# Equivalence of Navigation Widgets for Mobile Platforms

Amilcar Meneses Viveros[1], Erika Hernández Rubio[2], and Dario Emmanuel Vázquez Ceballos[2]

[1] Departamento de Computación, CINVESTAV-IPN, México D.F.
ameneses@cs.cinvestav.mx
[2] Instituto Politécnico Nacional, SEPI-ESCOM, México D.F.
ehernandezru@ipn.mx

**Abstract.** One of the main features of mobile applications is that its development should be focused on the user. For this reason the design of graphical interfaces must be usable, efficient and effective, among other properties of HCI. The acceptance of mobile applications and the device are affected by personality, cognitive abilities (memory, spatial ability and verbal ability), age, and experience in mobile technology by users. Even if there are methods and techniques to design graphics user interfaces, there is a limited styles for the interfaces for mobile applications. The proliferation of mobile devices has generated the emergence of various platforms. This variety of mobile platforms there has generated a several set of widgets. However, this set of widgets is not cross-platform. That is to say, not all widgets are available on all platforms. Developers have the problem that must generate native applications that have the same level of usability between different platforms or to generate cross-platforms applications that comply with the HCI properties: usability, effectiveness and efficiency. Moreover, if we want to achieve applications to automatically adapt its GUI to the mobile platform (OS and device), requires some equivalence between the widgets on each platform and also, you should know the styles of organization of widgets for each platform. One solution to this problem is to have usability equivalences between different widgets for each mobile platform. We propose the equivalence of widgets with two properties: functionality and usability. Possibly the most important widget sets are related to the navigation of mobile applications. In this paper we present an overview of the widgets of the main mobile platforms and a taxonomy of them. It also presents the study of some equivalences in widgets that allow navigation in mobile applications.

## 1 Introduction

One of the main features of mobile applications is that its development should be focused on the user [1][2][3]. For this reason the design of graphical interfaces must be usable, efficient and effective, among other properties of HCI [1][2][3][4][5]. The acceptance of mobile applications and the device are affected

A. Marcus (Ed.): DUXU 2014, Part II, LNCS 8518, pp. 269–278, 2014.

by personality, cognitive abilities (memory, spatial ability and verbal ability), age, and experience in mobile technology by users [1][2][4][6]. Even if there are methods and techniques to design graphics user interfaces, there is a limited styles for the interfaces for mobile applications [1][2][4][6][7][8][9].

The proliferation of mobile devices has generated the emergence of various platforms [7] [10]. Each platform has its own character [4][8][11][12][13][14]. For example, the way to have navigation applications on tablet is different from the smartphone. Another example is that the control of an application depends largely on the platform where it is implemented. This variety of mobile platforms there has generated a several set of widgets [1][7] [15]. However, this set of widgets is not cross-platform. That is to say, not all widgets are available on all platforms

Developers have the problem that must generate native applications that have the same level of usability between different platforms [1][3][5] [18] or to generate cross-platforms applications that comply with the HCI properties: usability, effectiveness and efficiency [1][7][9]. Moreover, if we want to achieve applications to automatically adapt its GUI to the mobile platform (OS and device), requires some equivalence between the widgets on each platform and also, you should know the styles of organization of widgets for each platform [1][9].

One solution to this problem is to have usability equivalences between different widgets for each mobile platform. Possibly the most important widget sets are related to the navigation of mobile applications. Android, for example, makes a distinction between deep and navigation navigation at the top, allowing you to use the navigation buttons to go BACK UP and moving up or you go back to the last activity that was used. In iOS these tasks are implicit in the navigation bar located on the top or bottom of the application.

In this paper we present an overview of the widgets of the main mobile platforms (iOS, Android, Windows Mobile and HTML5). It also presents the study of equivalence in widgets that allow navigation in mobile applications: menus, tabs, and scroll tabs, among others. The results of this study seek to provide a guide to developing cross-platform applications, or mobile application migration. This guide will give confidence to developers and interface designers on the usability of the same on multiple platforms.

## 2 Mobile Platforms

*"A mobile software platform is defined as the combination of an operating system for a collection of compatible mobile devices with a set of related software development libraries, application programming interfaces (APIs), and programming tools"* [10]. The need for mobile operating systems that enable the application development has increased significantly due to the mobile devices proliferation. The main purpose of these platforms is creating a mobile development environment where users and developers can make applications. Some of the most commercial platforms are: Android, iOS, Windows Phone, Symbian. Some of their characteristics will be described [16]:

**Android:** Android is an open source platform with an open source license for mobile devices based on Linux. This platform is built on Linux kernel, native

libraries, Android run time and Android application framework. The Linux kernel core services (including hardware drivers process and memory, security and power management) are handled by a 2.6 kernel. Libraries running on the top of the kernel, Android includes various C/C++ such as libc and SSL.

**iOS:** The operating system manages the device hardware and provides the technologies required to develop native applications. The iOS Software Development Kit (SDK) contains the tools and needed interfaces to develop, install, test and execute native applications. Native applications are built using the system frameworks and Objective-C language and run directly on iOS. At the highest level, iOS acts as an intermediary between the underlying hardware and the apps that appear on the screen. The applications communicate with the hardware through a set of well-defined system interfaces that protect the application from hardware changes. This abstraction makes easy to code applications that work consistently on devices with different hardware capabilities.

**Microsoft Windows Phone:** Introduces the ability to use C++ within XAML app and in games written using Direct3D. The Windows Phone API reference node encompasses the complete set of API available on Windows Phone 8. The following diagram shown in 4 illustrates the set of APIs that integrate the Windows Phone API. The .NET API contains classes and types from the main namespaces. Windows Phone Runtime is a subset of native API that is built into the operating system. It is implemented in C++ and projected into C#, VB.NET, and C++, making it easy to consume naturally in any language. In addition to these APIs, can access to some Win32 APIs that give access to low-level features of the platform.

**Symbian OS:** Symbian is a private, independent company that develops and provides maintenance to the Symbian OS. This was used by some mobile devices manufacturers such as Nokia, Ericsson, Sony Ericsson, Siemens and Samsung. Symbian is based on the EPOC operating system and was used primarily for PDAs developed by Psion. Symbian OS is designed to support a wide range of voice and data services, such as multimedia and data synchronization. Basic services are provided by a framework, which include APIs, drivers, system files, and a standard C ++ library. In the top layer are basic services, such as a set of communication services, multimedia services, PC connectivity, and services of the operating system. The EPOC operating system OS uses C ++, a pure object-oriented language, as the supporting programming language for both system implementations and application programming interfaces [10][15].

### 2.1 Mobiles Devices Interfaces

*Displays* The mobile computing refers to a wide range of computational operations that allow a user to access information from portable devices such as laptops, PDAs, cell phones, handheld computers and portable gaming devices, among others [16]. Each mobile device has its own characteristics, such as the resolution, e.g: the iPad has a resolution of 1024x768 pixels, while the iPhone or the Samsung Galaxy Ace has a resolution of 320x480 pixels, this affects the way

the output of the LMS is displayed in the mobile device screen, since these HTML outputs were designed for screen size and resolution of desktop computers.

The screen size of the mobile device is the diagonal measurement (inches) of the physical screen, while resolution is the number of pixels on the screen of the device.

## 3 A Proposal for Widget Taxonomy

In the literature there are various manuals and programming books for each development platform in mobile systems. Some of these manuals contain guidelines for user-centered design. Some other works solve this type of design through patterns. These design patterns are often related to how applications respond to user events (to trigger processes or navigation, to name a few), the way they interact with other applications or the operating system, among others. Usially, design patterns are very specific to each platform. When cross-platform applications are developed or mobile platform applications are migrated, you need to keep the requirements of usability and functionality in mobile applications.

To establish equivalence between widgets, we verify the functionality and usability between them is maintained. That is, if two widgets are equivalent, it means that its functionality and usability are equal. The strategy to achieve this equivalence is a classification of widgets through functionality. This classification contains more subclassifications. With this taxonomy, we can check if any subclassing widgets are equal in usability. Through this taxonomy, the next step is to check whether any subclassing widgets are equal in usability. In this way we will have the same widgets in functionality and usability, so we can establish that the widgets are equivalent.

To build functionality based taxnoma use design patterns. The functionality taxonomy of widgets we propose is the following:

- Widget
  - Navigation
    - Menu: Revealable Menu, Fixed Menu, Mega Menu, ...
    - Gallery: Grid, Carousel, Slider, ...
    - List: Vertical List, Thumbnail List, Fisheye List, ...
    - Panel: Revealable Panel, Fixed Panel, ...
    - Pagination: Page Carousel, Peel Away, Widget-based,...
  - Input
    - Forms: Sign In, Registration, Multi-step, ...
    - Text Field: Password Field, Search Field, Text Area, ...
    - Slider: Peel Away, Widget-based, ...
    - State Buttons: Diseabled, Radio Buttons, Button With Label, ...
  - Control
    - Buttons: Icons, Image Button, ...
  - Presentation
    - Tables: Pivot Table, Headerless Table, Editable Table, ...
    - Multimedia Content: Media Player, Music Player, Start Guide (Tour),...
    - Progress Indicators: Progress Bar, Ghost Progress Indicators, ...

Note that in this classification are considered simple and compound Widgets. A simple widget is a graphical element that is composed of one single element, such as a text field or button. The Widget compound consists of two or more widgets such as forms, tables, lists or menus. This taxonomy allows widgets can belong to two distinct subcategories

A special case presented navigation widgets. Because each platform generates navigation according to their programming paradigm has. Classic example is the way the navigation is proposed in Android (width and depth), or navigation tabs as iOS. Navigation patterns allow users to retrieve state information between applications or connect (if the OS supports it).

## 4 Study Case: Navigations Widgets

For case studies of two groups of widgets a user interface are used. The first group consists of two menu widgets, dropdown menu and fixed menu. The second group consists of two galleries, a carousel and a slider. In both groups is to identify an equivalence relation when these widgets are alternated in the user interface.

In the first case the widgets are alternated at the interface, depending on the space available for the menu. When space is reduced using dropdown menu and have more horizontal space used fixed menu. A goal is to identify whether the alternation of these widgets, has a negative influence on the user interaction with the menu items.

In the second case is similar to the previous case, when the display space allow to use a slider and a carousel. In this case the carousel controls the images displayed in the slider. When the display space is reduced slider is removed and the carousel absorbs its tasks.

### 4.1 General Prototype

To test the equivalence of the above widgets proposes a mobile application "Estiramientos" that addresses topics sports, are a series of pre-stretching (routines) to perform a physical activity.

The general prototype (see Figure 1) has a user interface composed mainly of a bar for the logo and a fixed menu of stretching routines, a carousel with images from the routine handling the current image of the routine slider. Current section of the application is indicated by the selected menu button, selecting a different

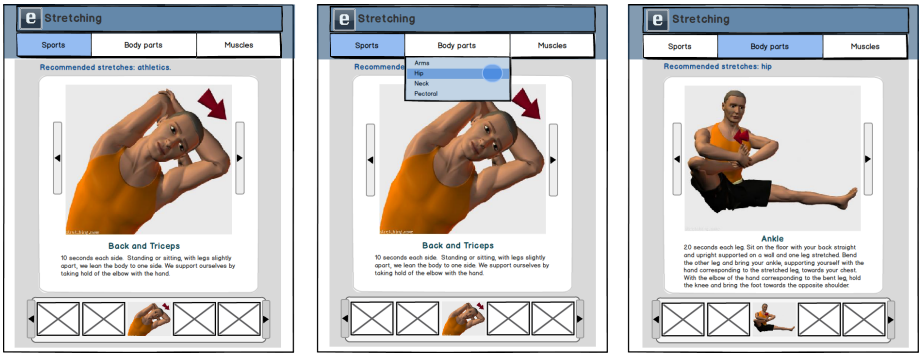


**Fig. 1.** General Prototype

menu option loads the new stretching routine with its images in the carousel and slider. The Slider is controlled by side buttons that provide access to the steps of the current stretching routine. The carousel can scroll through images of the routine through a swipe gesture. When the user selects an image carousel, the "slider" and the routine text is updated. By changing the orientation of the device the main content (slider, routine text and carousel) fits within the user interface.

### 4.2 Specific Prototypes

The application consists of two types of development: a native application in Android and Web development in HTML5 and CSS. Considered in developing Android devices with screen sizes of 4, 7 and 10 inches. HTML5 application for mobile devices are covered with similar screen size, whose Web browsers support CSS3 and jQuery v1.4[1]. Since the major Web browsers support desktop library jQuery Mobile v1.4, the application can display them correctly without incorporating other JavaScript libraries. The application presents orientations supports that allow mobile devices (landscape and portrait).

**Android Design for Small Displays.** The Android app for devices with screen size near 4 inches, in vertical orientation the user interface has the following widgets: top bar to the logo and dropdown menu that displays options routine using a spinner, a breadcrumbs indicating current section, a slideshow that displays the selected routine stretching and the main content of the routine controlled by the carousel.

In landscape orientation the user interface has the following changes: the main menu is replaced by spinners deploying routines related to sports, body pats and muscles. The carousel of images becomes more important and is placed in the central area of the application, displaying the current image and the content of the routine (ie the ImageView widget that displays the current image of the application in portrait orientation is eliminated).

**Android Design for Wide Screens.** For devices with screen size near 7 and 10 inches in the vertical orientation preserves the user interfaces widgets used in four-inch version see Figure 2. Current section of the application is indicated by the selected spinner, selecting a different menu option loads the new stretching routine with its images in the carousel.

In landscape orientation the user interface has the following changes: the routine text is placed to the right of the current image.

[1] Web browsers that support jQuery Mobile 1.4: Apple iOS 4-7, Android 4.4 (kitkat), Android 4.1-4.3(Jelly Bean), Android 4.0 (ICS), Android 3.2(Honeycomb), Android 2-1-2.3, Windows Phone 7.5-8, Blackberry 6-10, Blackberry Playbook (1.0-2.0), Firefox mobile18, Chome for Android18, Chrome Desktop 16-24, Safari Desktop 5-6, Firefox Desktop 10-18, Internet Explorer 8-10 y Opera Desktop 10-12.

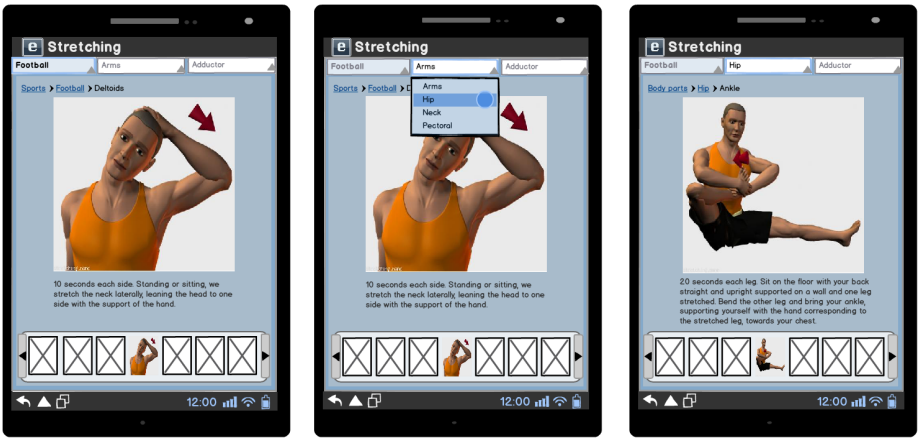


**Fig. 2.** Android prototype for wide screens

**Design for the Version in HTML5.** Web application developed in HTML5 and CSS3 for devices with screen size of 4 inches in both orientations and 7-inch screens in portrait orientation, presents a user interface with the following features: top bar to the logo and dropdown menu (using a nav structure, ul, li and) that displays sports routines, body parts and muscle, the central content slider and a routine text controlled by the carousel. The storyboard with the above items can be viewed in Figure 3.

For devices with screen size 7 inch and 10 inch horizontal orientation in both orientations, the user interface includes the following change: fixed horizontal menu (using a nav structure, ul, li and) deploying routines.

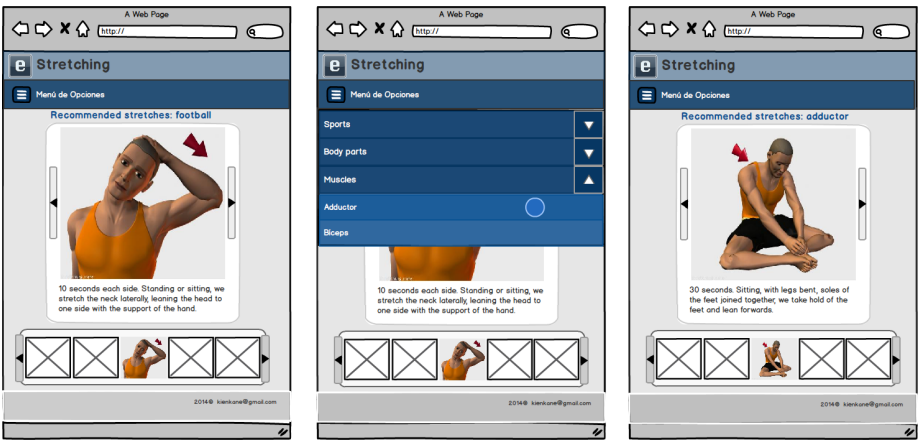


**Fig. 3.** Prototype for the Web version

### 4.3 Test

The main objective of the study is to find a functional equivalence relationship between types of widgets. In the first set of widgets we study the equivalence between the dropdown menu and set menu. In the second group studied the functional equivalence between an image-slider and a carousel. In both cases, the set menu is alternated by the dropdown menu for space on the screen and the same goes for the second set of widgets.

For the first group has hypothesized that users can use the application intuitively and naturally. No matter whether you are using the dropdown menu or horizontal menu fixed. That is, users can easily use the application in cases where the size or orientation of display device does not allow to use a fixed horizontal menu and chooses the dropdown menu. With this we can prove that the dropdown menu has the same usability as the fixed menu.

For the secong group of widgets has hypothesized that users can use the application intuitively and naturally. No matter whether you are using the image-slider or carousel. With this experiment, we can check the condition of the small screen to use a slideshow to replace both widgets (slider and carousel) without loss of usability. And therefore both widgets have the same usability.

To test the hypotheses were implemented and did usability testing on prototypes of "Estiramientos" application for Android and WEB HTML5/CSS. In addition, a questionnaire usability for mobile application was developed. A questionnaire for the Android version of the user interface for large screens and the HTML5 version. These tests were designed for Android devices 7 inch screen and Android devices 10 inch screen. The performances of these devices allow the user interface can switch them around we are interested to study.

The prototype test was applied to 56 individuals (25 women and 31 men). The test group had a mean age of 24.1 years old, with a minimum age of 18 years old and a maximum age of 44 years old. The results of usability testing on prototypes provided the following results:

1. 94.5% of the respondents believe that the text presented is easy to read, compared to 5.4% who said otherwise.
2. 96.4% of the respondents believe that the images can be viewed correctly compared to 3.6% who said otherwise.
   (a) When asked if the selection of images through Carousel is easy and intuitive. 94.6% responded affirmatively, compared to 5.4% who reported problems by selecting images in the carousel.
3. When asked if the navigation within the application was intuitive even to change the orientation of the device. 92.9 % responded affirmatively compared to 7.1 % who felt otherwise.

Relevant results when considering the orientation of device during the application of questionnaires.

1. 87.5% of the respondents believe that the control elements of the applications as menus, buttons and links, are intuitive and easy to access in the horizontal orientation of the application, compared to 12.5% who said otherwise.

2. When asked if the menu options are available to change the device orientation from horizontal to vertical. 91.1% of respondend affirmatively, compared to 8.9% who said otherwise.
3. When asked if the images were accessible on the carousel using the application in landscape orientation. 96.4% responded affirmatively, compared to 3.5% who said otherwise
4. When asked if the images carousel still accessible by changing the device orientation to vertical, all responded affirmatively.

Relevant results when considering the format of the application (Web or Native Android version) during the application of questionnaires.

1. Using the application in Web format , the following results were obtained:
   (a) 85.7% of the respondents said that the menu options are visible in landscape orientation, compared to 14.3% who felt otherwise.
   (b) When using the device in portrait orientation 82.1% responded that the menu options are visible, compared to 17.9% who felt otherwise.
   (c) 92.9% of the respondents felt that the images of the carousel in landscape orientation were accessible while 7.1% felt otherwise.
   (d) All of respondents felt that the images of the carousel remain accessible to change the orientation of the device.
2. Using Android application format , the following results were obtained:
   (a) 96.4% of the respondents said that the menu options are visible in landscape orientation, compared to 3.6% who felt otherwise.
   (b) When using the device in portrait orientation, 92.90% responded that the menu options are visible, compared to 7.1% who felt otherwise.
   (c) All of respondents felt that the images are visible in the landscape orientation of the device.
   (d) All of respondents felt that the images of the carousel remain accessible to change the orientation to portrait.

## 5 Conclusions

In this work we have studied the equivalence of widgets proposing as equality equivalence relation between functionality and usability. That is two widgets are equivalent if they have the same functionality and the same usability. The functionality is identified through the proposed taxonomy and usability is measured through tests conducted with prototypes on mobile devices. Because native applications have better performance than HTML5 rich-client applications, performance is not considered applications for equivalence relation.

We present two experiments to test the equivalence of some widgets. Two pairs of widgets that have the same functionality and usability testing were selected. A usability experiment throw one with two widgets, and another experiment it was concluded that the widgets did not have the same usability. In one hand, the test results show that the carousel widget used in Android application and

carousel slider used in HTML5 application are equivalent in landscape and portrait. In another hand, The usability of spinner widget Android App and HTML5 dropdown menu in portrait orientation is not the same. With this result we can suggest that to migrate an application in HTML5 to a native Android application, avoid spinner to replace HTML5 menu dropdown menu. Perhaps a better option would be a drop down list.

The carousel widget used in Android application has a fixed size regardless of the orientation of the device, while the carousel used in the HTML 5 version has an adaptability to the size of the screen. However, the test results show that this fact did not alter were equivalent both widgets.

With this type of analysis and testing, it would be possible to suggest the type of widgets that can be used to perform migration of applications between platforms while the functionality and usability is preserved.

## References

1. Love, S.: Understanding mobile human-computer interaction. Elsevier, Amsterdam (2005)
2. Jacko, J.A.: Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies, and Emerging Applications, 3rd edn. CRC Press, Inc., Boca Raton (2012)
3. Lee, V., Schneider, H., Schell, R.: Mobile Applications: Architecture, Design, and Development. Prentice Hall PTR, Upper Saddle River (2004)
4. Fogg, B.: Mobile Persuasion: 20 Perspectives on the Future of Behavior Change. Stanford Captology Media (2007)
5. Rogers, Y., Sharp, H., Preece, J.: Interaction Design: Beyond Human-Computer Interaction. John Wiley and Sons Ltd. (2002)
6. Pak, R., McLaughlin, A.: Designing Displays for Older Adults. Human factors and aging series. CRC Press, Inc., Boca Raton (2011)
7. Neil, T.: Mobile Design Pattern Gallery: UI Patterns for Mobile Applications. O'Reilly Media, Inc. (2012)
8. Inc., A.: iOS Human Interface Guidelines (2010)
9. Pilgrim, M.: HTML5: Up and Running. 1st edn. O'Reilly Media, Inc. (2010)
10. Zheng, P., Ni, L.M.: Smart phone and next-generation mobile computing. The Morgan Kaufmann series in networking. Morgan Kaufmann Publishers, San Francisco (2006)
11. Tzovaras, D.: Multimodal User Interfaces: From Signals to Interaction. In: Signals and Communication Technology. Springer (2008)
12. Mednieks, Z., Dornin, L., Meike, B., Nakamura, M.: Programming Android - Java Programming for the New Generation of Mobile Devices. O'Reilly (2011)
13. Galpin, M.D., Kappler, M.: Android in practice /. Manning, Shelter Island (c2012) Include index
14. Apple Inc.: Cocoa Fundamentals Guide (May 2010)
15. Firtman, M.: Programming the Mobile Web. O'Reilly Media (2013)
16. Gavalas, D., Economou, D.: Development platforms for mobile applications: Status and trends. IEEE Software 28(1), 77–86 (2011)