Improving the User Experience using an Intelligent Adaptive User Interface in Mobile Applications.

Conference Paper - November 2016
DOI: 10.1109/IMCET.2016.7777428

1 author:

Saeed Raheel
American University of Beirut
12 PUBLICATIONS 61 CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Deep Learning View project
Improving the User Experience using an Intelligent Adaptive User Interface in Mobile Applications

Saeed Raheel
Computer Science Department
American University of Science and Technology
Beirut, Lebanon

Abstract— nowadays, we witness an abundance in mobile applications and a richness in their functionality. However, many of them were developed with insufficient attention when it comes to usability and user friendliness. As such, we may find one application’s interface clogged with controls that hamper the user from using it properly or another’s interface designed without taking care of universal usability. The result is that the user experience with different applications goes from being a total satisfaction to a total frustration. Very good care should be taken while designing mobile applications since the mobile phone are constrained with a limited screen size and a few interaction mechanisms. Adaptive User Interfaces (AUIs) were proposed in order to mitigate the negative impact of these constraints and, as such, help improve the application’s usability. In this paper, we propose a number of intelligent adaptive mechanisms capable of monitoring the user’s behavior on a mobile phone and then adapting the interface accordingly. Our proposed mechanisms yield significant usability benefits and make the user’s experience with this application more satisfying.

Keywords—Human computer interaction, adaptive interfaces, user experience, mobile applications.

I. INTRODUCTION

Human-Computer Interaction (HCI) studies the interaction between user and a User Interface (UI). HCI is either explicit or implicit. Explicit HCI handles the direct manipulation of an interface by a certain user (e.g. pressing a button on the screen). The system reacts based on the action done directly by the user (touching the screen). On the other hand, the social and context information analysis allows for an implicit Human-Computer Interaction [1, 2]. This is done, mainly, by using the different sensors that the mobile phone is equipped with (e.g. determining the location (e.g. standing in front or far by an angle from the mobile phone) of the user with respect to the screen or determining the distance between the user and the screen). As more implicit data is collected and analyzed, the amount of explicit HCI required lessen thus making the application more user friendly. This has a direct positive impact on the user experience.

One major problem that HCI research tries to tackle is usability problems and these are mainly due to the physical constraints imposed by mobile devices. These constraints are mainly the result of the lack of flexibility and the complexity of the user interfaces [9]. One very useful approach that addresses these problems are self-learning and self-adaptive interfaces [7]. In other words, rather than having one universal design for all, the intelligent adaptability creates different versions of the same interface in order to satisfy the different needs that emerge by the diversity of its users. According to [13], there are two main approaches to intelligent interfaces: adaptable and adaptive. The earlier requires complex configuration while the latter is easier to apply and is more effective.

In this paper, we propose a number of mechanisms that are render interfaces adaptive to their users’ needs in an attempt to improve their usability. It is organized as follows. Section 2 defines Adaptive User Interfaces and states its advantages and disadvantages. Section 3 presents the design and evaluation of a case study that implements and evaluates different forms of adaptation. Section 4 draws conclusions from the evaluation results obtained in case study.

II. ADAPTIVE USER INTERFACE

AUl is defined as: “a software artifact that improves its ability to interact with a user by constructing a user model based on partial experience with that user” [9]. In other words, an AUl is capable of self-adapting to the user needs by changing its layout and elements [3]. This is done simply because there is a diversity in the types of users, their profiles, and their needs. As a result, it is the responsibility of the system to adapt to its user rather than the user adapting to the system. To do so, the system monitors the user’s behavior and tries to create a model that adapts to his/her needs.

AUl techniques collect explicit data (i.e. by asking the user to specify the options that fit his/her needs) and/or implicit data (i.e. by using sensors, frequently used elements, etc.) and then adjusts the interface accordingly. For example, the AUl might choose to keep and display the elements that are most frequently used or the ones that are mostly used and then hide the others. Another example of AUl is found in Apple’s ® iPhone 6’s perspective wallpaper which shifts and changes perspectives according to the user’s motion. A third one is simply how the mobile phone’s layout is adjusted when the latter is rotated.
AUI techniques try to address the constraints imposed by the screen size and the interaction options by carefully choosing what to show and how the chosen elements are shown. One other advantage that AUI might present in some interfaces is that it limits the number of explicit interactions with the mobile phone so as to allow the user to perform a certain task with the minimum effort possible.

A successful personalization and adaptation improves the usability of the application and hence its acceptance by a broad range of people. Ideally, what happens behind the scenes is that the mobile application records and remembers most of the interactions done by the user in order to customize the application in three ways: visually, touchably, and audibly. For this to work, the interface itself must learn and properly interpret each user’s action. It does this by creating a “user interaction profile” for its user and uses it to adapt the current interface. At a later stage, the profile can be propagated across all the different interfaces of the same application in order to adapt them as well.

AUI presents several advantages. First, by using an AUI that will record all interactions made by the user, the application determines what are the user’s needs and preferences e.g. the accessible screen area (i.e. the area that the user is most frequently able to access with ease), sound level, and orientation. In other words, the application now “understands” its user better. In addition, other applications can derive from the previous set of information that were collected from the original application that hosted the AUIs, in order to adapt themselves without the need to go through the “learning process” yet another time [6]. Second, by creating a “user interaction profile”, we improve the user’s skills needed to manipulate the current application, thus providing a well-designed and structured basis for all the other applications that are compatible (i.e. one that offers similar functionality) and are able to provide such an adaptability. Third, cross-platform adaptability may offer the highest benefits especially in terms of reducing the time required to develop and deploy an application that offers similar functionality since there are many common tasks that require the same pattern of interaction. As a result, if a new compatible application adopts the adaptability aspects of an already existing one then the new application can automatically adapt itself. This will spare the user the hassle of “teaching” the new application how to become adaptive according to his/her interaction profile. Finally, by allowing the interaction profile to propagate across different applications and platforms we are propagating consistency among different applications and across different platforms.

However, one should be careful when applying adaptability because, despite the fact that it improves the usability, self-adaptive interfaces do also pose significant challenges in terms of trust-worthiness and acceptability [7]. The major problem that the user might be facing is the feeling of losing control over the interface due to the fact the changes are being done automatically [13].

### III. CASE STUDY: IMEAD

iMeAid was designed as an example of a medical adaptive mobile application. The main objective of this application is to help the elderly remember taking their medications at specific times.

![Fig 1. Screenshot of iMeAid’s medication reminder](image1)

The application keeps track of the medical record of a certain patient (surgeries, treatments, vaccinations, etc.) in a “Health Book” and provides information about the different drugs and their substitutes as approved by the Lebanese Ministry of Public Health.

![Fig 2. Screenshot of iMeAid’s Health Book](image2)

The application also allows physicians to keep track of their patients’ medical records, check examination reports, create prescriptions, view all the medical history of the patient and manipulate the patient’s Health Book. For the sake of privacy, every physician has access to the above material directly related to him or authorized by someone else. The application provides, as well, a daily health related tip.
The AUI integration started in the reminder page that alerts the user that it is about time to take a medication (cf. Fig 1). We thought that a user who needs to be reminded of his/her medication is most probably an elderly person. Many elderly people have physical problems such as shaky hands, vision problems or limited hearing. The AUI was integrated to help them overcome the fact that they might not hear the alarm, locate or be able to properly hit any of the buttons (the “Snooze” or “Take it” buttons at the very bottom of Fig 1). The AUI’s goal became: “Let the user behave and the application adapt!”.

The conversion of the interface into an AUI was done in two stages. Stage 1 tackled the problem of correctly locating and touching an element on the interface (e.g. pressing one of the two buttons labeled “Snooze” and “Take it” in Fig. 1). The second stage tackled the problem of not being able to hear the alarm’s sound.

Stage 1 suggests two solutions: relocation (with awareness) and resizing:

1. **Relocation (with awareness):** To correctly relocate an element the screen is divided into horizontal zones (that we refer to as buckets) as shown in the figure below:

   ![Fig. 4 Mobile phone sections and distribution](image)

   Then we ask the user to locate and press a certain element on the screen. The above figure shows the attempts of a user trying to press a button that is located in bucket 9 (the GUI elements are hidden simply to show only the touched locations). It is clear that the user is mostly pressing on buckets 3, 7, and 8 (where the number of presses in buckets 3 and 8 is identical and larger than the ones in bucket 7). Therefore, the interface has to adapt to this scenario. The solution to this problem would be relocating the element to one of the two buckets 3 or 8. In order to choose which of the two buckets is the new location of the element, we calculated the standard deviation (SD) between the mean and the other points in each of the 9 buckets. We found out that the SD in bucket 3 is greater than the one in bucket 8. Being far from each other, that means that the user is probably holding the phone with both hands: one of them is touching the screen because it is holding it (the one touching bucket 3) and the other one is being used to touch the element (the one touching bucket 8). Therefore, the interface automatically relocates the element to bucket 8 by shifting it a bit higher.

   After integrating this logic into our interface, we tested it on another one. We added an image and a button labeled “Change my position” to that interface and we asked the user to try to press the button.

   Initially, the interface looks like what is shown in Fig. 5. Knowing that the user has shaky hands, we asked him to try to touch the button. The application stored the locations touched on the screen as shown in Fig. 6 (these locations are denoted to by stars). Accordingly, the button was relocated a number of pixels down to the location that the user was able...
to touch and that helped him touch the button easier than before as shown in Fig. 7.

Another AUI aspect that was integrated in our application is “awareness” among the different elements on a screen. Each element on the interface is cognizant of the other elements’ locations. As a result, if the relocation of an element overlaps or hides partially another one, the affected element will automatically relocate itself as shown in Fig. 8. Indeed, as soon as the button is relocated up, the image automatically looks for the best position and relocates to it. To do this, we classified the GUI elements as being either adaptable or dependable. An element that is adaptable (the button in our case) listens and reacts to the interactions made by the user. An element that is dependable (the image in this case) changes its position relative to the new location of the adaptable element. To classify the elements as being dependable or not, a dependency graph is constructed at runtime that stores for every element a dependability list that tells which element(s) is/are dependable on a given one.

2. Resizing: another solution for correctly locating a certain element is resizing. This approach follows the same methodology as the previous one but, this time, rather than relocating the element to a new location, it is resized to cover the area starting from its original location up/down to the designated location. By increasing the size of a button the use is now able to better see it and correctly locating and touching it.

Stage 2 integrates the audible adaptability in the interface as follows: when it is time to take a medication, the alarm starts with a certain volume level as shown in the figure below:

![Fig. 9 Medication Alarm (Initial Sound Level)](image)

After a short period of time, if the user does not react after 3 alarms then the alarm the volume starts slowly increasing as shown in the figure below:

![Fig. 10 The Alarm volume automatically increasing](image)

When the user reacts to the alarm, the current value of the volume is set as the default one for the future alarms as shown in the figure below:

![Fig. 11 A future alarm starts at the default set volume](image)

If the user wishes to lower the volume, he/she can do this manually and the interface recognizes this change and uses it as the default value for the future alarms. If in next alarm the user does react again after 3 alarm repetitions then the audible adaptability starts all over again. This adaptability helps people with impaired hearing hear the alarm clearly.

IV. CONCLUSION

This paper has shown that AUIs can be very effectively used to overcome some usability problems in mobile applications. The experiments show that integrating adaptive features in a mobile application improves usability, acceptance, and user experience. This study proved that the adaptability mechanisms were very useful in significantly improving usability. Relocation and resizing helped people with shivering hands or impaired vision and volume manipulation helped people with hearing problems.

REFERENCES


