# Smartphone Applications Usability Evaluation: A Hybrid Model and Its Implementation

Artur H. Kronbauer[1], Celso A. S. Santos[2], Vaninha Vieira[3]

[1] PMCC – UFBA, Av. Adhemar de Barros s/n sala 138,
Salvador – BA, Brazil
arturhk@gmail.com
[2] DI – CT – UFES, Av. Fernando Ferrari s/n sala 8,
Vitória – ES, Brazil
saibel@inf.ufes.br
[3] PMCC – UFBA, Av. Adhemar de Barros s/n sala 234,
Salvador – BA, Brazil
vaninha@ufba.br

**Abstract.** Evaluating the usability of smartphone applications is crucial for their success, so developers can learn how to adapt them considering the dynamicity of mobile scenarios. The HCI community recommends considering different requirements when evaluating those applications, such as quantitative data (metrics), subjective evaluation (users' impressions) and context data (e.g. environment and devices conditions). We observed a lack in the literature of approaches that support those three requirements combined into a single experiment; generally one or a pair of them is used. Besides, performing usability evaluation on real mobile scenarios is hard to achieve and most proposals are based on laboratory-controlled experiments. In this paper, we present our proposal for a hybrid usability evaluation of smartphone applications, which is composed by a model and an infrastructure that implements it. The model describes how to automatically monitor and collect context data and usability metrics, how those data can be processed for analysis support and how users' impressions can be collected. An infrastructure is provided to implement the model allowing it to be plugged into any smartphone Android-based application. To evaluate our proposal, we performed a field experiment, with 21 users using three mobile applications during a 6-month period, in their day-to-day scenarios.

**Keywords:** Usability Evaluation, Smartphone Application, Remote Usability Evaluation, Usability Testing.

## 1 Introduction

In recent years, the introduction of several technologies has revolutionized society's methods of communication, entertainment, and the execution of daily tasks. Simultaneously, the process of digital convergence has generated a great number of devices (PDAs, smartphones, tablets) capable of gathering different forms of user-computer interaction in an integrated way and with reasonable processing power.

Considering that the interaction of the user with this kind of device is extremely sensitive to the context in which the application is running, what constitutes the most adequate scenario for executing usability tests is a topic of great discussion. The central question is whether these experiments should be performed in the field, or in the laboratory [1].

Another important question has to do with the kind of data that must be present in a usability evaluation. Besides quantitative data used to determine the quality of the interfaces and the usability of the applications, subjective data should also be considered in order to draw conclusions about the users' satisfaction.

The main problem when evaluating mobile applications is the development of approaches whose evaluations integrate quantitative, subjective and context data in the same experiment. Nowadays, those three points of interest are investigated in isolation, which makes the relationship between those results difficult to determine [2].

Faced with this challenge, the objective of this study is to assess the new applications usability, their functionalities, how the users adapt to them to do their daily tasks, the satisfaction level provided and the level of external interferences.

In order to achieve evaluations with different kinds of information, we propose an approach to support the evaluation of smartphone applications based on the Android platform. This approach blends two strategies for data capture: the first one, known as Logging [3], is based on the data collection related to the user interaction with an application, allowing statistical analysis regarding usability; it allows the use of sensors available in the smartphones for contextual data collection, such as luminosity intensity and the device´s position [4]. The second one, known as Experience Sampling Method (ESM) [5], is based on the collection of users' feelings towards a specific product through questions.

The main contribution of this work is to propose a new approach that is able to relate different types of data to evaluate the usability of embedded applications on smartphones. The proposed approach can contribute to the implementation of field and laboratory experiments, whether they are supervised or not.

The remainder of this article is divided into six sections. Section 2 describes the proposed model. Section 3 presents an infrastructure that implements the model, to capture and evaluate usability data from Android applications. Section 4 presents a case study performed to evaluate the proposal and analyzes the results found on the experiments. Section 5 discusses related works to reinforce our contributions. Finally, in Section 6, we present conclusions and future work.

## 2 The UEProject: A Model for Usability Evaluation on Smartphone Applications

In our work, we propose the Usability Evaluation Project (UEProject), which is composed by a model and an infrastructure to support a hybrid approach for usability evaluation on Smartphone Applications. The proposed model enhances the capture of quantitative and subjective data, contextualized within the evaluation scenario and in conjunction with the identification of the users' profiles.

The intention is to go beyond the collection of statistical data, such as identifying the frequency of use, the error count or speed of execution of a task. The goal is to provide an infrastructure that enables the contextual factors associated with usability problems, and identify the user perception regarding the application.

The model uses a component-based architecture to support the reuse of the implemented resources and its redefinition according to evaluators' needs. Figure 1 presents the three high level components that represent the model and their relations: Mapping Unit, Traceability Unit and Assessment Unit. Arrows indicate information transfer between the components.

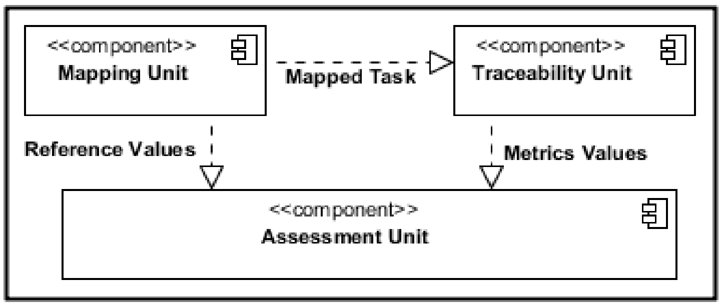

**Figure 1.** Main components of the model

The Mapping Unit is responsible to provide the necessary resources for mapping the tasks that will be investigated later in the application under evaluation. It generates the Mapped Tasks and Reference Values that provide information about the context in which the task was mapped and the perceptions of the person responsible for mapping it. The Traceability Unit provides a library of metrics that uses the mapped tasks, enabling an instrumentation of the application's source code with metrics for the capture of quantitative, contextual, and subjective data concerning the user profile. The values obtained during user interactions are sent to the Assessment Unit. The Assessment Unit is responsible for receiving the transmitted data, transferring it to a database and enabling evaluations. Next subsections present the internal components of the model's units.

## 2.1 Mapping Unit

This unit is subdivided into three components responsible for providing the task-mapping functionalities. In Figure 2 the components diagram can be seen, with its interfaces and doors for data transfer from one component to another.

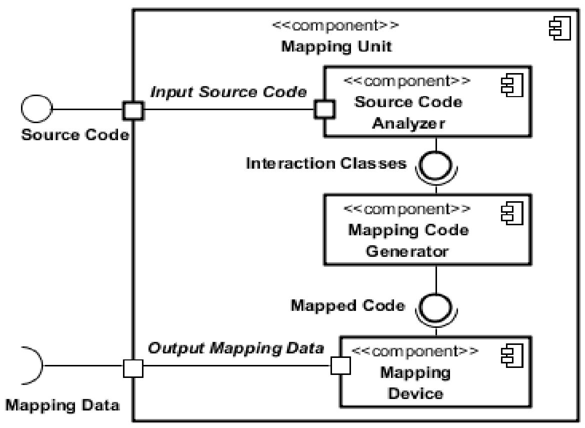

**Figure 2.** Mapping Unit Architectural Overview

Initially it has been planned that the applications' source code be made available. The objective of this action is to allow the Source Code Analyzer component to identify which classes refer to the treatment of users' interactions. These classes are identified and provided as a requirement for the Mapping Code Generator component so that it can attach information that enables the identification of tasks to the original source code. The result of this component's action is to perform the instrumentation of the application's original code, thereby allowing the tasks to be mapped. After the source code is prepared for mapping, it must be forwarded to the Mapping Device component where the tasks will be mapped. This device must enable the mapping of tasks and make them available to be used by other units in the model.

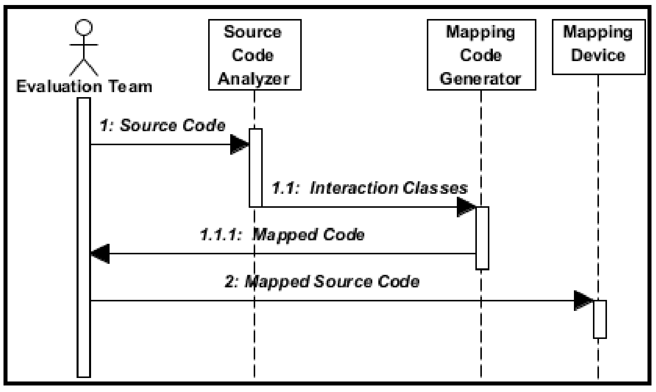

**Figure 3.** Steps for application instrumentation for task mapping

In Figure 3, the diagram portrays the sequence that enables the exchange of messages and data between the Mapping Unit components.

In Figure 4 the diagram of a simplified use case is presented, where the only two existing interactions are shown. In the first interaction, the Evaluation Team is responsible for providing the application's source code and, in the second, for performing the tasks' mapping for later assessment. It is worth noting that the tasks' mapping can be done by a test engineer, an experienced user, or even average users who are chosen according to their profile, thereby allowing future comparisons.

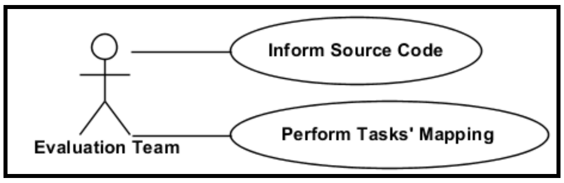

**Figure 4.** Use Case of the Mapping Unit

## 2.2 Traceability Unit

This unit is responsible for collecting different types of information: (i) user profile (e.g. education and age); (ii) data referring to the user's interaction, such as hits and time to complete a task; (iii) contextual data like luminosity and noise; and (iv) subjective data, related to the users' feelings regarding the applications' usage.

For the instrumentation of the source code to include the code for capture metrics and data collection, the Traceability Unit was divided into three internal components, as can be seen in Figure 5: Metrics Library, Metrics Generation and Interaction Device. The Metrics Library is the component that provides the structure to all capture metrics. It uses the mapped tasks to bring together the structure of the metrics with information about the application source code. It generates, as result, the metrics' structures adapted to the application under evaluation. In the development of the Metrics Library, the incorporation of three types of structure was expected: (i) the ones used to capture quantitative data from users interactions; (ii) the ones responsible for the subjective data (direct interactions with users) and; (iii) the metrics that allow instrumentation of the available sensors.

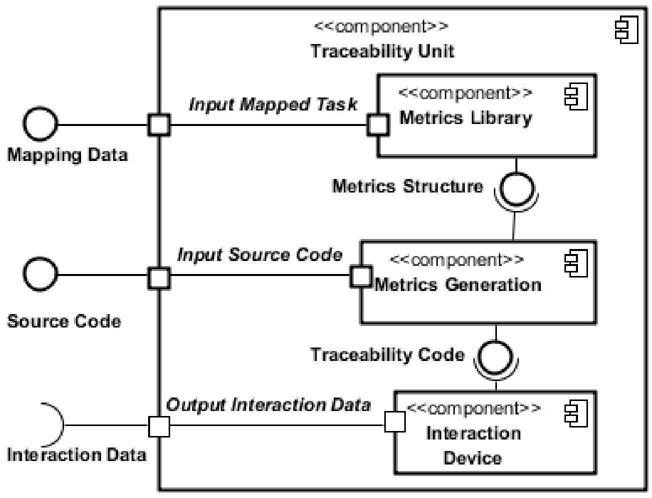

**Figure 5.** Traceability Unit Architectural Overview

The Metrics Generation component receives as input the application's source code and the adapted metrics structure to be attached to this code. As result it provides a new source code with the inserted metrics, the Traceability Code. The component Interaction Device requires the installation of the application's new code containing the metrics that will be used for data capture. This device should enable the user's interaction while allowing data to be captured.

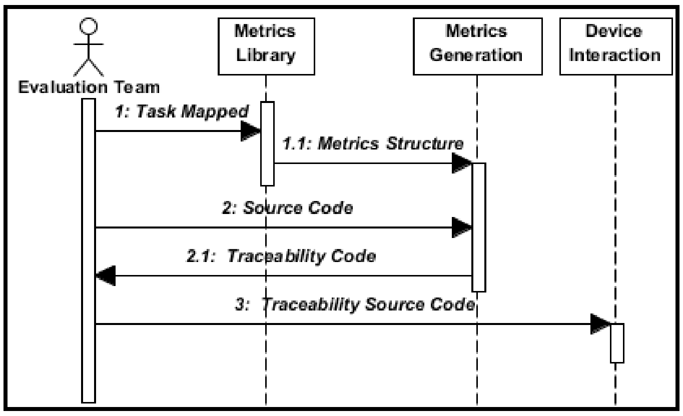

**Figure 6.** Steps for the instrumentation of an application with the metrics for capturing data

In Figure 6, the diagram presents the sequence of actions that permits the exchange of messages and data between the Traceability Unit components.

Figure 7 presents two use case diagrams concerning the Traceability Unit. The first actor (Evaluation Team) is responsible for providing files with the mapped tasks and the application's source code, enabling the creation of a new application with metrics for capturing data. The second actor (User) uses the application while data concerning usability, context, profile, and feeling about application usage are captured.

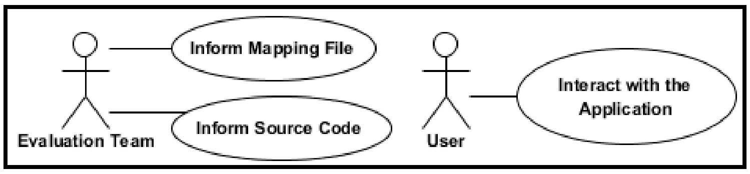

**Figure 7.** Use Cases Diagram of the Traceability Unit

## 2.3 Assessment Unit

This unit was structured in four different components that are responsible for infrastructure where data to be evaluated will be stored over time. Figure 8 presents an architectural overview of its components. To illustrate better its behavior, the sequence diagram in Figure 9 presents the exchange of messages and data between the components.

Initially, the Data Receiver component collects the data captured during the execution of the user's interactions within the application under evaluation. A second component, the Populate Database, performs the verification of new data arrival and populates it into the database. The Database component represents a Database Management System (DBMS) that stores persistent data. The Analysis Component has the responsibility to access existing data in the database and to provide resources that enable the extraction of usability information relevant to the evaluators.

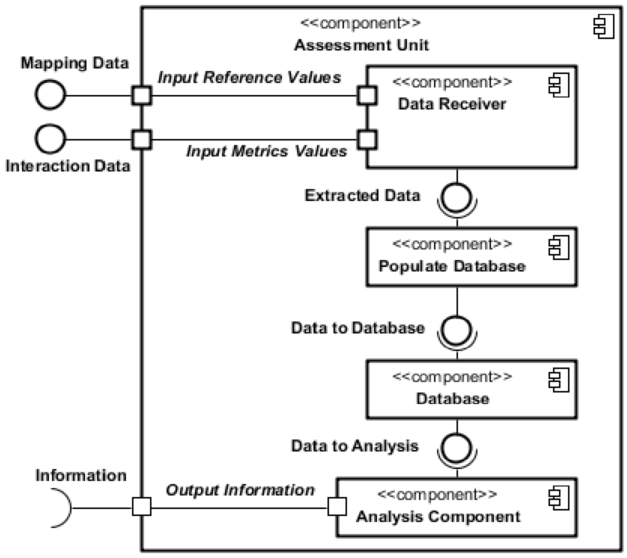

**Figure 8.** Assessment Unit Architectural Overview

To make the Assessment Unit easier to understand, the diagram that follows (Figure 9) illustrates the exchange of messages and data between components.

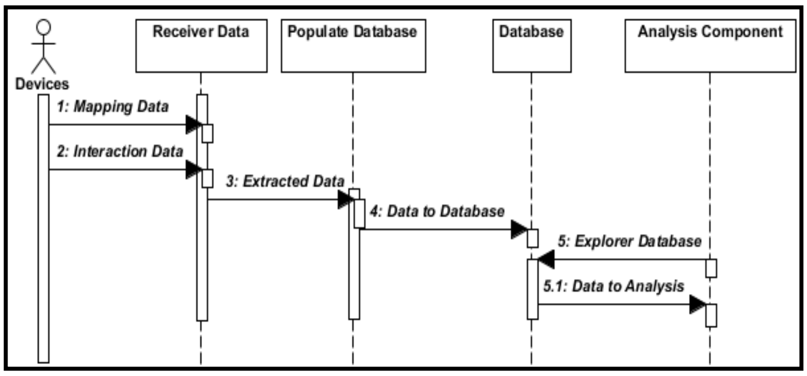

**Figure 9.** Actions performed by the components of the Assessment Unit

The only direct interaction of an actor with the devised components in this unit is at the moment of the assessments done by the Evaluation Team. This team is responsible for defining data, parameters, and configurations so the information can be extracted according to their needs.

## 3 The UEProject Infrastructure

In this section, we describe the infrastructure implemented to verify the feasibility of the UEProject model. The technological aspects and techniques used in the implementation of the infrastructure are discussed. Additionally, the section describes the tools created, or used from the market, to cover all the components according to the model proposal.

We defined four premises related to how technical choices will be made when planning the construction of the infrastructure, as follows.

1) To propose that the data can be collected anywhere the user is interacting. To facilitate this, the technique Logging was chosen. This strategy consists of automatically monitoring and gathering data concerning the applications' use.

2) To define that all used software licenses should be free. To this end, we decided to develop the components using the Java Language, since it is platform-independent and free. The components obtained from the market were also chosen according to this premise. The objective is to make the project economically feasible.

3) To determine that all collected data might be gathered from smartphone applications. For this reason, the Android platform was chosen because it uses Java and Android is widely used nowadays.

4) To suggest that the applications' source code to be instrumented should not be modified. Therefore the Aspect Oriented Program (AOP) [6] was chosen. AspectJ[1], an AOP plugin for Java is used to instrument the applications with codes considered as transversals. In this case, the codes are needed to map the tasks and the code related to the metrics for data gathering.

In this implementation of the infrastructure, the proposed way for task mapping was to capture the methods executed in the application after the interaction of the Evaluation Team. The group of interactions for a task execution is translated into a group of methods executed in the application. The mapped methods are used as Join Points so the metrics can be introduced in an application, without modifying the original source code. In this sense, Point Cuts are used to define the rules that validate the execution of the metrics built in the Advices.

The following subsections describe the tools used to implement the components of the three units defined in the Model.

### 3.1 Tools used in the Mapping Unit

The first developed tool looks at the Source Code Analyzer and the Mapping Code Generator. This tool was called the Mapping Aspect Generator (MAG). Figure 10 presents a workflow showing all the necessary steps for the preparation of an application to be mapped.

[1] Available in http://www.eclipse.org/aspectj/

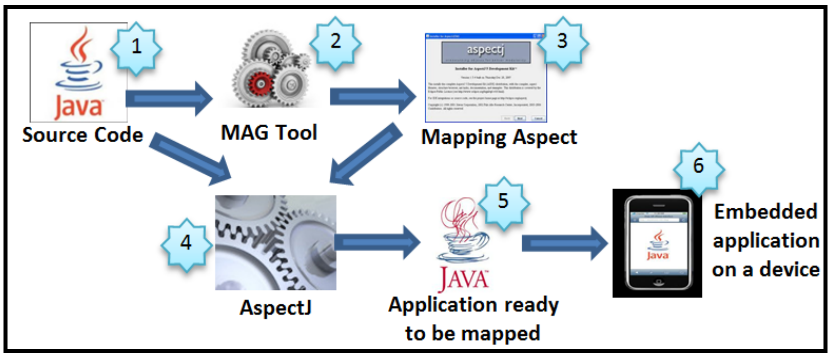
1
Source Code
2
MAG Tool
3
Mapping Aspect
4
AspectJ
5
Application ready to be mapped
6
Embedded application on a device

**Figure 10.** Workflow for the insertion of mapping aspects into an application

Initially the MAG tool must receive the application's source code to be mapped (step 1). After that, the tool goes through all the classes and creates a list of the classes that deal with users' interactions (step 2). Afterwards the tool uses the list of classes and generates an Aspect to implement the *onUserInteraction*[2] method in the classes that do not yet have it (step 3). This action enables the detection of users' interactions. The next action is to compile the application's source code with the generated Aspect, using AspectJ (step 4). The result will be that the application is ready to be mapped (step 5). To end the process, the application will be embedded on a device that allows the interactions to take place (step 6).

To allow the task mapping and the association of reference values and context during the interaction of the Evaluation Team, another tool was developed, called the Automatic Task Description (ATD). This tool must be embedded on the device on which the application will be mapped and executed simultaneously with the application (Device Mapping component). Therefore, as the Evaluation Team interacts with the application, the executed methods are automatically captured as the concluding steps of a task.

The main point of this tool is to act as a filter that identifies when a user interaction occurs. Also, the filter identifies which classes, methods, and application parameters were used.

### 3.2 Tools used in the Traceability Unit

The tool proposed to implement the Metrics Library and the Metrics Generation components was named Usability Metrics Generation. This tool contains a library that has the structure of the metrics that are available to perform measurements. Figure 12 shows a workflow of necessary actions for the automatic generation of the Aspects and its association with the source code of the application.

[2] Available in http://developer.android.com/reference/android/app/Activity.html

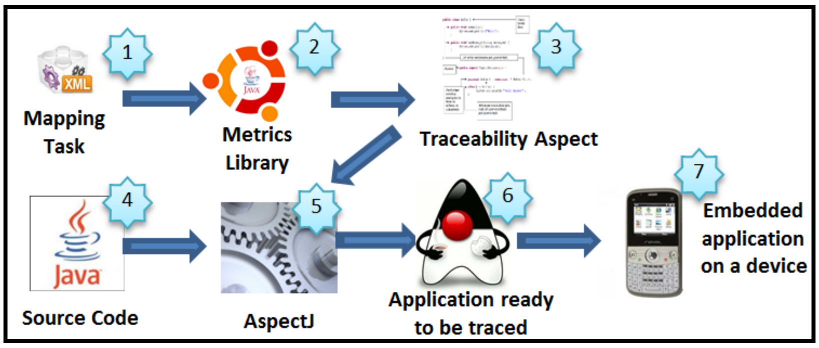

**Figure 12.** Workflow for the insertion of the aspects of traceability in an application

The tool receives as input the XML file generated in the Mapping Unit (step 1). Next, the existing methods in the XML, along with the information contained in the Metrics Library (step 2), will be used to generate the Aspects responsible for the capture, transmission, and persistency of the data (step 3). Afterwards, the source code of the application (step 4) and the generated Aspect (step 3) must be compiled using AspectJ (step 5). The result is the generation of an application with the metrics (step 6). Later, the Application will be loaded onto the device with which the user will interact.

The efficiency and effectiveness of the users' interactions are measured through usability metrics described in Table 1. All the measurements take into consideration the mapping of a task, where all the steps to its conclusion are described. Thus, the errors, for instance, are identified every time the user interacts with the application that is not in the previous mapping.

**Table 1.** Usability Metrics

| **Usability Metrics** | **Objectives** |
|---|---|
| Metrics for the capture of assertiveness | |
| Actions in conformity | Quantify the difficulty of interaction for the user in a task using the interface. |
| Actions in violation | |
| Metrics for the search and use of tasks | |
| Unfinished task | An unfinished task can mean that the user is lost or searching for a desired task.<br>The use/nonuse of a task indicates if the user found what s/he was searching for. |
| Finished tasks | |
| Finished and used tasks | |
| Finished and unused tasks | |
| Metrics related to performance | |
| Average time for task completion | Indicate the need for improvements that minimize the time and actions for the execution of tasks. |
| Average number of actions for task completion | |
| Number of tasks concluded without errors | |
| Metrics for the use of help | |
| Frequency of the use of help | Measure the level of difficulty faced by the user. Identify if the help available provides the desired effect. |
| Violations made after the use of help | |

The sensors used to capture the contextual data were the accelerometer (to capture the horizontal or vertical position), GPS (to capture movement), Luminosity Sensor (to capture the environment's luminosity), and the microphone (to capture noise in the environments).

It is important to stress that the infrastructure uncovers new metrics to be incorporated in the Metrics Library, which increases its adoption in different scenarios, contexts, and with the use of other sensors. For example, specific metrics could be incorporated in an application for the spatial orientation of people. Therefore, such metrics could be associated with the data provided by a mobile device's GPS, enabling the comparison of the user's interactions with information regarding positioning, speed, and route taken. In that case, the usability analysis would take into account not only how a user performs a task but also how the environment influences the user's interactions.

The subjective metrics are used to measure the emotional state of the users during their experiences with an application. The approach chosen for representing these metrics is known as the Experience Sampling Method (ESM) [5].

ESM was chosen because of the following aspects: (i) it is appropriate for use on devices with relatively small screens; (ii) it is intuitive and doesn't take much mental effort to be interpreted; and (iii) it is capable of being answered with entry modes provided by different mobile devices.

ESM measures two dimensions, the kind of emotion (positive or negative) and the intensity of the emotion. To do this, a group of pictures is displayed indicating emotional states associated with a question, as can be seen in Figure 13. The sequence of the pictures represents varying degrees of emotional intensity and can be interpreted from left to right as: very displeased, displeased, indifferent, pleased, very pleased. These questions are defined by the evaluators during the execution of the Usability Metrics Generation Tool.

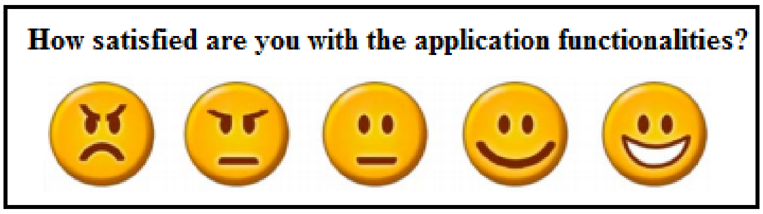

**Figure 13.** Example of an ESM

## 3.3 Tools used in the Assessment Unit

To devise the components defined in the Assessment Unit, the following processes were performed: (i) to create and configure a FTP and database server (DB) and to make them available on the internet; (ii) to model a database capable of storing the captured information about the usability of the instrumented applications; (iii) to create tools for detecting the presence of new files on the FTP server and populate the

DB; and (iv) to choose an Online Analytical Processing (OLAP) tool to analyze the data.

In order to transmit and store data on the mapping of tasks and the results of users' interactions, it is necessary to have a FTP and a DB available. For this, a micro instance of the service known as Amazon Elastic Compute Cloud (Amazon EC2) was contracted with Amazon, running on the Ubuntu operating system, version 11.10 Oneiric Server, with 613 Mbyte RAM memory and a 16 Gbytes hard drive.

To study the Data Receiver component, the FTP server ProFTPd, version 1.3.4, was installed to manage and control the transfer of files.

To devise the Database component, the DBMS MySql Community Server, version 5.1.58 was installed to store the data.

To devise the Populate Database component, a tool called Data Loader was developed. The steps executed by this tool are: (i) to detect the arrival of new files on the FTP server; (ii) to extract the data; and (iii) to load the data into the database.

To devise the Analysis component, the Pentaho Analysis Services, version 3.9.0, was chosen. It offers two possibilities for data analysis currently used in usability evaluations, OLAP and data mining tools.

## 4 Case Studies

This section describes the case studies performed to evaluate the functionalities and identify the potentials of the UEProject Model and the Infrastructure with the tools that give support to its implementation.

### 4.1 Methodology

The first step taken to carry out the experiments was the completion of an exploratory research on the internet with the objective to find applications with attractive functionalities that could be easily introduced into people's lives. Also, it was taken into consideration in the applications selection the compliance to the Android platform and the use of software modularization, allowing the application's source code to be instrumented with AOP.

Following those requirements, three applications were chosen: Mileage, ^3 (Cubed), and Shuffle[3].

The use of these applications in the case studies is based on two important aspects: (i) to show that the infrastructure can be used in usability experiments for Android applications; and (ii) to show that, although the originating process of the application has no connection with the proposed usability evaluation model, the aspects of mapping the tasks and the metrics can be easily integrated into its source code. To enable the investigations, tasks were chosen for each application that would be interesting to evaluate and that would offer more possibilities for interaction. Thus, the model can be tested with a varied group of actions.

The purpose of Mileage is to help users to control the money used on fuel and automotive maintenance services. Eight tasks were mapped in this application to complete the experiments. Some of the mapped tasks for the execution of the Mileage experiments were, for example: to register a vehicle, to add a new maintenance control, and to visualize the graph showing fuel price variation.

^3 Cubed is a music and video clip manager. In this application, six tasks were mapped. Some of the mapped tasks in the experiments were: to choose a song from a playlist, to change the appearance of the application, and to activate the equalizer.

The Shuffle application is an activity scheduler that can manage daily tasks. Ten tasks were used from this application in completion of the experiments. Some examples of mapped tasks in this application are: to insert a new activity, to download a copy (backup) of the data, and to create a new project so that new activities can be linked to it.

It is important to note that, independent of the tasks chosen for the evaluation, the user should not perceive any change in the functionality or appearance of the application. The only change implemented is the presentation of subjective questions to the user.

The applications instrumented with the traceability metrics were made available for download on the internet by users who are interested in collaborating with the experiments[3].

The experiments started in December 2011. About thirty two users downloaded the applications; however, only twenty one of them are frequent and contribute effectively to the investigations. They were categorized according to age, educational grade, field of academic study, work type and social class. The context data being collected to associate to the users' experiences are: environment luminosity, device position (vertical, horizontal and mixed) and user´s movement. In addition to these data, the screen size and resolution of the used device are identified.

The first results of the analyses indicate the possibility for different methods of analysis, such as: to verify the applications usability, to analyze the experiences provided for the users and to relate their interactions to the context of use.

To exemplify, we would like to answer three questions in this preliminary study:

1) What can be concluded regarding the users´ performance over time?
2) What kind of information the context can offer to improve the usability analysis?
3) What is the users´ satisfaction level with the applications proposed in the experiment?

## 4.2 Analyzing Users' Performance on Interacting with the Applications

To analyze users' performance over the period when the experiments were carried out, the average speed to execute the tasks was evaluated. Figure 14 presents the average time, in seconds, used to conclude the tasks available in the three applications.

[3] Available in ueproject.no-ip.org

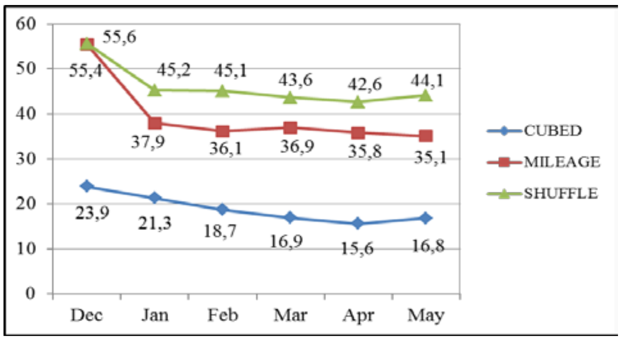

**Figure 14**. Average time to execute tasks (in seconds)

It is possible to see that, in the first month, there is a discrepancy in the average time when set against the other months. In December, we can see that the users need more time to conclude the tasks than in the following months. This draws to the conclusion that, after the first month, the users show they are more familiar with the available resources.

## 4.3 Analyzing the Potential of Considering Contextual Information

It was observed in the analyzed data that the occurrence of errors in the low-income class is greater than in the other classes. To find an explanation to this result, the type of device used by those people was investigated.

It was found that the error rate was not related to the users' purchasing power, but to the low resolution of the devices used. So, as the low resolution devices are more frequently used in the low-income class, a wrong result might be found without the evaluation of a contextual factor.

In Figure 15, the relation between the purchasing power and the smartphones screen resolution is presented. It shows that the errors are more frequent when low resolution smartphones are used, regardless the purchase power.

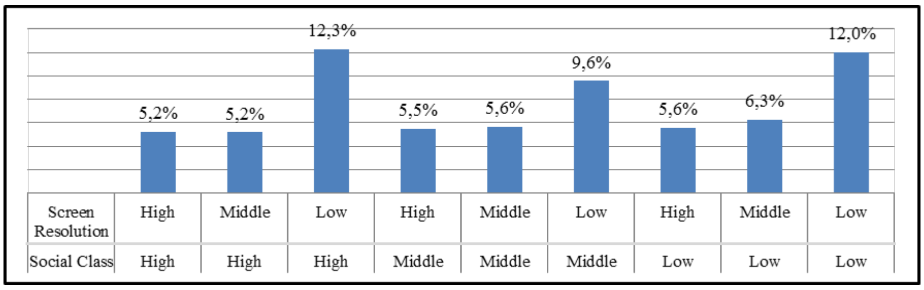

**Figure 15**. Relation between the error rate and the social class and screen resolution

Another relevant factor regarding the applications' context of use is that approximately 70% of the interactions occur when the users are still, having the device in a single position and under normal environment luminosity. However, when those contextual factors change, the users make more mistakes and take longer to execute the tasks. This information suggests that the applications should, for example: (i) make the interaction impossible in positions where there is greater probability of errors, forcing the user to interact in the proper position; (ii) detect the external luminosity and try to balance the luminosity radiated by the device in order to guarantee a good visualization; and (iii) identify the user movement and make the most usual functionalities available, decreasing the visual pollution.

### 4.4 Analyzing Users' Feelings Related to Using the Applications

In order to cover the last question proposed in this study, the ESM technique was used, presenting the questions in Table 2. The aim was to collect subjective data from the users' feelings related to using each application. The analysis of such kind of information allows to detect problems that generate either discomfort or dissatisfaction to users, as well as to learn what attracts users the most to insert applications into their daily routine.

**Table 2.** Questions submitted to users.

| ID | Questions |
|---|---|
| 1 | How satisfied are you with the interface? |
| 2 | How satisfied are you with learning new functionalities? |
| 3 | How satisfied are you with using this application in your daily routine? |
| 4 | How satisfied are you with the application functionalities? |
| 5 | How will you feel when using this application again? |

Figure 16 presents the percentages concerning users' answers to the questions made in the experiment for the application Cubed, according to the description and identification (ID) presented in Table 2.

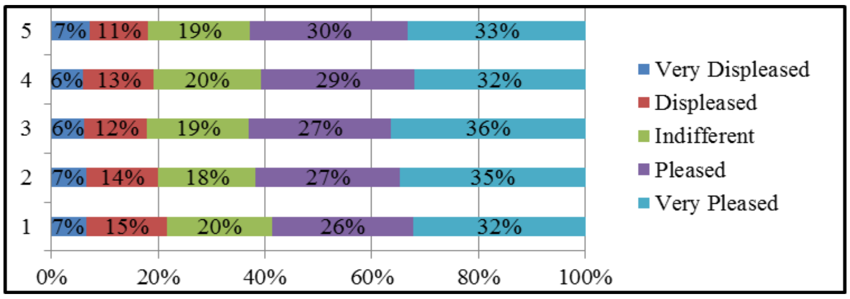

**Figure16**. ESM answers associated to the application Cubed.

A fact that deserves to be highlighted is that the distribution of satisfaction level is maintained constant and increasing from very displeased to very pleased. This trend is observed in all questions submitted to users in the three applications.

Another relevant aspect is that the group of pleased and very pleased users is higher than 50% in all items evaluated. This reveals one of the reasons for the success of the experiment in maintaining the users utilizing the applications during the six months. It is essential that the applications are interesting, useful, attractive, rewarding and easy to interact with. After the findings presented, it is possible to say that the users are satisfied with the applications used in the experiments, answering, thus, the last question asked.

## 5 Related Works

In this section, we describe and discuss some works related to our research that uses at least one of the chosen collecting data used in our proposal to support evaluating mobile applications that is *Logging*, Devices' Sensors and Online Interviewing with users.

The systems evaluation through Logging is not a new strategy as it has been used for over two decades [3]. However, it has gained strength over the last years with the use of AOP to develop it [7][8][9] and the possibility to be applied in smartphones [4][10][11].

The data collection through Logging and sensors in smartphones is the focus of many tools [12][13][14]. These tools, in general, aim to identify the users' behavior and to associate contextual information to its use. The captured data are sent to available servers in the Web for later analysis. The tools LiveLab [11], SystemSens [12] e AnonySense [14] are examples of systems that encompass uncountable logs records, with the objective of monitoring the incoming calls, network traffic, data sharing, battery level, user location, frequency of application use, and CPU and memory use. MobileSens [13] differentiates itself from the previous tools because it uses log records to optimize some smartphones resources in real time such as, reconfiguring the system in order to minimize the battery waste when it is low and indicate more appropriate networks to be used based on the user location. Although the described tools are useful to evaluate users' behavior, they do not enable Online Interviewing with users to identify their satisfaction with the used applications.

The MyExperience [4] stands out in this sense since it uses direct interactions to obtain subjective data that refers to users' satisfaction. The study proposed by Ickin et al [15] presents a mixed methodology to evaluate the user experience, using three different resources: (i) a tool named CSS Application, to capture the log records related to the system use; (ii) ESM, to identify the satisfaction and obtain information about the user interaction context, such as location, social context and mobility level (still, walking, motorized) and (iii) DRM (Day Reconstruction Method), a questionnaire that helps identifying the relationships and results of the Quality of Experience (QoE). Although those tools allow evaluating the user satisfaction, they do not enable identifying specific usability problems, as they don't encompass particular tasks of available applications.

In this context, Paterno et al. [16] proposed a methodology and environment that allows the remote evaluation of mobile applications. The system features a Mobile Logger that collects the information from the mobile device, and a tool called MultiDevice RemUsine that processes the logged information and provides the necessary visualizations to analyze the application usability.

Another similar approach by Au et al [17] was done to test usability of handheld applications. Different aspects considered in testing the usability of handheld device applications and a proposed a list of functional requirements that automated usability testing tools should have in order to be effective were discussed. The Handheld device User Interface Analysis (HUIA) testing framework was then developed and meets most of the requirements proposed.

The EvaHelper [10] framework is another proposal for usability evaluation. Its structure is divided in three aspects: preparation of applications to be tracked, extraction of data obtained and results evaluation. Whilst its structure is similar to the one proposed in this paper, many differences can be highlighted: (i) it does not use AOP, which generates direct intrusion in the application original source code; (ii) it does not have a tool to automatically map the tasks; (iii) it does not have a tool to instrument the code automatically, without the knowledge of APIs and programming in the Android platform and, finally, (iv) it is restricted to statistical evaluations, that is, it does not make the contextual and subjective data capture possible.

## 6 Conclusions

This article presented a hybrid model to capture and assess data from smartphones applications. The units of the model were structured in components that can be redefined according to evaluators' needs. Basically, the model can be described in three units: Mapping Unit, Traceability Unit, and Assessment Unit.

The proposal has as main contributions the possibility to evaluate the smartphone applications' usability by integrating, in the same experiment, metrics data, contextual data and users' feelings according to the used applications. In the literature, there is a lack of proposals that relate those three types of data and that support the performance of field experiments with users using the applications in their daily lives.

Other contributions that can be highlighted are:

- The implementation of the components that interfere in the source code of the applications using AOP, so the aspects can be inserted and removed according to evaluators' needs.
- The creation of tools that enable the use of the infrastructure and the instrumentation of applications without the need for programming.
- The provision of a database that allows continuous assessments in order to identify usability problems with a large number of users and for long periods of time.

For future research, new metrics must be developed, prioritizing common attributes of certain groups of applications. Moreover, there is an intention to incorporate to the infrastructure interactions with users that permit to identify their location and social context. Other studies will be published to present the results of the experiments carried out with the objective of contributing to new findings in the Mobile HCI area.

## References

1 Queiroz, J. E. R. and Ferreira, D. S.: A Multidimensional Approach for the Evaluation of Mobile Application User Interfaces. Human-Computer Interaction, New Trends. Lecture Notes in Computer Science (2009)

2 Zhang, D. and Adipat, B.: Challenges, Methodologies, and Issues in the Usability Testing of Mobile Applications. International Journal of Human-Computer Interaction, 18(3), pp. 293--308 (2005).

3 Ivory, M. Y., Hearst, M. A.: The state of the art in automating usability evaluation of user interfaces. ACM Computer Survey, vol. 33, pp. 470--516 (2001)

4 Froehlich, J., Chen, M., Consolvo, S., Harrison, B. and Landay, J.: MyExperience: A System for In Situ Tracing and Capturing of User Feedback on Mobile Phones. Mobile Systems, pp. 57--70. Application ACM Press (2007)

5 Meschtscherjakov, A., Weiss, A. and Scherndl; T.: Utilizing Emoticons on Mobile Devices within ESM studies to Measure Emotions in the Field. Proc. MME in conjunction with MobileHCI'09, pp. 3361--3366. Bonn, Germany. ACM (2009)

6 Kiczales, G., Lamping, J., Mendhekar, A., Maeda, C., Lopes, C. V., Loingtier, J. M. and Irwin, J.: Aspect-oriented programming. Proceedings of the European Conference on Object-Oriented Programming (ECOOP). Vol. LNCS 1241. Springer-Verlag (1997)

7 Moldovan, G. S. and Tarta, A.: Automatic Usability Evaluation using AOP. In: IEEE International Conference on Automation, Quality and Testing, Robotics, vol. 2, pp. 84--89. Los Alamitos. IEEE Computer Society (2006)

8 Tao, Y.: Toward Computer-Aided Usability Evaluation for Evolving Interactive Software. Proc. ECOOP – Workshop on Reflection, AOP and Meta-Data, pp. 9--16 (2007)

9 Bateman, S., Gutwin, C., Osgood, N. and McCalla. G.: Interactive usability instrumentation. In EICS '09: 1st ACM SIGCHI symposium on Engineering interactive computing systems, pp. 45--54. ACM (2009)

10 Balagtas-Fernandez, F. and Hussmann, H.: A methodology and framework to simplify usability analysis of mobile applications. In: ASE'09: International Conference on Automated Software Engineering, pp. 520--524. IEEE Computer Society (2009)

11 Shepard, C., Rahmati, A., Tossell, C., Zhong, L. and Kortum, P.: LiveLab: measuring wireless networks and smartphone users in the field. ACM SIGMETRICS Performance Evaluation Review, p. 15--20 (2011)

12 Falaki, H., Mahajan, R. and Estrin, D.: SystemSens: a tool for monitoring usage in smartphone research deployments. In. 6th ACM Int. Work on Mobility in the Evolving Internet Architecture (2011)

13 Guo, R.; Zhu, T.; Wang, Y. and Xu, X.: MobileSens: A Framework of Behavior Logger on Android Mobile Device. 6th International Conference Pervasive Computing and Applications (ICPCA), pp. 281--286 (2011)

14 Shin, M., Cornelius, C., Peebles, D., Kapadia, A., Kotz , D. and Triandopoulos, N.: AnonySense: A System for Anonymous Opportunistic Sensing. Pervasive and Mobile Computing, pp. 16--30 (2011)

15 Ickin, S., Wac, K., Fiedler, M., Janowski, L., Hong, J. and Dey, A. K.: Factors Influencing Quality of Experience of Commonly Used Mobile Applications. IEEE Communications Magazine, pp. 48--56 (2012)

16 Paterno, F., Russino, A. and Santoro, C. Remote evaluation of mobile applications. Task Models and Diagrams for User Interface Design, pp. 155--169 (2007).

17 Au, F. T. W., Baker, S., Warren, I. and Dobbie, G.: Automated Usability Testing Framework. In Proceedings of the 9th Australasian User Interface Conference. Australian Computer Society, Inc. Volume 76 (2008)