Shortening User Interface Design Iterations through Realtime Visualisation of Design Actions on the Target Device

Jan Meskens Kris Luyten Karin Coninx

Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2, B-3590 Diepenbeek, Belgium
{jan.meskens,kris.luyten,karin.coninx}@uhasselt.be

Abstract

In current mobile user interface design tools, it is time consuming to export a design to the target device. This makes it hard for designers to iterate over the user interfaces they are creating. We propose Gummy-live, a GUI builder for mobile devices allowing designers to test and observe immediately on the target device each step they take in the GUI builder. This way, designers are stimulated to iteratively test and refine user interface prototypes in order to take the target device characteristics into account.

I. Introduction

Today, various Graphical User Interface (GUI) builders such as Visual Studio or NetBeans can be used to create User Interfaces (UI) for mobile devices. These tools are geared towards designers, either developers or graphical designers, and allow the creation of GUIs by the use of a visual language that contains the user interface primitives. This visual language hides complex programming constructs and allows designers to concentrate on the look and feel of the design.

To test mobile applications, most GUI builders are tightly coupled with device emulators that imitate how the design will behave on the target device. The major drawback of these emulators is that they do not reveal *all* runtime characteristics of a designed UI. Their usage is often very different from using the real device; e.g. emulating touch based interaction with mouse clicks does not reveal the sensitivity and precision of a mobile device's touch screen. In order to observe and test all these runtime characteristics, designers have to *deploy* the design as a "live" GUI to the target device. Because of the time and effort it takes to deploy an application, a *gap* exists between the design inside the GUI builder and the live GUI on the target device. Consequently, designers may encounter difficulties to determine what the live GUI will look like.

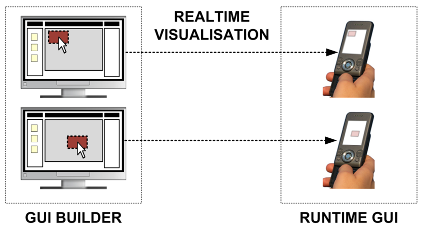

Figure 1. The Gummy-live design approach

Instead of creating designs in a design tool, *"live editing"* [1], [7], [10] approaches allow designers to edit a GUI directly on the target device while the GUI is running. This way, designers receive instant feedback about the implications of their design decisions which bridges the aforementioned gap. It becomes challenging, however, to enable live editing on mobile devices with constraints such as limited input or low screen resolutions.

In this paper, we explore the integration of the target device as part of the GUI design environment. This is implemented as a new version of the multi-device Gummy [9] GUI builder called Gummy-live. While designers create user interfaces inside our GUI builder (see Figure 1, left), every design action is immediately reflected in a live UI located on the target device (see Figure 1, right). This allows the designers to continuously test and observe the impact of their design decisions while using a traditional GUI builder. Since all design operations are done in a GUI builder, Gummy-live can target a mobile device regardless of its in- or output constraints.

II. Motivation

To get a better understanding of how designers currently design UIs for mobile devices, we interviewed four UI designers across four companies. Based on these interviews, we identified some problems designers faced during the design of user interfaces for mobile devices. All these issues were caused by mismatches designers experienced between the design environment and the actual computing platform. The three most important mismatches designers mentioned during the interviews are summarised below.

1. 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 mobile device's screen.

2. Occluded area mismatches: one designer found it hard to estimate which areas would be occluded by her fingers when touching the screen of the mobile device. 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.

3. Colour mismatches: another designer noted differences between the colours of the GUI inside the design environment and the colours of the runtime GUI.

The findings in these interviews are consistent with the *"information barrier"* in end-user programming systems as identified by Ko et al. [6]. In UI design, this barrier represents several properties of the design environment that make it difficult for designers to acquire *information* about their design's runtime characteristics. Properties that typically influence the height of this barrier are the number of steps and the amount of time it takes before a designer can acquire the runtime characteristics of the designed GUI.

III. Gummy-live

Gummy-live consists of two major parts that are carefully integrated as depicted in Figure 2: the Gummy *design workspace* (designer's Pc) and a *live-view* (target device). The design workspace is always in edit mode and structured similarly to traditional GUI builders: there is the *toolbox* (see Figure 2, part 1) showing images of the available user interface elements, the *canvas* (see Figure 2, part 2) to create a GUI design using direct manipulation and the *properties panel* (see Figure 2, part 3) to change the properties of the user interface elements on the canvas. The Gummy design workspace supports all design operations of traditional GUI builders and is fully described by Meskens et al. [9].

The live-view component is always in run mode and harvests the currently designed user interface. This live-view allows observing the runtime characteristics of the designed GUI. A live-view component was preferred to the live editing of a user interface on the target device for two reasons. First, not every computing platform is suited to act as a development platform due to the available screen size, limited processing power or limited set of input devices (e.g. a low performance keyboard or mouse). Second, a live editing mode would require designers to learn a new design method that is suited for the selected target device. When targeting a new device, designers might be forced to learn another more suitable methodology. Using our approach, designers can always use the same design environment while observing the result on the target device through the live-view.

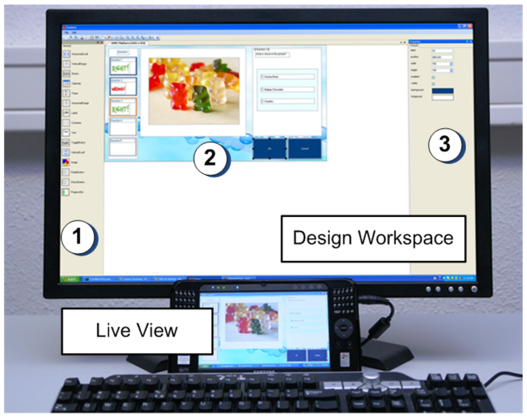

Figure 2. The Gummy-live setup.

When a designer changes the GUI design in the workspace, each change is immediately visualised in the running GUI of the live-view. 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 view on the target device 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 will reduce the information barrier, since designers are not forced to invest additional time or effort in acquiring the runtime characteristics of their design.

IV. Implementation

When a designer makes changes to the GUI design in the design workspace, every change is encoded and immediately sent to the live-view. When receiving a design change, the live-view decodes and visualises this change in the live GUI. In our current implementation, the platform independent User Interface Markup Language (UIML) [5] is used to encode design changes. This language is completely hidden for the designer and could be replaced with other languages such as *XUL* or *HTML+CSS* as long as the live-view can interpret and visualise every design change.

The communication between the design workspace and live-view is established using the Extensible Messaging and Presence Protocol (XMPP) which allows exchanging messages through a (local) XMPP server. There are currently many implementations of this protocol available for several platforms including Windows Mobile, Symbian, Javascript, iPhone, Java ME, etc. For now, we have a C# implementation of the live-view that runs on Windows and Windows Mobile. In order to prove the flexibility of our approach, two *proof-of-concept* implementations for Java ME and Adobe Flash were created.

V. Informal User Study

To evaluate the work presented in this paper, we conducted an informal user study. The purpose of this study was to check if the subjects were able to create user interfaces for an Ultra Mobile PC (UMPC) in Gummy-live. More specifically, we wanted to assess whether the participants took the output of the live-view into account and changed their design decisions according to what they perceived in this live-view.

Seven subjects participated in our study, having different backgrounds and experiences. Five of the subjects were experienced programmers, two of the subjects had a non-technical background (a graphical designer and a usability engineer). The majority of our test audience only had limited experience with an UMPC.

During the evaluation, the participants were asked to create a "home automation system" user interface for an UMPC. The UI was described by means of a written list of functionalities the resulting UI should contain: changing the temperature of the heating, controlling the lights in several rooms and activating the alarm. During the experiment a think aloud protocol was used to understand the actions and decisions of the participants. Additional questions were asked by the observers if necessary.

We observed that five participants changed their design decisions according to the feedback they got from the live-view. Two subjects first selected a trackbar to change the temperature of the heating. Immediately after they dragged the trackbar to the canvas, they tested its runtime characteristics and discovered it was difficult to manipulate this widget using the UMPC's touch screen. This made them decide to use another widget. While one subject replaced the trackbar with two larger buttons to increase or decrease the temperature, the other one opted for a textfield where the temperature can be entered using the UMPC's keyboard.

Another subject immediately tested the runtime behaviour of the button she dragged to the canvas. This way, she discovered difficulties to hit the button through the touchscreen because it was quite small. Next, she increased the button size until it became easy to hit. At later stages of the design this subject used buttons as well, but now she omitted testing and immediately changed the size of these buttons to the size of the first one. One subject first opted for a toggle button to switch the lights on or off. In the live view, however, she immediately noticed the toggle button's feedback was not clear enough according to her opinion. Thus, she replaced the toggle button with a normal button and a label that mentions the status of the light (on or off).

One of the participants preferred to give the designed user interface colours that match to the UMPC's colours. For establishing this, she updated the colours of the user interface in the design environment and immediately checked these colours in the live-view. From her findings in this live-view, she updated the colours again in the design workspace until she found a good colour scheme.

VI. Discussion and Future Work

During our study, the majority of the participants regularly tested their intermediate designs. Moreover, they changed their design decisions according to what they perceived on the target device. This can be explained by the fact that Gummy-live minimises the effort needed to deploy the intermediate design to the target device. These results are consistent with our intuition that Gummy-live stimulates designers to test and verify there design decisions.

Some participants provided suggestions that could further improve the integration of the target device and the design environment. Two participants mentioned that is would be nice to diminish the physical distance between the live-view on the UMPC and the canvas. Another participant suggested the integration of a distributed drag and drop to drag elements from the toolbox directly to the target device. A prototype of a distributed drag and drop technique for designing mobile UIs is described in [8].

We have observed that our approach works for the design of mobile user interfaces, and we believe it can be extended to support other computing platforms as well. For example, it should be possible to implement a live-view that runs on a multi-touch table or a digital TV and to connect such a target device to the GUI builder. Gummy-live can be extended to target devices with wide varying physical sizes. It is important here to explore how the live-view and design environment are positioned within the designer's environment.

Future research will explore how Gummy-live can be used for designing interface behaviour besides the graphical interface design. We will evaluate how designers or programmers can benefit from the live-view when they are adding behaviour to their user interface designs. This allows designers to create more complex and complete UIs such as interactive visualisations.

VII. Related Work

A design framework closely related to the design methodology embodied in Gummy-live is presented by de Sá et al. [3]. Their framework allows creating software prototypes for mobile and desktop devices, and to polish these prototypes while the UI is running. However, contrary to our approach, they do not involve the target device during the whole design process.

Several tools developed for commercial or research purposes such as *SketchiXML* [2], *Gummy* [9] and *Qt designer* provide designers with a GUI builder workspace where they can polish a static representation of their user interface designs. In order to acquire all runtime characteristics of the designed UI, designers have to deploy the design explicitly to the target device. In Gummy-live, all characteristics are immediately revealed by integrating the target devices as part of the design environment.

User interface adaptation tools allow designers or end-users to edit user interfaces while they are running. *Pagetailor* [1], *User Interface Façades* [10] and the work presented by Demeure et al. [4] are examples of such approaches. Compared with our Gummy-live approach, most of these user interface adaptation systems only support a limited set of design operations and are often hard to use when designing user interfaces from scratch.

Morphic [7] allows designers to change the attributes, structure and behaviour of user interface components by pointing at their graphical presentation while the UI is running. The idea behind Morphic maps with the idea behind our approach. However, we believe that not all mobile devices are suited to act as a complete development or design platform due to the input and/or output constraints. The distributed nature of our approach can solve this problem and allows designing on a desktop pc while making every design change immediately visible on the target device.

VIII. Conclusion

We have presented Gummy-live, a GUI builder for mobile devices that immediately reflects each step a designer takes in the design tool on the target device. Gummy-live encourages designers to test and observe the results of their design decisions throughout the whole design. This way, they are guided to create optimal user interfaces for the target device. Based on the current findings, we will continue the development of Gummy-live. Additional, thorough validation is needed to estimate the value of Gummy-live for UI designers and developers.

IX. Acknowledgements

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). We also thank Jeffrey Nichols for his useful insights and detailed comments during the writing of this paper.

References

[1] N. Bila, T. Ronda, I. Mohomed, K. N. Truong, and E. de Lara. Pagetailor: reusable end-user customization for the mobile web. In *Proceedings of MobiSys '07*, pages 16–29, New York, NY, USA, 2007. ACM.

[2] A. Coyette, S. Kieffer, and J. Vanderdonckt. Multi-fidelity prototyping of user interfaces. In *Proceedings of INTERACT'07*, 2007.

[3] M. de Sá, L. Carriço, L. Duarte, and T. Reis. A mixed-fidelity prototyping tool for mobile devices. In *Proceedings of AVI '08*, pages 225–232, New York, NY, USA, 2008. ACM.

[4] A. Demeure, J. Meskens, K. Luyten, and K. Coninx. Design by example of graphical user interfaces adapting to available screen size. In *Proceedings of CADUI'08*, pages 1–6, 2008.

[5] J. Helms and M. Abrams. Retrospective on ui description languages, based on eight years´ experience with the user interface markup language (uiml). *International Journal of Web Engineering and Technology (IJWET)*, 4(2), 2008.

[6] A. J. Ko, B. A. Myers, and H. H. Aung. Six learning barriers in end-user programming systems. In *Proceedings of VLHCC '04*, pages 199–206, Washington, DC, USA, 2004. IEEE Computer Society.

[7] J. H. Maloney and R. B. Smith. Directness and liveness in the morphic user interface construction environment. In *Proceedings of UIST '95*, pages 21–28, New York, NY, USA, 1995. ACM.

[8] J. Meskens, K. Luyten, and K. Coninx. Plug-and-design: Embracing mobile devices as part of the design environment. In *Proceedings of EICS'09*, 2009.

[9] J. Meskens, J. Vermeulen, K. Luyten, and K. Coninx. Gummy for multi-platform user interface designs: Shape me, multiply me, fix me, use me. In *Proceedings of AVI'08*, 2008.

[10] W. Stuerzlinger, O. Chapuis, D. Phillips, and N. Roussel. User interface façades: towards fully adaptable user interfaces. In *Proceedings of UIST '06*, pages 309–318, New York, NY, USA, 2006. ACM.