An Abstract User Interface Framework for Mobile and Wearable Devices

Claas Ahlrichs, Universitaet Bremen, Germany
Michael Lawo, Universitaet Bremen, Germany
Hendrik Iben, Universitaet Bremen, Germany

ABSTRACT

In the future, mobile and wearable devices will increasingly be used for interaction with surrounding technologies. When developing applications for those devices, one usually has to implement the same application for each individual device. Thus a unified framework could drastically reduce development efforts. This paper presents a framework that facilitates the development of context-aware user interfaces (UIs) with reusable components for those devices. It is based on an abstract description of an envisioned UI which is used to generate a context- and device-specific representation at run-time. Rendition in various modalities and adaption of the generated representation are also supported.

Keywords: Abstract UI, Context-Aware User Interfaces, Mobile Devices, Unified Framework, Wearable Devices

INTRODUCTION

Desktop computers present no longer the only affordable technology with reasonable computing power. Over the last decades a new trend in computing has emerged: mobile computing. In recent years, mobile computing platforms have become available to the broad masses. Their price has decreased and today almost everybody can profit from devices, like smartphones or personal digital assistants (PDAs).

Nowadays many people use mobile phones, MP3 players or digital cameras in their daily lives. All of these devices provide functionality which can be accessed via a UI. They are typically graphically represented and must ensure an appropriate level of usability and information presentation in order to let users control the provided functionality. However, usability and information presentation are rather often neglected. E.g. using applications of mobile phones may be a burden as their UIs are still frequently based on desktop applications. Instead of creating an application that is specialized to solve a (single) problem, desktop applications are usually rather general purpose and feature-rich. While this may be a useful approach for desktop computers, it is not for
mobile and wearable devices as they significantly differ from stationary systems (e.g., in terms of size, usage, computing power, etc.). Unnecessary features tend to hinder usability than being a useful extension when being in a mobile setting.

Not having expert knowledge on creating UIs for wearable and mobile scenarios makes it hard to develop applications for such devices. The heterogeneous nature of the hardware of these devices intensifies this problem as existing UIs are unlikely to be reused. A unified framework, that offers reusable components and facilitates UI development, could reduce development efforts and decrease programming errors.

UIs could be described in an abstract manner, so called abstract user interfaces (AUIs). AUIs can be used to transport information just like regular UIs. They are used to specify what is presented rather than how information is presented. Thus no concrete representation or visualization is defined instead a single AUI description can be represented in multiple ways. Furthermore an AUI can easily be adapted according to user preferences, device constraints or contextual information.

RELATED WORK

Several frameworks for automatic UI generation already exist. Three of them are listed and summarized as follows. Most frameworks can also be used to create wearable and mobile computing applications.

- **WUI Toolkit**: Witt et al. (2005) introduced “a toolkit for context-aware UI development for wearable computers” called wearable user interface (WUI) toolkit. It was designed and developed to meet requirements of wearable computers and aimed to ease development of WUIs. The toolkit first utilized reusable UI components and was based around a model-driven approach. It support self-adapting UIs without being limited to specific interaction devices or graphical UIs (Witt, 2005; Witt, Nicolai, & Kenn, 2007).
- **Huddle**: Huddle is a system that uses an abstract description language for automatic generation of task-based UIs for appliances in a multi-device environment (e.g. a home theater or presentation room). It makes use of an XML-based language for describing functionalities of appliances in those environments (e.g., televisions, DVD players, printers or microwave ovens). Huddle has been used to generate graphical and speech interfaces for over thirty appliances on mobile phones, handhelds and desktop computers (Nichols & Myers, 2007; Nichols, Myers, Litwack, Higgins, Hughes, & Harris, 2004; Nichols, Rothrock, Chau, & Myers, 2006).
- **SUPPLE**: SUPPLE is an alternative to creating UIs in a hand-crafted fashion. Instead UIs are automatically generated with respect to a person’s device, abilities and preferences. It is based around an abstract UI describing its functionality rather than its representation. The actual generation of UIs with SUPPLE is interpreted as an optimization problem (Gajos & Weld, 2004; Gajos, Weld, & Wobbrock, 2010).

AbstractUI

The AbstractUI framework eases development of applications using AUIs. It has been specifically designed to meet requirements of mobile and wearable devices. Furthermore development efforts are reduced as the same application can be used on multiple devices. A context-aware representation of an AUI can be generated at run-time.

Application developers describe what is to be displayed rather than how an envisioned UI is displayed. While doing so developers make use of several UI components and can interact with them just like they would with standard UI components of other toolkits and frameworks. Once an application is completed, it is passed to a renderer which in turn takes care of the concrete representation.
The requirements and limitations of the AbstractUI framework are listed in the following. Furthermore a general overview of the framework is given.

**Requirements**

The AbstractUI framework has been designed and implemented in consideration of several requirements.

- **Device independent UI description**: The UI is to be described in a way that allows the rendering on a wide range of devices and usage of UI toolkits or frameworks. It should be specified on an abstract level that is independent of a particular rendering software (e.g., AWT, SWT, GTK+, etc.) or device (e.g., PDA, head-mounted display (HMD), etc.).

- **Reusability of components**: Reusability of UI components allows a more productive and effective development of wearable and mobile applications. Instead of creating an application from scratch with a specialized interface, a list of default UI components (that are likely to be reused) should be identified and provided for re-usage. The identified components are not expected to be exhaustive, meaning that they can be used to create a wide range of applications but certainly not all possible ones.

- **Support for integration of context**: Gathered contextual information (by sensors, user, wearable) can help to optimize the rendering of a UI. The usage should not be restricted to internal software components but also permitted to be directly used within applications developed with the AbstractUI framework. A global storage place for context information would allow the propagation of newly gained or changed contextual information.

- **Extensibility**: New ways of displaying information emerge from time to time and create the necessity to adapt the rendering process. Existing UI components may be required to be rendered in a different way or new UI components may need to be integrated into the toolkit. The identified list of UI components is not exhaustive and therefore the AbstractUI framework must permit the addition of new UI components.

- **Support for distribution of toolkit components**: Wearable and mobile systems are typically very limited in terms of available computing power and energy consumption. Therefore the possibility to distribute an application across multiple systems in a network should be considered e.g., a wearable system could act as a display of a wearable application and communicate with a second system (with more computing power) on which the actual wearable application is executed.

- **Support for multi-modal information presentation**: UIs of wearable and mobile devices are not restricted to graphical representations. In fact there are cases in which a graphical representation is not the preferred way to display information e.g. when walking, crossing the street or in general when a user’s visual attention is occupied by a real world task. Some information can equally well be displayed using tactile or auditory interfaces. The UI components should be designed in a way that allows the possibility to render them in a non-visual modality.

**General Overview**

The implementation of the AbstractUI framework makes frequent use of design patterns found in Gamma, Helm, Johnson, and Vlissides (1994) e.g., the Abstract Factory (Gamma, Helm, Johnson, & Vlissides, 1994, p. 87) and the Observer (Gamma, Helm, Johnson, & Vlissides, 1994, p. 293) design pattern have been applied several times.

A set of five UI components is provided by the AbstractUI framework, which was chosen as their concrete representations cover a broad range of commonly used UI components. Even though this set is not exhaustive, it enables application developers to create a fairly large...
amount of applications. The UI components of the AbstractUI framework are summarized as follows.

- **Text**: Is used to display text-related information. Depending on the context, it could be represented as a label, text box or text area. Thus also allowing user inputs.
- **Trigger**: Enables users to execute actions (e.g., send form or save file). It will typically be represented as a labeled button, but could also be displayed as part of a menu.
- **Choice**: Is responsible for displaying selection-related information (e.g., month, weekday, etc.). The most obvious representation is probably the combo box. However depending on the context it is being used in, a check box, radio buttons or a list might also be suitable.
- **Container**: Is a component that allows the addition of child components. It will usually be represented as some sort of panel or dialog box.
- **Screen**: Is a top level container and will typically represent as a window.

The Composite design pattern (Gamma, Helm, Johnson, & Vlissides, 1994, p. 163) has been applied to all of the above components and allows treating of individual components and compositions in a uniform way. Compositions and individual components do not need to be distinguished, thus reducing development efforts.

Each UI component represents a certain kind of data (e.g., a text, an action or a choice) and thereby has an individual data model. The data models were designed in a way that allows easy interchangeability e.g. the same data model can be implemented to store its information in the device it is being used on or retrieve its data from a remote computing machine.

Contextual information (e.g., environmental information, user’s mental state, etc.) can be represented in the AbstractUI framework. It provides a globally accessible storage point for such information. The context information is represented as a simple key-value pair (e.g., ‘illumination’ → ‘bright’). Furthermore, all UI components are equipped with a way to represents the context they are being used in.

Having used above components to create an AUI, one can pass it to a renderer and have a concrete representation build for it.

**Limitations**

The design of the framework is not limited to a specific programming language nor does it suggest one in particular. Nonetheless, the implementation of the AbstractUI framework has been done in Java and therefore can only be used on Java-enabled devices. Future versions will target additional programming languages and may include support for the .Net programming environment or the iPhone/iPad; thus extending the range of supported devices. Projects like XMLVM (2011) could be utilized for such a purpose.

Although UIs for wearable and mobile devices are not limited to graphical representations, the primary focus of the proposed framework will be on graphical output. The existence of other output modalities (e.g., auditory and tactile) was kept in mind during the development and renderers using them can be created.

AbstractUI provides five basic UI components (**Text**, **Trigger**, **Choice**, **Container**, **Screen**) that can be used to create applications. Even though many applications can be created with them, there are still a lot that cannot be represented with those five components. However, missing UI components can be added to the AbstractUI framework.

Even though the AbstractUI framework supports distribution of framework components across multiple machines no actual implementation exists that does so. The default data models and event providers do not utilize network capabilities. However, further families of data models and event providers can be created to do so.
**Evaluation**

**Description of User Study**

The idea of the user study was to compare Witt’s WUI toolkit (Witt, Nicolai, & Kenn, 2007) to the AbstractUI framework from the viewpoint of an application developer. Consequently, a part of the conducted user study was to use both toolkits and create applications with them.

A total number of 12 subjects took part in the user study. Most of them were students at University Bremen (10 subjects) while the remaining ones were local staff members (2 subjects). 10 of them were male and 2 female. All subjects were between 23 and 31 years old (average age was 25).

At the beginning of the user study, each subject was asked to fill out a simple questionnaire. They reported their age, gender and whether or not they had used the toolkits beforehand. Furthermore subjects rated their Java-skills on a scale from 1 to 10 (1 meaning very poor to 10 meaning very good) as well as their English-skills also on a scale ranging from 1 to 10.

Subjects were given several minimalistic examples demonstrating the use of key components for both toolkits (e.g., Trigger in AbstractUI framework and ExplicitTrigger in WUI toolkit). The examples for the AbstractUI framework and WUI toolkit contained the same functionality. One example for each key component and toolkit was presented to the subjects (4 exercises * 2 toolkits = 8 examples). Furthermore an overview of the four key components and where to find them in the corresponding toolkit was given to the subjects.

Subjects could then look through the resources and ask questions at their own ledger. They were told that that all resources were allowed to be used throughout the entire user study. Once they felt comfortable with them they started with the programming exercises.

A total number of four exercises (one for each key component), which had to be completed with both toolkits, were part of the user study. Subjects were told to have about 5 to 8 minutes to solve a given problem in a particular toolkit. Once the exercise was completed in one toolkit, they were asked to fill out a NASA-Task Load Index (TLX) form. The same exercise was then completed using the remaining toolkit and the subjects were asked to fill out another NASA TLX form. Having completed the exercise with both toolkits, subjects were asked to rate how well they could solve the exercise in each toolkit on a scale from 1 to 10. This was repeated for all exercises.

The programming was done using Eclipse IDE (Eclipse Foundation, 2011) with preconfigured Java-projects. The projects contained two sets of empty Java-classes, one set for each toolkit.

The order of all exercises was changed with each subject as well as the order of the toolkits for each exercise. The sequence of both was balanced across all subjects.

Having completed all exercises the subjects were asked whether or not they felt to have mixed up the toolkits. They were also asked to rate their overall experience with both toolkits on a scale from 1 to 10. The user study was then finished with an informal interview in which all subjects were given a chance to comment on everything they felt noteworthy.

**Results**

The subjects’ NASA TLX and usability ratings were analyzed using a paired t-test (two-tailed). The findings will be depicted in the following.

The NASA TLX rating of the AbstractUI framework and the WUI toolkit was statistically significant (p < 0.01). The average NASA TLX rating for both toolkits can found in Figures 1 and 2. The latter illustrates the average NASA TLX rating for each subject, whereas the first shows the overall NASA TLX rating of both toolkits. The average NASA TLX rating for the AbstractUI framework was 26.80 and 43.43 for the WUI toolkit (0 meaning a very low task load index and 100 meaning a very high task load index).

The usability rating of both toolkits was also statistically significant (p < 0.01). The
average usability rating for each subject is illustrated in Figure 4. Figure 3 shows the average usability rating for both toolkits. Subjects’ usability ratings averaged for the AbstractUI framework at 9.28 and the WUI toolkit at 4.92 (1 meaning very poor to 10 meaning very good).

Discussion

All 12 subjects completed the entire user study without complications. Most of them rated their own Java- and English-skills to be good (rating $\geq 7$). The average subject required about an hour in order to complete the user study. The fastest subject required about 40 minutes while the slowest subject needed almost 90 minutes.

Some subjects commented on the toolkits during the informal interview session. Several of them were complaining about the inconsistent interfaces and weird naming conventions of the WUI toolkit. All of them commented on how easy and straight forward the AbstractUI framework was when comparing to the WUI toolkit. One subject was so pleased that he was already looking forward to programming with the AbstractUI framework when having to complete an exercise using the WUI toolkit. Other subjects felt that they spent 80% of their time understanding and programming with the WUI toolkit while the remaining 20% were used for the AbstractUI framework and paperwork. Furthermore, several subjects told the interviewer that they were fans of the slim source code they produced while completing the exercises with the AbstractUI framework.

From observation of the subjects, it became quite clear that the AbstractUI framework is simpler and more intuitive to use. The results support this. The NASA TLX ratings of all subjects for both toolkits were statistically significant and show that the AbstractUI frame-
work requires less workload than the WUI toolkit. Statistically significant was also the usability rating of both toolkits and indicate that the AbstractUI framework is more usable than the WUI toolkit.

The results of the experimental evaluation indicate that the AbstractUI framework outperforms the WUI toolkit in terms of usability.

CONCLUSION AND FUTURE WORK

A framework for developing context-aware UIs on mobile and wearable devices has been introduced. Reusable UI components and an abstract description are used in order to generate concrete representations. Furthermore, a user study has been conducted which compared the WUI toolkit to the AbstractUI framework from the viewpoint of an application developer. The results indicate that the AbstractUI framework outperforms the WUI toolkit in terms of usability.

In the course of working on the AbstractUI framework, several ideas were generated e.g. porting the AbstractUI framework to different programming languages (e.g., dotNet, Objective-C, JavaScript, Ruby, etc.) or comparing the AbstractUI framework with further frameworks in the ‘Related Work’ section. The first would allow the use of the AbstractUI framework on various devices (e.g., iPhone, Windows Phone, etc.). Furthermore, the implementation of UI generation as an optimization problem is a very interesting approach (Gajos & Weld, 2004) and will certainly be looked into.

REFERENCES


Claas Ahlrichs is a student at the University Bremen. He enrolled in 2007, started to study computer science and recently finished his diploma thesis. He is working as student assistant at the TZI (Center for Computing and Communication Technologies) in the field of wearable computing.

Michael Lawo is with TZI (Center for Computing and Communication Technologies) of Universitaet Bremen since 2004. He is professor for applied computer science involved in numerous projects of wearable computing and artificial intelligence. He is a 1975 graduate of Ruhr-Universitaet-Bochum, got his PhD in 1981 from Essen University and became professor there in 1992. He has more than 15 years of experience in the IT industry in different management positions, and is author, co-author and co-publisher of eight books and more than 120 scientific papers on numerical methods and computer applications also in healthcare, optimization, IT-security and wearable computing.

Hendrik Iben is with TZI (Center for Computing and Communication Technologies) of University Bremen since 2007. He is a PhD candidate involved in wearable computing projects and lectures. He is a 2007 graduate of University Bremen.