Supporting Designers in the Creation of Multi-Device User Interfaces

Jan Meskens
Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
jan.meskens@uhasselt.be

ABSTRACT
The design of User Interfaces (UIs) for several computing devices with different characteristics and constraints is a complex activity. Designers often have to switch between several design tools and alternate between run and edit modes in order to polish the UI designs for each individual device. I am exploring novel design techniques where designers can create UIs directly on the target device while the UI is running: (1) the design mirror, (2) the continuous mouse pointer and (3) the design toolglass. These three techniques allow designers to polish UIs for each device without having to switch between design tools nor run and edit modes.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

General terms: Design, Human Factors

Keywords: Multi-Device User Interfaces, Gummy, GUI Design Tools, Live Editing.

INTRODUCTION
People tend to use a diverse set of computing devices such as mobile phones, desktop PCs and game consoles for reading their email or browsing the web. This increases the need for applications that are available on multiple devices. However, designing a user interface (UI) manually for different computing platforms is far from simple due to the differences in form factor, input and output devices and user interface toolkits between these platforms.

I am investigating a multi-device UI design approach that empowers the UI designers. Essential to this approach are appropriate tools that do not force designers to considerably change their current work practices.

The research I present in this paper encompasses three interrelated parts:

• A survey on the current needs and issues when creating UIs for other devices than desktop PCs [12].
• The multi-device GUMMY [14] Graphical User Interface (GUI) design tool that allows designers to create the visual design of a GUI in a similar way as when dealing with traditional GUI builders. GUMMY can translate a UI designed for one platform into an initial design for another platform as well.
• Three live UI editing techniques that are built on top of GUMMY and allow designers to interact with the UI on the target device while it is being designed: the design mirror [12], the continuous multi-device mouse pointer [13] and the design toolglass.

MOTIVATION
To get a better understanding of how designers currently create UIs for other devices than desktop PCs, four UI designers were interviewed across four companies [12]. Artefact and project walkthroughs were employed to identify problems designers faced when using design tools. Most of the problems that the designers mentioned were caused by mismatches they experienced between the design environment and the actual computing platform. Below, the most important mismatches the designers noted during the interviews are summarized:

• Size and position mismatches: two of the interviewed designers often had problems when estimating the size and position of user interface components due to the high DPI of the target device’s screen. These designers found it also difficult to forecast the necessary amount of white space across screens with different sizes and DPIs.
• Occluded area mismatches: one designer found it hard to estimate which areas would be occluded by her fingers when interacting with a touchscreen. After deploying a GUI design, she often experienced that the feedback on a certain action was not always visible because it appeared in an occluded area.
• Colour mismatches: one designer noted differences between the colours of the GUI inside the design environment and the colours of the runtime GUI.
• Device emulator mismatches: since it is time consuming to deploy a UI design to the actual target device, designers often rely on device emulators to test their designs. However, emulators do not reveal all runtime characteristics of a designed UI. For example, emulating touch-based interaction with mouse clicks does not reveal the sensitivity and

1Dots Per Inch
For several years, Multi-Device UI Design towards UI designers are other multi-device design approaches that are more oriented behind the curtain, but never exposed to the designers. Visual design of a UI. Abstract models and languages are used to control the look and feel of the resulting UI. In Gummy, high quality UIs. However, it remains difficult for designers to control the hardware prototype and observe live in the d.tools design environment how the UI changes. This is different from my approach, where the designer’s activities can be observed live on the target device.

Multi-Device UI Design
For several years, model-based UI design techniques have been applied to create multi-device UIs [15, 5]. Rather than specifying a visual UI design from scratch for each computing platform, these systems allow designers to specify a UI only once by means of an abstract model. The platform-specific UIs are then generated automatically from this abstract description. Even though model-based UI design can simplify the development of UIs for multiple devices, this technique can be daunting for designers. Designers have to master a new language to specify the high-level models and cannot accurately predict what the resulting user interface will look like [16].

Model-based UI design certainly has its benefits. Several model-based approaches such as SUPPLE [6] and the Personal Universal Controller (PUC) [17] are able to produce high quality UIs. However, it remains difficult for designers to control the look and feel of the resulting UI. In Gummy, this problem is solved since designers can directly alter the visual design of a UI. Abstract models and languages are used behind the curtain, but never exposed to the designers.

Other multi-device design approaches that are more oriented towards UI designers are Damask [10] and SketchiXML [4]. These tools allow designers to create UI prototypes by sketching. In Damask, designers can select the UI patterns that occur in their designs. These patterns are then used to translate prototypes from one platform to another. SketchiXML maps sketches directly to UI elements to create either a low-, mid- or high fidelity prototype. Transformation algorithms are applied to transform prototypes between different levels of fidelity. In contrast with these approaches, Gummy does not employ sketching but allows designers to create high fidelity prototypes by dragging concrete widgets from a toolbox. This improves the visual feedback and does not force designers to learn sketch gestures.

Live UI Editing
Live UI editing concerns editing a GUI directly on the target device while the GUI is running. Notable examples of this technique are UI adaptation systems such as PageTailor [2], User Interface Façades [19] and Arnauld [7]. Most of these systems only support a limited set of design operations and are often hard to use when designing user interfaces from scratch. This is in contrast with the Morphic [11] UI construction environment, which is an example of a complete live development platform that allows designers to change the attributes, structure and behaviour of user interface components by pointing at their graphical presentation while the UI is running.

Live editing techniques are often difficult to apply in a multi-device design approach. First, not every device (e.g. a smartphone with hardware buttons) can be used to design UIs due to the different input and/or output constraints of each device. Secondly, different devices may require different interaction techniques to edit a UI. For example, designing a UI on a PDA with a stylus is very different from creating a UI on an iPhone with touch screen.

THE GUMMY MULTI-DEVICE GUI BUILDER
As a general framework for experimenting with UI design tools, I created the GUMMY [14] multi-device GUI builder. GUMMY’s workspace is structured in a similar way to traditional GUI builders. It contains a toolbox showing the available user interface elements, a canvas to build the actual user interface and a properties panel to change the properties of the user interface elements on the canvas. After selecting the desired target platform, GUMMY adapts its workspace to support UI design for this platform. While designers shape the visual representations of a UI, GUMMY maintains a synchronised high-level UIML2 [9] description for this visual design. This avoids designers from having to deal with abstract descriptions or models.

When designers want to make a switch to another computing platform, GUMMY uses a transformation algorithm to generate an initial design for this new platform. Designers can then refine this initial design. My current implementation uses a basic rule-based algorithm, similar to the one described by Collignon et al. [3]. Based on the underlying abstract UI description, the algorithm tries to select appropriate renditions for each widget depending on the screen size and installed toolkit on the new computing platform. To also take into account more advanced device characteristics such as the provided interaction mechanism (e.g. a mouse or a touch screen), a more robust algorithm such as SUPPLE [6] might be used.

To show the visual design in the canvas, GUMMY relies on a UIML interface renderer for the selected target platform. Each action a designer takes in GUMMY implicitly affects the underlying UI description. This description is sent to an interface renderer, which returns a set of bitmaps representing the components employed in the design. GUMMY then uses these bitmaps to build up the concrete view of its canvas. Currently, GUMMY can communicate with UIML renderers for Java (ME). .NET (on Mono, Windows and Windows Mobile), Adobe Flash and Android.

2User Interface Markup Language
LIVE-UI DESIGN INTEGRATION

In my research, live UI editing will be combined with multi-device UI design to prevent mismatches between the UI design environment and the running UI. Properly integrating live UI editing in GUMMY requires communication between the target device and design tool. For this reason, I rebuilt the GUMMY architecture around the extensible messaging and presence protocol (XMPP). As shown by Pierce et al. [18], this Instant Messaging protocol provides the required functionalities and flexibility to build distributed UIs. In the networked GUMMY architecture, a target device runs a client application that hosts a running version of the currently designed UI. Both the target device and GUMMY join an XMPP chat session and use this session to communicate (see Figure 1(a)).

In live UI editing, it is important to have an appropriate interaction technique that allows designers to easily intertwine edit mode and run mode. When applying live UI editing in a multi-device design approach, this interaction technique should also scale over many different devices. The remainder of this section discusses three live design interaction techniques that I created using GUMMY's networked architecture: the design mirror, the continuous multi-device mouse pointer and the design toolglass.

The Design Mirror

The design mirror [12] extends GUMMY by streaming each action a designer takes in GUMMY directly to the running UI on the target device (see Figure 1(b)). For example, if a designer moves or resizes a component in the design workspace, the runtime version of this component is moved or resized in the live UI simultaneously. By visualising every change on the target device immediately, designers do not have to switch between run and edit mode to test the runtime implications of design changes. This way, designers are not forced to invest additional time or effort in acquiring the runtime characteristics of their design, which can help to avoid mismatches.

Continuous Multi-Device Mouse Pointer

Another live UI editing technique is the continuous multi-device mouse pointer [13]. This is a special mouse pointer which can for example start on the desktop PC of the designer, cross the bezel of the screen and end on the screen of the mobile device (see Figure 1(c)). The mouse pointer can be used for repositioning and resizing UI elements, adding new elements by dragging an element from the toolbox on the designer’s PC to the running UI on the target device and changing the properties of UI elements. While this design pointer is employed to edit a UI design, the normal interaction method (e.g. pressing on a touch screen, the normal mouse cursor, etc.) can still be used to interact with the design.

The Design Toolglass

The design toolglass (see Figure 2) is a see-through interface [1] that is placed on top of a UI running in a device emulator. If there is a target device available, the actions the designer takes in this design toolglass are immediately streamed to a UI on this device as well.

Similar to GUMMY, the design toolglass’s major components are a toolbox with UI elements that are suited for the selected target platform (Figure 2, A), a canvas (Figure 2, B) and a properties panel (Figure 2, C). The canvas is a transparent overlay that triggers design actions on top of a running UI design. This UI is hosted by an application that communicates with the toolglass and can either run in a device emulator (e.g. the Android or Windows Mobile emulator in Figure 2) or directly on the PC of the designer (e.g. Adobe Flex in a webbrowser). To select a certain UI element, the designer has to click through the canvas on this element. This results in a bounding rectangle that is drawn on the canvas around the selected element (Figure 2, D). The properties of the selected component can be observed and adapted through the
properties panel. New components are added by dragging an element from the toolbox to the canvas. Each action a designer takes in the design toolglass is streamed directly to a running UI on the actual target device. However, if the actual device is not present, designers can minimize the toolglass and test their design on the emulator.

ACHIEVEMENTS AND RESEARCH VISION

For the creation of multi-device UIs, designers often have to switch between different design tools, and alternate between edit and run modes to estimate if the designed UI fits the target device constraints. In order to facilitate the design of multi-device UIs, I developed the GUMMY multi-device GUI builder together with three live UI editing techniques: (1) the design mirror, (2) the continuous multi-device mouse pointer and (3) the design toolglass. These techniques allow to interact with a UI on the target device while it is being designed.

The current prototypes of the presented design techniques have evolved through informal experiments with a small number of designers (see e.g. [12]). According to my intuition, both the continuous multi-device mouse pointer and the design mirror can help to target mobile devices but might not be helpful for designing UIs for large displays. In contrast with mobile devices, large horizontal or vertical displays are difficult to align with the designer’s PC. The position of the target device with respect to the designer’s PC screen can complicate tasks like reading text or manipulating objects. The design toolglass might scale better over a wide range of devices since designers can always rely on a device emulator if a device cannot be placed at an optimal position with respect to the designer’s PC.

Although first experiments were conducted to refine the developed prototypes, a thorough validation will be needed to evaluate the value of these techniques for UI designers. I plan to conduct a study in which designers apply our live design techniques to create UIs for several platforms: a mobile device, a web application and a large display. Based on the feedback from this study, I will iterate over the design techniques in order to create a better multi-device live editing system. Rather than forcing designers to use a certain technique, it will allow to use a combination of the different techniques. This system will also allow designers to create parts of the multi-device UIs in concert.

In a future version of my approach, I plan to support designers in specifying interface behaviour as well. By integrating my approach in a programming environment, I plan to study if designers programmers can benefit from the live running UI when they are adding behaviour to their UI designs. For example, extending the programming environment with a visualisation of the events that occur in the live UI might help designers to empirically discover the events that are interesting for specifying a certain interaction.

ACKNOWLEDGEMENTS

I would like to thank Professors Karin Coninx and Kris Luyten for advising me on this work. Part of the research at EDM is funded by ERDF (European Regional Development Fund) and the Flemish Government. The AMASS++ (Advanced Multimedia Alignment and Structured Summarization) project IWT 060051 is directly funded by the IWT (Flemish subsidy organization). I also thank Jeffrey Pierce for his useful insights and comments on my research.

REFERENCES