How to Validate Mobile Crowdsourcing Design? Leveraging Data Integration in Prototype Testing

Chu Luo
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
chu.luo@ee.oulu.fi

Miikka Kuutila
M3S
University of Oulu
Oulu, Finland
miikka.kuutila@oulu.fi

Simon Klakegg
Denzil Ferreira
Huber Flores
Jorge Goncalves
Vassilis Kostakos
Center for Ubiquitous Computing
University of Oulu
Oulu, Finland
{firstname.lastname}@ee.oulu.fi

Mika Mäntylä
M3S
University of Oulu
Oulu, Finland
mika.mantyla@oulu.fi

Ubicomp/ISWC'16 Adjunct, September 12 - 16, 2016, Heidelberg, Germany

ACM 978-1-4503-4462-3/16/09…$15.00
DOI: http://dx.doi.org/10.1145/2968219.2968586

Abstract
Mobile crowdsourcing applications often run in dynamic environments. Due to limited time and budget, developers of mobile crowdsourcing applications sometimes cannot completely test their prototypes in real world situations. We describe a data integration technique for developers to validate their design in prototype testing. Our approach constructs the intended context by combining real-time, historical and simulated data. With correct context-aware design, mobile crowdsourcing applications presenting crowdsourcing questions in relevant context to users are likely to obtain high response quality.

Author Keywords
Smartphones; ubiquitous computing; mobile crowdsourcing; ambient intelligence; software testing and debugging.

ACM Classification Keywords
H.5.m. Information interfaces and presentation (e.g., HCI): Miscellaneous;

Introduction
Mobile crowdsourcing applications usually operate smartly in a specific context. Therefore, this kind of

context-aware applications requires elaborate quality control for truthful responses. Prototype testing is proven to be an effective way in validation of software design [2]. However, this task can be extremely challenging [12] for mobile context-aware applications. First, since mobile context-awareness often involves multiple devices, various data sources, a large amount of data and complex algorithms, it is difficult for developers to conduct static testing by reading the source code. Second, due to limited time and resources, developers may have no chance to completely test these systems in real world situations (e.g., specific time, locations, device models or users) where these systems should operate. For example, mobile apps may collect and process diverse types of data (e.g., location, application usage or emotions) from hardware sensors, software or the Experience Sampling Method (ESM) [7] for a long period of time [13].

To enable the testing of mobile context-aware applications, researchers have proposed context management tools, testing techniques (e.g., white-box and black-box testing) and platforms [3,10,12]. Despite numerous efforts in previous work, a systematic testing framework supporting heterogeneous data sources across multiple devices is still lacked.

This paper first outlines the challenges identified from the state of art. To validate mobile crowdsourcing applications, we then present and discuss the approach that uses data integration in prototype testing. The integrated testing data comprises a combination of real-time, historical and simulated data to construct the intended context where these systems should operate. Our aim is to ensure that mobile crowdsourcing applications recognise their context correctly through effective and efficient prototype testing. Thus, presenting crowdsourcing tasks in relevant context to users can improve the response quality.

Related Work

To test mobile applications, the simplest way is to let users write reviews for a testing version. Besides, developers can also test their systems using specialised tools. For example, Monkey [1] is a black-box testing tool within Android SDK. It can generate user events or system events on devices or emulators. Monkey monitors application responses to look for crashes, unhandled exceptions and unresponsiveness. Similarly, Android Studio provides Monkeyrunner [9] for higher test automation. Developers can specify commands and events using its API. Monkeyrunner can present and store the testing results as screenshots.

However, the testing of context-aware systems normally requires real context. In most cases (e.g., longitudinal environment monitoring), developers may not manage to test their systems in real context. Prior work attempted to alleviate this problem from many aspects. ContextViewer [3] is a context management tool that visualises and preprocesses historical contextual data from mobile devices. Developers can select relevant contextual information using ContextViewer to replay context with other tools. ContextSimulator [4] is an example of context replay tools to test mobile context-aware systems. It can fetch and replay data at a certain speed from a mobile context-aware middleware, AWARE [5], designed for Android. It allows developers to specify an existing database with historical contextual data. However, a limitation is that it does not support a query for multiple rows of data at the same time from the target application. Similarly, MobiPlay [10] is a remote testing tool that records and replays data for Android

applications. These applications run on a server, while MobiPlay on mobile devices acts as a client showing the GUI of the target application. Other context replay tools with similar features include RERAN [6]. A weakness of RERAN is that it cannot record or replay GPS and microphone data.

In addition, collaboration and communication within mobile devices has been trending in modern context-aware systems. For example, Riboni and Bettini [11] constructed a situation of smartwatch and smartphone to evaluate an activity recognition technique based on ontological and statistical Reasoning. However, testing techniques for mobile cross-device context-aware scenarios are quite scant, although a number of studies have investigated the testing of cross-device UI interactions.

Overall, the testing of mobile crowdsourcing applications requires more advanced features in existing tools and techniques.

Challenges in Mobile Context-Aware Testing

Although existing techniques and tools are able to help developers to conduct necessary testing in most cases, several issues remain in the testing of mobile crowdsourcing applications:

1. **Participant recruitment.** This concerns applications whose results may vary among different groups of users, such as people with different personalities. Testing these applications requires representative samples according to demographic features. Due to limited budget and time, the sample size of users is often small unless the target application is sufficiently appealing to attract volunteers.
2. **Timeliness of historical data.** Historical contextual data often contains a large sample size of users. It can also reduce the cost of participant recruitment. However, historical contextual data may lack some important data sources since new hardware sensors are increasingly emerging on mobile devices.
3. **Uncommon context.** Several kinds of contextual data, such as app crashes, are too rare for a dataset to include. In addition, hardware sensors, such as GPS, may be unavailable at some moments which are critical to test the system robustness.
4. **Device heterogeneity.** Mobile devices usually have different hardware, operating systems and features. Testing on all possible mobile devices is expensive and impractical. If the target system involves communication within devices, the possible combination and permutation will make testing on real devices even more impractical.

Leveraging the Combination of Real-Time, Historical and Simulated Data

To validate the design of mobile crowdsourcing applications, developers typically build prototypes and conduct prototype testing. Thus, we propose a data integration approach for developers to construct intended context in prototype testing. Figure 1 depicts the testing flow from data integration to testing environments. For data integration, the testing tool should be able to:

- capture real-time data;
- fetch and replay historical contextual data;
- simulate data.

Figure 2 compares these three data sources with regard to cost and realism. The criteria used in deciding the data source in prototype testing are quite

straightforward. Historical contextual data is a good choice to reduce the development cost when it is available. If the historical data cannot reshape the intended context, developers have to choose real-time data or simulated data. As the ideal way to reproduce high fidelity context, real-time data can be obtained directly from participants' devices. For instance, the testing tool obtains heart rates directly from a suitable phone if there is no historical data. However, an experiment with participants can be expensive and inconvenient. To construct uncommon context, developers may rely on data manipulation, which is inexpensive. For example, the testing tool simulates a crash information of a specific application to test the target application because it is hard to witness a real crash. The drawback of simulated data is that it may contain distortion compared to the original context.

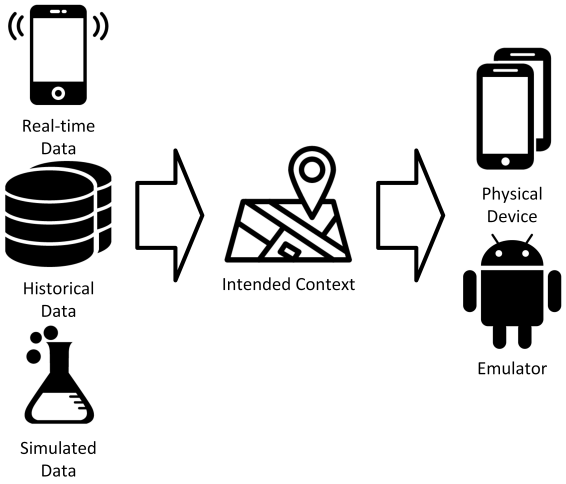

Figure 1: Data integration can assist developers to construct intended context in prototype testing.

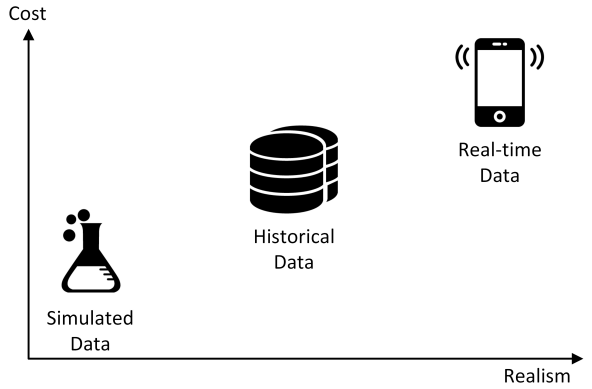

Figure 2: The comparison of data sources in data integration.

For a mobile crowdsourcing application, developers often have to validate its design in different usage scenarios. With suitable data integration, developers can build the intended context of these scenarios to conduct testing in relevant environments (i.e., physical device and/or emulators). Furthermore, developers, with the help of the testing tool, need to consider the cases where certain types of data are missing. Rather than crashing or generating erroneous results, robust mobile crowdsourcing systems should be aware of these cases.

Implementation Example

From literature, we can see that a number of techniques and tools can realise the idea of data integration. In this section, we take AWARE [5] and ContextSimulator [4] as examples to implement such an idea.

Real-Time Data Collection

Although Android SDK provides APIs for sensor data collection, it requires numerous efforts for developers

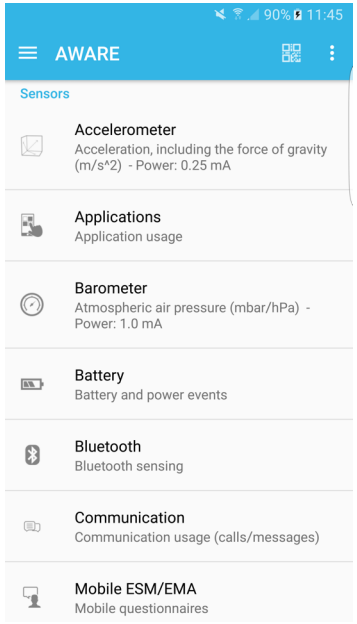

Figure 3: The AWARE interface on a smartphone.

to merge various sensors into one application. Instead, AWARE works as a hub to manage all the sensors, as shown in Figure 3. It can collect and store sensor data from three categories:

1. Hardware including motion sensors (e.g., accelerometer), location sensors (e.g., location from GPS or cellular tower), environmental sensors (e.g., barometer), and multimedia modules (e.g., microphone and dual-cameras).
2. Software including operating system information (e.g. cellular data usage) and application data (e.g. application notifications).
3. Human input that users manually produce using smartphone-based surveys and the Experience Sampling Method [8].

Note that a crowdsourcing task is often completed through human input. Developers can launch crowdsourcing questions using AWARE's ESM function.

Historical Data Record and Replay

Since AWARE can also store sensor data, we can reuse such data as historical data in future replay. This feature leads to the other tool, ContextSimulator. Developers can fetch historical data by specifying an AWARE database in ContextSimulator. To replay the data, developers must set replay speed, the start timestamp and device of data. Figure 4 shows the example code to start replaying historical data in ContextSimulator.

```
DataSource ds = new MysqlDataSource("localhost", 3306,
    "root", "toor", "aware");
AwareSimulator sim = new AwareSimulator(ds, 1391062684000L,
    UUID.fromString("92d47d9d-a600-4309-b340-b58314c2e429"));
sim.setSpeed(1000.0);
sim.applicationsForeground.addListener(
    new AwareSimulator.Listener<ApplicationsForeground>() {
    public void onEvent(ApplicationsForeground event) {
        System.out.println(event.applicationName);}});
sim.start();
```

Figure 4: The example code provided by ContextSimulator to replay data from an AWARE database.

Data Manipulation

To achieve data manipulation easily, we create a dummy application based on AWARE. Developers can generate simulated data by inputting values, such as application crashes (Figure 5), into this dummy application. Then this application automatically uploads the data to its AWARE database. During testing, developers can replay such data using ContextSimulator, as illustrated in the previous section.

```
public void createApplicationsCrashes(long timestamp, String device_id, String package_name,
                                       String application_name, long application_version,
                                       String error_short, String error_long,
                                       int error_condition, int is_system_app) {
    ContentValues data = new ContentValues();
    data.put(Dummy_AWARE_Data_Applications_Crashes.TIMESTAMP, timestamp);
    data.put(Dummy_AWARE_Data_Applications_Crashes.DEVICE_ID, device_id);
    data.put(Dummy_AWARE_Data_Applications_Crashes.PACKAGE_NAME, package_name);
    data.put(Dummy_AWARE_Data_Applications_Crashes.APPLICATION_NAME, application_name);
    data.put(Dummy_AWARE_Data_Applications_Crashes.APPLICATION_VERSION, application_version);
    data.put(Dummy_AWARE_Data_Applications_Crashes.ERROR_SHORT, error_short);
    data.put(Dummy_AWARE_Data_Applications_Crashes.ERROR_LONG, error_long);
    data.put(Dummy_AWARE_Data_Applications_Crashes.ERROR_CONDITION, error_condition);
    data.put(Dummy_AWARE_Data_Applications_Crashes.IS_SYSTEM_APP, is_system_app);
    Intent applications_crashes = new Intent();
    applications_crashes.setAction(ACTION_AWARE_PLUGIN_DUMMYAWARE);
    applications_crashes.putExtra(EXTRA_DATA,data);
    sendBroadcast(applications_crashes);
    getContentResolver().insert(Dummy_AWARE_Data_Applications_Crashes.CONTENT_URI, data);}
```

Figure 5: The function to create and save application crash information.

Conclusion and Future Work

We present a data integration technique for developers to validate their design in prototype testing. This approach can construct the intended context by combining real-time, historical and simulated data. With high confidence in context-awareness, mobile crowdsourcing systems can present crowdsourcing questions in relevant context to users which improves response quality. In future work, we plan to build a middleware as a testing tool for this approach. It will work on Android mobile devices and emulators.

Acknowledgements

This work is partially funded by the Academy of Finland (Grants 276786-AWARE, 285062-iCYCLE, 286386-CPDSS, 285459-iSCIENCE), European Commission (Grants PCIG11-GA-2012-322138, 645706-GRAGE, and 6AIKA-A71143-AKAI) and University of Oulu (Grants ITEE-2016-SA-13, and ITEE-2016-SA-20).

References

1. Application Exerciser Monkey. Retrieved 17/12/2014 from http://developer.android.com/tools/help/monkey.html
2. Larry Bernstein. Foreword: Importance of software prototyping. *Journal of Systems Integration* 6, 1-2: 9-14. http://dx.doi.org/10.1007/BF02262748.
3. Szymon Bobek, Sebastian Dziadzio, Paweł Jaciów, Mateusz Ślażyński and Grzegorz J. Nalepa. 2015. *Understanding Context with ContextViewer – Tool for Visualization and Initial Preprocessing of Mobile Sensors Data.* Springer International Publishing.
4. ContextSimulator. Retrieved 18/05/2016 from http://glados.kis.agh.edu.pl/doku.php?id=pub:software:contextsimulator:start
5. Denzil Ferreira, Vassilis Kostakos and Anind K. Dey. 2015. AWARE: mobile context instrumentation framework. *Frontiers in ICT* 2, 6: 1-9.
6. Lorenzo Gomez, Iulian Neamtiu, Tanzirul Azim and Todd Millstein. 2013. RERAN: Timing- and touch-sensitive record and replay for Android. *International Conference on Software Engineering (ICSE)*: 72-81.
7. Reed Larson and Mihaly Csikszentmihalyi. 1983. The Experience Sampling Method. In *Flow and the Foundations of Positive Psychology* (eds.). Wiley Jossey-Bass, San Francisco, 15, 41-56.
8. Reed Larson and Mihaly Csikszentmihalyi. 2014. *The Experience Sampling Method.* Springer Netherlands.
9. monkeyrunner | Android Developers. Retrieved 03/05/2016 from http://developer.android.com/tools/help/monkeyrunner_concepts.html
10. Zhengrui Qin, Yutao Tang, Ed Novak and Qun Li. 2016. MobiPlay: A Remote Execution Based Record-and-replay Tool for Mobile Applications. In *Proceedings of the 38th International Conference on Software Engineering*, ACM, 571-582.
11. Daniele Riboni and Claudio Bettini. 2009. *Context-Aware Activity Recognition through a Combination of Ontological and Statistical Reasoning.* Springer Berlin Heidelberg.
12. Ralf Tonjes, Eike S. Reetz, Marten Fischer and Daniel Kuemper. 2015. Automated Testing of Context-Aware Applications. *Vehicular Technology Conference (VTC Fall)*: 1-5.
13. Niels van Berkel, Chu Luo, Denzil Ferreira, Jorge Goncalves and Vassilis Kostakos. 2015. The Curse of Quantified-Self: An Endless Quest for Answers. In *Adjunct Proceedings of International Joint Conference on Pervasive and Ubiquitous Computing Adjunct*, 973-978. http://dx.doi.org/10.1145/2800835.2800946.