An Analysis of HCI in Mobile Platforms

COS30007–Creating Data-Driven Mobile Applications

Swinburne University of Technology

Alex Cummaudo  1744070

Semester 1, 2014

Abstract

Mobile platforms today have changed more in the last five years than since the dawn of mobile devices. The success of major mobile platforms today can be attributed to their revolutionary UIs that shifted traditional HCI with new HCI. What is it about the design that makes these platforms successful? Do different major platforms differ in their implementation of HCI, and how does this change the user experience between them? On inspection of design guidelines and critical differential analysis popular apps between the iOS and Android platforms, it was found that iOS has a greater focus on content, while Android keeps its user’s needs paramount. A good UI on iOS is considered clean, non-competitive with content and largely should play a supporting role to the content. That of Android is more emotionally involved with the user, provides features that aids users more and allows greater customisability options. Ultimately, the user experience is similar between the two, but the subtle differences in the respective design principles are just enough to change what the focus of the HCI is between varying platforms.
Contents

1 Introduction 3
   1.1 Background .................................................. 3
   1.2 Motivation .................................................. 3

2 Method 4
   2.1 Human Interface Guidelines .................................. 4
   2.2 App Analysis .................................................. 5

3 Discussion 5
   3.1 Core Design Principles ....................................... 5
   3.2 Analysis of Common Apps ..................................... 9

4 Conclusion 17
1 Introduction

1.1 Background

With the introduction of the iPhone in 2007 and iPad in 2010, Apple sparked a revolution in the field of mobile HCI, finally delivering a mobile interface that overcame the limitations of the hardware it ran on. For years beforehand, the literature of mobile HCI has always been predicting a boom of mobile technologies, yet “the pains and complexities of interacting through very limited input and output facilities” seemed to hinder this boom [Chittaro, 2004].

While touch-screen technology has been available with touching, dragging, and drawing since the early 1990s [Shneiderman, 1991, p.94], an assortment of various touch-screen products simply never took off to the extent where the mobile market is today. Fundamentally, a successful implementation of an appropriate interaction with all user groups just could not be grasped within these devices, which led to their ultimate demise. Yet, as the iPhone and iPad were introduced, the ‘Post-PC’ era, a phrase popularised by then-Apple CEO Steve Jobs, began to take off—primarily powered by a new interaction method with users.

Ultimately, mobile platforms should not share common UI elements with the traditional point-and-click interfaces of desktop computing. Smaller screen sizes, touch interfaces and in-hand mobility are three key aspects as to why HCI within mobile devices should be different to the traditional HCI in desktop computing, though some aspects are retained yet slightly varied (e.g., rather than clicking, users tap instead).

1.2 Motivation

As [Ebner et al. 2010, p.2] described, part of the iPhone and iPad’s success was a shift from traditional UI elements to new interaction and usage paradigms, which made an impact on the way end users can seamlessly and ubiquitously interact with all of Apple’s iOS devices. The growth of the Android user-base since iPhone’s introduction is also a second case where a touch-screen interface for a mobile device has been largely successful. Whilst there are slight differences

Figure 1.1: During Q2 2013 an average of 52% of smartphone owners in the US had a handset that runs the Android OS. Source: Neilsen.
between the two major platforms, the UI is largely similar to ‘fit’ with the hardware it is wrapped around. The successes of both platforms have been from this deviation from the path of traditional HCI.

This research report will compare and contrast both Android and iOS, chiefly in regards to how the UI components, feedback and elements all culminate together to provide some of the best examples of HCI on the mobile-market. Android and iOS are the largest mobile platforms at the time of writing this report; by market share on US smartphones, Android leads by 52% ahead of iOS at 40% (refer to Figure 1.1), making both suitable candidates UI technique comparison.

In addition, an analysis of differences between the two will propose ways in which the user-experiences changes between both platforms, factoring in considerations from third-party literature about what makes a ‘good’ mobile user-interface for apps on these devices. The design guidelines for iOS and Android produced by Apple and Google, respectively, will outline the factors that need to be considered when developing for either platform.

2 Method

2.1 Human Interface Guidelines

Both Apple and Google provide a set of Human Interface Guidelines, or HIGs. HIGs are sets of rules that go behind the design of a specific product or family of products. The purpose of HIGs, as summarised by Stone et al. [2005] p.168, are to guide developers into creating apps that suit and are consistent for platform that they are developed on. Usually, HIGs provide a suite of principles and suggestions that keep apps on the platform consistent, and are developed from both academic research and experience of usability practitioners. Developers may not always adhere to these recommendation, and may choose to deviate from the ‘standards’ of the platform, though this will tend to conflict with the conformity of apps on the platform. Creating an app that keeps user’s familiar with the platform it’s running on is key to an app’s success.

An analysis of the HIGs provided by Google and Apple for their respective platforms allowed for an insight into how Apple’s and Google’s design principles differ, and how they are similar. Key to this insight was how and why the same app on both platforms may have a different user experience, in order to conform with the HIGs provided, and to discover
what iOS and Android uphold as paramount principles to the user experience. Comparing principles in similar concrete examples between both platforms lead to contrast discussion.

### 2.2 App Analysis

An analysis of both Facebook and Twitter versions of Android and iOS were assessed to find how they comply with the respective iOS and Android HIGs. Differing UI elements and techniques were also investigated to discuss how the UI elements themselves differ, and how this may affect the user experience if a user switches from iOS to Android and vice-versa.

### 3 Discussion

#### 3.1 Core Design Principles

<table>
<thead>
<tr>
<th>iOS</th>
<th>Android</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Deference</strong></td>
<td><strong>Enchantment</strong></td>
</tr>
<tr>
<td><em>Content</em> is at the heart of the UI. The UI helps users understand and interact with the content, but never competes with it.</td>
<td><em>Joy</em> of the user is heart of the UI. The UI should adapt to suit the user, and should constantly satisfy user’s emotions.</td>
</tr>
<tr>
<td><strong>Clarity</strong></td>
<td><strong>Simplification</strong></td>
</tr>
<tr>
<td><em>Focus</em> motivates functionality and ensures content is paramount. The UI should be crisp, subtle and appropriate to bring out the most in the content.</td>
<td><em>Brevity</em> motivates functionality and ensures users aren’t disturbed but still kept informed. The UI should be consistent, decisive and quiet.</td>
</tr>
<tr>
<td><strong>Depth</strong></td>
<td><strong>Empowerment</strong></td>
</tr>
<tr>
<td><em>Layers</em> heighten users understanding on where they are. Hierarchy and position informs users of relationships between elements.</td>
<td><em>Encourage</em> the user to learn new things on their own. Let the user feel in control, and pick up features quickly with encouragement form the UI.</td>
</tr>
</tbody>
</table>

Both iOS and Android’s HIGs display differing Core Design Principles which demonstrate that the user experience is different between both platforms. These are contrasted
in Table 3.1, iOS has a far greater emphasis on the content user’s expect to see within their apps, whereas Android’s focus is to keep the user as delighted as possible regardless of what they’re doing. iOS suggests that the UI needs to be as subtle as possible, and play a supporting role in the app.

![Figure 3.1: Comparing Deference and Enchantment in iOS and Android; focus on content is the iOS-approach, while focus on user emotions is the Android approach.](image)

Deference vs. Enchantment The UI should never compete with the content in the app. Android’s suggestion is that the UI needs to be joyful, and connect to the user’s emotions. Figure 3.1 compares the two in a concrete example; the Weather app in iOS clears as much of the UI elements as possible, giving as much focus on the data presented as possible. The Android weather example uses as much as the content on screen to provide a weather animation to emotionally convey the current weather in San Francisco. While there is not as much detail in the centre of the screen as per iOS, the weather is emotionally conveyed; users can tell it is sunny in San Francisco just by looking at the animation. iOS too provides an animation, though this is presented in the background, keeping the location, temperature and forecast in the foreground. Unlike Android which retains buttons for refreshing, adding
locations and changing the forecast scope, iOS has almost no UI buttons at all, except for a deferred locations button in the bottom right of the screen.

Ultimately, iOS does have the upper hand here. Users can quickly grasp the content they are looking for at a glance, with it’s very clear and large typeset. The UI has deferred to give a greater focus on the content itself, which pays off here. Android, though trying to enchant the user with animation, is worse off in this regard as the very same content is visibly hard to see in the upper left-hand corner.

![Sign In](image)

**Figure 3.2:** Signing in on iOS and Android shows different approaches to simplifying user decisions.

**Clarity vs. Simplification** Keeping things as simple and as clear as possible are similar principles in iOS and Android, though subtle differences arise on analysis. A good example of this arises when downloading apps on the respective platforms, as shown in Figure 3.2. Both contribute more or less the same to the user experience, but in varying ways. Android keeps its sign in message brief and encourage all messages to be shortly phrased with simple words. iOS’s approach is to ask the question in the action; rather than having ‘Yes’ or ‘No’-styled responses, responses themselves should be descriptive, verbal and convey meaning, thereby eliminating the need for any message at all. However, by eliminating the message, iOS can’t provide the suggestion hints that Android does (e.g., ‘If you use Gmail, answer Yes’).

In essence, both have their pros and cons. Apple’s approach for naming the buttons other than the standard ‘Yes’, ‘No’ is useful to give a verb to whatever the action of the button does. However, Android’s simple dictation (“If you use Gmail, answer Yes”) can really avoid users questioning themselves and helps guide users in a very explicit way.
(a) Tapping a list in the reminders app will provide a zoom-up transition on the list tapped, using the animation as an indication that they have drilled down from all lists to the tapped list.

(b) Swipe gestures are a common pattern in Android apps; users can discover this as a way to transition from one screen to the next and often do so by using muscle memory from other apps.

Figure 3.3: iOS uses animations and transition to inform users of how deep (hierarchically) they are within interface. Android empowers users to learn and get used to patterns frequently used from other Android apps.

**Depth vs. Empowerment**  
Android encourages users to find things out for themselves, using visual patterns and muscle memory from other Android apps to help achieve this. In particular, encouragement should be ‘sprinkled’ amongst apps (e.g., an outline or glow as an app icon is being moved into a folder), thereby giving users a heightened sense of achievement when they perform tasks on the device. This leads to a satisfactory user experience, where the user feels in control and also feels like an expert when using their device. iOS partially adapts empowerment in its HIG, though to keep the user informed of their situation, they shouldn’t need to figure things for themselves; the app should give clear indications as to where the user is and how they can go back from the previous screen, and depth is a way in which app hierarchy can be visually represented by the use of layers (as opposed to patterns in Android). iOS devices do not have the physical back button, and so layers can be used to overcome this physical limitation; users remain informed and still feel in control since the animations that transition from one layer to the next help users form a cognitive model of how deep they have ‘drilled-down’ into the interface, and know exactly where they are, where they came from, and how to get back. Figure 3.3 and 3.12 illustrate these two principles in detail.
In this regard, iOS’s approach is perhaps more preferential in its approach to always provide a user with a level of hierarchy. The depth itself keeps the user always visually informed as to wherever they are, which is empowering in itself as the user will feel in greater control knowing whereabouts (depth-wise) they are in the app.

3.2 Analysis of Common Apps

(a) When the Login button is tapped, an Activity Indicator is placed at the far right of the button in iOS.

(b) Android uses an Activity Circle mixed with a Progress Bar. Visual feedback of completion is given.

Figure 3.4: The signin process on iOS (left) and Android (right).

Figure 3.5: Suggested progress bars from Apple (left) and Google’s (right) HIGs indicate both activity progress and task duration.

Signing In To begin with, the Facebook app immediately differs from the login screen on either platform (see Figure 3.4). Signing in on the Android version gives an indication of completion time with an Activity Circle that has been morphed with a Progress Bar. The progress bar “are for situations where the percentage completed can be determined” (Google, Inc. 2014), however, this has been presented in the form of a circle. iOS only uses an Activity Indicator instead, and therefore does not give an indication as to how long it will take for the logging in process to finish, though it does “reassure users that [the logging in process] hasn’t stalled.” (Apple, Inc. 2014)—thus having some feedback rather than no feedback at
all is essential (chiefly since the Activity Indicator is better used for tasks where progress cannot be determined, such as how much a file has been uploaded). UX in the iOS version is slightly worse off since it does not provide any form of how much longer it will take for sign-in to complete. Better still, the suggested progress indicators, see Figure 3.5, from Apple and Google may have been more suitable in either case, as this keeps consistency with other apps on either platform.

Ultimately, neither is ‘better’ than the other as the standardised progress indication bars have not been used. Though Android’s approach is slightly better off, incorporating the progress bar into the activity indicator, this will not be consistent with other progress indicators throughout the platform. Nonetheless, Android’s approach still provides some form of progress, albeit in an inconsistent method to the platform.

**Home View Layout** The home view layouts in both screens vary between platforms also. Whilst the content remains central to the screen, positioning of the tab bar controller and the actions swap. Key to Android’s interface is the ability to swipe left and right to switch between tabs, and so therefore users less frequently tap the tab bar controller icons. This is in contrast with iOS, where users need to tap the tab bar controllers (as opposed to swiping left and right) in order to switch views associated with the tab bar. As such, the positioning of action elements are at the bottom in Android to make them closer to the home buttons (where the user’s thumb spends most of the time) as users are more likely going to swipe between tabs and use the post buttons more. In iOS, users are encouraged to switch tabs instead, and, as Apple’s HIGs stipulate, tab bars “always appear at the bottom edge of the screen” (Apple, Inc., 2014). As such, since iOS users will be tapping the tab bar buttons more frequently then tapping posts, improving UX is done by placing these buttons at the bottom near the home button.

Android’s swipe to switch tabs is certainly more convenient and fluid in its nature, and this is a reflection of the empowerment design force in practise, as discussed in Section 3.1. Such a faster method does empower users to make context switches within the app (i.e., from one tab view to the next) with just a swipe of the thumb, which gives rise to better UX fundamentally due to its fast nature. As such, the rearrangement from iOS in the Android home view layout does support this nature of swipe-to-switch tabs, and is better in this regard than iOS’s layout.
(a) Users more frequently use the tab buttons to switch tab views in iOS. Post buttons placed in less frequently used on-screen area instead. (b) Users more frequently swipe left and right to switch tab views in Android. As such, the post buttons are placed near toward the home buttons instead.

Figure 3.6: Comparing positioning of elements of the home screen in iOS and Android. The area towards the bottom (near the home button) is good for key/frequent action elements.

Tab Bar The tab bar controllers in both platforms differ. As aforementioned, Android provides the ease of switching between tabs by swiping from left to right, which is why it is not positioned at the bottom near the home button as per iOS. The flexibility of the swiping gestures in Android improves UX for expert users by improving speed at which content can be navigated through. However for novice users who are unable to decipher the icons or are unaware of the swipe functionality, this is where iOS’s approach to tab bar labelling and icon use excels. A drawback to the iOS approach is that only 4 tabs can be used on a standard iPhone screen; a designated ‘More’ tab should be used to customise or have access to hierarchically equal features (i.e., views of the app just as important as each other) as per the Apple HIG. Android’s tab bars can be scrollable (Figure 3.8), so as many tabs can be
Figure 3.7: Varying tab bars in both iOS (left) and Android (right) have their benefits and disadvantages.

Figure 3.8: Scrollable Tab bars are only available in Android. Swiping right-to-left shows even more available tabs.

Figure 3.9: Similar to Scrollable Tabs, Page Controls are only available in iOS. Swiping left to right moves users from one page to the next, as used in the iOS home screen.

used as necessary, thereby removing the need for a ‘more’ bar. However, iOS does introduce Page Control indicators (Figure 3.8) to provide a similar feature for page-able content—i.e., they “don’t show how views are related to each other and it don’t indicate which view corresponds to each dot” (Apple, Inc., 2014). In addition, both the iOS and Android tab bars support badge capabilities; this improves UX by leading users to new content (e.g., a new message was received in the example provided in Figure 3.7).

Perhaps Android’s ability to provide so many tabs may indeed complicate users as to where they want to go next, thereby leading to an adverse UX. Too many options will complicate user’s thought patterns, and fragments the app into a lot of tab views, which perhaps can be consolidated into just a few. For example, the image posed in Figure 3.8 shows Top Paid and Top Free—these could just be consolidated into a Top tab that has segmented sections within that view of Paid and Free. Apple’s approach at keeping limited tabs on the screen forces content to be in fewer locations, making a user’s ability to find such content easier for them and ultimately upholding the UX to a greater standard than Android.
3 DISCUSSION

Figure 3.10: Rather than using modal alerts, subtle error messages indicate lack of internet connection to give feedback.

(a) Using a modal Alert view captures user’s attention more  
(b) Using a system notification may be overlooked by users

Figure 3.11: Twitter notifies users that posts fail, either with a Modal alert (iOS) or system notification (Android).

Dealing With Limited Network In the event of a lack of network connection, both apps inform the user by use of a subtle message, as shown in Figure 3.10. Whilst download progress ceases throughout the app (e.g., no more posts can be downloaded until network activity resumes) uploading is still permitted—creating new posts will be queued until network activity is available once more. When network activity is available, all queued posts will be posted at the time stamp when the ‘draft’ was saved when posted without the internet connection. This is a pattern endorsed both by Apple and Google’s HIGs to improve user experience; lack of an internet connection should not stop the functionality of content posting as preventing content from being written (and then cached) will lead to adverse user experiences. Similarly, all content downloaded thus far should be cached and reused the next time the app is started; that way if the app opens without an internet connection, rather than displaying *no* data at all, at least some data is presented (albeit outdated). Gracefully
limiting the functionality is implemented in both versions of the app, thereby providing users with some of the features they expect, rather than stalling all features entirely (retaining a better UX since partial functionality is still present).

However, Twitter takes a different approach to failed tweets (see Figure 3.11). In addition to queuing posts, some form of notification is used to alert users that the post failed. Android’s Simplification Design Principle stipulates:

“Like a good personal assistant, shield people from unimportant minutiae. People want to stay focused, and unless it’s critical and time-sensitive, an interruption can be taxing and frustrating.”

- (Google, Inc. 2014)

Because of this principle, the Android Twitter app only notifies users of failed tweets by a simple system notification, as Figure 3.11b shows, may prove to be a frustrating UX design choice since users can easily miss the failed tweet notification (e.g., they had their phone down after they posted the tweet). Should they miss the notification, this will make them believe that the app did post their tweet, even though they haven’t yet realised that it isn’t pushed to Twitter’s servers. Apple’s iOS approach forces users to read the Modal alert in order to continue using the app—in this version of the app, the view is critical enough not to be dismissed as a system notification and is used to ‘bug’ the user into reading why their tweet failed. While this may cause the app to have more noise, it does improve UX since users will definitely see the alert view, and therefore will be informed of the failed tweet. In this regard, the iOS approach, though bugging the user, is a safer approach—it’s far better to assume the user didn’t see the Android system notification and then get angry (leading to bad UX) when they later find out that their Tweet did not post at all.

**Back and Up Navigation**  All Android devices are equipped with a system back button that allows traversal back to the previous view that the user last saw. All illustration of this process is provided in Figure 3.12 and as Comis (2013) describes, “the Back button is used to navigate through the history of screens the user has recently navigated through, even outside of the current app”. Most app navigation in iOS follows an Android ‘up’ navigation pattern, whilst the ‘back’ navigation functionality is non-existent in iOS and is only available in Android. Tapping a back button on-the-go is also a far more fluid and dynamic way to
restore previous page content, a another reason why it improves UX for users. Users in iOS can, however, swipe-left to go up when the view is used in a UINavigationController pattern. These respective navigation methods can be found in both the Twitter and Facebook apps on either platform.

Therefore, Android’s back button is a very fast and convenient way to return back to the previous screen the user saw. However, this does come at a hefty price, notably the need to include a new, physical button the device itself. This will ultimately complicate the hardware IO in the long run, and Apple’s approach at making the device have one, big physical button only greatly simplifies its device hardware IO. However, this is open to interpretation, and, while some users may be confused by the extra physical buttons, others may not—there is no clear answer as to which platform ‘does it better’.

**Search Bars** The extra physical back button proves useful for eliminating cancellation buttons (see Figure 3.13), since the user can simply tap the physical button to leave whatever task they have begun. This is a motivating reason why iOS must maintain the Cancel button next to its search field, and why Android removes the cancel button—by eliminating the cancel button, extra room is available for the search query, and users can immediately tap the back button right next to the on-screen keyboard to go back, improving the speed of the
search experience on Android perhaps more than it does in iOS.

**Figure 3.13:** Search in iOS (left) requires a Cancel button. Android users can simply use their system back button.

![Figure 3.13: Search in iOS (left) requires a Cancel button. Android users can simply use their system back button.](image)

**Figure 3.14:** Action Views are an invasive, though depth-wise important, way to change what the app will do next. Spinners achieve the same functionality but do so without being so invasive.

![Figure 3.14: Action Views are an invasive, though depth-wise important, way to change what the app will do next. Spinners achieve the same functionality but do so without being so invasive.](image)

**Spinners vs. Action Sheets** A significant alteration in UI is the use of providing additional functionality that is concealed underneath a button (Figure 3.14). While iOS achieves this through action sheets (i.e., tapping a button conceals the content with a modal Action Sheet) Android achieves the same thing, albeit un-modally, with a non-invasive spinner—retaining the content that was on-screen. Consider the log-out procedure in the Twitter app; Android requires a tap of the user’s icon to expand a spinner *over* the original content. The spinner, once expanded, is non-invasive, and still allows surrounding content to be readable. It can also be used to change a field’s content—e.g., ‘Reply’ to ‘Reply To’ or to ‘Forward’
etc. iOS’s Action Sheet is a modal view; it takes up the entire screen once it is invoked, and forces the user to make a decision as to what to do next. The invasive nature of the Action Sheet upholds Apple’s Depth Design Principle—since the user is about to make an important decision as to what screen they will see next, it is imperative that the depth be heightened with a new layer introduced over the original content so that users are aware that the next step will take them elsewhere. This heightened sense of UI has its pros and cons, chiefly it gives attention to the Action Sheet (as the action sheet will result in some critical functional behaviour), though it does cover the content is sits in front of. Either the use of spinners or action sheets retain the same overall functionality, yet, as discussed, their implementation leads to differing user experiences overall.

4 Conclusion

For years preceding the launches of iOS and Android, touch-screen mobile platforms have not been largely successful. The key to this success has been the paradigm shift away from traditional desktop UI components and design principles and the developent of new ones that suit the hardware of the mobile device.

This research report has discussed and analysed the effectiveness of Android and iOS’s design principles and design elements. Differing components in both operating systems lead to a slightly different user experience; while there is a greater focus on the content users interact with in iOS—such that the UI should be unnoticed by the user—the focus in Android’s UI is to keeping the user as satisfied as possible. A different perspective to what the platform sees paramount in its design (i.e., content vs. user) has lead to key differences in UX. Swiping abilities from tab-to-tab is a feature to allow Android users navigate faster for instance, or Action Sheets in iOS to keep the user focused on what task they will perform next is an imperative implementation of keeping depth in the app.

Future extensions of this research should discuss other platforms, particularly the growing Windows Phone market, and also discuss how physical hardware buttons have a greater impact on the UX (for instance, volume buttons as camera buttons and what implications to UX this may have). Fundamentally, though, most platforms are largely similar and provide a similar UX overall—the tiny differences between both systems are simply up to the user’s taste, and a critical analysis of these differences highlights a user’s taste in greater detail.
References


