An Android Toolkit for Supporting Field Studies on Mobile Devices

Clemens Holzmann
University of Applied Sciences Upper Austria
Department of Mobility & Energy
Softwarepark 11, 4232
Hagenberg, Austria
clemens.holzmann@fh-hagenberg.at

Andreas Riegler
University of Applied Sciences Upper Austria
Department of Mobility & Energy
Softwarepark 11, 4232
Hagenberg, Austria
andreas.riegler@fh-hagenberg.at

Dustin Steiner
University of Applied Sciences Upper Austria
Department of Mobility & Energy
Softwarepark 11, 4232
Hagenberg, Austria
dustin.steiner@fh-hagenberg.at

Christian Grossauer
University of Applied Sciences Upper Austria
Department of Mobility & Energy
Softwarepark 11, 4232
Hagenberg, Austria
christian.grossauer@fh-hagenberg.at

MUM 2017, November 26–29, 2017, Stuttgart, Germany
ACM 978-1-4503-5378-6/17/11.
https://doi.org/10.1145/3152832.3157814

Abstract
Evaluating mobile user interfaces in the field is a time-consuming and cumbersome task. In order to figure out how users interact with mobile apps or devices over an extended period of time, an automated logging of device usage and context information is necessary. In this paper, we present an Android app called automate toolkit for logging such data across arbitrary apps in a convenient and customizable way. It allows to trace usage information like visited screens and performed gestures as well as information about the context of use like device orientation and light conditions. The data is stored on the device for further analysis with statistical software. We made the toolkit available as open source software in order to support developers, designers and researchers in conducting field studies on Android devices.

Author Keywords
Field study; usability evaluation; software toolkit; Android app.

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

Introduction

Conducting user tests with the aim to improve the usability of mobile applications is a critical competitive factor in this fast-moving market. Especially the evaluation of prototypes can show valuable measures about the quality of mobile applications, which helps designers and developers in finding possible improvements or errors that should be fixed. Users are observed how they use a specific application, in order to calculate usability metrics such as the percentage of successfully executed tasks, the time required to perform certain tasks or the frequency of errors, such as those caused by wrong entries or navigation problems.

In addition to the necessary expertise, primarily technical obstacles such as the heterogeneity of mobile platforms and the high resource requirements (time required, number of subjects, extensive infrastructure, etc.) are reasons why usability testing of mobile applications can be found mostly in large companies. Last but not least, it is the current lack of techniques and methods on mobile platforms which avoids a flexible and efficient way of testing in the field under real-world conditions. Furthermore, commercial evaluation tools only provide insufficient information about usability as they tend to only consider commercial key figures regarding user loyalty, purchases or demographics.

We present a novel way to automatically collect relevant usability-related data of mobile applications that is able to log all user interactions as well as their context of use during the entire lifecycle of the application unter test. As a major improvement to existing frameworks, our toolkit can be used without changing the application source code, which makes it more flexible and scalable. With the logged data, our toolkit is able to detect issues such as unused functions and navigational problems under certain contextual circumstances.

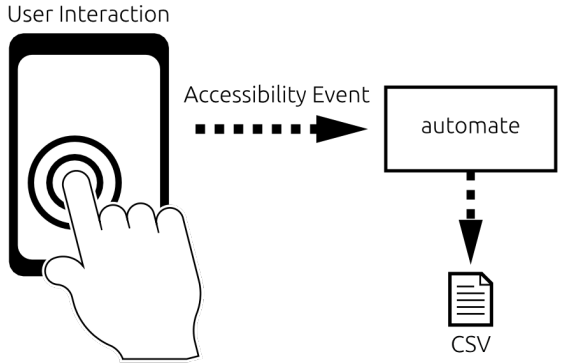

Figure 1: A simple overview of the automate toolkit.

The introduced toolkit for Android is a logging framework which allows to track mobile device usage and context information across arbitrary Android apps in a convenient and customizable way. A variety of different kinds of data can be tracked, like e.g. user interactions on widgets, interaction types (tap, scale, fling etc.), device information (device type, carrier name, OS information etc.) and also some context information (language, current location etc.). The app can be used for starting and stopping the tracking process, changing the settings and triggering the file export. The collected data will be solely stored locally in CSV files which can be accessed using the built in export function.

In order to collect and log the usage data, the users just have to install the Android app on their device and grant rights to run as an accessibility service. The data logged by the app will be stored locally on the device in comma-separated values (CSV) files. Figure 1 shows this process.

Related Work

Identifying interactions of users when they open an app on their mobile device is a key issue in mobile usability analysis. Usability evaluation is an essential aspect for application development. However, in the mobile domain, collecting the relevant data is different from traditional desktop environments. Moreover, evaluating mobile user interfaces is done in a platform-specific way. [2]

In the following, several projects that contribute to the work within this paper are described and related to our approach.

In [4], the authors also used Android's accessibility service to retrieve data from the device, although their approach focuses on visual data (screenshots) rather than on interaction data. Each screenshot is analyzed based on a set of metrics related to the balance and density of elements, used colors, typography and consistency.

Ma et al. [3] propose a toolkit for automatically capturing user interactions in mobile applications. The collected data is sent to a remote server for either automated or manual usability analysis. However, access to the application's source code is needed, as the toolkit has to be integrated into the app in order to capture user interactions and upload them to the remote server.

Balagtas-Fernandez et al. [1] propose an Android-based logging framework. It should simplify usability evaluations on mobile devices by collecting a variety of usability-related metrics. The logging framework must also be included into the target app's source code as the proposed approach involves a manual coding of logging calls in the application.

AWARE[1] is an Android framework for the purpose of instrumenting, inferring, logging and sharing mobile context information. The main purpose of the AWARE framework is to collect data from a series of sensors on the mobile device and infer context information from them. The framework provides plugins to detect contexts such as conversations and ambient noise. The main difference to our proposed toolkit is that AWARE focuses on presenting and explaining context information, while our toolkit emphasizes on the logging of interactions such as opened apps, visited pages and number of interactions per page.

There are also commercial products such as Flurry Analytics[2] and Localytics[3]. Such commercial frameworks intend to get a deep audience insight. They provide statistics based on metrics like average active users per day, new users per week or user loyalty. The framework code must be integrated with the existing target application. The developers are responsible for using the framework's code snippets at the right place in their applications.

The Automate Toolkit

Our toolkit tracks data by acting as an accessibility service and extracting the needed information from incoming accessibility events. The idea behind the toolkit is to help researchers or mobile app developers to understand how users are interacting with the app. This is possible without any modifications of the source code (e.g. by adding logging code) by using Android's accessibility service. The accessibility features have to be activated manually by the user. This ensured that accessibility events are only recorded with the knowledge of the users.

The automate toolkit can easily be configured through the automate app. This app displays the current status of the tracking process and can be used to activate or deactivate

[1] http://www.awareframework.com

[2] https://y.flurry.com

[3] https://www.localytics.com

Tracked Data:

App Usage Data
- Visited pages ("screens" within an app)
- Dwell times (per page)
- Number of interactions (per page)
- Device orientation

Interaction Data
- Type of interaction (touch, scroll, long touch)
- Interaction target

Network Data
- Network type
- Network subtype
- Roaming status

Device Data
- Device description
- Country of origin
- Language
- Network operator
- Operating system
- Screen resolution

Battery Data
- Battery level
- Charging status (charging/discharging)
- Temperature
- Voltage

Context Data
- Light condition

Orientation Data
- Device orientation

the tracking of certain types of data (e.g. number of interactions, used apps, light sensor data, etc.). Since not all recorded data might be needed by the app developers, certain data types can be ignored to be captured.

The automate toolkit is capable of tracking mobile device usage and context information. Which kind of data is tracked can be customized in the toolkit app. All data that is recorded by automate is shown in the sidebar of this page.

Furthermore, the system architecture allows for an easy extension of functionalities. The main components are responsible for starting/stopping the tracking process, receiving accessibility events, extracting the necessary data and storing it locally in CSV files. The processing of the data is handled by so called managers. Each manager handles a certain kind of related data (e.g. the device info manager handles the tracking of the set language and the country of origin).

Accessibility conformant apps are sending accessibility events that are triggered by user interactions. The automate toolkit is listening to these events and distributes them across the registered managers. If data like the GPS location or the surrounding noise should be recorded as well, a new manager can easily be added. This process is visualized in Figure 2.

Figure 3 shows the architecture of the automate toolkit, which is split into five repositories, in more detail.

Managers

Managers are used to gather data (e.g. device or interaction information etc.) and convert them into different kinds of structured data that can be analysed later. In order to get access to the data, a manager or a couple of them have a corresponding file export handler, which will create struc-

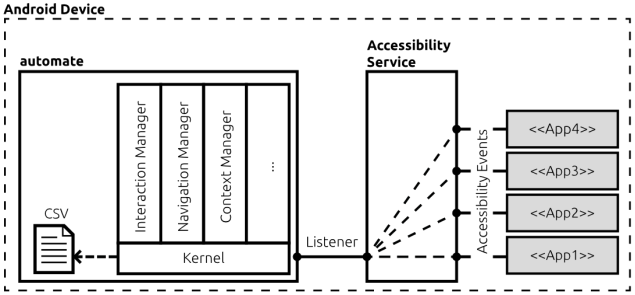

Figure 2: The workflow of collecting usability data using the automate toolkit.

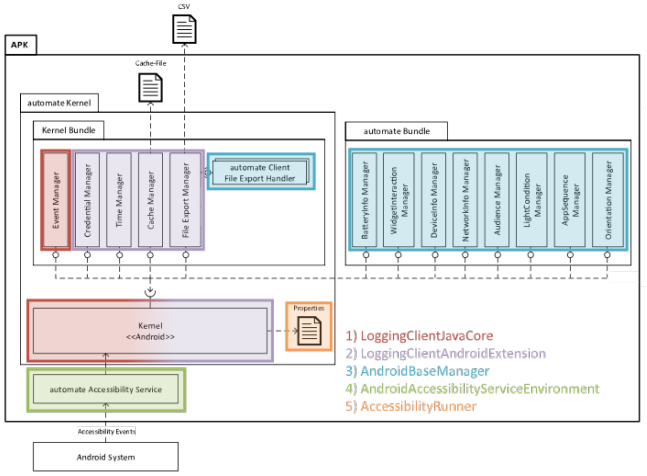

Figure 3: Components of the automate toolkit.

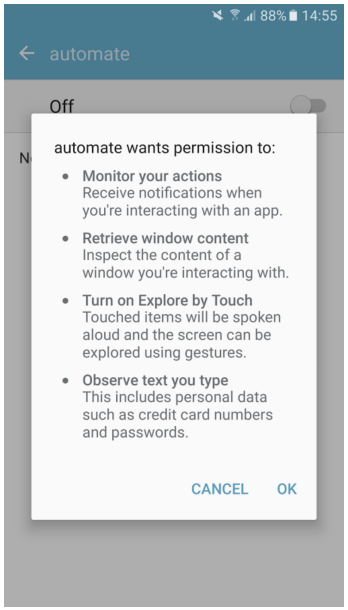

Figure 4: Accessibility service permission request dialog.

tured CSV files from the internal data representation. The following sections go into the details of the managers that are already included in the automate toolkit, including information about the gathered data and its structure. All CSV files have a number of general columns (deviceId, sessionId, sequenceNr, projectId and appVersion), which are described in the following.

App Sequence Manager Listens for events of app screen visit changes and app interactions (clicks, scrolls etc.), and creates a structured XML document out of one session. A session starts when the screen is turned on and the device returns from sleep mode, and it lasts until the device returns to sleep mode again (which usually means the display is off again) or when the device is turned off completely.

Listing 1: XML structure of an app session.

```
<session>
    <appUsage packageName="..." name="..."
        startTime="...">
    <state name="..." className="..."
        duration="..." interactionCount="..."
        orientation="..."/>
    </appUsage>
</session>
```

Listing 1 shows the basic XML structure of an app session created by the App Sequence Manager. Every time an app is opened, there is an *appUsage* tag with the *packageName* of its package and the *startTime* as a timestamp. Inside every *appUsage* there is a list of state tags which stand for a change in the view (usually and Android activity). They include the *name* and *className* of the view if returned by the Android accessibility environment as well as the *duration*, device *orientation* and *interactionCount* in that state.

Widget Interaction Manager Collects the different interaction events for the current screen. A screen is usually a view like the Android activity. Interactions are characterized by the interaction type (click, scroll, long click, context click, selected), class name and text, as well as a content description if available. Additional information are the time when the interaction occurred as well as the screen bounds of the widget that was interacted on. Due to restrictions on Android, some data may be missing.

Network Info Manager This manager waits for changes to the network connection and updates the information accordingly. The relevant data is the network type and subtype that Android reports as well as a flag if the network connection is roaming or not.

Device Info Manager This manager collects the basic device information that rarely changes, like device name and carrier, OS version, resolution and locale.

Battery Info Manager Whenever the Android battery system manager reports changes to the battery status, this manager compares and updates the stored data.

Light Condition Manager This manager records changes to the classification of the light condition based on an article by R. Soneira[4].

Orientation Change Manager This manager stores the amount of time the device was in one of the possible screen orientations.

[4] http://www.displaymate.com/Smartphone_Brightness_ShootOut_1.htm

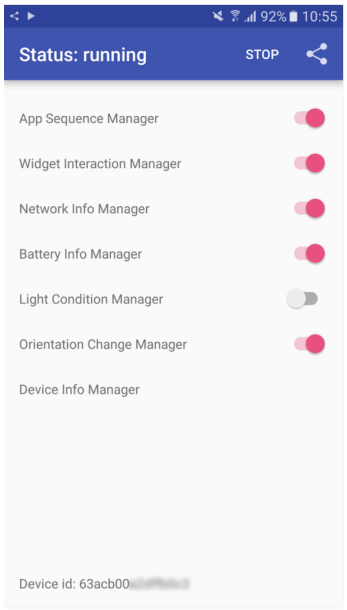

Figure 5: Settings screen of the automate application.

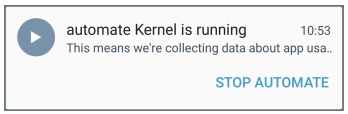

Figure 6: Notification view while the automate framework is tracking data in the background.

Using the Toolkit

The automate toolkit has been published as open source project under the GNU General Public License (GPL) version 3[5] (or later). It is hosted on GitHub[6] and contains everything needed for tracking mobile device usage and context information on Android phones. The automate toolkit app, available via Google Play[7], provides a convenient and customizable way to track the usage of arbitrary applications on Android.

In order to use the toolkit, the following steps are involved:

1. **Install the APK file** from Google Play.
2. **Enable the accessibility service** when the app is started for the first time. The accessibility service option has to be activated in the Android settings to allow the automate service to receive accessibility events. Figure 4 shows the permission dialog that must be accepted in order to let the automate framework track the device usage data. After the permissions are granted, the tracking starts automatically and can be modified in the tracking settings.
3. **Modify settings**, such as enabling and disabling the various managers. Figure 5 shows the settings screen where it can be customized what types of data should be tracked. On the bottom, the individual device ID is displayed. Each manager can be enabled and disabled at any time. The whole tracking process can also be stopped/resumed anytime by pressing the start/stop button of the top bar. Additionally, if the tracking is in process, the notification bar shows quick access settings as displayed in Figure 6.
4. **Interact with the device** When the tracking is running, automate automatically collects the device usage and context data in the background, depending on the chosen configuration.
5. **View tracked data** The tracked data can be exported as a ZIP archive of CSV files. Depending on the size of the data, the export process can take some time. The ZIP archive only contains tracked data since the last file export (or since the initial start).

[5] https://www.gnu.org/licenses/gpl.html
[6] http://mint-hagenberg.github.io/automate-documentation/
[7] https://play.google.com/store/apps/details?id=at.fhhagenberg.mint.automate.accessibilityrunner

Conclusions

We presented a toolkit for automating the logging of mobile device usage and context information on Android, which we consider especially useful for field studies. It captures data related to user interactions as well as their context of use, works across arbitrary apps and does not require any instrumentation of the application source code. The toolkit is available as open source software for Android.

Acknowledgements

The research presented has been conducted within the Austrian project “AUToMAte - Automatic Usability Testing of Mobile Applications” funded by the Austrian Research Promotion Agency (FFG) under contract number 839094.

REFERENCES

1. Florence Balagtas-Fernandez and Heinrich Hussmann. 2009. A Methodology and Framework to Simplify Usability Analysis of Mobile Applications. In *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering (ASE '09)*. IEEE Computer Society, Washington, DC, USA,

520–524. DOI:
http://dx.doi.org/10.1109/ASE.2009.12

2. Florian Lettner, Christian Grossauer, and Clemens Holzmann. 2014. Mobile Interaction Analysis: Towards a Novel Concept for Interaction Sequence Mining. In *Proceedings of the 16th International Conference on Human-computer Interaction with Mobile Devices anxd Services (MobileHCI '14)*. ACM, New York, NY, USA, 359–368. DOI:
http://dx.doi.org/10.1145/2628363.2628384

3. Xiaoxiao Ma, Bo Yan, Guanling Chen, Chunhui Zhang, Ke Huang, Jill Drury, and Linzhang Wang. 2013. Design and Implementation of a Toolkit for Usability Testing of Mobile Apps. *Mob. Netw. Appl.* 18, 1 (Feb. 2013), 81–97. DOI:
http://dx.doi.org/10.1007/s11036-012-0421-z

4. Andreas Riegler and Clemens Holzmann. 2015. UI-CAT: Calculating User Interface Complexity Metrics for Mobile Applications. In *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia (MUM '15)*. ACM, New York, NY, USA, 390–394. DOI:
http://dx.doi.org/10.1145/2836041.2841214