Damask: A Tool for Early-Stage Design and Prototyping of Cross-Device User Interfaces

James Lin
Group for User Interface Research, EECS Department
UC Berkeley
Berkeley, CA 94720-1776
jimlin@cs.berkeley.edu

ABSTRACT

We are creating a system called Damask to better support UIs that run across several types of devices. With Damask, the designer will design a UI for one device, by sketching the design and by specifying which *design patterns* the interface uses. The patterns will help Damask generate user interfaces optimized for the other target devices. The generated interfaces will be of sufficient quality so that it will be more convenient to use Damask than to design each of the other interfaces separately, and the ease with which designers will be able to create designs will encourage them to engage in iterative design.

KEYWORDS

cross-device user interfaces, multi-channel user interfaces, mobile computing

INTRODUCTION

People often use a variety of computing devices, such as PCs, PDAs, and cell phones, to access the same information. The user interface to this information needs to be different for each device, due to different input and output constraints. Currently, designers designing such cross-device UIs either have to design a UI separately for each device, which is time consuming, or use a program to automatically generate interfaces, which often leads to awkward interaction.

We believe that there is a way that will allow designers to design and prototype cross-device UIs that are appropriate for each device, yet take much less time than designing each design-specific UI separately. Specifically, a tool that uses design patterns to bridge the gap between device-specific UIs will enable designers to create cross-device UIs with the same quality as if the designer designed each device-specific UI separately, but in much less time.

To test this hypothesis, we are creating a system called Damask [9] that aims to combine the advantages of designing multiple interfaces from scratch with the speed of automatically generating interfaces. Designers using it will be able to create user interfaces highly optimized for several devices, much faster than if they created each of them from scratch.

With Damask, the designer will design a UI for one device, by sketching the design and by specifying which design patterns the interface uses. As the designer creates an interface, Damask will use the sketches and patterns to construct an *abstract model*, which captures aspects of the UI design at a high level of abstraction. When the designer is ready to create interfaces for the other devices, Damask will use the abstract model to generate the other device-specific interfaces, which the designer could refine if he or she wanted. The generated interfaces will be good enough so that it would be more convenient to use the tool than to design each of the other interfaces separately.

Damask will be aimed at designers who want to design and prototype a UI targeted at three types of interfaces: the web via a desktop PC, cell phone displays, and prompt-and-response style voice interfaces.

Through our work in Damask, we expect to better understand how designers currently design cross-device UIs, develop algorithms for retargeting a UI sketch from one device to other devices, implement those algorithms within Damask, and then evaluate Damask showing that designers using it can create cross-device UI designs that are at least as good as designing each device-specific interface separately, in less time.

RELATED WORK

Our work is closely related to the concept of *model-based user interfaces*, designing user interfaces based on an abstract model of the interface rather than visual appearance (see [16] for a summary and retrospective). This allows for rendering of the user interface in multiple ways, such as using a drop-down list or presenting a voice menu instead of radio buttons.

However, model-based UI tools often force designers to think at a high level of abstraction too early in the design process. Designers are accustomed to thinking about concrete interfaces at the beginning. In addition, specifying models often looks like programming, at which most designers are not skilled, so specifying models impedes their main task of designing user interfaces.

There has been much work on automatically transforming interfaces meant for one device or modality to another. Much of it has focused on transforming existing, finished desktop web interfaces to handheld interfaces at run-time [4, 7, 11], which unfortunately often results in awkward interaction. Others have worked on converting GUIs to audio interfaces [13, 15], mostly to benefit the blind and visually impaired. With most of these tools, designers cannot modify the results of the transformation process. Since our tool is not meant for the final implementation of UIs, designers are free to modify the generated UI design.

There are several model-based projects [2, 5, 6] that specifically address the issue of creating user interfaces targeted at multiple devices. These projects directly expose models to the UI designer, while our tool will avoid doing this.

PIMA [3] and Microsoft's ASP.NET mobile controls [12] allow designers to design cross-device web applications. A designer using either of them describes the application's user interface in an abstract representation, by laying out abstract widgets linearly in a constrained Visual Basic-like form designer. The representation is then converted into concrete device-specific UIs. However, these tools are not appropriate for early-stage design, because designers tend to think about concrete user interfaces, not abstract representations.

There are several projects that specify platforms for creating universal remote controls (*e.g.*, [14, 18]). These platforms use high-level descriptions of a remote control's user interface which can then be realized on a variety of hardware devices, such as PDAs or Braille readers. The target domain of universal remote controls is narrower than Damask's, but the UIs that are rendered from the abstract description must be appealing and useful immediately, without additional tweaking. Our work, on the other hand, is targeting a broader set of user interfaces, but the generated interfaces will most likely be modified by the user interface designers before being released.

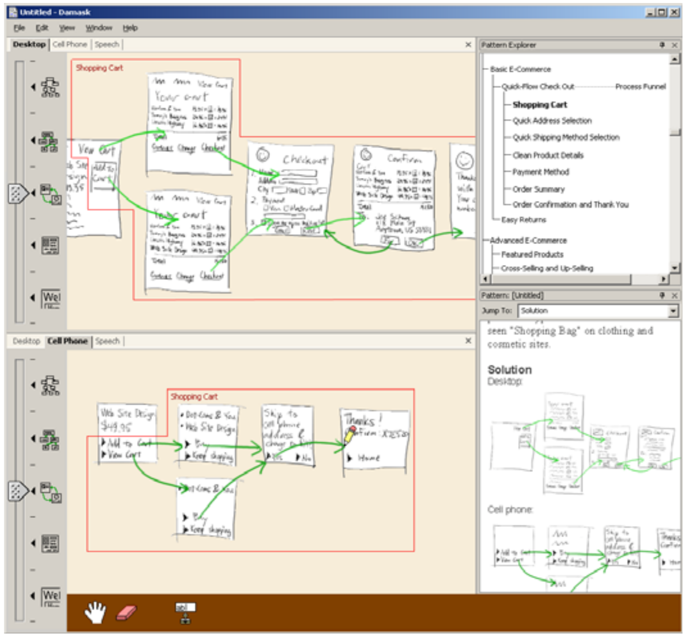


Figure 1. Damask's proposed user interface.

OVERVIEW OF DAMASK'S APPROACH

At a high level, Damask will include a catalog of design patterns from the book *The Design of Sites* [17] that designers can use in their designs. Each design pattern will have specific *examples* of how the pattern has been used in other projects, and several generalized *solutions* capturing the essence of the examples. Each design pattern will have a separate solution for each device.

Designers will create their UI designs by sketching and by adding instances of patterns to their design for one device. Damask will take that design and generate UI design sketches for the other two devices, which the designers can go back and modify if desired. Finally, designers can use Damask (or SUEDE [8] for voice interfaces) to run their designs in a device simulator, so that they can interact with their design sketches.

DAMASK'S PROPOSED USER INTERFACE

Damask's proposed user interface consists of several regions (Figure 1). The canvas is where the designer will sketch the actual user interface design. The design will include which patterns it is using, as denoted by a red outline and the name of the pattern. There will be tabs above the canvas where designers will choose which target device they are viewing. To view several device-specific UIs at the same time, the designer will be able to split the canvas or view the design in multiple windows.

Damask will also have a *Pattern Explorer* sidebar, where designers can browse for patterns to be instantiated in their designs, and a *Pattern* sidebar where designers can find details about a particular pattern, instantiate a pattern, and create their own patterns. Each pattern will have eight parts, which correspond to the structure of patterns found in several pattern languages such as [1] and [17]:

- name
- sensitizing image
- background
- problem
- forces
- examples
- solution
- references

Two of the sections warrant more elaboration. The Examples section will contain real examples of the pattern in use. It will also be constantly updated: whenever a pattern is instantiated, that instance will be added to the Examples section and will be continuously updated whenever the designer modified the instance.

The Solution section will contain generalized solutions for the pattern. Like the canvas, the Solution section will also be divided into three sections, with one solution for each device supported by Damask.

CREATING CROSS-DEVICE INTERFACES

Here is how we envision a designer using Damask to design a UI, for example, an e-commerce web site for the PC and cell phone. The designer decides to first target the PC, so he sketches out some web pages for the PC version of the web site. Here is one such page:

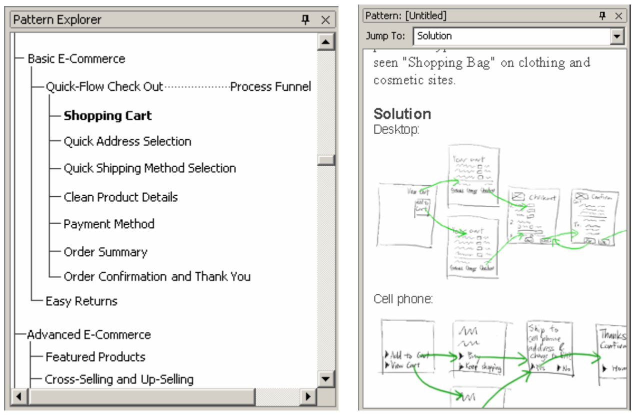
Web Site
Design

Instead of sketching out all of the pages from scratch, the designer takes advantage of the patterns built into Damask. He brings up the Pattern Explorer to browse through the patterns, and comes across the SHOPPING CART pattern (Figure 2). He sees that there are two generalized solutions for the SHOPPING CART, one for a PC and one for a cell phone (see Figure 2, right).

The structure of the designer's UI sketches and the pattern's solutions follow a visual language similar to DENIM [10] and SUEDE [8]. A *page* represents a web page, cell phone screen, or voice prompt. A designer can sketch or type in a page. An *arrow* between two pages represents an action that the end-user performs to go from one page to the other. In a web page, the source of the arrow represents the hyperlink the end-user clicks on to go to the target page. In a cell phone screen, it represents a menu item that the end-user selects. In a voice interface, the arrow is annotated with the response that the end-user says to go to the target voice prompt.

The designer picks the PC version of the SHOPPING CART solution and drags the leftmost page of the solution onto the canvas, bringing the rest of the solution along. Then he drops it on top of the page that he first sketched. This merges the contents of that pattern page with his sketched page and adds the rest of the pattern to his design. SHOPPING CART has now been *instantiated* in his design (Figure 3).

The pattern that the designer has just instantiated is generic, for example, having mostly text placeholders instead of actual text. The designer now customizes the pattern instance to fit his own project. He replaces the text placeholders with actual text, moves widgets around, and adds his own images. He could even add pages and change the arrows if he decides that is appropriate. As the designer customizes the pattern instance, Damask keeps track of his customizations. The pattern is now fully



seen "Shopping Bag" on clothing and cosmetic sites.

Figure 2. *Left:* The Pattern Explorer with the SHOPPING CART pattern highlighted. *Right:* The Pattern sidebar containing the SHOPPING CART pattern.

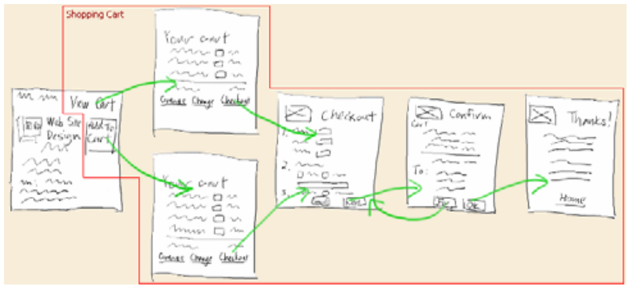


Figure 3. The PC version of the e-commerce web site, with shopping cart merged into it.

integrated into his design (Figure 4).

At some point, the designer decides he is ready to work on the cell phone version of the web site. So he clicks on the Cell Phone tab just above the canvas. Damask first makes the cell phone-specific design by copying the PC-specific design. Then it goes through the design and finds which parts of the design are pattern instances and which are not.

Damask modifies the parts of the design that are not pattern instances by applying traditional model-based UI techniques. For example, it will rearrange widgets to compensate for the smaller screen, and replace sets of radio buttons with drop-down boxes. Both perform the same task, but drop-down boxes take up less space.

Damask replaces the pattern instances, which have been PC specific up to now, with the corresponding cell-phone pattern solutions. It then takes the customizations that the designer applied to the PC-specific versions and applies them to the cell-phone versions. This results in a pattern instance specific to the cell phone but customized to the application that is being designed (Figure 5).

Not all of the customizations will necessarily be applied. For example, if the designer moves a widget in the PC version, Damask will not apply that customization to the cell-phone version, since the displays of cell phones are so limited that the designer would most likely have to move the widget again. One of the biggest research challenges is deciding which customizations to take from one device-specific instance and apply it to the others.

The instances of SHOPPING CART within this project are automatically added to the Examples section of the SHOPPING CART pattern within Damask's pattern library.

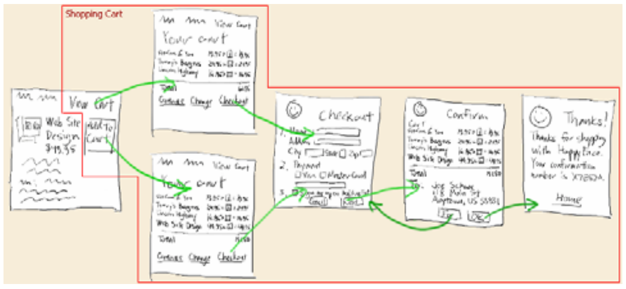


Figure 4. The PC version of the e-commerce web site, with shopping cart customized.

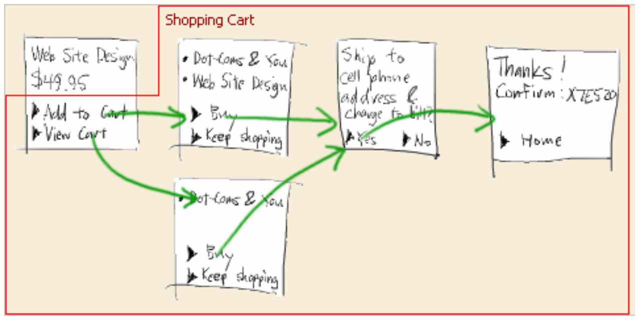


Figure 5. The cell phone version of the e-commerce web site generated by Damask.

This encourages reuse of designs and could decrease the time and effort spent on future projects.

CREATING CUSTOM PATTERNS

Creating new design patterns in Damask consists of several steps:

- Choosing the fragments of a design from which to create a pattern.
- Generalizing those fragments to create generic pattern solutions.
- Showing how the device-specific solutions of the pattern relate to each other.

For example, suppose Damask did not have a SHOPPING CART pattern, and the designer wanted to create one from his design. To create SHOPPING CART, the designer first opens up the Pattern Explorer sidebar, opens a context menu, and chooses New Pattern. An empty Pattern sidebar is created. The designer selects the part of the design he wants to become part of the SHOPPING CART pattern and drags it into the Pattern sidebar. Damask puts the fragment into the pattern's Solution and Examples sections and marks the design with the new pattern.

The specific shopping cart that the designer dragged into the Pattern sidebar has actual text and other details that are not appropriate for a general solution of SHOPPING CART. To make the solution more general, the designer edits the solution, replacing actual text with placeholders, and so on.

Finally, the designer takes the device-specific pattern solutions and shows how they relate to each other. This is so that when Damask generates a UI for another device, it knows how to take the customizations the designer applied to the first device-specific pattern instance and apply them to the second device-specific pattern instance. The mechanism for specifying this is subject to further research.

SUMMARY

We believe that combining the concepts of model-based UIs and design patterns is a promising approach for creating cross-device UIs. Our goal is to demonstrate this by building Damask, which will take advantage of the design patterns used in one UI sketch to create interactive UI sketches optimized for other target devices, allowing designers to create high quality cross-device UIs in less time than current practices.

REFERENCES

1. Alexander, C., S. Ishikawa, M. Silverstein, M. Jacobson, I. Fiksdahl-King, and S. Angel, *A Pattern Language*. New York: Oxford University Press, 1977.
2. Ali, M.F. and M.A. Pérez-Quiñones. Using Task Models to Generate Multi-Platform User Interfaces while Ensuring Usability. In *CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 670-671, April 20-25, 2002.
3. Bergman, L.D., G. Banavar, D. Soroker, and J. Sussman. Combining Handcrafting and Automatic Generation of User-Interfaces for Pervasive Devices. In Proc. *CADUI'2002*. Valenciennes, France: May 15-17, 2002.
4. Buyukkokten, O., H. Garcia-Molina, A. Paepcke, and T. Winograd, Power Browser: Efficient Web Browsing for PDAs. *CHI Letters: Proc. CHI 2000*, 2000. **2**(1): pp. 430-437.
5. Calvary, G., J. Coutaz, and D. Thevenin. A Unifying Reference Framework for the Development of Plastic User Interfaces. In Proc. *EHCI 2001*. Toronto, Canada: Springer-Verlag. pp. 173-192, May 11-13, 2001.
6. Eisenstein, J., J. Vanderdonckt, and A. Puerta. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. In Proc. *IUI 2001*. Santa Fe, NM: ACM Press. pp. 69-76, January 14-17, 2001.
7. Fox, A., I. Goldberg, S.D. Gribble, D.C. Lee, A. Polito, and E.A. Brewer. Experience With Top Gun Wingman: A Proxy-Based Graphical Web Browser for the 3Com PalmPilot. In Proc. *Middleware '98*. Lake District, UK, Sept. 15-18, 1998.
8. Klemmer, S.R., A.K. Sinha, J. Chen, J.A. Landay, N. Aboobaker, and A. Wang, SUEDE: A Wizard of Oz Prototyping Tool for Speech User Interfaces. *CHI Letters: Proc. of UIST 2000*, 2000. **2**(2): pp. 1-10.
9. Lin, J. and J.A. Landay. Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. In Proceedings of *The 8th Intl. Conf. of Distributed Multimedia Systems (2002 Intl. Workshop on Visual Computing)*. San Francisco, CA. pp. 573-580, Sept. 26-28, 2002.
10. Lin, J., M. Thomsen, and J.A. Landay, A Visual Language for Sketching Large and Complex Interactive Designs. *CHI Letters: Proc. CHI 2002*, 2002. **4**(1): pp. 307-314.
11. Lopez, J.F. and P. Szekely, Web Page Adaptation for Universal Access, in *Universal Access in HCI: Towards and Information Society for All (Proc. of 1st Intl. Conf. on Universal Access in HCI, New Orleans, LA, August 8-10, 2001)*, C. Stephanidis, Editor. Lawrence Erlbaum Associates: Mahwah, NJ. p. 690-694, 2001.
12. Microsoft, *ASP.NET Mobile Controls*. Microsoft Corporation: Redmond, WA. http://msdn.microsoft.com/vstudio/device/mobilecontrols/
13. Mynatt, E.D. and W.K. Edwards. An Architecture for Transforming Graphical Interfaces. In Proc. *UIST '94*. Marina del Rey, California. pp. 39-47, November 2-4, 1994.
14. Nichols, J. Informing Automatic Generation of Remote Control Interfaces with Human Designs. In *CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 864-865, April 20-25, 2002.
15. Olsen, D.R., S.E. Hudson, R.C.-M. Tam, G. Conaty, M. Phelps, and J.M. Heiner. Speech Interaction with Graphical User Interfaces. In Proc. *INTERACT2001*. Tokyo, Japan: IOS Press, 2001.
16. Szekely, P. Retrospective and Challenges for Model-Based Interface Development. In Proc. *DSV-IS'96*. Namur, Belgium. pp. 1-27, June 5-7, 1996.
17. van Duyne, D.K., J.A. Landay, and J.I. Hong, *The Design of Sites*. Boston: Addison-Wesley, 2002.
18. Zimmermann, G., G. Vanderheiden, and A. Gilman. Prototype Implementations for a Universal Remote Console Specification. In *CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 510-511, April 20-25, 2002.