# Extending Mobile App Analytics for Usability Test Logging

Xavier Ferre[1,2(✉)], Elena Villalba[2], Héctor Julio[2], and Hongming Zhu[3]

[1] High-end Expert at Tongji University, Shanghai, China
xavier.ferre@upm.es
[2] DLSIIS, Universidad Politecnica de Madrid, Madrid, Spain
evillalba@fi.upm.es, h.julio@icloud.com
[3] School of Software Engineering, Tongji University, Shanghai, China
zhu_hongming@tongji.edu.cn

**Abstract.** Mobile application development is characterized by reduced development cycles and high time-to-market pressure. Usability evaluation in mobile applications calls for the application of cost-effective methods, specially adapted to such constraints. We propose extending the Google Analytics for Mobile Applications basic service to store specific low-level user actions of interest for usability evaluation purposes. The solution can serve both for lab usability testing, automating quantitative data gathering, and for logging real use after application release. It is based on identification of relevant user tasks and the detailed events worth gathering, instrumentation of specific code for data gathering, and subsequent data extraction for calculating relevant usability–related variables. We validated our application in a real usability test by comparing the automatically gathered data with the information gathered by the human observer. Results shows both measurements are statistically exchangeable, opening promising new ways to perform usability testing cost-effectively and at greater scale.

**Keywords:** Automated usability evaluation · Usability testing · Log file analysis · Usability evaluation of mobile applications

## 1 Introduction

Smartphones include the most advanced computing technology and are available to a growing user base. They combine a variety of sensors, high-speed Internet connection, good quality camera and touch-screen, offering additional possibilities for application developers. In 2015, 1.43 billion smartphones were shipped worldwide [1], while PC shipments amounted to a total of just 249 million [2]. Mobile apps give users the opportunity to carry out all sorts of activities on the move. In fact, they are coming to be the device that a rapidly growing user base chooses in order to use software and access the Internet. This is especially true in developing countries, where smartphones are the first Internet-connected computing device that a high number of citizens will have access to.

R. Bernhaupt et al. (Eds.): INTERACT 2017, Part III, LNCS 10515, pp. 114–131, 2017.
DOI: 10.1007/978-3-319-67687-6_9

There are native and web-based mobile apps. Native apps are developed directly for a given platform, and, according to a mobile developer survey [3], they are developers' preferred option, whereas the use of web-based apps is declining [4]. There is a third option, hybrid apps, which wrap web content with native code. However, the main apps on the market are moving away from the hybrid to the native approach [5]. We will focus on native mobile apps in this paper. Henceforth, the term 'mobile app' refers to 'native mobile application'.

Usability and user experience (UX) are quality attributes with special relevance for mobile development, since competition in the app market is fierce: the two main mobile app stores, App Store for iOS and Google Play for Android, offer 2 and 2.2 million apps respectively [6]. Mobile software development teams face the challenge of this dynamic environment [7], with reduced time-to-market schedules and a continuous delivery paradigm to change and improve the app in order to get better user retention. Low-quality apps can have devastating consequences for mobile app development companies. As a result, there is a strong accent on usability testing [5].

User-centered design (UCD) is the approach applied to achieve a good level of usability in software systems [8]. There are a variety of UCD methods for achieving a good usability level (Ferre identified 95 different methods from six books [9]), most of them focused on usability evaluation. A careful selection of usability evaluation methods is essential in this field, as it faces the challenge of the changing and varied contexts of use: a mobile app can be used inside a building or outside under a very bright sunlight, with either no ambient noise or in a busy environment, and, possibly, the tasks can be interrupted and resumed several times before fulfilling the user objectives.

Traditional lab-based usability testing can offer valuable insight into possible usability problems in a pre-release stage, but it cannot possibly cover aspects of UX related to the changing contexts of use. Alternatively, field studies can offer rich and relevant findings [10], despite the challenge of measuring usability under real-life conditions in the mobile arena [11]. But the cost of field testing at a pre-release stage may be too high for average mobile development projects. Within this ongoing debate lab vs. field studies in the mobile usability evaluation community, we believe that field testing after release offers a compromise solution, and a complement to pre-release lab testing. Our proposal aims to cover these two evaluation approaches, with code instrumentation that serves both targets.

Field studies with selected users can consume a lot of resources. Automated usability evaluation is a potentially good alternative, offering the possibility to gather information from actual app usage. In particular, data capture for automated usability evaluation involves using software that automatically records usability data (e.g., logging UI usage) [12]. While usability testing provides timely feedback about user reactions to a new design in a controlled environment, automated usability logging can complement such basic method with evaluation in the actual context of use, shortening at the same time usability evaluation activities in the continuous delivery effort for mobile app development.

Google Analytics is a widely used cloud service offered by Google that tracks and reports website traffic, behaviors and purchases. It provides a limited web tool to access the reports and tracking data, but it also offers a public API to automate complex reporting and configuration. Such API can be used to extract the raw data and create custom reports, with some limitations in its Standard (free) version. This powerful tool is primarily designed for marketing purposes, such as the measurement of ad campaign success or ad revenue maximization. It also aims to measure UX aspects like user engagement, but it requires careful tweaking if it is to be useful for full-scale usability evaluation. In 2009, Google released a variant of this service specifically addressed to mobile apps: Google Analytics for Mobile Applications (GAMA) [13, 14].

We propose a logging mechanism for user actions that can serve both for usability testing and for continuous usability logging, based on GAMA. Our model is based on task modeling to establish the UI events to be tracked, automating the data capture side of usability evaluation, whereas the gathered data are analyzed and critiqued by usability experts.

In this paper, we present our logging mechanism and its validation in a lab usability test of a real application (available in both Google Play & iOS App Store).

Section 2 presents existing literature on the issue. Section 3 details the overall logging approach: (1) how task modeling and usability goal setting define the UI events to track; (2) the logging strategy injected into the application code; and (3) how lab usability testing is planned and the results of tests are interpreted. Section 4 details the results of the validation, where our proposed method has been applied for a usability test. Finally, Sect. 5 discusses the results obtained; and Sect. 6 outlines the conclusions and future lines of research.

## 2 Related Work

Usability evaluation is a cornerstone of the UCD process. In particular, according to ISO 9241-210 [8], feedback from users during operational use identifies long-term issues and provides input for future design.

Ivory and Hearst [12] presented the state of the art of automated usability evaluation back in 2001, classifying automated usability evaluation techniques into a taxonomy. They highlighted the fact that automation within usability testing has been used predominantly in two ways: automated capture of use data and automated analysis of these data according to some metrics or a model (log file analysis in Ivory and Hearst's terminology). They differentiated between methods for WIMP (windows, icons, pointer, and mouse) and for Web interfaces, but they did not take into account mobile interfaces. The first fully-fledged smartphone, the original iPhone, appeared in 2007. As Ivory and Hearst's survey was published well before this modern kind of mobile app came into being, it does not account for the peculiarities of this kind of software applications.

Automated website usability evaluation [15, 16] has been extensively applied to extract usability-relevant information from website logs. The reason is that web servers automatically log every single page access, creating a wealth of information for usability purposes. On the contrary, user events in mobile applications are not typically logged, except for some basic information gathered by GAMA for any mobile app using its services. Therefore, for a similar level of automated usability evaluation as in the web domain, additional explicit code instrumentation is needed. From the existing web mining strategies, our proposal goes in line with the web usage mining approach, where the steps are identification of users, identification of single sessions, and identification of the navigation paths within these sessions [15].

Every mobile platform offers a different UX [5, 17], and mobile UX has distinct features from the desktop experience [18]. Mobile usability and UX evaluation requires specific focus on issues like the changing context, smaller screen size, etc. For example, the number of clicks necessary to fulfill a task is very relevant as a usability attribute in the web domain, while number of taps is comparatively less important for mobile users (due to the much faster response times in mobile apps vs websites). For these reasons (among others), automated usability evaluation of mobile apps, while taking advantage of the wealth of experience distilled from years of application in the web and WIMP domains, cannot directly apply it to the mobile domain. It is necessary to carefully consider how traditional usability evaluation methods can be applied to mobile app development.

Paternò et al. [19] present an extension addressed to mobile apps of an existing tool that remotely processes multimodal user interaction in desktop applications. This extension allows remote evaluation of mobile applications, tracking both user actions and environmental conditions. It is based on a comparison of planned and actual user behavior, using CTT (Concurrent Task Trees) for task modeling.

Carta et al. offer a tool supporting remote usability evaluation of mobile websites [20]. They dismiss Google Analytics as being limited by the number of events that can be captured, which our approach has extended by means of custom dimensions (see Sect. 3.2.2). Carta et al. focus exclusively on the webpage interaction style, accounting mainly for page loads, whereas native mobile apps use a different method of interaction.

There exist a lot of services for automatic website usability evaluation, but only a handful for evaluation of native mobile apps in Android and iOS. Most of the latter ones, like validately.com or lookback.io, are based in video-recording of testing sessions, and they only offer remote usability testing (not evaluation of free use).

Balagtas-Fernandez and Hussmann present their EvaHelper tool and framework for collecting usability data in mobile apps [21]. It is based on generating usage logs stored in the mobile device. These logs are then processed to generate navigation graphs that can identify usability problems. Balagtas-Fernandez and Hussmann focus on the work of the developer to ensure adequate UI instrumentation. Kluth et al. [22] extend the Balagtas-Fernandez and Hussmann four-stage model considering an additional critique stage.

Lettner and Holzmann propose a toolkit for unsupervised user interaction logging in mobile apps oriented to usability evaluation [11]. They log user navigation between screens and on-screen interaction which is sent to a cloud-based web server for further analysis. They combine low-level metrics, like button-clicks, or session times, with device statistics. In this sense, their solution could have been applied using GAMA. But Lettner and Holzmann acknowledge that their approach has the limitation of not having pre-defined tasks.

Feijó et al. [23] propose a toolkit for automated emotion logging for mobile applications. It is able to trace emotional reactions of users during usability tests and relate them to the specific interaction that is taking place. These reactions are collected using the front camera of the mobile device.

Leichtenstern et al. [24] follow a similar approach, with their EVAL tool that records video-logs of users using a mobile application, and relates the videos to the specific screen that was being used at that time.

Porat et al. [25] present the MATE tool to support usability experts in the definition of the task scenarios to use in a usability test, allowing the combination of results of different kinds of usability testing methods across devices and operating systems. It is focused on performance metrics that they categorize as efficiency, effectiveness and frequency. The usability experts they have used for evaluation emphasize the need to particularize such metrics for each project.

According to Hulo and To [26], Sony has been using GAMA for some of their decoupled applications. Hulo and To's work extends GAMA basic service with a UsageTracker service that operates in the background to log UI events in the Telephone app of a Sony mobile phone. What they consider tasks are actually events or individual steps to log, therefore lacking a proper task analysis.

Of these proposals, only [21] and [22] actually record user events in a similar way to our approach. The EvaHelper tool [21] is very similar in terms of preparation difficulty to our approach, since it is necessary to include calls to an API. The gain in using GAMA as logging server is that it is very common that app development companies already use GAMA for its benefits in terms of marketing issues. Therefore, if GAMA is already used, the extra burden of using our instrumentation for specific usability-related event logging is lighter than when using an alternative API that requires handling the logging mechanism itself (starting the logging activity in the app, making sure the server is up and running, etc.).

About the second work, Kulth et al. [22] present a capture framework for just iOS, with no easy extension to Android, since it is based on specific characteristics of Objective-C programming language.

In short, while there exist other automated logging methods for desktop and web-based systems, to the best of our knowledge there are only 2 methods published for native mobile apps, and they either offer additional burden to our GAMA proposed approach, or they have not been applied to Android apps.

## 3 Our Logging Approach

Our approach for automated UI logging to support usability tests is based on considering tasks at the beginning of the process, and focusing the usability-related instrumentation and analysis on such tasks expressed as detailed use cases. This is combined with the usage of GAMA for the collection of relevant user events.

Our proposal is based on a three-stage approach (see Fig. 1):

1. **Identification of the user tasks of interest to be evaluated, task modeling and selection of the specific events to log in each task:** As part of the User & Task Analysis activity, user tasks are identified and prioritized, selecting the most relevant ones and building their task models. Such models are linked to specific

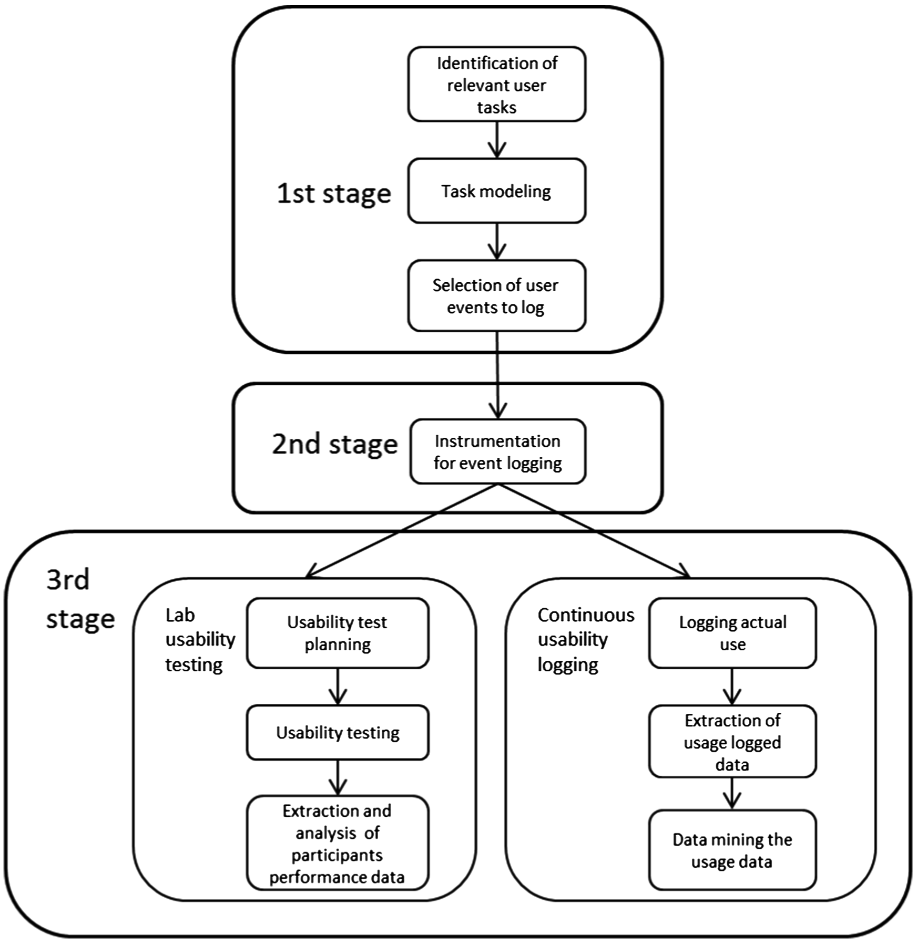
Identification of relevant user tasks
Task modeling
Selection of user events to log
1st stage
2nd stage
Instrumentation for event logging
3rd stage
Lab usability testing
Usability test planning
Usability testing
Extraction and analysis of participants performance data
Continuous usability logging
Logging actual use
Extraction of usage logged data
Data mining the usage data

**Fig. 1.** Stages of the proposed logging approach

elements in the User Interface (UI). The usability evaluation strategy needs to be materialized in this stage, with the selection of specific events to be tracked through the logging mechanism.

2. **Instrumentation for event logging:** Implementation of the data gathering functionality in the app code to ensure that the selected events are logged in the GAMA servers. The UI is instrumented with calls to functions of the GAMA API (Application Programming Interface), sending the usability-relevant data gathered from user actions.
3. **Usability evaluation**
   a. **Lab usability testing:** First, it is necessary to plan the usability tests with specific instructions for test facilitators, so that each user task enactment is recorded and the User Unique IDentifier (UUID) is stored. After we carry out the usability test, we need to extract performance data from GAMA servers, differentiating the information corresponding to each test participant.

b. **Continuous usability logging:** As users use the app, GAMA gathers information such as device model and operating system version, duration of user sessions, country, etc. Along with this general GAMA data, the code instrumentation also logs the specific user actions chosen, with no extra effort by the development team. For every period we want to analyze, we extract the log data to analyze it through the application of data mining techniques. We aim to obtain usage patterns that can be contrasted with the expected usage, uncovering possible usability problems that need to be investigated further.

Analyzing the gathered data provides detailed information on user performance and task execution paths, to serve as basis for identification of usability problems. Such identification requires a certain degree of usability expertise, possibly calling for the application of complementary usability evaluation methods to more precisely identify the problems and find a way to fix them.

If we consider usability testing alone, it would not be necessary to store information in GAMA servers. Considering the typically low number of users participating in a usability test, a more modest approach would be probably enough, storing usage logs locally in the phone (as in [21]). But we aim to have a UI logging mechanism that serves both for usability testing and for evaluation of the app real usage. Even if we describe in this paper a validation of just the application to lab usability testing, the UI logging implementation can also serve for continuous usability logging, as mentioned above.

## 3.1 First Stage: Task Modeling and Selection of User Events

Usability needs to be tackled throughout the software development cycle, starting with an early focus on users and tasks [27]. Usability testing requires a previous user & task analysis that defines the intended users, their goals and the tasks they carry out to fulfill such goals.

Once tasks have been identified, they can be modeled to serve as basis for the interaction design, that is, the design of the information exchange between the user and the system [28], including the design of the UI. The UI is designed to help users perform the tasks of each major user profile.

Task models serve as the standard for optimal task execution from the designer point of view. When user actions depart from such path, it may be an indication of a divergence between the envisioned design model and the user mental model. The actual steps followed by a user to complete a task can be compared to the intended path of action detailed in the task model. Identified differences between both are relevant and must be studied in order to enrich the specification of the context of use and make design changes. We will call them in this paper ‘user errors’, to be in line with usability testing literature, understanding that they reflect possible design errors in any case.

Use case descriptions detail interaction design in term of user steps and system responses, and are built from the results of user and task analysis. Use cases are widely used in object-oriented design, but they also serve for UX design, according to Ivar Jacobson, the creator of the use case technique [29]. But we need UCD-oriented use cases that take into account user goals and usability, so they can serve as a basis for usability evaluation.

We have chosen use cases as technique for task modeling because they serve as an ideal bridge between HCI experts and developers. We need the detailed version of use cases (according to Constantine and Lockwood terminology [30]), so that user tasks are clearly linked to specific UI elements, and their activation can be adequately logged.

The different app screens are designed to comply with the design underlying the use cases. When changes are made in the actual UI code, use cases will have to be updated accordingly, to be able to maintain the coherence between model and code.

This stage identifies key information for usability purposes. If detailed usability goals are established, they are identified according to the tasks modeled, and they are taken as a basis for the identification of the UI elements to log. Usability test design will be then focused on measuring the variables that are described in the usability goals.

If no specific usability goals are considered, the UI elements to log are typically the ones that update the elements shown in a screen, or the ones that lead to a different screen.

Annotations are added to use cases stating the UI element that needs to be tracked, and the relevant context information to be considered. For example, it may be of interest for an application to know if the user moves repeatedly from one menu tab to another in a short period of time, because this may be a sign that the user does not know where to find the functionality that he or she is looking for. The elements to track and their interpretation in terms of usability will be specific to each project and domain. Figure 2 shows an example of an annotated use case for the app used in the validation (see Sect. 4).

## 3.2 Second Stage: Instrumentation for UI Logging

### 3.2.1 Google Analytics for Mobile Applications

The Google Analytics for Mobile Applications standard online tool offers aggregated data about screens traversed, scrolls, buttons tapped or similar actions. The GAMA services offer a report system that can serve to obtain the information of interest in a machine-readable format, but GAMA reports do not directly allow downloading any kind of information shown in the web tool. Particularly, they don't allow to download screen flow information.

Two main concepts of GAMA are dimensions and metrics. Dimensions are qualitative variables (e.g., city), while metrics are measurable quantitative values (e.g., total number of visits). We can combine at least one dimension and one metric to generate custom reports. Note that not every combination of dimensions and metrics is allowed. If we carefully choose the combination of dimensions and metrics, we can gather enough information for our purpose.

In our approach, annotated use cases serve as the basis for the instrumentation of the UI with appropriate calls to the tracking server (GAMA), providing details on key user actions, like buttons tapped or visited screens. We can use custom dimensions to include such information along with other basic information included in reports.

In any case, the screen flow information automatically tracked by GAMA is too coarse-grained for an accurate usability-focused analysis of usage interaction. We need a method for operating on aggregated data to identify unexpected patterns of usage, but we also need to be able to drill down to the log of actions carried out by specific

Use case: Search for a professor's email
User Profile: Student and Novice Student.

1. The user opens the school app.
2. The system displays the icons on the main screen.
3. The user taps on "Subjects".
   *Track user tap on the icon button.*
4. The system displays the list of subjects for the default degree[*].
5. The user taps on the degree name.
   *Track user tap on the spinner.*
6. The system displays a list with all the available degrees.
7. The user selects a degree.
   *Track user tap on a list item.*
8. The system displays the list of subjects for the chosen degree.
9. The user taps on a subject.
   *Track user tap on a list item.*
10. The system displays information about the subject.
11. The user looks for the professor section and taps on the respective email address.
    *Track user tap.*

*: Users are not directly presented with a list of degrees, to save one tap for the users interested in the default degree (the main bachelor degree in the school where the majority of users are enrolled). The user needs to tap on the degree name to change to a different degree.

**Fig. 2.** Example of annotated use case used in the validation (*annotations highlighted in italics*)

individual users. We need both kinds of information (fine-grained and coarse-grained) to be able to apply data mining processes.

### 3.2.2 GAMA for Mobility Logging

Our proposal aims to carry out UI event logging for both lab usability testing reporting and for logging actual use, and both strategies can coexist in the same project. When the lab usability test is carried out after system deployment, the logging mechanism is already submitting usage information to GAMA servers, and we will then need to tell apart the usability test participants information from the gathered information about the rest of users. User identification plays an important role in this issue.

There are two GAMA parameters that we can use in order to identify specific user actions: Client ID and User ID. Both are unique user identifiers. While the Client ID is automatically assigned by GAMA and anonymous (using an UUID), the User ID is taken from the internal user profile and can be assigned by the app (in apps with a user profile). We have chosen Client ID as user identifier for our custom reports in order to keep the usability evaluation anonymous, and thus respect users' privacy rights. But having the user uniquely identified is not enough to analyze his/her steps when traversing the application. In order to complete our purpose, we need to add a timestamp as a second custom dimension. We use the current UNIX time in milliseconds as custom timestamp.

We use Client ID and our custom timestamp, combining them with other dimensions or metrics of interest, to define GAMA custom reports, thus obtaining user behavior flows and step-by-step user action tracking. These two custom dimensions are defined in the GAMA account in order to log usability-relevant actions. Every time information is sent to GAMA servers, the Client ID is specified and a timestamp is added. Figure 3 shows sample code for this kind of submission for Android. We can use this information to order user steps, allowing us to compare and analyze the paths and behaviors of test participants.

```
Tracker mTracker
=  analytics.newTracker(R.xml.analytics_global_tracker);


String cid = mTracker.get("&cid");// Getting Client ID
String dimensionValue1 = cid;

Calendar c = Calendar.getInstance();
long ms = c.getTimeInMillis();
Log.i("TIME", Long.toString(ms));
String dimensionValue2 =  Long.toString(ms);


// Build and Send the Analytics Event.
mTracker.send(new
HitBuilders.EventBuilder().setCategory(category).setAction(action).setLa
bel(label).setValue(value).setCustomDimension(1,
dimensionValue1).setCustomDimension(2, dimensionValue2).build());
```

**Fig. 3.** Sample code for fetching Client ID and custom timestamp for the app Android version

## 3.3 Third Stage: Usability Evaluation

We will focus in this section in the application of the proposed logging mechanism just to lab usability testing, which is the option validated in the current paper.

It is typically recommended that usability testing not be carried out by a lone experimenter, but to involve more people that take care of the different roles. The two more common roles are data gatherer/note taker (to take relevant notes about user actions and errors), and time keeper (keeping track of each task starting and elapsed time) [31], and we aim to automate these two roles with the usage of our UI logging mechanism. Thus, with our approach only one usability expert is needed to take care of the usability test, acting mainly as facilitator [32].

But apart from having a proper code instrumentation, usability testing needs to be carefully planned to ensure that participants' tasks are correctly registered for further analysis.

### 3.3.1 Usability Test Planning

The tasks chosen for the usability test must have been between the ones modeled and instrumented for usage logging.

Apart from the task events instrumentation, there is a need for some special event that the test facilitator must activate when the task is about to start (for example going to the Home screen) and another special event to signal the end of the task. These actions must be logged as well, to be able to calculate the starting and ending time for each task in the subsequent test data extraction.

The test may be carried out on the participant's own mobile phone, to avoid user errors coming from lack of familiarity with the specific flavor of the operating system or device buttons. In such case, it is also necessary to access the mobile phone ClientID and register it for the later data extraction.

The test protocol must include the exact steps to be taken by the test facilitator to:

1. Install the app to test in the participant's mobile phone.
2. Ensure that the mobile phone has access to Internet, so that actions can be recorded in GAMA.
3. Register the device ClientID.
4. Indicate each task start and end.

### 3.3.2 Test and Interpretation of Logged Data

Tests are carried out, following the protocol previously established, and then the data logged needs to be extracted trough the GAMA API and analyzed.

We carry out the extraction of data through a set of Python language scripts that connect to GAMA servers and download the test data. The scripts query metrics and dimensions in a specific time interval for a ClientID.

In order to extract the path followed by the participant, the requested metric comprises all the events, and the specific dimensions are as follows:

- Our custom timestamp.
- ClientID.
- Event – Action.
- Event – Category.
- Day/Hour[1].
- Minute (See footnote 1).

[1] These two time-related dimensions are included just for manual double-check purposes, they are not strictly necessary if we have the timestamp.

We can order participant actions thanks to the timestamp. We then compare this course of actions with the optimal path previously identified in the use case, and the discrepancies are highlighted to identify possible design errors or different user behaviors that allow user analysis refinement. Lack of fluency in task enactment may also serve as an indicator of mismatch between the designers model and the user mental model. Task completion time is calculated subtracting the agreed upon task starting event timestamp from the ending event one.

Additional data can be collected, such as task completion rate and number of taps to complete the task. Optimal values for efficiency and effectiveness measures can be obtained from the measurement of designers performance with the application. Target values can be then defined considering such optimal values and the context(s) of use considered for the application. These values act as baseline measures for the usability test.

## 4 Validation Results

We validated our proposed logging mechanism by supporting a lab usability test for an app oriented to university students in our school, with more than 500 total installs in the Google Play store by the end of 2016.

A new map feature was designed for the app, and the usability test presents 6 tasks to carry out focused on such new feature. Code has been instrumented to ensure that such events are registered in the GAMA servers.

The usability tests have been carried out by a facilitator who measured time taken per task and number of user errors manually, as traditionally done in any usability test. Afterwards, both manually measured results and automatic measurements extracted from the logging mechanism have been compared using non-parametric tests (Mann-Whitney).

The usability test was announced through twitter and the forum of an HCI University course. Volunteers that participated in the test received a small gift.

29 participants were recruited, all of them University students between 18–25 years old and users of an Android phone. In the introduction to the usability test, participants were told that they could abandon the test at any time and they would still receive the gift.

### 4.1 Usability Test Procedure

For each test participant, the facilitator followed the following steps (trigger actions signaling task start and end appear highlighted in **bold face**):

1. Read the introductory text.
2. Install the app version to test in the participant's mobile phone, by activating the developer mode and the USB debugging mode in the phone, connecting it to the lab computer and running the test app.
3. Open the app and cancel the tutorial.
4. Go to the Settings section of the app and write down the ClientID displayed.

5. Read the task to the participant.
6. **Take the app to the screen where the task would be started** and ask the participant to carry out the task with the app. Start the manual timer at the same time.
7. When the participant says he/she has finished the task (or abandons), stop the timer and **go to Settings section of the app**.

   Steps 5–7 are repeated until all the tasks have been carried out.
8. Personal data and SUS (System Usability Scale) questionnaires are handed out to the participant to fill in.

The facilitator wrote down throughout the test task completion times, participant comments, relevant observations, number of participant errors, and if the participant successfully completed each task. Besides, test participants performance data were successfully recorded in Google Analytics servers, to be later compared to the manually registered data.

The ClientIDs registered for every participant served for the extraction of each test data from GAMA servers. All user events recorded in the logging mechanism were ordered by time using the timestamp custom dimension, and the special events used for signaling starting and ending time for each task were also looked for in the list of events. The start of the task was signaled by tapping the back button when in the Settings screen, and the end of the task was marked going to the Settings section when the participant says he/she had finished the task or abandoned. The timestamps of both events allowed for the calculation of task completion times.

## 4.2 Statistics Results

A total of 25 subjects were analyzed. 4 subjects' data were not included since the protocol was not strictly followed, or they did not complete the test.

First, a descriptive statistic analysis was done. Later, a non-parametric test (U test Mann-Whitney [33]) was used to determine if GAMA measurement is comparable to the gold standard of usability test, which is performed by a trained observer. Thus, our null hypothesis was that both distributions are replaceable. Significance was established at 0.05.

Tables 1 and 2 present the descriptive statistics of the time and errors, both manually and automatically measured through GAMA. Results are presented as average and standard deviation with a confident interval of 95%.

From the descriptive data we observe that per task both measurements are similar, both in terms of average measure and standard deviation. In order to test if they are statistically replaceable, we perform non-parametric tests, concretely U test Mann Whitney, since the distribution of data are not normal.

Table 3 shows the results of the p-values for all the models we fitted in the U test Mann Whitney. All p values are above 0.05, which is the threshold to consider that the null hypothesis is true. That is, that both distributions are equal in the time task and number of errors, implying that both methods to measure usability can be replaceable.

**Table 1.** Time to perform each task. Measured in seconds. CI: Confident interval. LL: Low level. UP: Upper level

| | Average | CI 95% | | Standard deviation |
|---|---|---|---|---|
| | | LL | UL | |
| Manual Task 1 | 42.70 | 33.32 | 52.08 | 20.034 |
| GAMA Task 1 | 49.00 | 35.80 | 62.20 | 28.200 |
| Manual Task 2 | 49.25 | 36.36 | 62.14 | 27.535 |
| GAMA Task 2 | 52.60 | 37.75 | 67.45 | 31.725 |
| Manual Task 3 | 30.30 | 17.54 | 43.06 | 27.255 |
| GAMA Task 3 | 28.80 | 17.59 | 40.01 | 23.955 |
| Manual Task 4 | 47.75 | 35.64 | 59.86 | 25.872 |
| GAMA Task 4 | 44.40 | 33.03 | 55.77 | 24.295 |
| Manual Task 5 | 39.50 | 28.01 | 50.99 | 24.560 |
| GAMA Task 5 | 39.70 | 28.66 | 50.74 | 23.591 |
| Manual Task 6 | 41.90 | 32.40 | 51.40 | 20.300 |
| GAMA Task 6 | 41.25 | 32.28 | 50.22 | 19.169 |
| Manual total time | 251.40 | 217.12 | 285.68 | 73.246 |
| GAMA total time | 255.75 | 225.70 | 285.80 | 64.206 |

**Table 2.** Number of errors per task. CI: Confident interval. LL: Low level. UP: Upper level

| | Average | CI 95% | | Standard deviation |
|---|---|---|---|---|
| | | LL | UL | |
| Manual Task 1 | 1.75 | 1.18 | 2.32 | 1.209 |
| GAMA Task 1 | 1.85 | 1.26 | 2.44 | 1.268 |
| Manual Task 2 | 1.55 | 0.75 | 3.35 | 1.701 |
| GAMA Task 2 | 1.60 | 0.76 | 2.44 | 1.789 |
| Manual Task 3 | 0.50 | 0.03 | 0.97 | 1.000 |
| GAMA Task 3 | 0.50 | 0.03 | 0.97 | 1.000 |
| Manual Task 4 | 0.85 | 0.34 | 1.36 | 1.089 |
| GAMA Task 4 | 0.85 | 0.36 | 1.34 | 1.040 |
| Manual Task 5 | 0.75 | 0.16 | 1.34 | 1.251 |
| GAMA Task 5 | 0.65 | 0.10 | 1.20 | 1.182 |
| Manual Task 6 | 1.00 | 0.37 | 1.63 | 1.338 |
| GAMA Task 6 | 1.15 | 0.47 | 1.83 | 1.461 |
| Manual total error | 6.40 | 4.90 | 7.90 | 3.202 |
| GAMA total error | 6.60 | 4.90 | 8.30 | 3.633 |

**Table 3.** Results of the U test Mann Whitney with a significant level 0.05

| Time per task (seconds) | | Errors per task | |
|---|---|---|---|
| Task 1 | 0.528 | Task 1 | 0.603 |
| Task 2 | 0.956 | Task 2 | 0.981 |
| Task 3 | 0.959 | Task 3 | 0.926 |
| Task 4 | 0.652 | Task 4 | 0.962 |
| Task 5 | 0.954 | Task 5 | 0.982 |
| Task 6 | 0.796 | Task 6 | 0.875 |
| Total | 0.831 | Total | 0.799 |

## 5 Discussion

We have demonstrated through a non-parametric test that our automatic GAMA-based logging mechanism of user actions measurement and the traditional manual measurement are exchangeable. Usability testers of mobile apps can then profit of our logging approach to focus on observing user reactions, and thus avoiding the need to employ two experimenters for the test.

One of the test facilitators did not follow the protocol correctly, since he did not visit the Settings section of the app when appropriate (therefore his 4 tests were not included in the validation). This requirement might be too complicated for the test facilitator, as he/she does not receive any feedback from the app about the correct activation of the task starting and ending events. We need to explore the usability of this activation mechanism for test facilitators, and to provide an alternative way that is: (1) easier and faster; and (2) offers some user feedback.

We may use voice recognition for signaling task start and ending. In this case, the Settings section would have a way to turn on the testing mode, thus activating the voice recognition feature. A sound signal could be used for feedback in order to use the same channel of communication. A second option would be to use a non-common gesture (like a three-finger swipe) to indicate task start or ending. A screen blink could provide the user feedback in this case. We need to be very careful in the choice of the specific gestures for this purpose, because there is always the risk that they override a gesture already captured in the app to test.

Usability testing with usability experts will be necessary to establish the best way of signaling the start of the overall test and the start and ending of each task asked to the test participant.

For further analysis of our work, we have to refine the automated tracking mechanism to consider possible alternative paths. In most apps there is not only one way to carry out a task, but a variety of them which must be taken into account. Then, we can consider the number of user actions as an additional performance criterion. In this case, user actions departing from the optimal path would not be considered user errors if they follow an alternative success path.

Our logging mechanism serves both for lab testing and for evaluation of real usage. We have just tested the former application, lab usability testing, where user tasks are defined in advance. In the automated logging of user events on the field, a key issue for the analysis of the gathered data will be the identification of the user tasks.

We have started by gathering objective data related to basic usability attributes such as efficiency and effectiveness. However, additional variables could be studied in the future, such as advancement in the learning curve (how much usage time requires a user to reach a pre-defined level of performance), regularity/irregularity of app usage across time or number of times the help subsystem is accessed, or different paths users take to reach the same results.

Different variables are relevant depending on the stage of development the project is: defining the product concept, pre-release, or evolution of an already released app. Task completion times and number of errors may be variables of interest in the pre-release stage, while qualitative data may be more relevant in the early phases of development, or when the development team is working on a major redesign of an existing app.

UX also has a special relevance in mobile applications, and we intend to study how our proposed logging mechanism could gather data about additional variables affecting UX. When we have specific user profiles clustered and recognized through data mining, we will be able to include in the app code ad-hoc attitude questionnaires addressed to specific profiles.

## 6 Conclusions

We have presented a proposal for automated logging of UI events through the GAMA service for usability evaluation purposes, serving for both lab usability testing and for logging actual use. For lab usability testing it can allow for a higher number of participants or more ambitious test plans. When used for logging actual use, our GAMA-based proposal may allow for an easy management of a high volume of usage data, without the need of maintaining a dedicated server. The reduced cost of automated or semi-automated solutions make them especially suited to the short development cycles present in mobile app development projects, increasing the cost-effectiveness of usability evaluation activities.

We have validated the logging mechanism applying it to lab usability testing of a real application, proving that the results of the automated version and the manual measurements are exchangeable. Despite this, some improvements are necessary to ensure that the test facilitator can more easily indicate to the logging mechanism when a new task is starting or ending.

We are currently working on the extraction and analysis of the user events logged through our proposed mechanism, applying data mining techniques in order to identify possible usability defects and specific user profile behavior. Validation of this alternative application of our automated logging proposal will be necessary for fully acknowledging the advantages of the overall solution.

Extensive study of usage logs could provide an identification of typical usability problems in mobile apps, and possible solutions could be compiled in the form of guidelines for developers or interaction designers.

Finally, we plan to facilitate the task modeling and annotation for mobile app developers by building a plugin to ease use case annotation and code instrumentation.

**Acknowledgements.** We would like to thank the anonymous reviewers for their valuable feedback that allowed us to greatly improve the paper.

## References

1. IDC: Apple, Huawei, and Xiaomi Finish 2015 with Above Average Year-Over-Year Growth, as Worldwide Smartphone Shipments Surpass 1.4 Billion for the Year (2016). http://www.idc.com/getdoc.jsp?containerId=prUS40980416
2. Gartner: Worldwide Device Shipments to Grow 1.9 Percent in 2016, While End-User Spending to Decline for the First Time (2016). http://www.gartner.com/newsroom/id/3187134
3. Appcelerator/IDC: Voice of the Next-Generation Mobile Developer, Appcelerator/IDC Q3 2012 Mobile Developer Report (2012). http://www.appcelerator.com.s3.amazonaws.com/pdf/Appcelerator-Report-Q3-2012-final.pdf
4. Flurry: Apps Solidify Leadership Six Years into the Mobile Revolution (2014). http://flurrymobile.tumblr.com/post/115191864580/apps-solidify-leadership-six-years-into-the-mobile
5. Joorabchi, M.E., Mesbah, A., Kruchten, P.: Real challenges in mobile app development. In: 2013 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2013, Baltimore, MD, USA, 10–11 October 2013, pp. 15–24. IEEE (2013). doi:10.1109/ESEM.2013.9
6. Statista: Number of apps available in leading app stores as of June 2016 (2017). https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/
7. Flora, H.K., Chande, S.V.: A review and analysis on mobile application development processes using agile method. Int. J. Res. Comput. Sci. **3**(4), 9–18 (2013). doi:10.7815/ijorcs.34.2013.068
8. ISO. 9241-210: Ergonomics of Human-System Interaction - Part 210: Human-Centred Design for Interactive Systems. ISO 9241-210 (2010)
9. Ferre, X., Juristo, N., Moreno, Ana M.: Improving software engineering practice with HCI aspects. In: Ramamoorthy, C.V., Lee, R., Lee, K.W. (eds.) SERA 2003. LNCS, vol. 3026, pp. 349–363. Springer, Heidelberg (2004). doi:10.1007/978-3-540-24675-6_27
10. Coursaris, C., Kim, D.: A meta-analytical review of empirical mobile usability studies. J. Usabil. Stud. **6**(3), 117–171 (2011)
11. Lettner, F., Holzmann, C.: Automated and unsupervised user interaction logging as basis for usability evaluation of mobile applications. In: Proceedings of 10th International Conference on Advances in Mobile Computing & Multimedia – MoMM 2012, vol. 118 (2012). doi:10.1145/2428955.2428983
12. Ivory, M.Y., Hearst, M.A.: The state of the art in automating usability evaluation of user interfaces. ACM Comput. Surv. (CSUR) **33**(4), 470–516 (2001). doi:10.1145/503112.503114
13. Path, M.: Introducing Google Analytics for Mobile Apps. Google Mobile Ads Team (2009). http://googlemobile.blogspot.com.es/2009/11/introducing-google-analytics-for-mobile.html
14. Google: Google Analytics for Mobile Apps. https://developers.google.com/analytics/solutions/mobile. Accessed 2017

15. Tiedtke, T., Märtin, C., Gerth, N.: AWUSA–a tool for automated website usability analysis. In: Proceedings of 9th International Workshop on Design, Specification and Verification DSV-IS 2002 (2002)
16. Ivory, M.Y.: Automated Web Site Evaluation. Researchers' and Practitioners' Perspectives. Springer Science+Business Media, Dordrecht (2003)
17. Angulo, E., Ferre, X.: A case study on cross-platform development frameworks for mobile applications and UX. In: Proceedings of XV International Conference on Human Computer Interaction - Interacción 2014, Puerto de la Cruz, Tenerife, Spain, pp. 1–8 (2014). doi:10.1145/2662253.2662280
18. Mendoza, A.: Mobile User Experience: Patterns to Make Sense of It All. Morgan Kaufmann, Waltham (2014)
19. Paternò, F., Russino, A., Santoro, C.: Remote evaluation of mobile applications. In: Winckler, M., Johnson, H., Palanque, P. (eds.) TAMODIA 2007. LNCS, vol. 4849, pp. 155–169. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77222-4_13
20. Carta, T., Paternò, F., Santana, V.: Support for remote usability evaluation of web mobile applications. In: Proceedings of 29th ACM International Conference on Design of Communication - SIGDOC 2011, pp. 129–136 (2011). doi:10.1145/2038476.2038502
21. Balagtas-Fernandez, F., Hussmann, H.: A methodology and framework to simplify usability analysis of mobile applications. In: ASE2009 - 24th IEEE/ACM International Conference on Automated Software Engineering, pp. 520–524 (2009). doi:10.1109/ASE.2009.12
22. Kluth, W., Krempels, K.H., Samsel, C.: Automated usability testing for mobile applications. In: WEBIST, vol. 2, pp. 149–156 (2014)
23. Feijó Filho, J., Valle, V., Prata, W.: Automated usability tests for mobile devices through live emotions logging. In: Proceedings of 17th International Conference on Human-Computer Interaction with Mobile Devices and Services Adjunct (MobileHCI 2015), pp. 636–643. ACM, New York (2105). doi:10.1145/2786567.2792902
24. Leichtenstern, K., Erdmann, D., André, E.: EVAL-an evaluation component for mobile interfaces. In: Proceedings of 10th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 483–484. ACM, September 2008
25. Porat, T., Schclar, A., Shapira, B.: Mate: a mobile analysis tool for usability experts. In: CHI 2013, Extended Abstracts on Human Factors in Computing Systems, pp. 265–270. ACM, April 2013
26. Hulo, A., To, J.: Developing real time tracking of user behavior, with Google analytics for mobile phone devices. Master thesis, Lund University (2015). https://lup.lub.lu.se/student-papers/search/publication/5305701
27. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction, 3rd edn. Wiley, Chichester (2011)
28. Ferre, X., Juristo, N., Windl, H., Constantine, L.: Usability basics for software developers. IEEE Softw. **18**, 22–29 (2001). doi:10.1109/52.903160
29. Jacobson, I., Spence, I., Kerr, B.: Use-case 2.0. Commun. ACM **59**(5), 61–69 (2016). doi:10.1145/2890778
30. Constantine, L.L., Lockwood, L.A.D.: Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, New York (1999)
31. Rubin, J., Chisnell, D.: Handbook of Usability Testing. How to Plan, Design and Conduct Effective Tests, 2nd edn. Wiley, Indianapolis (2008)
32. Rettig, M.: Prototyping for tiny fingers. Commun. ACM **37**(4), 21–27 (1994). doi:10.1145/175276.175288
33. Mann, H.B., Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other. Ann. Math. Stat. **18**(1), 50–60 (1947). doi:10.1214/aoms/1177730491