This paper proposes sets of patterns useful in designing scalable mobile app user interfaces targeted at multiple mobile platforms. These sets of patterns, called mobile-UITS patterns, are derived from mobile platform vendors’ style guidelines and the common practices found in existing mobile apps. The resulting patterns provide solutions in specific contexts for the transition of mobile app UI design and interaction from one platform to the other ones, and also tackle the scaling issues that arise when moving a mobile app from small-size smart devices (phones) to large-size smart devices (tablets) and vice versa. In this work, we focus mainly on the categorization of these patterns and provide a template for writing the pattern description.


1. INTRODUCTION

Nowadays, the number of people using smart devices (i.e., smartphones and tablets) is increasing rapidly day by day. In addition, they are increasingly being used for conducting mobile activities – including entertainment, commerce (m-commerce), or learning (m-learning). In order to reach to a broad pool of potential users, companies have to support several of the existing mobile platforms for each of their mobile applications (commonly abbreviated as mobile apps or just apps). Moreover, users of a specific mobile platform expect that all these apps follow the standard structural and interaction concepts of the underlying platform to allow them to enjoy a consistent user experience (UX) on the mobile device. Addressing users’ expectations and reaching a broad pool of potential users means that companies need to provide several versions of the same app for different mobile platforms and device types. This requires great effort on the part of the designers. Moreover, ignoring the required structural and interaction concepts of the underlying platform may result in reducing the usability and UX in the resulting version, which are, in fact, major success factors [Gorlenko and Merrick 2009] of an app; hence, such a decrease increases the risk of the app becoming a failure product.

Targeting the challenge of designing mobile app UI for multiple platforms and device types, we propose specific patterns in order to minimize the efforts on the part of the designers and reduce the possibility of making mistakes. These patterns, called mobile-UITS (User-interface Transition & Scaling) patterns, are aimed at solving those problems that arise when transitioning mobile app UI design and interaction from one platform to another as well as the scaling issues when moving from small-size smart devices (phones) to large-size smart devices (tablets). We derive these patterns from the mobile platform vendors’ style guidelines (e.g., Google Android UI Guidelines, iOS human interface guidelines, Microsoft Windows Phone, etc.) as well as the common practices found in existing mobile apps. In this work, our focus is on the categorization of these proposed patterns. Moreover, we provide a pattern template for writing the detailed description of these patterns. This work advances the state-of-the-art in the current mobile domain in three directions: a) dealing with the issue of transitioning app UI design and interaction from one platform to others, b) tackling the scaling issues when moving from one device to another size, and c) providing a formal template. The last section highlights this in a more elaborated way.

The remainder of the paper is structured as follows: In Section 2, we highlight some related work. In Section 3, we explain different mobile-UITS pattern categories. In Section 4, we provide the formal pattern template with some examples. Finally, we
discuss the benefits of our approach and shed light on future directions in Section 5.

2. RELATED WORK

In their famous work, Alexander et al. (1977) originally illustrated the pattern concept for use in the architecture domain. The patterns are supposed to provide the solution to a recurring problem in a specific context by capturing existing knowledge (Alexander et al. 1977; Gamma et al. 1995; Borchers 2001). Initially, Gamma et al. (1995) introduced the concept in software construction by defining design patterns. Afterwards, many people used the concept in different domains of human computer interaction (HCI); e.g., for usability engineering (Mahemoff and Johnston 1998), interaction design (Borchers 2001; Tidwell 2005; van Welie 2008), UI design (UI Patterns; van Welie et al. 2000), decision support (de Moel and van der Veer 2011), etc.

Most of the mobile platform vendors – like Apple, Google, or Microsoft – offer style guidelines for using a particular UI element or interaction type for specific purposes; however, van Welie et al. (2000) provide a fruitful discussion targeting the importance of proven solutions in a specific context provided by a pattern compared to the general rule given by a guideline.

Targeting this concern, many collections of mobile-specific UI and interaction patterns have also sprung up. There are several online mobile patterns libraries available; e.g., Pttms [pttms.com] categorizes screenshots of different real iOS apps according to the set criteria, Inspired-Ul [inspired-ui.com] provides UI styles in different categories taken from real mobile apps for iOS and Android, Android Pttms [androidpttms.com] works like Pttms but for the Android platform, etc.

In the literature, few examples can also be found in the same direction. Neil (2012) provided screenshots of real apps in well-organized categories, mainly focusing on iOS but also including some examples of other platforms. However, many of the given examples are not widely used and do not follow the underlying platform style guidelines. Nilsson (2009) provided a collection of UI design patterns for mobile apps while focusing on the pattern collection structure, in order to provide suggested solutions to problems grouped into set of problem areas. Roth (2002) presented a set of mobility patterns, which were derived from mobile apps to allow designers for reusing design elements as building blocks. Few others applied patterns in mobile domain in some specific context, e.g., Tesoriero et al. (2008) proposed an HCI design patterns language and used it for building mobile apps targeting cultural environments.

3. THE SETS OF MOBILE-UI TS PATTERNS

Our proposed sets of mobile-UiTS patterns provide solutions in specific contexts for the transition of UI design elements or interaction concepts from one platform to another that adhere to the design guidelines and interaction schema of the target platform. Moreover, they also tackle the scaling issues of UI elements when moving from one size of mobile devices to another size.

We group the proposed mobile-UiTS patterns into two groups, transition patterns and scaling patterns, according to their nature. We are currently considering three platforms: Google Android, Apple iOS, and Microsoft Windows Phone. The following subsections provide details on both pattern groups.

3.1 Transition Patterns

Transition patterns can be divided into three categories: style patterns, interaction patterns, and structural patterns:

- **Style Patterns:** These patterns address the issues and problems related to the visual design (such as displays, themes, colours, textures, etc.) of the app UI. When transitioning a mobile app interface – designed for a particular platform – to other platforms, each version’s interface needs to possess the style factors of the underlying platform in order to remain consistent with the underlying platform style guidelines and provide the same user experience.

- **Interaction Patterns:** These patterns address the issues and problems related to the actual user interaction (such as interactive gestures, app layout, navigation rules, app settings, etc.) in the given context that may result from the transition of the app UI from one platform to another one. These patterns aim at providing consistent user interaction for the target app in accordance with the interaction paradigm of the underlying platform.

- **Structural Patterns:** These patterns address issues and problems related to common user interface elements (such as Tabs, Lists, Scrolls, Buttons, Spinners, Text Fields, Bars, Switches, Dialog, Content View, etc.). They aim at providing a platform-consistent positive user experience by being compliant to the existing platform standards when transitioning a UI element from another platform.

In terms of usage, the transition patterns have two layers of complexity. The first one is the simple and straightforward transition level; here, a visual
design, user interaction technique, or UI element from the source platform is transitioned directly to the same level of visual design, user interaction technique, or UI element in the target platform. In this case, the source and the target platforms are supposed to provide the same level of functionality and interaction for the underlying concept while remaining consistent with their own environments. In this context, the related pattern provides solutions for selecting the correct corresponding concept in the target platform. Some simple examples: The iOS standard provides three basic styles (i.e., gray, opaque black, and translucent black) for the overall app UI theme. Android also provides three different basic styles (i.e., holo light, holo dark, and holo light with dark background). So in this case, each theme in iOS can be directly transitioned to a theme in Android (e.g., gray style to holo light) and vice versa. Regarding gestures, the iOS tap gesture can be directly transitioned to the Android touch gesture. The iOS buttons can also be transitioned directly to Android buttons, and vice versa, with the same theme. The only exceptions are buttons that deliver a functionality that is supported by the Android default buttons (i.e., back, menu, search, and home buttons). These differences need to be covered directly by the target pattern.

The second level is more complex as the transition from the source to the target platform may not be straightforward. This is either because the target platform does not provide the same level of functionality and interaction for the underlying concept or because it provides the functionality and interaction in another way. In such a scenario, the relevant pattern indicates the best option on the target platform that can replicate the desired design, interaction concept or UI element as closely as possible according to the source platform in order to provide the same functionality and interaction across the different versions of the app while remaining consistent with the user experience of each underlying platform. For example, the iOS platform provides flick and swipe gestures, while the Android platform provides only the swipe gesture. For transitioning flick and swipe gestures from iOS to Android, extra care is necessary when defining the required interaction actions at design time as well as during implementation time. Therefore, in this case it must be specified what kind of swipe version the designer wants in the Android-based version compared to the iOS flick and swipe gestures.

4.2 Scaling Patterns

Scaling patterns deal with the scalability issues that occur when moving from one size of mobile device to another size. Generally, these patterns are used when resizing the common user interface elements or when performing changes in the layout structure – overall or in particular UI elements – due to the changes caused by the size of the device and the resolution. The focus in these patterns is on describing under which circumstances and how much we need to scale an interface element based on its meaning, as well as on tackling the issue of restructuring the overall layout while moving from one device size to another one. We divide these patterns into two categories: element-resizing patterns and layout patterns.

- **Element-Resizing Patterns**: These patterns address issues and problems regarding the changes that happen in UI elements – such as changes to buttons, text fields, bars, switches, content views, etc. – when moving the mobile app from small-size devices (i.e., phones) to large-size devices (i.e., tablets) and vice versa. The aim is to provide a better user experience in the target device, in accordance with the underlying device screen size and resolution. For example, when we talk about a button in the app UI in the context of the scaling issue from phones to tablets, the related pattern highlights the circumstances in which the button would have the same size and those circumstances in which it would be enlarged, along with the possible proportion of such an enlargement.

- **Layout Patterns**: These patterns address issues and problems regarding the changes that happen to the overall layout structure or to some particular group of UI elements as a result of a change in device screen size and resolution. Since more space is available on tablets than on phones (and since the normal orientation in tablets is horizontal compared to the vertical orientation in phones), sometimes there is a need to rearrange the layout structure in order to provide a better user experience in line with the device environment. For example, when moving an app UI from phones to tablets, a decision must be made regarding the default orientation (vertical or horizontal) and the corresponding layout. In such cases, the relevant patterns provide the solutions that fit best in a particular context for the rearrangement of specific UI elements or of the overall structural layout.

5. THE MOBILE-UITS PATTERNS TEMPLATE

Table 1 shows the template that we designed for writing the proposed mobile-UITS patterns description. The first column shows the item name while the second column provides the description in the context of the underlying pattern. The origin of the template is based on: analysing patterns templates from the literature (Gamma et al. 1995; Mahemoff et al. 1998; Tesoriero et al. 2008;
Patterns for Designing Scalable Mobile App User Interfaces for Multiple Platforms
S. R. Humayoun ● S. Hess ● F. Kiefer ● A. Ebert

Tidwell 2005; UI-Patterns (van Welie et al. 2000), studying the industry practice and vendors’ style guidelines (e.g., Google Android UI Guidelines, iOS human interface guidelines, Microsoft Windows Phone, etc.), and targeting patterns for the current mobile paradigm.

Table 1: Template for mobile-UiTS patterns description.

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pattern ID</strong></td>
<td>A unique ID for each pattern.</td>
</tr>
<tr>
<td><strong>Pattern Name</strong></td>
<td>An expressive unique name.</td>
</tr>
<tr>
<td><strong>Hierarchy</strong></td>
<td>Hierarchy in the pattern catalogue.</td>
</tr>
<tr>
<td><strong>Problem</strong></td>
<td>Stating the problem in which the pattern provides the solution.</td>
</tr>
<tr>
<td><strong>Context</strong></td>
<td>The background details (e.g., the situation in which the pattern can be applied).</td>
</tr>
<tr>
<td><strong>Solution</strong></td>
<td>The pattern solution in the light of the described context. In the case of transition patterns, the possible transition solution for the target platform. In the case of scaling patterns, the possible new size and layout of the UI element(s) or the overall new structure of the app UI in the target device.</td>
</tr>
<tr>
<td><strong>Resulting Context</strong></td>
<td>This is for the details provided afterwards to show the situation after applying the solution.</td>
</tr>
<tr>
<td><strong>Examples</strong></td>
<td>Illustrating the pattern with example(s). Each pattern can reference a catalogue containing several implemented examples.</td>
</tr>
<tr>
<td><strong>Design Suggestions</strong></td>
<td>Important suggestions that need to be considered while transitioning the design to the target platform or scaling the UI elements and layout structure to the target device.</td>
</tr>
<tr>
<td><strong>Alternative Patterns</strong></td>
<td>List of patterns that could also be implemented in this situation.</td>
</tr>
<tr>
<td><strong>Inverse Patterns</strong></td>
<td>List of patterns that provide the reverse solution (e.g., for a gesture transition pattern from iOS to other platforms, this will contain the references from those platforms to iOS).</td>
</tr>
<tr>
<td><strong>Optional</strong></td>
<td>Any relevant optional information.</td>
</tr>
</tbody>
</table>

Example 1: This transition pattern is about the overall theme of the app UI for three platforms (i.e., iOS, Android, and Windows Phone):

- **Name**: Overall Theme Style Transition
- **Hierarchy**: MobileUiTS/Transition/Style/Theme/
- **Problem**: The designer/programmer wants to convert the main theme of the app UI from one platform to another one.
- **Context**: This pattern applies to the overall theme of the app UI rather than to any particular building block.
- **Solution**: (Note: the platforms in this solution are iOS, Android, and Windows Phone, respectively.)
  1. gray style (default) – holo light – light
  2. opaque black – holo dark – dark,
  3. translucent black – holo light with dark action bars – customized.
- **Design Suggestions**:
  - Need to concentrate on changing “Translucent black” to “Holo light with dark background” while transitioning from iOS to Android.
  - As Windows Phone does not have a theme comparable to iOS “Translucent black” or Android “Holo light with dark action bars”, the designer needs to take care of this while converting from iOS and Android to Windows Phone and vice versa. The action bar should be black when converting from iOS and Android to Windows Phone.

Example 2: This transition pattern is about the gesture used in three platforms (i.e., iOS, Android, and Windows Phone) for dragging freely an element:

- **Name**: Drag Element Gesture
- **Hierarchy**: MobileUiTS/Transition/Interaction/Gestures/
- **Problem**: The designer/programmer is looking for the specific gesture in order to drag an element freely in the frame.
  - The designer/programmer wants to implement a drag-and-drop kind of gesture for moving an element from one place to another place anywhere in the frame without any restriction to the direction.
- **Context**: This pattern is applied when the Drag gesture for moving an element from one place into another place works freely anywhere in the frame.
  - The pattern is not applied if the designer/programmer wants to move the object in a particular direction or place.
**Solution:**

<table>
<thead>
<tr>
<th></th>
<th>iOS</th>
<th>Android</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>(to drag an element)</td>
<td>Drag</td>
<td>Drag</td>
<td>FreeDrag</td>
</tr>
</tbody>
</table>

- In the iOS platform, this gesture is represented by the *Drag* gesture and is used for dragging an element. The gesture works by a long press, move, and then lift the finger.
- In the Android platform, this gesture is represented by the *Drag* gesture. The gesture works by a long press, move, and then lift. It is same as re-arranging data within a view or moving data into a container.
- In the Windows Phone, this gesture is represented by the *FreeDrag* gesture. This gesture works by touching the object with a finger and then moving it in any direction.

**Design Suggestions:**

- **Windows Phone platform** provides a *VerticalDrag* gesture for moving object in up or down direction and a *HorizontalDrag* gesture for moving object in left or right direction. However, iOS and Android platforms provide only one *Drag* gesture for all kinds of dragging. Therefore, it is required to use only *FreeDrag* gesture in Windows Phone if the corresponding *Drag* gesture in iOS or Android represents free dragging.

**Example 3:** This scaling pattern is about resizing the “OK” button when moving from mobile to tablet devices:

- **Name:** OK Button Resizing
- **Hierarchy:** MobileUiTS/Scaling/Element-Resizing/Buttons/
- **Problem:** The designer/programmer wants to resize “OK” button on tablet devices.
- **Context:** Many apps have a main OK button for finalizing some task. This button normally appears at the bottom of the app. This pattern tells when to scale up and when not to scale up while moving from mobile to tablet devices.

**Solution:**

- **Tablet in vertical view:** In this case, the button will be scaled up according to the proportion of the tablet screen size compared to the mobile screen size.
- **Tablet in horizontal view with single frame:** In this case, the button will be scaled up according to the proportion of the tablet screen size in vertical view, but will be placed at the bottom-center of the view.
- **Tablet in horizontal view with two frames (where the left frame provides other options):** In this case, the button will not be scaled up, but will be placed at the bottom-center of the right side view.

**Design Suggestions:**

- Need to concentrate on coordinates when the tablet device is in horizontal view with two frames.

**DISCUSSION AND CONCLUDING REMARKS**

The design and development of mobile apps differs from that of conventional desktop applications in a lot of ways, due to many factors (e.g., single-task focused goal, multi-touch interaction paradigm, etc.). Most of the general HCI patterns approaches (Borchers 2001; de Moel and van der Veer 2011; Mahemoff and Johnston 1998; Tidwell 2005; UI-Patterns; van Welie et al. 2000; van Welie 2008) focuses on only desktop environments; hence, their suitability to the current mobile domain is applicable only to a limited extent. We distinguish them by focusing on the current mobile domain and by dealing with the scalability issues on different mobile device sizes.

The work targeting mobile environments, on the other hand, provides libraries with just screenshots of mobile apps in different categories (Neil 2012; Ptttns; Inspired-Ul; Android Ptttns), or for creating only a specific mobile app UI style (Nilsson 2009; Roth 2002), or for applying patterns in a specific area (Tesoriero et al. 2008). None of these patterns deals with the challenge of designing and developing mobile apps for multiple platforms. Moreover, the online libraries do not follow any formal approach and lack any guidelines for using such collections. Furthermore, the vendors’ style guidelines (e.g., Google Android UI Guidelines, iOS human interface guidelines, [Microsoft Windows Phone]) fail to describe the transition of the app UI design and interaction to/from another platform. Compared to the previous pattern-based work for mobile environments, our mobile-UlTS patterns are different and they advance the state-of-the-art due to many reasons: first, they deal with the issue of transitioning the mobile app UI design and interaction from one platform to another platform, which is not properly covered by any previous work; secondly, they tackle the scaling issues when moving from one size of smart device to another size (e.g., from smartphones to tablets and vice versa); thirdly, the provided template helps in writing these patterns in a formal way, which is the de factor way in the state-of-the-art patterns literature (Gamma et al. 1995; Mahemoff and Johnston 1998; UI-Patterns ;van Welie et al. 2000; van Welie 2008).

There are several benefits of our proposed mobile-UlTS patterns, such as: they tackle the issue of providing consistent user experience and usability
across all versions of the target mobile app; they minimize the effort required on the part of the designers; they provide a solution combining the vendors’ style guidelines and common practices found in existing mobile apps.

We also envision providing prototyping tool support in order to facilitate the building of interactive prototypes for multiple platforms and device types based on using the rules and guidelines provided by these mobile-UiTS patterns. Most of the current prototyping tools for mobile apps – such as Antetype1, Axure2, Balsamiq3, iMocBuilder (Humayoun et al. 2013), etc. – do not respect patterns; hence, the creation of the app UI and the interaction concepts is up to the designer. In the case of designing mobile apps for multiple platforms and device types, these tools do not offer much help to the designers in terms of getting the same user experience and usability across all designed versions. We plan to extend the work of a mobile mock-ups framework (i.e. Humayoun et al. 2013) in order to use these mobile-UiTS patterns to support the creation of interactive prototypes. This framework extension will thus help the designers to follow the patterns when designing the app UI for multiple platforms and device types.

In the future, we intend to extend our patterns catalogue in order to accommodate changes in upcoming style guidelines as well as to enhance our patterns for other platforms. We also intend to perform empirical studies to analyse the effect of our patterns on designers and developers during their work.

6. REFERENCES


Android Ptrtns: http://androidpttrns.com/


de Moel, N. and van der Veer, G. C. Design pattern based decision support. ECCE '11, ACM, 93-96, 2011.

Gamma, E., Helm, R., Johnson, R. and Vlissides, J. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, 1995.


Inspired-UI: Mobile Apps Design Patterns [iPhone], http://inspired-ui.com/


Microsoft Windows Phone: http://dev.windowsphone.com/en-us/


Ptttns: http://ptttns.com/


---

1 http://www.antetype.com/
2 http://www.axure.com/
3 http://balsamiq.com/