Effectiveness Measurement Framework for Field-Based Experiments Focused on Android Devices

Ivan Pretel and Ana B. Lago

Deusto Institute of Technology - DeustoTech
MORElab – Envisioning Future Internet
University of Deusto, Avda. Universidades 24, 48007 - Bilbao, Spain
{ivan.pretel, anabelen.lago}@deusto.es

Abstract. Most of the mobile phones have turned into full-connected devices. This provides companies with a perfect channel to interact with their potential clients and employees. The quality of the experience with these applications can directly affect the profits of the company it represents. Focusing on the mobile field and its extremely dynamic context, the quality of the experience can highly fluctuate. Inside this field, several methods and tools have been developed by defining a context of use. However, current methods can only capture it through adding external capture tools (added cameras, human observers...) that can change the experience. The main contribution in this article is a new approach to automatically measure effectiveness through a tiny but powerful mobile tool that can capture interaction metrics and the surrounding context without biasing the measured experience.

Keywords: Mobile HCI, evaluation, effectiveness, context, quality, usability, mobile services, framework

1 Introduction

In the last few years mobile devices are gaining more and more importance to perform tasks not only in our leisure time but also at work. Companies are progressively increasing the number of services connected to the virtual world through the mobile devices. Thanks to these devices they can expose their business models to everywhere by developing and deploying a tiny mobile application. The main aim of this kind of applications is to enable potential users to interact with the business model of the company from everywhere.

Testing tools have drastically changed and have been focusing on the web domain. However, software applications must be focused not only on the web domain but also on mobile devices. According to the last Cisco Visual Networking Index [1], the average smartphone usage has nearly tripled in 2011 and the

[1] Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf

average amount of traffic per smartphone in 2011 was 150 MB per month, up from 55 MB per month in 2010. Paying attention to this report, software testing tools should evolve into web and mobile at the same time.

The quality of mobile applications can extremely fluctuate depending on the context. Therefore, capture information without biasing the context is very important.

In this article we expose a mobile-based tool to automatically evaluate the effectiveness of interactions and capture metrics of the surrounding context. It is formed by a tiny Android library developed for mobile applications used to log interactions and a context model. It is also formed by a web server to remotely store all information. A preliminary version of the system which can capture the majority of the mobile context model was developed and validated through a tiny mobile game.

Firstly, the main definitions of effectiveness, usability and quality based on standards are studied in Section 2. Secondly, a context model focused on mobile devices is explained and a new approach to capture it is defined through the Section 3. In Section 4, the existing systems to capture the context model are studied and the capture tool is presented. After describing it, a brief experiment and its results are shown in Section 5. Finally, the research is concluded and further work is discussed in Section 6.

2 Usability, Quality and Effectiveness

According to the ISO 9241-11 standard [3] usability is the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use. This standard defines effectiveness as the level of accuracy and completeness with which users achieve specified goals. As we will see later, it uses the same definition provided in ISO 9126-4 [4] to define the effectiveness but does not provide a general rule for how measures should be chosen or combined. In fact, it delegates the responsibility for developing the proper metrics to the product developers. This is because the importance of components of usability depends on the context of use and the software which is going to be tested. According to the ISO/IEC 9126 standard, quality represents a property of the software product defined in terms of a set of interdependent attributes (such as usability, security, reliability, performance, complexity, readability, reusability) expressed at different levels of detail and also taken into account the particular context of software use.

The different attributes can measure three different quality aspects: Internal Quality, External Quality and Quality in Use. Internal Quality is the totality of attributes of the software product from an internal view (e.g. spent resources, analysability). It is measured and improved during the code implementation, reviewing and testing. External Quality is the quality when software is running in terms of its behaviour (e.g. number of wrong expected reactions).

Quality in use is the quality of software that user can perceive when the software is used in an explicit context of use. It measures the extent to which

users can complete their tasks in a particular environment. It is measured by four main capabilities of the software product in a specified context of use: effectiveness, satisfaction, productivity and safety.

Each capability is made up by several metrics which can be measured through designing and performing experiments. This work is centred on measuring the effectiveness inside the quality in use aspect, where the effectiveness is formed by three main metrics (see Table 1): Task Effectiveness (TE), Task Completion (TC) and Error Frequency (EF). These metrics measure the accuracy and completeness with which goals can be achieved.

Table 1. Effectiveness metrics defined by ISO/IEC 9126

Metric	Formula	Definition
Task Effectiveness (TE)	$\|1 - \sum Ai\|$	What proportion of the goals is achieved?
Task Completion (TC)	TCM/TA	What proportion of the tasks is completed?
Error Frequency (EF)	E/T	What is the frequency of errors?

Task Effectiveness measures the quantity of the goals achieved by a user. It is measured summing the number of errors (Ai) appeared during the task. Many errors could be more important than others. In order to solve so, each kind of error has its associate weight. TEs value will be between 0 and 1, the closer to 1 the better.

Task Completion measures the level of success the user achieves performing tasks. In contrast to the Task Effectiveness, this metric assumes that the tasks can be performed without the chance of being partially completed. In this case, it is calculated by the number of tasks completed (TCM) divided by the number of tasks attempted (TA). TCs value will be between 0 and 1, the closer to 1 the better.

Finally, Error Frequency measures the number of times that an error is made within a given period. It is calculated dividing the number of errors (E) by the task time or the number of total steps (T). This metric is very useful for making comparisons if errors have equal importance, or are weighted. EFs value will be between 0 and 1, the closer to 0 the better.

Centring on the usability and quality in use fields, effectiveness does not take account how the goals were achieved, only the extent to which they were achieved. Effectiveness, which defines and forms usability and quality in use, can drastically affect to the quality and usability. Furthermore, the defined effectiveness may be biased by the surrounding context of use.

3 Context Model focused on Mobile Devices

Throughout the different definitions of effectiveness the context of use has been appearing as an element which can bias effectiveness, usability and quality. Many

decades ago the context of use has been taken into account and has been defined several times by a lot of researchers, experts and communities.

Several studies maintain that a context is just the physical location [10]. Others add to the location more attributes such as weather [2] to achieve a more accurate definition of the environment and physical context. Other studies expand the limit of the context of use adding the community [6] and the stakeholders [7]. They also maintain that context should be defined by answering where the user is, who the user is with, and what resources are nearby. Others [9] define the context by enumerating more parameters such as goals, attention, connection

After seeing the different definitions of contexts and paying special attention to the work by Abowd et al. [1] we have defined a preliminary context of use model focused on the mobile field. We conclude that components which define the context (see Fig. 1) are: the user, the mobile device and the environment (physical, ambient, technical and sociocultural). User is defined by four main groups of attributes: personal information, knowledge, skills and attitudes. The mobile device is formed by six groups: connections, body, inputs, outputs, battery and software features. Finally, the environment is made by four main groups: physical, ambient, technical and sociocultural groups.

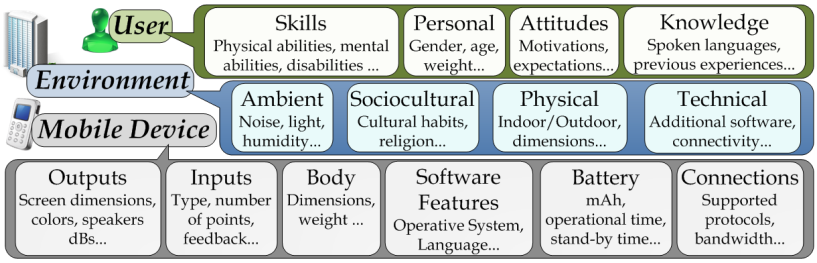

Fig. 1. Components which define the context

4 Context Model Capturer

Once the context of use model is defined the next step is to design the capturing method. To capture the context model and the interactions, it is essential to capture objective information using the mobile device in real environments. This can hardly be captured with a lab-based framework (such as Morae Observer[2]) which logs information in a highly controlled environment using specific devices and users. The field-based evaluation frameworks [8] [5] can provide more objective information because they are performed in real environments, but the added agents such as cameras and invasive evaluation methods (e.g. surveys

[2] Morae Observer - http://www.techsmith.com/morae.html

during tasks) have to be removed. Therefore, the best way to capture interaction data and the context model is by registering information through a mobile device using a tiny capture tool. So as to implement a capturer without biasing the context we have to use only elements that make up the context (in this case, the mobile device). This tool should capture the context model via the built-in mobile sensors and logging interaction events. Although the context model has been defined, the preliminary version only can capture a small subset of data and assumes all the errors have the same relevance. The purposed system (see Fig. 2) is formed by a tiny Android library developed for mobile applications and a server to store and log the performed interactions. First, users should download an application-to-test (ATT) from the server. They sign up for the platform, use the application and upload their interactions to the platform. The ATT should be integrated with the developed library. This library automatically captures context and interaction information through a context model module and stores it in a local database using an interaction store module. When the device has internet connection and its owner wants it, all the information is uploaded.

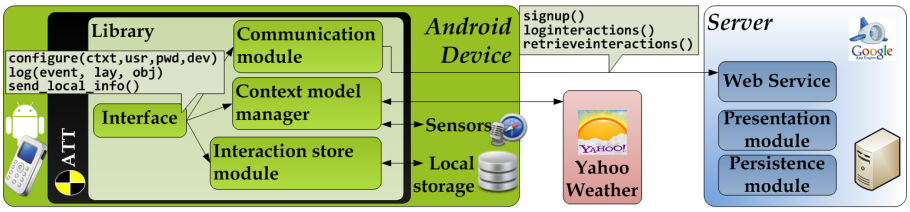

Fig. 2. Context Model Capturer Architecture

4.1 Server

The server is developed by the Google App Engine technology [3]. Its main aim is to be the main gate between users and experimenters. It remotely stores all the interaction data in the cloud. It offers several services through a web interface. Through these services users can sign up, log interactions and see their logged interactions. All the information is stored in the cloud by Google Cloud SQL [4].

4.2 Android Library

In order to capture information from the context model this library allows programmers to abstract the capture. Developers only need to implement their own application and insert small log lines inside their ATT logic to log interactions.

[3] Google App Engine: build and host web application - http://developers.google.com/appengine
[4] Google Cloud SQL - http://cloud.google.com

This library only exposes three instructions. The CONFIGURE instruction sets up the capturer with the needed parameters to work. It does not start the capture, only sets the username, pass, registered id of the device and the context (interface provided by Android to see the global information about the application environment). SEND_LOCAL_INFO sends the captured and locally stored information to the server. LOG captures the interaction timestamp, context and the object with witch user is interacting.

A task (see Fig. 3) can pass through four main states: When a task is not started yet (NOT STARTED), when a task is started and its user is interacting to achieve the goal of the task (STARTED), when a task is started but its user is not interacting to achieve it (PAUSED) and when the task is finally ended (ENDED). During a task performance a user can trigger two main events: START_TASK and END_TASK. Additionally there exist two others: if user leaves the task (e.g. phone call) the PAUSE_TASK event is triggered. Where user decides to continue the task RESUME_TASK is produced. When a task is started two events related to the interaction of the user can be triggered. The INTERACTION event means that a user is interacting in the right way. This event should be triggered when a user is achieving little microchallenges inside the goal of the task. The ERROR event means that a user has made a mistake.

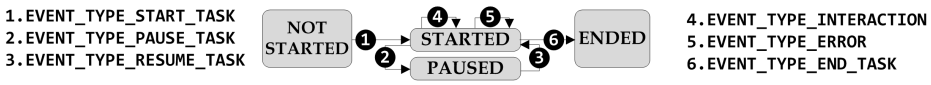

Fig. 3. States of a task and its events

The context model is captured by using all the built-in sensors and the application programming interfaces (APIs) provided by Android. The information related to the user is retrieved during the signup because the changing frequency of this information is very low. The nickname, gender, birthday, height, weight, if the user is left-handed, right-handed or ambidextrous and the European level of several languages are stored. Mobile device information is captured in two different phases: during the sign up (DeviceID, OS version, productID, display model, country and its configured language) and during the task performance. When the user is performing tasks information related to the sound level of outputs is captured (i.e. alarm volume, ringtone volume, if headphones are used). Information related to the battery (battery level, charging/unplugged) as well as the application display properties (density, height, and width) are also captured. Information related to the environment is captured during the task performance because of the high frequency of changes. Noise and light levels are captured by sensors. Location and connection information (coordinates, the connection type...) is captured by internal services provided by Android. If the device is connected the current weather conditions are requested to Yahoo Weather[5]).

[5] Yahoo! Weather Developer Network - http://developer.yahoo.com/weather/

5 System Validation

The preliminary version of the system was validated performing a brief experiment. 4 subjects have been signed up from the platform and have played a memory game application which contains the library. They have been playing during one day and they have generated more than 1400 logged lines. They played in four contexts: at home (H), walking down the street (S), travelling (PT) by a public transport and at work (W), more concretely, in an office.

First, users have to sign up on the testing platform and complete information related to them and their device. Finally, user presses play button to start the game, selects a context name(H, S, PT or W) and the game starts. It is worth taking into consideration that users must manually select the context name. This is because the framework captures the conforming context variables so as to analyse how they affect to the experience, and thus it is not its aim to address the context inference issue. The memory game application is a card game in which a player deals out a set of cards face down. In one turn, the player flips over two cards (one interaction). If they match, the player leaves them face up and solves a simple add to continue playing (it is one interaction). If they do not match (error), the player flips the cards back face down.

Focusing on the effectiveness measurement, the task to perform is clear: to end up with all of the cards flipped face up in less than 15 turns. There are 8 pairs of cards; it means the best round is made by only 8 interactions and 0 errors. If a user spends more than 15 turns, it loses and this task is not completed. Task Completion (TC) is calculated counting all the rounds won by a subject divided by all rounds this subject has played. The Task Effectiveness (TE) metric is calculated subtracting 1 to the number of made errors during the task multiplied by the weight of the error. The weight of an error is 1/7 because the maximum number of errors you can make is 6 with 8 correct interactions. If you make 7 errors the TC should be 0 (1 (7*1/7) = 0). Error Frequency (EF) is calculated dividing the error number by the number of total turns.

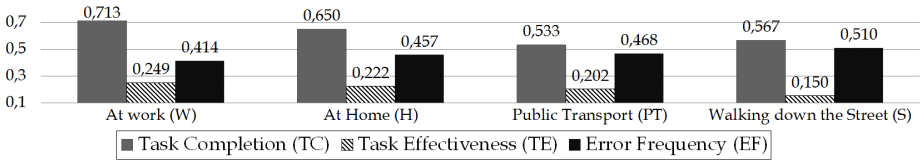

Fig. 4. Effectiveness results grouped by contexts

The results grouped by contexts (see Fig. 4) shows the outdoor contexts (PT and S) affect subjects and lead them to be less effective than in the indoor contexts (H and W). TE indicates that the S context is the context where subjects have the worst results. Although EF exposes that is in W where subjects make more mistakes, it is also the most efficient context.

6 Conclusion

Through this work we have defined effectiveness metrics, which can bias the usability and the quality of mobile applications. We have also studied context models and we have defined one based on mobile devices. This context model is formed by several groups of attributes which have to be captured and logged. Based on the study of the lab-based and field-based capturing methods and focusing the design on mobile environments we conclude the best way to capture interaction data and the defined context model is through the used mobile device.

The purposed tool is formed by a tiny Android library used to log interactions and the context model and a server to store all the captured data. Finally, a preliminary version of the system which can capture the majority of the context model and a tiny application to test it were developed. The results of the exposed experiment demonstrate the effectiveness and the context can be automatically measured by an automatic tool without biasing the interaction with external agents. The next step is to study all the captured attributes of the context model to calculate correlations with the effectiveness as well as keep enhancing the captured attributes and the system performance.

References

1. Abowd, G.D., Dey, A.K., Brown, P.J., Davies, N., Smith, M., Steggles, P.: Towards a better understanding of context and context-awareness. In: Handheld and ubiquitous computing. pp. 304–307. Springer (1999)
2. Brown, P.J., Bovey, J.D., Chen, X.: Context-aware applications: from the laboratory to the marketplace. Personal Communications, IEEE 4(5), 58–64 (1997)
3. ISO (ed.): ISO 9241-11:1998(E): Ergonomic requirements for office work with visual display terminals (VDTs) Part 11: Guidance on usability (1998)
4. ISO (ed.): ISO/IEC TR 9126-4:2004(E): Software engineering Product quality Part 4: Quality in use metrics) (2004)
5. Jensen, K.L.: Recon: capturing mobile and ubiquitous interaction in real contexts. In: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services. p. 76. ACM (2009)
6. Kankainen, A., et al.: Thinking model and tools for understanding user experience related to information appliance product concepts. Helsinki University of Technology (2002)
7. {National Institute of Standards and Technology}: Common industry specification for usability requirements (NISTIR 7432) (Jun 2007)
8. Raento, M., Oulasvirta, A., Petit, R., Toivonen, H.: Contextphone: A prototyping platform for context-aware mobile applications. Pervasive Computing, IEEE 4(2), 51–59 (2005)
9. Ryan, N.S., Pascoe, J., Morse, D.R.: Enhanced reality fieldwork: the context-aware archaeological assistant. In: Computer applications in archaeology (1998)
10. Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. Network, IEEE 8(5), 22–32 (1994)