized Discounted Cumulative Gain index [7] over the two ordered lists of peaks. It uses as relevance a numerical score referred to the gold frame (3, 2 and 1 for peaks in the 1st, 2nd and 3rd page; 0 elsewhere).

IV. Evaluation

For space reasons, we comment the evaluation of PeakLens only for four gold sequences and four app releases. The first 4 columns of Table I report the indicators for a release with a regression error affecting peak ranking (version A). In general, Perceived Quality is the most representative metric at first sight, because it summarizes all the other ones. However, in version A the diminished value of PRE shows that a defect related to peak ranking has been introduced. The next 8 columns of Table I refer to versions B and C in which, respectively, a scale factor and a vertical offset projection problem manifested. PQ decreased strongly in both cases, with sensible angular error increase and loss of both precision and recall. Sequence replay permitted us to locate the wrongly positioned peaks and to remove the defect. Finally, the last 4 columns of Table I refer to version D, a release featuring a new algorithm to track peaks during movement. In this case, PQ has not been affected considerably and therefore the performance of the release was considered acceptable. Overall, the regression testing, coupled with the easy replay of complex outdoor conditions, gave effective feedback on the new versions, and an average PQ value below 90% proved to be a good predictor of the insurgence of defects after a change.

V. Conclusions

We have presented a capture and replay framework for regression testing of mobile applications exploiting input output correlations. We have shown the impact of the proposed framework on the development of an augmented reality application, with a particular focus on non-function requirements.

References

[1] J. Steven, P. Chandra, B. Fleck, and A. Podgurski, "jrapture: A capture/replay tool for observation-based testing," in *ISSTA*, 2000, pp. 158–167. [Online]. Available: http://doi.acm.org/10.1145/347324.348993

[2] S. Joshi and A. Orso, "SCARPE: A technique and tool for selective capture and replay of program executions," in *23rd IEEE International Conference on Software Maintenance (ICSM 2007), October 2-5, 2007, Paris, France*. IEEE, 2007, pp. 234–243. [Online]. Available: http://dx.doi.org/10.1109/ICSM.2007.4362636

[3] L. J. White, "Regression testing of gui event interactions," in *Software Maintenance 1996, Proceedings., International Conference on*. IEEE, 1996, pp. 350–358.

[4] O. El Ariss, D. Xu, S. Dandey, B. Vender, P. McClean, and B. Slator, "A systematic capture and replay strategy for testing complex gui based java applications," in *Information Technology: New Generations (ITNG), 2010 Seventh International Conference on*. IEEE, 2010, pp. 1038–1043.

[5] L. Gomez, I. Neamtiu, T. Azim, and T. Millstein, "Reran: Timing-and touch-sensitive record and replay for android," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 72–81.

[6] R. Fedorov, D. Frajberg, and P. Fraternali, "A framework for outdoor mobile augmented reality and its application to mountain peak detection," in *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*. Springer, 2016, pp. 281–301.

[7] K. Järvelin and J. Kekäläinen, "Discounted cumulated gain," in *Encyclopedia of Database Systems*, L. Liu and M. T. Özsu, Eds. Springer US, 2009, pp. 849–853. [Online]. Available: http://dx.doi.org/10.1007/978-0-387-39940-9_478