Plug-and-Design: Bringing a Design Environment to a Mobile Device

Plug-and-Design: Embracing Mobile Devices as Part of the Design Environment

Jan Meskens Kris Luyten Karin Coninx

Hasselt University – tUL – IBBT
Expertise Centre for Digital Media
Wetenschapspark 2, 3590 Diepenbeek, Belgium
{firstname.lastname}@uhasselt.be

ABSTRACT

Due to the large amount of mobile devices that continue to appear on the consumer market, mobile user interface design becomes increasingly important. The major issue with many existing mobile user interface design approaches is the time and effort that is needed to deploy a user interface design to the target device. In order to address this issue, we propose the plug-and-design tool that relies on a continuous multi-device mouse pointer to design user interfaces directly on the mobile target device. This will shorten iteration time since designers can continuously test and validate each design action they take. Using our approach, designers can empirically learn the specialities of a target device which will help them while creating user interfaces for devices they are not familiar with.

Categories and Subject Descriptors

H.5.2 [**Information interfaces and presentation**]: User Interfaces – Graphical user interfaces, Prototyping

General Terms

Design

Keywords

Design tools, Mobile UI design, GUI builder

1. INTRODUCTION

User Interface (UI) designers often adopt a *"thinking-by-doing"* [11] design approach in which they try a set of design alternatives in search of the best solution. The iterative production of intermediate UI prototypes provides the crucial element of surprise, unexpected realizations that the designer could not have arrived at without producing a concrete manifestation of her ideas. Intermediate prototypes also reveal the characteristics of the target device that need to be taken into account, such as the sensitivity of a mobile device's touch screen or the screen resolution of a tablet PC. Consequently,

EICS'09, July 15–27, 2009, Pittsburgh, Pennsylvania, USA.

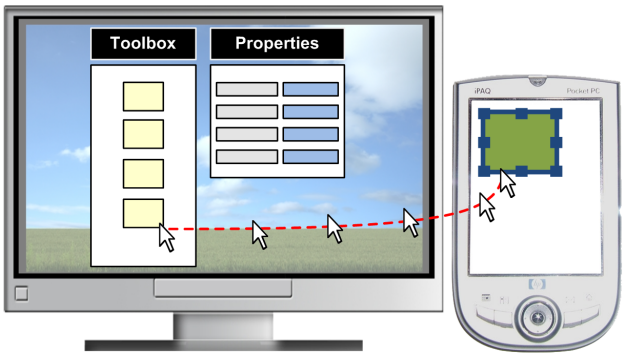


Figure 1: The plug-and-design system uses a continuous mouse pointer to design UIs directly on the target device.

prototypes created during design iterations provide invaluable input to end up with a user interface that is optimized for the targeted device(s).

For the creation of mobile UI prototypes, Graphical User Interface (GUI) builders such as those provided in traditional software development environments can be used. These tools hide complex programming constructs and allow designers to concentrate on the look and feel of the design. After deploying a design as a "live" UI to the target device, the runtime characteristics of this prototype can be observed. Due to postponing the deployment on the target device to the final stage, there is a large gap between the design inside the GUI builder and the live GUI on the target device, often defined as the *gulf of evaluation* [14]. Our approach closes this gap by using a live editing approach without losing the power of a traditional design environment.

Live UI editing approaches [5, 9, 12, 20], allow designers to edit a GUI directly on the target device while the GUI is running. This way, designers get instant feedback about the implications of their design decisions which bridges the aforementioned gulf of evaluation. Live UI editing avoids switching between run and edit mode and thus reduces iteration time and effort. This will stimulate designers to test and verify their design ideas continuously throughout the whole design process. However, based on our own experiences, we noticed that it becomes challenging to build live editing environments for mobile devices that are as powerful as traditional design tools [7]. The major problem here is that many operations have to be supported while mobile devices have constraints such as limited input, screen space limitations or low screen resolutions.

In this paper, we present the *plug-and-design* system that combines live editing and graphical user interface design tools for mo-

bile devices. The cornerstone of this approach is a continuous multi-device 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). After *plugging* a target device into our system, designers can use the mouse to start *designing* a UI directly on this target device. A traditional 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.

The main contribution in this paper is the tool support for plug-and-design, which is a novel live editing approach for creating mobile UI prototypes. After describing related work, we present a scenario of use that illustrates how our plug-and-design approach works. Finally, the implementation and underlying architecture of the plug-and-design system is discussed.

2. RELATED WORK

In this section we discuss related work from two different areas that are combined in our approach: live UI design and multi-display systems. To the authors' knowledge, the uniqueness of our approach lies in the combination of work from these two areas.

2.1 Live User Interface Design

A design framework closely related to the design methodology embodied in our plug-and-design environment is presented by de Sá et al. [6]. Their framework divides the design activity in two phases. The first phase allows designers to create sketches or software prototypes for mobile and/or desktop devices inside a design tool. After deploying these designs to the target device, the second phase allows designers to polish the UI using direct manipulation on the target device itself. By providing an integrated environment, our approach blurs the distinction between these two phases which means designers can build UIs directly on the target device during the whole design process.

User interface adaptation tools allow designers or end-users to edit user interfaces while they are running. *Pagetailor* [5], *User Interface Façades* [20] and the work presented by Demeure et al. [7] are examples of such approaches. While *Pagetailor* allows end-users to adapt the layout of a website on a mobile device using direct manipulation, User Interface Façades is limited to desktop computers and allows users to copy screen elements from one running program to another or replace interactors by other ones. The tool presented by Demeure et al., on the other hand, allows end-users to specify at runtime how a user interface should scale over various screen sizes. Compared with our plug-and-design approach, most UI adaptation systems only support a limited set of design operations and are often hard to use when designing user interfaces from scratch.

The *Morphic user interface construction environment* [12] introduces the concepts *liveness* and *directness* as a way user interface designers can examine or change the attributes, structure and behaviour of user interface components by pointing at their graphical presentation (directness) while the user interface is running (liveness). These properties were applied during the development of the *Parks PDA* [15], which showed that doing small amounts of programming directly on the target device saves development effort. More recently, the concepts of liveness and directness were integrated in Sun's lively kernel [9]. The idea behind the aforementioned live editing construction environments matches with the idea behind our approach. However, we believe that not every computing platform is suited to act as a complete development or design platform due to the limited availability of input devices or screen space. The distributed nature of our approach can solve this problem since parts of the UI design tool are located on the PC of the designer.

2.2 Multi-display Systems

Notable examples for doing cross-display operations in a multi-display environment are *Stitching* [8], *Pick-and-Drop* [17] and *HyperDragging* [18]. These approaches allow to use a pen or a mouse to e.g. drag elements from a tablet Pc to a Personal Digital Assistant (PDA). Besides traditional input devices like a pen or mouse, researchers have proposed other ubiquitous input devices such as the *Smart Phone* [2] or *Soap* [4]. To manage how different input devices can be used within (large scale) multi-device and multi-display environments, architectures like *iStuff* [3], *PointRight* [10] and *Synergy* [1] have been proposed. Compared with our approach, the aforementioned systems do not focus on the design of user interfaces. We adopt a multi-display mouse pointer as a mean to design user interfaces for different types of mobile devices while they are running.

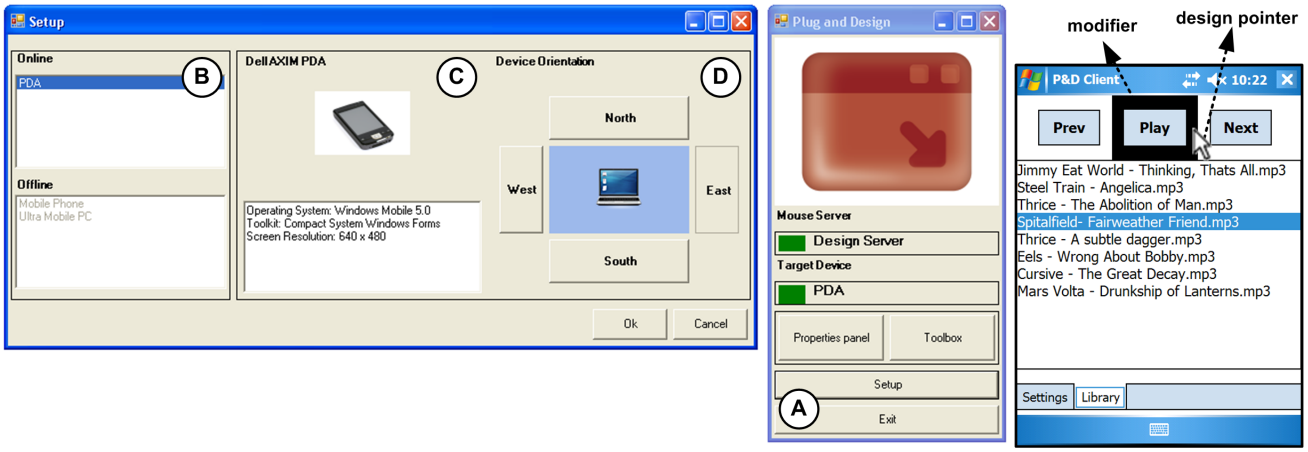
3. SCENARIO OF USE

We illustrate the use of our system by walking through an example scenario in which Emma creates a music player UI for a PDA using the plug-and-design system.

The plug-and-design environment running on Emma's PC (see Figure 2(a), A) contains two main parts: a setup dialog to plug the target device in the system and a design tool to manipulate a user interface design that is running on the target device. Emma starts by connecting the PDA to the local network and runs a plug-and-design client application on this PDA. As she starts this client, the PDA appears in the list of online device of the setup dialog (see Figure 2(a), B). She can now select this PDA in order to get some more information about device characteristics such as the installed operating system, toolkit and screen resolution (see Figure 2(a), C). The next step is to specify where the target device is located with respect to Emma's PC screen (see Figure 2(a), D), which is on the east side in this example scenario (see Figure 1). The property defined here is used by the underlying system to compute how the mouse pointer can move between the PC and the target device.

After selecting the target device, Emma can start using the design tool. This tool provides a toolbox with all user interface elements for the design and a properties panel to change the properties of these elements. The design itself is located on the target device and can be reached by the mouse of Emma's PC. When she moves her mouse pointer over the right edge - as specified in the setup dialog - of her PC screen, the mouse pointer appears on the PDA and is considered as a *design pointer* to add, select and manipulate elements of the running design. The actions performed with this design pointer will not affect the internal state of the user interface elements and thus clearly separates design actions from the UI behaviour. For example, *clicking* on a checkbox element with the design pointer will not check or uncheck this element (= design action) whereas *pressing* on the same checkbox through the PDA's touchscreen will check or uncheck this element (= UI behaviour).

Emma adds a button by dragging a button element from the toolbox to the design on the PDA. She immediately tests this button and discovers that it is too small and occluded by her finger when pressing on it. Using the mouse, and thus the design pointer, Emma now selects the button which makes a black modifier visible around this control (see Figure 2(b)). This modifier allows to resize and move the control with the design pointer using direct manipulation. She now enlarges the button until it is large enough to perceive its feedback. Emma also changes the label of this button into *"Play"*



(a) The plug-and-design system allows to plug a device into the environment using a setup dialog.

(b) The plug-and-design client hosts the running UI.

Figure 2: Screenshots of the plug-and-design environment on the PC of the designer (a) and the target device (b).

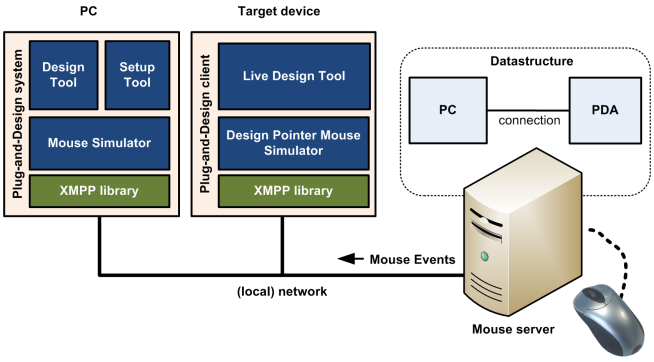


Figure 3: The plug-and-design system is implemented as a distributed system on top of an XMPP layer.

through the properties panel. After adding the play button, Emma adds two other buttons for selecting the previous and next song. She also inserts two tab pages: one for changing the settings of the player, and another one that shows the complete play list. The final music player UI created with the plug-and-design environment is shown in Figure 2(b).

4. IMPLEMENTATION

The plug-and-design software is distributed over three devices: a mouse server, the target mobile device and the PC of the designer (see Figure 3). The Extensible Messaging and Presence Protocol (XMPP) [19] is used as the communication protocol between these devices. As shown by Pierce et al. [16], this Instant Messaging (IM) protocol provides the functionalities and flexibility to build multi-device infrastructures. To participate in the plug-and-design environment, each device has to authenticate itself through the network to an XMPP server.

The mouse server maintains a data structure that describes the spatial relation between the target device and the designer's PC. This data structure is based on the space topology description as suggested in the PointRight system [10]. It consists of *screens*, represented by their respective sizes, and *connections* to specify mouse transitions that are possible between these screens. For example, Figure 3 shows the data structure specifying that the mouse pointer can "jump" from the PC screen to the PDA screen when it passes the right border of the PC screen. Designers can configure the spatial relation between the target device and PC using the setup menu that runs on the PC (see Figure 2(a)).

The mouse used by the designer is directly connected to theÂă-mouse server using e.g. Bluetooth or USB. Each time this mouse is manipulated, the server queries the aforementioned data structure to detect the device that should get mouse input. Next, the mouse pointing position is rescaled in order to fit within the screen size of the selected device. Finally, the mouse server encodes the mouse event into an XMPP message and sends this to the right device. The mouse events that are currently sent by our system are *mouse move*, left/right *mouse down* and left/right *mouse release*.

Both the target device and PC contain a *mouse simulation layer* which listens for mouse messages and uses these to steer the local mouse cursor. On the PC side, the mouse simulator generates *real* mouse events using the P/Invoke .NET API[1]. The simulator that is located on the target device steers the design pointer. This custom made pointer is closely integrated into the live design tool and enables UI editing by direct manipulation on the target device. For now, we have a C# implementation of this live design tool that runs on all .NET distributions. On the other hand, several *"proof of concept"* implementations are under development for Java ME[2] and Adobe AIR[3].

A connected target device describes its characteristics such as the installed operating system and screen resolution in a *vCard* which is accessible for all other participating machines through the *vcard-temp* XMPP extension[4]. While the setup tool uses this vCard to show details of the target device to the designer (see Figure 2(a)), the mouse server queries the vCard to set the screen size of the target device in its data structure. This vCard also contains a visual and textual description of the UI elements that can be provided by the target device's UI toolkit. Using this description, the design tool can build up its toolbox that shows icons of all these UI elements. When a designer *drags* an element from the toolbox to the

[1] http://www.pinvoke.net
[2] http://java.sun.com/javame/
[3] http://www.adobe.com/products/air/
[4] http://xmpp.org/extensions/xep-0054.html

target device, an identifier representing this UI element is sent to the live design tool on the target device. After releasing the mouse in this live design tool, the UI component that corresponds to this identifier is added at the position of the design pointer.

5. DISCUSSION AND FUTURE WORK

In the plug-and-design system, mobile user interfaces are designed by using a continuous mouse pointer that connects a design tool, running on the PC of the designer, directly with the UI on the target mobile device. Mouse transitions between the PC screen and the mobile device screen are done by passing the mouse pointer over the borders of these screens. This interaction method is similar to how a mouse is used in dual screen setups that are currently common place for desktop PCs and notebooks. We believe this similarity prevents designers from having to learn new interaction techniques for using our design environment.

Most mobile devices have a Pc-based emulation layer for design purposes. These emulators allow designers to layout mobile UIs on their Pc using a traditional mouse and keyboard. However, since these emulators run on a Pc, they do not provide the same interaction experience as the actual mobile target device. Using our approach, design actions and runtime actions can be intertwined on the actual target devices. This way, designers can immediately test and verify the impact of each design step directly on the target device.

Currently, the plug-and-design approach works for the design of mobile user interfaces. We believe this can be extended to other computing platforms too. For example, it should be possible to implement a plug-and-design client that runs on a multi-touch table or a digital TV and to connect such a target device to our design tool. An interesting area of future research here is to explore how the live design tool has to be positioned within the designer's environment. This is an important issue since the position of the target device with respect to the designer's PC screen can complicate tasks like reading text or manipulating objects [13].

The data structure that is currently maintained by the mouse server is flexible enough to define mouse transitions between an unlimited set of devices [10]. This opens up the possibility to plug multiple target devices simultaneously into the plug-and-design system. Further investigations might explore how designers could benefit from this when designing applications that have to run across multiple devices. For example, it would be interesting to support designers when creating a media player that is optimized for as well a mobile phone as an ultra mobile PC.

In our current plug-and-design architecture, mouse events are streamed by the mouse server using the XMPP protocol. However, depending on the network load, we noticed there could be too much latency to track cursor motion at adequate rates. In order to optimize this, we are currently integrating a direct socket communication channel between the mouse server and the other devices based on XMPP's *Jingle*[5] extension. In future versions of our tool, this channel will be used to stream consecutive mouse events such as mouse movements.

6. CONCLUSION

We have presented the plug-and-design system, a GUI builder for mobile devices that uses a continuous multi-device mouse pointer to design directly on the target device. Because designers are working directly on the running UI, they can test and observe their design decisions throughout the whole design process. This immediate feedback will reduce iteration time and guides designers to create optimal user interfaces for the target device. Plug-and-design is built on top of a networked architecture with the potential value to target a wide set of mobile platforms. Based on the current status of our work, we will continue the development of the plug-and-design system. Additional, thorough validation is needed to estimate the value of our system for UI designers and developers.

[5] `http://xmpp.org/extensions/xep-0166.html`

Acknowledgments

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).

7. REFERENCES

[1] Synergy. `http://synergy2.sourceforge.net`.

[2] R. Ballagas, J. Borchers, M. Rohs, and J. G. Sheridan. The smart phone: A ubiquitous input device. *IEEE Pervasive Computing*, 5(1):70, 2006.

[3] R. Ballagas, M. Ringel, M. Stone, and J. Borchers. istuff: a physical user interface toolkit for ubiquitous computing environments. In *CHI '03: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 537–544, New York, NY, USA, 2003. ACM.

[4] P. Baudisch, M. Sinclair, and A. Wilson. Soap: a pointing device that works in mid-air. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 43–46, New York, NY, USA, 2006. ACM.

[5] N. Bila, T. Ronda, I. Mohomed, K. N. Truong, and E. de Lara. Pagetailor: reusable end-user customization for the mobile web. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 16–29, New York, NY, USA, 2007. ACM.

[6] M. de Sá, L. Carriço, L. Duarte, and T. Reis. A mixed-fidelity prototyping tool for mobile devices. In *AVI '08: Proceedings of the working conference on Advanced visual interfaces*, pages 225–232, New York, NY, USA, 2008. ACM.

[7] A. Demeure, J. Meskens, K. Luyten, and K. Coninx. Design by example of graphical user interfaces adapting to available screen size. pages 1–6, 2008. In Proc. CADUI 2008, Albacete, Spain, June 11-13, 2008.

[8] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith. Stitching: pen gestures that span multiple displays. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 23–31, New York, NY, USA, 2004. ACM.

[9] D. Ingalls. The lively kernel: just for fun, let's take javascript seriously. In *DLS '08: Proceedings of the 2008 symposium on Dynamic languages*, pages 1–1, New York, NY, USA, 2008. ACM.

[10] B. Johanson, G. Hutchins, T. Winograd, and M. Stone. Pointright: experience with flexible input redirection in interactive workspaces. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 227–234, New York, NY, USA, 2002. ACM.

[11] S. R. Klemmer, B. Hartmann, and L. Takayama. How bodies matter: five themes for interaction design. In *DIS '06: Proceedings of the 6th conference on Designing Interactive systems*, pages 140–149, New York, NY, USA, 2006. ACM.

[12] J. H. Maloney and R. B. Smith. Directness and liveness in the morphic user interface construction environment. In *UIST '95: Proceedings of the 8th annual ACM symposium on User interface and software technology*, pages 21–28, New York, NY, USA, 1995. ACM.

[13] M. A. Nacenta, S. Sakurai, T. Yamaguchi, Y. Miki, Y. Itoh, Y. Kitamura, S. Subramanian, and C. Gutwin. E-conic: a perspective-aware interface for multi-display environments. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 279–288, New York, NY, USA, 2007. ACM.

[14] D. A. Norman. *The design of everyday things*. Doubleday, New York, 1990.

[15] Y. Ohshima, J. Maloney, and A. Ogden. The parks pda: a handheld device for theme park guests in squeak. In *OOPSLA '03: Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 370–380, New York, NY, USA, 2003. ACM.

[16] J. S. Pierce and J. Nichols. An infrastructure for extending applications' user experiences across multiple personal devices. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 101–110, New York, NY, USA, 2008. ACM.

[17] J. Rekimoto. Pick-and-drop: a direct manipulation technique for multiple computer environments. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 31–39, New York, NY, USA, 1997. ACM.

[18] J. Rekimoto and M. Saitoh. Augmented surfaces: a spatially continuous work space for hybrid computing environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 378–385, New York, NY, USA, 1999. ACM.

[19] P. Saint-Andre. Extensible messaging and presence protocol (xmpp): Core. Internet RFC 3920, October 2004.

[20] W. Stuerzlinger, O. Chapuis, D. Phillips, and N. Roussel. User interface façades: towards fully adaptable user interfaces. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 309–318, New York, NY, USA, 2006. ACM.