

A FRAMEWORK TO AID USABILITY TESTING AND USER EXPERIENCE MEASUREMENT ON MOBILE DEVICES

Ákos Nagy¹, Bence Kővári²

B.Sc. Student, Assistant Lecturer

Budapest University of Technology and Economics

Faculty of Electrical Engineering and Informatics

Department of Automation and Applied Informatics

ABSTRACT

When developing mobile applications, developers must not only produce robust and functional code, but also respect their users' habits and provide a great user experience. To do this, they have to collect and analyze data regarding these habits, but this is a time-consuming and expensive process. In this paper we present the theoretical basis and a concrete implementation of a framework that helps developers by automating the collection phase. The acquired data is suitable to deduct information about user habits regarding the use of the device and the application, to monitor the utilization of application features and to support other types of usability tests. The framework is also suitable to supervise functional testing of an application.

INTRODUCTION

Forecasts of the International Data Corporation indicated that in 2012, more than 106 million tablets will have been sold by the end of the year, and in 2016 almost 200 million tablet shipments are expected [1]. Statistics published by PortioResearch show an even bigger growth in the market for smartphones [2]. The numbers warn developers that the demands of mobile device users must be heard.

This growth also enforces a change in software development paradigms. Designing and testing the user experience (UX) is nowadays an important part of the software engineering lifecycle. But while designing and testing functional requirements is supported by a variety of tools and methods, in the case of user experience both the toolset and the methodology is still very new, incomplete and the results are not guaranteed.

The introduction of different sensors into the mobile devices – such as the inclinometer, the accelerometer, the touch screen and so on –, can give us a new way to monitor user habits and deduct information about them. This data, combined with other measurements regarding utilization of different features can provide ideas about how to improve software to make it more user-friendly and can also be used to monitor other testing processes or to detect poor coverage.

RELATED WORK

With the aggressive spreading of touch-based mobile devices, many studies have been carried out to measure and observe different aspects of their use.

Measuring UX usually involves giving different tasks to a set of users and monitoring them, while recording their performance or asking them questions about the experience afterwards. Quantitative testing usually involves a formal protocol, which uses mostly statistical methods to analyze and compare results. Qualitative testing is less formal, focuses mostly on the impressions and opinions of users about the software. The following studies usually perform qualitative tests to determine usability.

In [3], experiments are described to determine muscle fatigue in the function of the tilt angle when using a device with pen-based input. The authors concluded that the best overall comfort level is at a tilt angle of 25°. Similar studies in [4] show, that the most suitable angle is 30°.

Analyzing gestures has also a crucial role in UX measurement. In [5], the authors present a gesture recognition using the accelerometer of the device. [6] examines how the environment affects the users' ability to perform different gestures and conclude that some gestures require less attention to carry out than others.

Finally, in [7] the authors present a library to automate testing of gesture-based applications. While this work focuses more on how to test functional code, the described gesture recording can be used for other purposes.

Most of the works mentioned above focus on the analysis of the data collected via traditional methods. The main contribution of this paper is the introduction of a framework that can automate the process of collecting this data, which can be analyzed and compared to the results of the previous studies.

CONTRIBUTIONS

First, the theoretical basics of the framework are presented. An implementation is also discussed for the new Windows 8 platform, but respecting the described principles, implementations can be developed for other platforms.

The main requirements are simple: on one hand, the module must be easy-to-insert into existing applications. This means that the least possible new code has to be introduced at the least possible number of locations. On the other hand, not just the initialization process, but also the collecting process must be highly automatized, i.e. after initialization no further modifications should be required in the application. In practice, these tenets mean that an implemented framework:

- has to be easy to initialize with a few lines of code in one place
- automatically hook up to the available sensors and log their data
- automatically process the user interface (UI) and hook up to the events of the different controls to log them and their parameters
- save the logged data in an efficient form to the user's device
- periodically send the data to a server

- all of the tasks above must be done efficiently to not interfere with the user experience of the application

In the Windows 8 implementation, the easy initialization was achieved: depending on whether application lifecycle events are logged or not, 6 or 9 lines of code are enough to set up the framework. This code segment is always the same and has to be inserted at one location.

Upon initialization, the framework determines which sensors are available on the platform and automatically initializes the measurement process, it is transparent to the developers using the module.

Pages are also processed automatically when they are displayed. The UI of a page can be represented with a graph, which is called the visual tree. The nodes of this tree represent the controls of the UI and edges stand for container-control relationships. This tree is traversed and the controls are examined for events using reflection.

Events and measurements are temporarily saved to the device in a compact binary form. When the size of the saved the data reaches a limit, it is sent to a server using a simple WCF service. The service is also aware of this binary format, so it can deserialize the information and write it into a database for future use.

When possible, all the steps of the tasks mentioned above are done asynchronously, so the logging and sending does not cause the UI to become unresponsive, maintaining the UX of the monitored application.

CASE STUDY

In this section we present how the data collected using the framework described above can be analyzed and prove its usefulness.

For the first test run of the module, we inserted it into a Windows 8 application and used it throughout the beta testing phase; almost 40000 entries were collected during a period of one month. While these testers are not everyday users, from an ergonomic point of view having experience with IT applications makes no difference, thus not making our conclusions less valid.

Table 1 shows how much time the testers spent testing each page of the application.

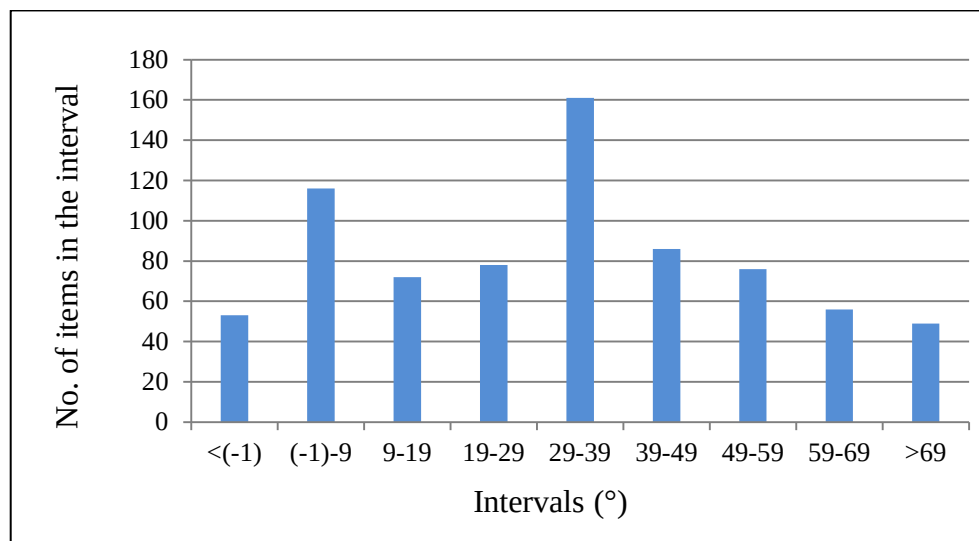
Table 1
Time spent on pages

Page	Avg. time	Total time	Total time share
BuyPage	0:01:03	1:28:58	13,49
ExceptionPage	0:01:40	0:35:06	5,32
GalleryPage	0:00:08	0:00:15	0,04
InterestedProductsPage	0:00:44	0:14:33	2,21
MainPage	0:00:37	5:58:17	54,31
NoServicePage	0:00:49	0:26:44	4,05
ProductPage	0:00:10	0:13:49	2,09
ProfilePage	0:01:03	0:17:57	2,72
SearchProductsPage	0:00:22	1:33:34	14,18
UserPage	0:00:19	0:00:57	0,14
WatchedProductsPage	0:00:14	0:09:34	1,45

Table 1 shows that the testers spent a small amount of time on the GalleryPage and the UserPage, thus making these pages more error prone in the final version of the application. Had the module been used in a production environment, the results would mean that only few users visited the GalleryPage and maybe it is necessary to reorganize the content to attract more visitors or remove the page entirely. Though not groundbreaking in concept, there is no similar framework for the platform this time that can do such measurements.

The second main contribution of the framework is the analysis of sensory data. As an example, Figure 1 visualizes collected information about the tilt angle of the tablet.

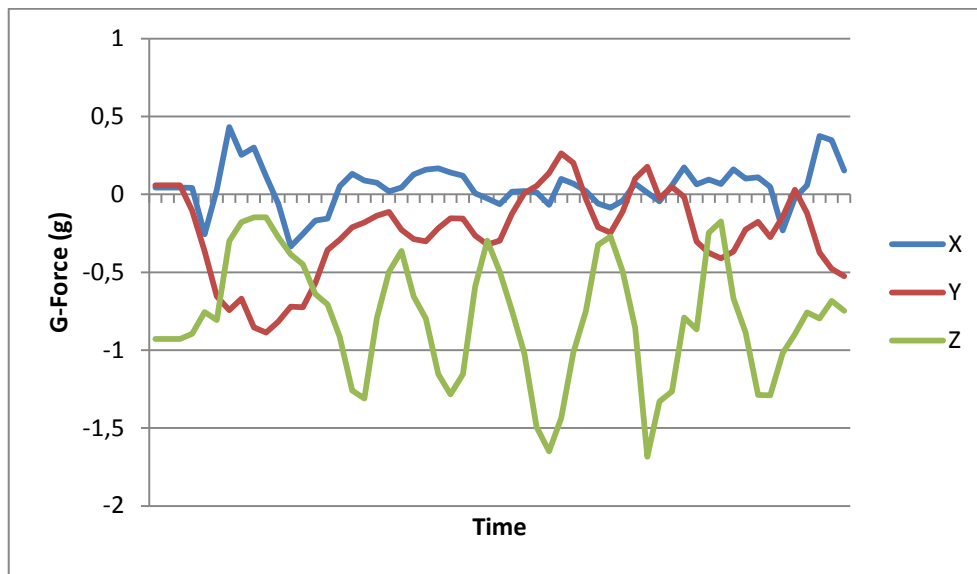
Figure 1
Histogram of tilt angle



Examining this histogram shows that the most widely used category (29-39) corresponds with the one determined in [3] and [4], making our application comfortable to use (the other category ((-1)-9) involves the scenario when the tablet was on lying on a flat surface).

Sensory data can also show if a user walks with the device. Figure 2 shows the accelerometer measurements during walking, which conforms with the measurements of [8].

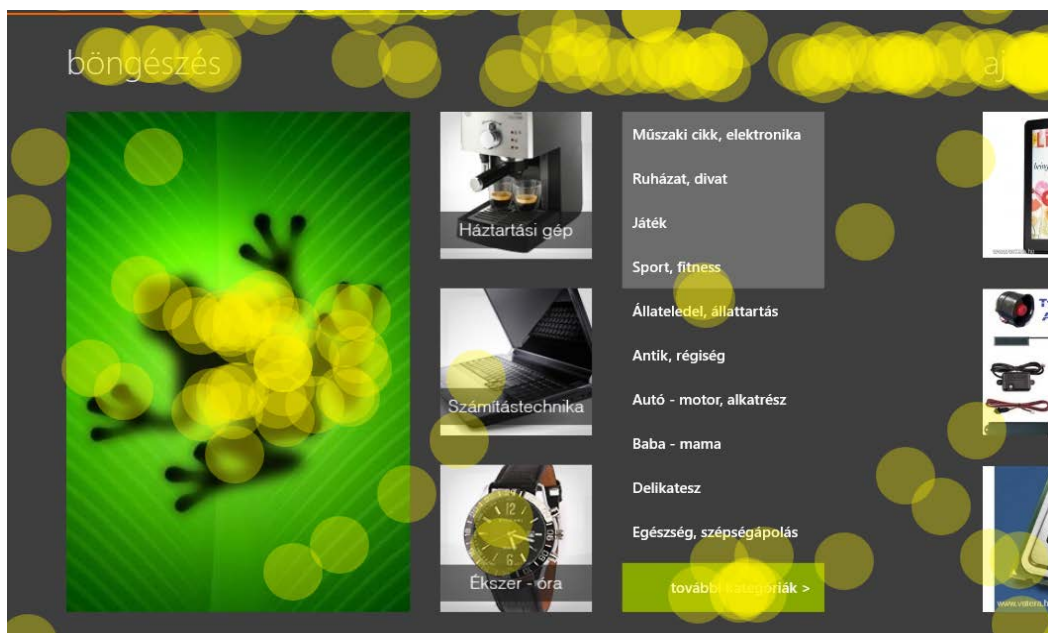
Figure 2
Walking with the device



Examining how often users walk when using the application is important, because this means that they hold the tablet in one hand, making that hand unavailable to perform gestures.

Figure 3 shows the heat map – this is a graphic representation of how active different sections of the UI are (i.e. how many times users tap on the given areas).

Figure 3
Heat map



Examining the heat map helps to identify the areas that users expect to be interactive but are not, and vice versa.

Other analysis options include gesture recognition, navigation graph drawing, determining the most frequent input device, examining user habits regarding scrolling, examining most frequent orientations.

CONCLUSIONS AND FUTURE WORK

This paper discussed a framework aiding the development of more UX-aware applications. We discussed the theoretical and technical requirements of such a framework, then briefly presented an implementation for the Windows 8 platform. Finally we highlighted some usage scenarios for the collected data.

The most important task in the future is to determine which other mobile platforms provide the necessary functionality to implement a framework meeting the requirements presented in this paper, developing the module for those platforms and expand our current findings to incorporate the habits of the users of those platforms.

REFERENCES

- [1] "Media Tablet Shipments Outpace Fourth Quarter Targets," 13 march 2012. [Online]. Available: <http://www.idc.com/getdoc.jsp?containerId=prUS23371312> [Accessed 11 january 2013].
- [2] "Portio Research Mobile Factbook," Portio Research, 2012.
- [3] C.-C. Yen, "A Survey of Physical Fatigue during Use of a Tablet LCD Monitor," *Journal of C.C.I.T.*, pp. 49-62, may 2011.
- [4] D. Bao, Y. Xin and X. Ren, "Effect of Tilt Angle of Tablet on Pen-based Input Operation Based on Fitts' Law," *Proceedings of the 2010 IEEE International Conference on Information and Automation*, pp. 99-104, 20 june 2010.
- [5] V.-M. Mäntylä, J. Mäntyjärvi, T. Seppänen and E. Tuulari, "Hand gesture recognition of a mobile device user," *2000 IEEE International Conference on Multimedia and Expo, 2000. ICME 2000.*, pp. 281-284, 2000.
- [6] A. Bragdon, E. Nelson, Y. Li and K. Hinckley, "Experimental Analysis of Touch-Screen Gesture Designs in Mobile Environments," *ACM CHI Conference on Human Factors in Computing Systems*, pp. 403-412, 2011.
- [7] S. H. Khandkar, S. M. Sohan, J. Sillitio and F. Maurer, "Tool Support for Testing Complex Multi-Touch Gestures," *Proceedings of the 2010 ACM International Conference on Interactive Tabletops and Surfaces*, pp. 59-68, 2010.
- [8] S. Sinofsky, "MSDN Blogs: Supporting sensors in Windows 8," 24 january 2012. [Online]. Available: <http://blogs.msdn.com/b/b8/archive/2012/01/24/supporting-sensors-in-windows-8.aspx> [Accessed 14 january 2013].