Evaluation of Descriptive User Interface Methodologies for Mobile Devices

Michael Tschernuth, Michael Lettner, and Rene Mayrhofer

Upper Austria University of Applied Sciences
FH OOE Forschungs & Entwicklungs GmbH, Softwarepark 11,
4232 Hagenberg, Austria
{michael.tschernuth,michael.lettner,rene.mayrhofer}@fh-hagenberg.at
http://www.fhooe.at

Abstract. The difficulty application developers in the mobile phone environment are facing is the variety of devices on the market. Since there are a few relevant global players with different operating systems and devices, it is getting more expensive to adapt applications once implemented from one hardware platform to another. To keep the costs at a minimum, a structured and reusable software architecture is crucial. Especially the maintenance of the user interface (UI) is a big challenge, thus it is beneficial to have a platform independent description of it. This paper aims at the upper layers of the software which are responsible for the user interface. A few established user interface declaration technologies and an own approach will be investigated. It is shown, that it could be beneficial when the logic—which is responsible for screen changes—is contained in the user interface description.

1 Introduction

Developers of mobile phone applications are facing the problem of implementing the same application for a variety of devices. Since there are a few relevant global players with different operating systems ([6]), more effort is needed to port applications once implemented from one platform to another. A structured and reuseable software architecture is crucial for minimizing costs for different platforms. Separating the user interface from the application logic itself can be a useful means to optimize the architecture.

A declarative approach to define the user interface independently of the application logic and following the model view controller pattern, which was introduced by Trygve Reenskaug [1], is a well established method to achieve such a separation. The implementation of the view layer is the concern of the paper. *Android OS*, *IPhone OS (iOS)*, *Windows Phone 7 (WP7)* as well as the *Qt Meta-Object Language (QML)*—which is used in Qt Quick (Qt UI Creation Kit)—all provide such an architecture but the interpretation is different.

Due to a variety of screen sizes of mobile devices a UI description which is independent of the program logic, is essential if an application should be reused

R. Moreno-Díaz et al. (Eds.): EUROCAST 2011, Part II, LNCS 6928, pp. 520–527, 2011.

for different devices. Furthermore it can raise the quality of the software as layout changes do not affect the source code and the maintenance of the user interface can be done independently from the code.

The idea to separate the concerns is not a new one. Well established solutions exist in the desktop environment [7]. This research investigates declarative solutions for some of the most relevant—in terms of market share—smartphone platforms. An analysis of several mobile platform solutions and methodologies will reveal the similarities and differences regarding the user interface layer and what actually is contained in the UI description.

An own XML definition—which is used for the architecture of an embedded feature phone [5]—shows that it could be beneficial to have the information of how the screens are connected in the user interface declaration.

Which field of application our XML UI description offers is shown in Section 4, where tools—targeting the mobile software development process—of a low resource phone are presented which utilize this additional information. On of the tools is a translation tool which is used to translate the completed product into different languages. The other one shows how the description can be used to transform the UI definition to another mobile platform in a practical example. As cross-platform solutions are numerous (e.g., *RhoMobile*[1], *Titanium*[2], *WidgedPad*[3], *PhoneGap*[4], *MoSync*[5]) and already provide features for a generic user interface implementation, this should merely serve as a transformation case study and not as an actual cross-platform product.

2 Related Work

There are different approaches on how to create a user interface description. This paper describes various declarative interface description technologies with the main focus on methodologies in the mobile phone domain. In this section a short introduction into existing solutions in general is given. As an additional objective it is investigated if it is possible to enhance the UI layer of the mobile phone platforms with the structural information. If that holds true, our tools presented in Section 4 can be utilized by this platform. The interface description languages based on XML are numerous [3]. However this paper investigates user interface descriptions which are used for the common available software development kits for mobile phones because it is a main objective to translate the approach proposed in Section 3 to the available platforms as a future work. Implementations which are based on modeling the user interface (e.g.,[2]) are not discussed, as there is no practical applicability in combination with our approach.

[1] RhoMobile: `http://rhomobile.com/`
[2] Titanium von appcelerator: `http://www.appcelerator.com/`
[3] WidgetPad:`http://widgetpad.com/`
[4] PhoneGap: `http://www.phonegap.com/`
[5] MoSync: `http://www.mosync.com/`

2.1 Android

Android uses an XML based layout file to define the user interface. The description is hierarchically organized, so every element (e.g., a Button) or *View* needs a corresponding layout type (*ViewGroup*) where it is embedded (Fig. 1). One *ViewGroup* can contain multiple *View* objects. During runtime the layout is loaded by the *Android* class that it was built for [10].

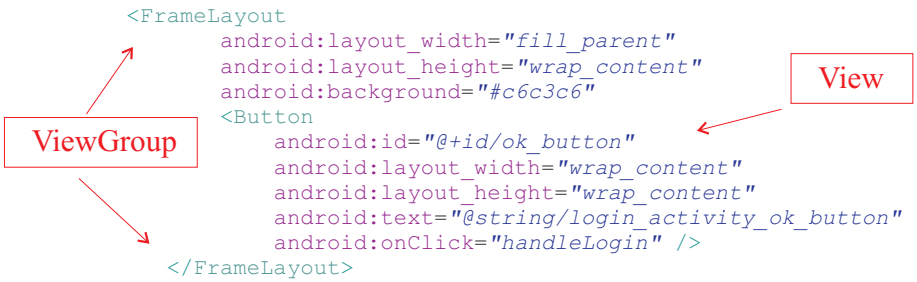

Fig. 1. XML definition of a Button (View) which is embedded in a layout (ViewGroup)

Connection of Screens. The relation between different screens cannot be derived from the XML file. It is done by starting a new Activity. During the starting process, the user interface description of the new class is loaded. The XML description of one screen is not aware of any following screens as the connections are implemented in source code (Controller). *Android* follows the MVC pattern very strictly, thus it is *not* possible to put additional information in the view layer.

2.2 IPhone

IPhone uses the *Interface Builder* for user interface creation. Outcome of this process is an XML file following the XIB syntax [11]. The description of a basic button element is depicted in Fig. 2.

```
<object class="IBUIButton" id="197068213">
        <reference key="NSNextResponder" ref="766721923"/>
....
</object>
```

Fig. 2. Basic button definition

Connection of Screens. The architecture is similar to the *Android* concept. The user interface description has no knowledge about how the screens are connected and how the overall structure of different screens look like. Screen transitions have to be done in source code.

2.3 Windows Phone 7

The windows phone platform uses the *Extensible Application Markup Language (XAML)* as a user interface definition. The difference to the other XML descriptions is that each XAML element is a representation of a .NET object [4].

Connection of Screens. The typical way to implement screen changes is to manipulate the code-behind file from a UI element (e.g., a Button) and set the target XAML file in source code. However it is possible—although not common—to to create links between several XAML files in the XAML file itself.

2.4 *Qt Quick* for Symbian

The only not XML based UI definition in this investigation scenario is *QML*, which is part of *Qt Quick*. In *QML* it is possible to declare a user interface using *JavaScript*. Due to the usage of *JavaScript* it is possible to build more sophisticated user interfaces which already contain behavior. Compared to *Android* or *IPhone OS*, *QML* does not provide user interface elements. However, it is possible to create these elements using *JavaScript*. A set of built-in components is currently under development and will be available in future releases aka Qt Quick Components [13,14].

Connection of Screens. As *QML* can utilize *JavaScript* to introduce functionality in the user interface declaration it is possible to connect a series of screens without using the underlying platform[6]

2.5 Evaluation

The discussed platforms have been evaluated regarding their applicability for our tools presented in Section 4. Table 1 shows a comparison of the investigated declarative approaches for mobile devices. Microsoft's user interface description can be—although it is not the intended approach—adapted to work with our tools. *Android* and iOS cannot be used that easy because the controller logic has to be extracted from the source code which imposes a challenge. Qt is not XML based and therefore not applicable. The following section describes our approach where navigational context and content description are available in the XML declaration file.

3 Alternative XML description

During our project for a feature phone an XML description was developed to represent the user interface layer [5]. The previously mentioned XML descriptions are too generic to meet the requirements for the scarce resource phone. The

[6] Layout Manager: `http://wiki.forum.nokia.com/index.php/Implementing_a_simple_View_Manager_with_QML`.

Table 1. Comparison of several declarative approaches for mobile devices

Platform	XML	navigational context in the view
Android	Yes	No
iOS	Yes	No
WP7	Yes	Yes
Qt	No	Yes

developed XML descriptions only contains elements of the specific system and uses an adapted MVC approach where additional information was introduced to the view layer. This section briefly describes the developed XML user interface description which contains the content of the screens as well as the information about how the screens are connected. The investigated XML descriptions from Section 2 are not—except XAML where structure information can be added to the declaration—designed to support this alteration of the MVC approach. A practical example of that idea is given in the following subsections.

3.1 Declaration of the Content

All graphical objects that appear on the screen are contained in the definition file. Elements always have to have a parent *state*, which is one single screen, where they are attached to. How the elements of the real screen are transformed into an XML description is shown in Fig. 3.

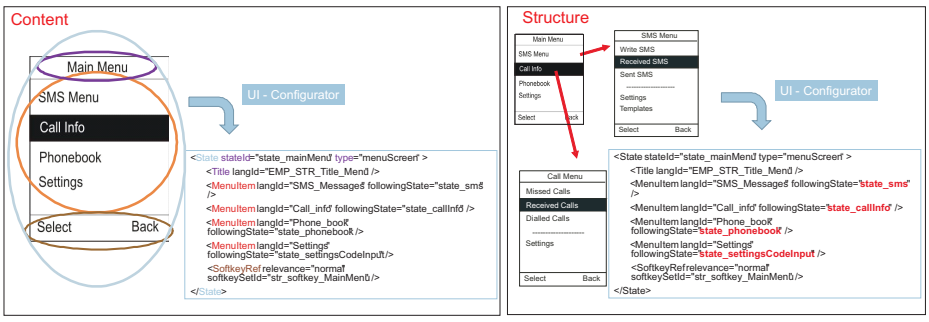

Fig. 3. Example of the content and structure definition

3.2 Declaration of the UI Structure

The connection between different *states* is achieved by adding an additional attribute to every element which is able to initiate a transition between two screens (Fig. 3).

Including the structure in the XML description offers new possibilities which resulted in a few tools which support the software engineer during the development phase. The creation of a prototype for translation purposes is one of them. The next section will present some software tools based on this XML definition.

4 Applications Based on Our XML Description

Although the description of the user interface is used as a basis for the feature phone it can be used independently of the underlying layers as well, to support the development process and to present different views of the whole system. Having navigational context available in the user interface definition resulted in a variety of tools which utilize this additional information. Within this section these tools are explained in detail.

4.1 Language Translator

For localization purposes a tool was developed to support a native speaker in translating the software into different languages. Due to the restricted screen sizes on mobile devices translations it is sometimes not sufficient to provide an as is translation. The string length has to be taken into account to match the current screen area. Under that circumstances it is beneficial that the *context* in which the current screen appears *is known*, thus the expert is able to optimize the translation for this limited space requirement.

Our developed XML description provides the navigational context, therefore it is possible to create a representation of the UI declaration for translation purposes. Fig. 4 shows how a visual representation looks like in the translation tool. Every user interface element is aware of its following screen thus it is possible to navigate along the defined structure. Ids of the translatable items for each screen are shown in the tool. The user has the possibility to change the translation for an id. This translation is stored in a separate XML file, whereas the XML structure definition remains unchanged. The structure itself only serves as a plot of the user interface which is beneficial for the translation process.

Field of Application. This tool offers a convenient way of translating applications which are based on the proposed XML description. The UI description and the translation are stored in XML files therefore these are the only elements that have to be exchanged to translate a new product.

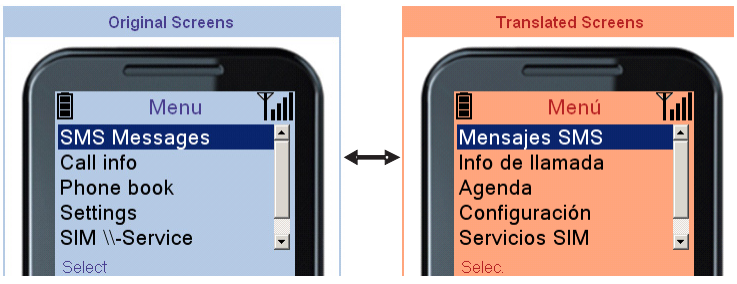

Fig. 4. Screenshot of the translation tool

4.2 Transformation to Android

As the information contained in our XML structure is a super set of the *Android* view it is possible to generate code for the *Android* platform to represent our user interface. The developed transformation tool is able to generate the XML files needed for the Android user interface on the one hand and the sourcecode for navigating between the defined screens on the other hand.

Example. Fig. 5 shows the transformation from the an XML file (*StateContent.xml*) to the Android XML description plus the source code for the navigation. The example illustrates that it is possible to transform the XML content file to Android because all the information is available in a defined format.

The other way around is more difficult as the information is not only contained in the user interface description files. The interface files lack the navigational context which has to be extracted from the sourcecode. This imposes a problem as there are numerous way to induce a screen change.

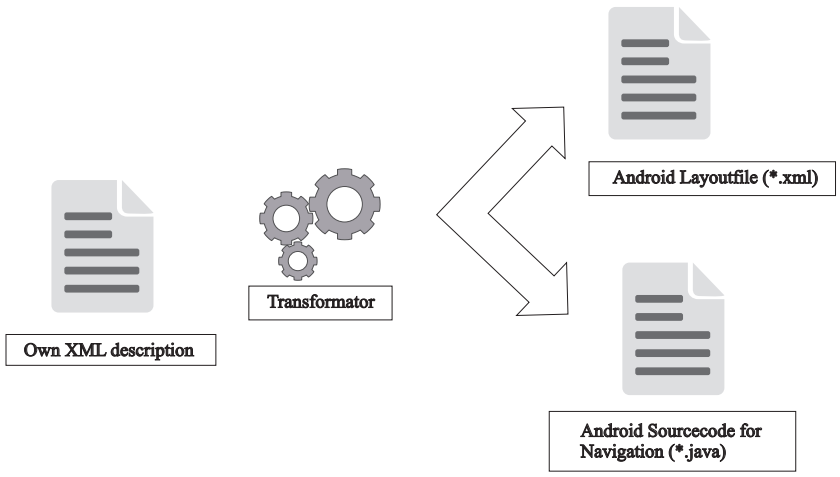

Fig. 5. Transformation overview

Field of Application. This software application serves as a case study for look and feel tests of the user interface definition on *Android*. An additional benefit this transformation tool provides is the acceleration of *Android application* development if an XML structure definition is available, because the source for the screen connections is generated automatically.

5 Future Work and Conclusion

The software tools presented in Section 4 are already used in practice and the software development is supported by their usage. To utilize those tools for the

development on *Android* a structure extractor (Extracturer) is in development. With this tool it will be possible to retrieve the relevant controller parts from the source code and pack them into the view layer. An additional field of application for the user interface definition can be for documentation purposes, as the structure could be exported into a document where all the screens and their connections are printed. The transformation to other programming languages is another aspect which will be investigated in the future to benefit from the already developed toolchain.

References

1. Trygve Reenskaug. The original MVC reports (2007)
2. da Silva, A.R., Saraiva, J., Silva, R., Martins, C.: XIS UML Profile for eXtreme Modeling Interactive Systems (2007)
3. Souchon, N., Jean, V.: A review of xml-compliant user interface description languages (2003)
4. Charles, P.: Programming Windows Phone 7 (2010)
5. Lettner M., Tschernuth M.: Applied MDA for Embedded Devices: Software design and code generation for a low-cost mobile phone (2010)
6. Statistic: mobile phone market shares, `http://www.gartner.com/it/page.jsp?id=1434613`
7. Technology Report, `http://xml.coverpages.org/userInterfaceXML.html`
8. Developer Center - XUL, `https://developer.mozilla.org/En/XUL`
9. Android Developers, `http://developer.android.com/guide/`
10. Arno, B., Marcus, P.: Android 2 (2010)
11. Dave, M., Jeff, L.: Beginning iPhone Development: Exploring the iPhone SDK
12. QML - Reference Guide, `http://doc.qt.nokia.com/`
13. Fitzek, F.H.P., Mikkonen, T., Torp, T.: Qt for Symbian. Wiley & Sons Ltd, United Kingdom (2010)
14. Qt Quick, `http://qt.nokia.com/qtquick/`