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

James Lin and James A. Landay
Group for User Interface Research, EECS Department
UC Berkeley
Berkeley, CA 94720-1776
{jimlin, landay}@cs.berkeley.edu

Abstract 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 multi-device user interfaces either have to design a UI separately for each device, which is time consuming, or use a program to automatically generate interfaces, which often results in interfaces that are awkward. We are creating a system called Damask to better support multi-device UI design. 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.

1 Introduction

The experience of using a computer is increasingly diverse. Interaction with a PC in a home or office is now augmented with a variety of devices, such as handheld personal digital assistants (PDAs), cell phones, pagers, and even telematics systems in cars. Companies as varied as Amazon, TV Guide, and Microsoft are starting to allow their customers to access their services through such a variety of devices. For example, you can find out which theaters are playing a particular movie and at what time through a voice-based phone interface, a PDA web site, or a desktop web site. However, due to the attributes and limitations of each device, the interfaces across devices are often drastically different. This makes the task of designing a user interface (UI) for a service that targets several devices difficult, because you essentially need a distinct UI for each device.

If UI designers want to target several devices for an application, they generally face two alternatives. One option is to design a user interface for each targeted device. This process results in interfaces that are optimized for each device, but it has several drawbacks. Designing several user interfaces is very time consuming, and the more devices the designer targets, the more time and effort the designer must spend. It is also hard for designers to keep the designs coordinated across devices. A designer could add a feature to one device-specific UI, and then easily forget to at least investigate the possibility of adding that feature to another device-specific UI. Also, a different person may design each device-specific UI, exacerbating this problem.

The other option is to design an interface for only one device and let special-purpose programs automatically generate the interfaces for other devices. This cuts down development time but leads to interfaces that are awkward to use. Consequently, they are only used as a last resort by end-users who have no other way to access the information or perform the task provided by that UI.

The difficulty of designing for multiple devices discourages designers from iteratively refining and prototyping their designs. One of the best ways to create a good user interface is to continually design, test, and analyze a user interface idea [15]. If creating a design in the first place is difficult, designers will not want to try multiple designs or drastically change their initial design, which may impact the quality of the final design. Tools that make early-stage design, prototyping, and testing multi-device user interfaces easier could dramatically improve the usability and usefulness of those interfaces.

We believe that there is a way that will allow designers to design and prototype multi-device UIs that are appropriate for each device, yet take much less time than designing each design-specific UI separately, by using *design patterns* [1, 31]. We believe that there are patterns in user interfaces for multiple devices, and that the structure of these pattern solutions can be dramatically different, depending on the devices' characteristics. A shopping cart checkout pattern solution for a desktop web site could consist of several pages, asking for the buyer's name, address, credit card number, and so on. Entering all of this information would be extremely tedious on a cell phone. Instead, a pattern solution for the cell phone could be a single screen that asks, "Ship to cell phone address and charge to cell phone bill? Yes/No".

To demonstrate this, we are designing such a tool, called *Damask*, which will support the early-stage design and prototyping of multi-device interfaces. Damask aims to combine the advantages of designing multiple interfaces from scratch with the speed of automatically generating

interfaces. Designers using it could create user interfaces highly optimized for several devices, much faster than if they created each of them from scratch.

With Damask, the designer would design a user interface for one device, by sketching the design and by specifying which design patterns the interface uses. As the designer creates an interface, Damask uses the sketches and patterns to construct an *abstract model* [9], 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 uses 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 also provide a Run mode in which designers interact with their design sketches in a browser that will roughly simulate the devices they are targeting. This will allow designers to get quick feedback about their design from other team members or even their target users, which will inform any modifications they want to make to their design.

In the rest of this article, we first discuss two of the main concepts that Damask embodies, model-based user interfaces and design patterns, and related work in more detail. We then describe our preliminary ideas for Damask, including how a designer would use it to design multi-device user interfaces.

2 Related Work

The next section contrasts Damask's approach to other related work, including model-based UI tools, tool support for patterns, combinations of model-based and pattern-based approaches, and tools to transform UIs from one device or modality to another.

2.1 Model-Based UI Tools

Damask's underlying representation of UI designs and patterns would be based on the concept of *model-based user interfaces*, designing user interfaces based not just on visual appearance but also on an abstract model of the interface [9]. The model describes the interface at a higher level of abstraction than the actual widgets. For example, instead of describing a dialog box as having three radio buttons and two check boxes, an abstract model would describe it has having one part where the user can select one of three items, and two other on-off selections. This level of abstraction allows the possibility of rendering the user interface in other ways, such as using a drop-down list or presenting a voice menu instead of radio buttons. Using patterns for describing interfaces would further increase the level of abstraction and allow even more radical differences in interfaces across devices.

While model-based user interfaces have the promise of creating flexible interfaces that can adapt to their environment, they have not been widely adopted in the commercial software development world, which has instead gravitated towards visual interface builders. We believe one reason for the lack of acceptance is the fact that many model-based UI tools do not match or augment the work practices of designers. They often force designers to think at a high level of abstraction too early in the design process, by making them design in terms of abstract widgets (*e.g.,* [27, 29, 34]), or by specifying a task model which is then transformed into a concrete UI (*e.g.,* [9, 25]). Designers are accustomed to thinking about concrete interfaces at the beginning. In addition, specifying models often requires the designer to deal with preconditions, postconditions, and conditionals. This starts to look like programming, which most designers are not skilled at, so specifying models impedes their main task of designing UIs.

We believe that Damask's approach could allow UI designers to specify their designs at a more abstract level, but with a vocabulary that designers understand, via sketches and design patterns. We also believe that design patterns could give Damask information that would allow it to generate interfaces that are more appropriate for the targeted devices.

The philosophy of most model-based UI research is that the model-based tools would be the primary way to create the finished user interface, although many tools expect the user interface to be modified somewhat by the designer. In contrast, Damask is targeted towards prototyping. We do not expect the designer to use Damask to create the final user interface, nor do we expect Damask's generated user interfaces to be used without modification. Since we are targeting prototyping, the generated user interface does not need to be ideal, since in the early stages of design, the designer is concerned more with the user's interaction flow rather than the details of the interface [33].

2.2 Tool Support for Patterns

Patterns were first introduced by Christopher Alexander and his colleagues in the field of architecture. He states, "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice." [1] This basic definition has become popular in the software engineering (e.g., [11]) and human-computer interaction (HCI) fields [4, 30-32].

In computer science, patterns have made the most impact in software engineering, especially object-oriented programming [3, 11]. Consequently, software tools that support patterns have mostly targeted object-oriented software development (e.g., [5, 19, 26]).

While these tools only address software engineering issues, there are some aspects of these tools which address issues that any pattern-based tool needs to support, such as browsing and searching for patterns, and customizing patterns for a particular application. However, customizing patterns with these tools usually involves a form-based interface, which would fit awkwardly with Damask's

sketch-based interface. Also, if a developer wants to use a pattern, some of these tools force the developer to change his solution to make it fit the pattern. Damask will not do this. One of the most important aspects of patterns is their flexibility: using the Alexandrian definition of patterns, a developer should be able to use a pattern many times but never the same way twice. Designers using Damask will be free to greatly modify how a pattern is used in their particular design, on which we will elaborate later.

There has been much discussion about using design patterns in human-computer interaction (HCI) [4, 30-32], but few HCI tools have been created that support patterns. Paternó [24] describes extending a task and architecture model editor to support patterns that are made up of model fragments themselves. Paternó focuses on abstract task and architecture patterns. A task pattern describes what steps a user performs to execute a particular task, such as searching, independent from a particular user interface. An architecture pattern describes how the program implements a task, such as how a program accesses a database to perform a search. On the other hand, Damask will focus on more concrete UI design patterns, since designers will be creating concrete UI designs using Damask.

2.3 Combining Models and Patterns

Other groups have proposed combining patterns and model-based approaches (*e.g.*, Paternó [24]). Hussey and Carrington [13] discuss designing user interfaces by starting out with an abstract UI specification, and then methodically applying transformation patterns to it to create a concrete UI specification. In contrast, designers using Damask interact with concrete UI specifications that contain UI design patterns.

2.4 User Interface Transformation Tools

There has been much work on automatically transforming interfaces meant for one device or modality to another. Many of these projects have focused on transforming existing, finished desktop web interfaces to PDA interfaces at run-time [6, 10, 18]. However, shrinking interfaces from large desktop displays to such small PDA displays often results in awkward interaction. Others have worked on converting GUIs to audio interfaces [21, 23], mostly to benefit the blind and visually impaired. With most of these tools, designers cannot modify the results of the interface transformation process. Since Damask is a prototyping tool, not a tool to create final UIs, designers will be free to modify the generated user interface design.

Ultraman [28] provides a way for designers to control the transformations, but it assumes they are comfortable with the concept of trees, grammars, and writing code in Java. Damask is targeting a different audience for a different part of the design cycle: designers who have little or no experience programming, and early-stage design, before any interface is completely specified and ready to run.

There are several model-based projects that are specifically addressing the issue of creating user interfaces targeted at multiple devices. Eisenstein, Vanderdonckt, and Puerta [8] describe using MIMIC [25] to create models which describe multi-device user interfaces. Their methodology involves mapping common tasks in a task model to presentation models optimized for the task. However, this work does not directly address the case of when the UIs for performing the same task on more than one device are very different. Damask will use patterns to address this issue.

Ali et al [2] discuss designing a multi-device UI using four types of models: a task model, an abstract logical model, physical family models, and platform-specific UI descriptions in UIML. In contrast, Damask will avoid directly exposing models to the UI designer.

There have been several projects that aim to create a platform for creating universal remote controls (*e.g.*, [22, 35]). These projects envision appliances that export high-level descriptions of a remote control user interface to a device, such as a PDA or a Braille reader, which then renders that description into a concrete UI. The UI would take the user's input to the remote control UI and send it back to the appliance for processing. There are two important distinctions between the problems these projects are solving and Damask's problem area. The target domain of universal remote controls is narrower (remote controls for appliances vs. web interaction), but the UIs that are rendered from the abstract remote control description must be appealing and useful immediately, without additional tweaking. Damask, on the other hand, is targeting a broader set of UIs (*e.g.*, general web-style interaction on PCs), but the interfaces that are generated will most likely be modified by the UI designers before being released.

Calvary, Coutaz, and Thevenin [7] discuss a process framework for developing *plastic interfaces*, which can adapt to different devices. In addition to the typical model-based approach, in which a designer creates a series of models from top-level abstract models to a concrete interface, the framework also covers translations between platforms, which may happen at any model abstraction level. This framework provides a useful way of thinking about how to develop multi-device UIs, although with Damask, top-level abstract models are not directly exposed, so the framework is not directly applicable.

In MUSA [20], multi-device services are described with an event graph, which abstractly describes the navigational structure of a service and how it interacts with the services' logic. MUSA dynamically generates UIs at run-time. This differs from Damask, which focuses on the UI *design* process, before the UI is ready for final deployment.

3 Overview of Damask's approach

We believe that a tool that uses design patterns to bridge the gap between device-specific UIs will enable designers to create multi-device UIs with the same quality as if the designer designed each device-specific UI separately, but in much less time.

To test this, we will create a tool called Damask aimed at designers who want to design and prototype a UI targeted at three types of interfaces: the web accessed through a desktop, cell phone displays, and prompt-and-response style voice interfaces. We have picked these three because they represent the "extremes" of the range of devices that are widely used. For example, simply shrinking a screen designed for a desktop PC will not result in a good cell phone interface.

Damask will take an interactive sketch for a user interface for one device and the design patterns used in that sketch, and will create interactive user interface sketches for the other devices. These generated designs will be of sufficient quality and usefulness such that the designer will spend less effort modifying the generated sketches than creating them from scratch, and that the resulting designs will be at least as good.

At a high level, Damask would include a catalog of design patterns that designers can use in their designs. Each design pattern would 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 would 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 [14] for voice interfaces) to run their designs in a device simulator, so that they can interact with their design sketches.

First, we will describe Damask's proposed user interface. Then we will walk through an example of how a designer would design and run his UI design, and finally how he would create his own design pattern.

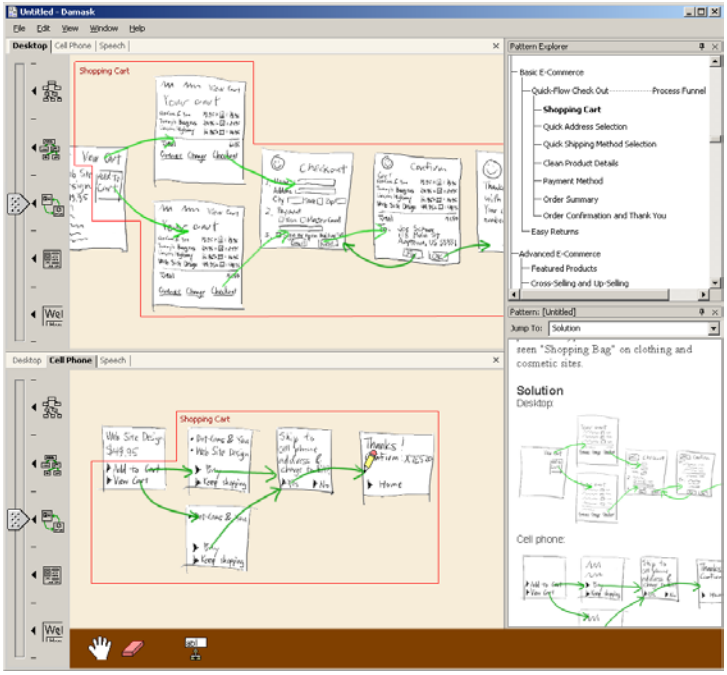

Figure 1. Damask's proposed user interface.

3.1 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 would choose which target device they are viewing. To view the different device-specific UIs at the same time, the designer will be able to split the canvas or view the design in multiple windows.

Damask would also have a *Pattern Explorer* sidebar, where designers could browse for patterns to be instantiated in their designs, and the *Pattern* sidebar where designers could find the details about a particular pattern, instantiate a pattern, and create their own patterns. Each pattern would have eight parts, which correspond to the structure of patterns found in several pattern languages such as [1] and [31]:

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

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

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

3.2 Creating Multi-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:

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).

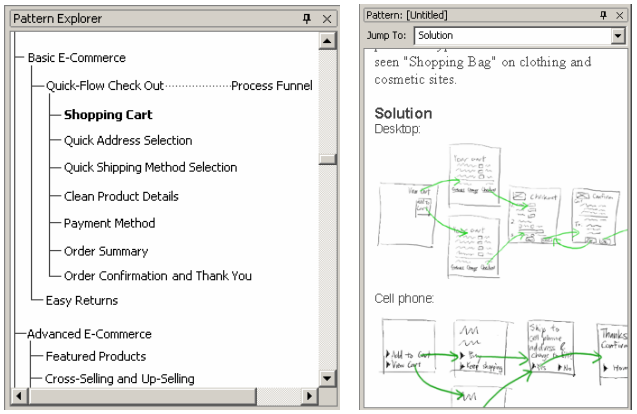

Figure 2. *Left:* The Pattern Explorer with the SHOPPING CART pattern highlighted. *Right:* The Pattern sidebar containing the SHOPPING CART pattern.

The structure of the designer's UI sketches and the pattern's solutions follow a visual language similar to DENIM [17] and SUEDE [14]. 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 pattern into the canvas, bringing the rest of the pattern along. Then he drops it on top of the page that he first sketched. This merges the contents of that the 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 very 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 integrated into his

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

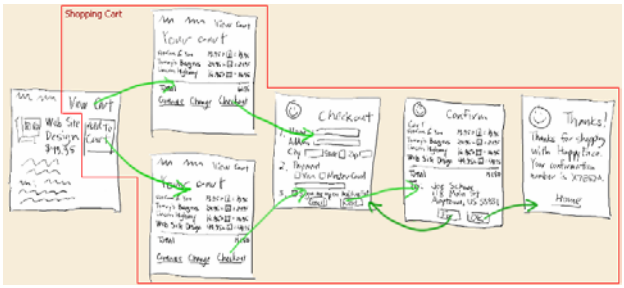

Figure 4. The PC version of the e-commerce web site, with SHOPPING CART customized.

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 abstract 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 anyway. 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. This encourages reuse of designs and could decrease the

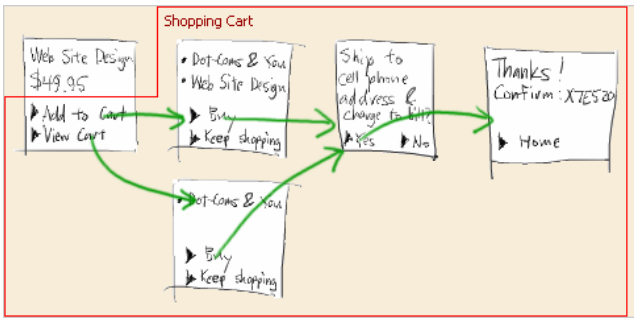

Figure 5. The cell phone version of the e-commerce web site generated by Damask.

time and effort spent on future projects.

3.3 Managing Consistency in Multi-device Interfaces

Damask will allow a designer to edit one device-specific UI without necessarily changing the other device-specific UIs. Figuring out which edits will propagate from one device-specific UI to the others is a key research question. The following is the approach we are proposing to take.

Changes within a page, such as adding and removing elements, will not be propagated across devices. However, adding and removing pages on one device will cause pages to be added or removed in the other devices. The idea is that the particular layout and detailed content within a page are not usually the same across devices, but adding and removing pages indicates significant structural changes that should be reflected in all device-specific UIs.

Often the designer will want the information in one page for the desktop to be in several pages on a cell phone or in many prompts and responses for a voice interface, even though the overall structure is the same. In these cases, the designer will be able execute a Split command on one page to split it into several pages, or execute a Merge command on several pages to merge them into a single page. Splitting and merging pages in one device-specific UI will not result in pages being added or removed in other device-specific UIs. When a designer mouses over or edits a particular page, the corresponding page or pages will be highlighted in the other device-specific UIs, so that the designer can keep track of the overall structure across devices.

In some parts of a UI design, the page structure will be very different across different devices. Patterns will take care of many of those cases, but when this occurs outside of a pattern, the designer will be able to mark off a region in the design, inside which no edits will be propagated to the other device-specific UIs. Thus, a designer can create or delete pages within the region, and no corresponding pages will be created or deleted in the other device-specific UIs.

3.4 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.

We will illustrate this with an 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. (See Figure 6.)

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. If the designer does not specify these relationships in the pattern, then when Damask uses the pattern as the basis for automatically generating another interface for a second device, the solution specific to the second device will be used without applying any customizations to it.

One proposal for showing these relationships is to draw lines between the related parts. In this case, the designer views both the PC and cell phone SHOPPING CART solutions and draws lines between them to show how they are related. For example, he draws a line from the shopping list on the first page of the PC solution, to the shopping list on the first page of the cell phone solution. This way, when the designer fills in the list in the shopping cart in a PC design, and then asks Damask to generate a cell phone design, Damask knows to take the contents of the shopping

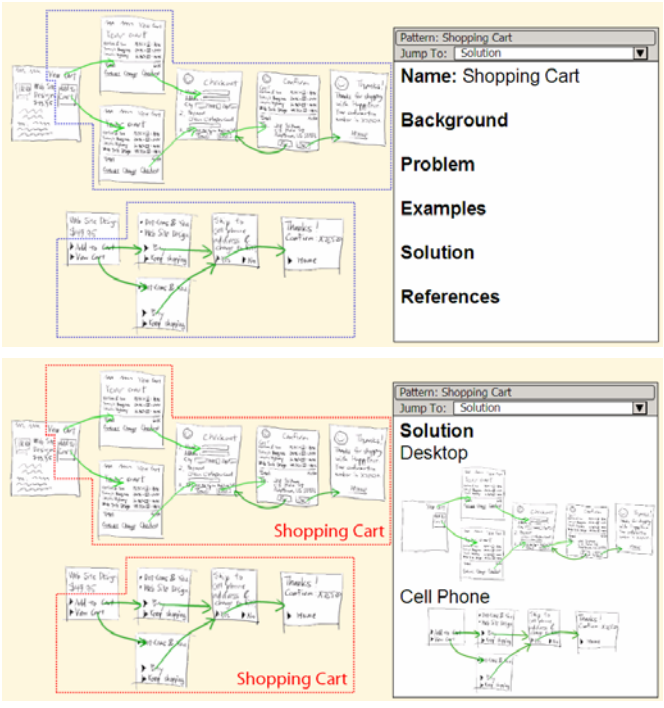

Figure 6. *Top:* Highlighting the portion of a design from which to create a pattern. *Bottom:* The results of dragging the highlighted sections to the pattern sidebar.

list in the PC version, and put them into the corresponding list in the cell phone version.

There are many research questions to be answered here, such as what happens if the designer relates two parts that are not exactly the same (such as a list with three elements with a list with one element), whether such a simple mechanism is sufficiently powerful in enough cases, and how to design such a mechanism that does not overwhelmingly clutter the UI sketches.

4 Next Steps

The following is a discussion of the methodology we will use to conduct our research into designing and prototyping multi-device user interfaces.

4.1 Survey of Existing Multi-device UI Design Practices and Design Patterns

We would like to get a more complete picture for how designers currently design multi-device UIs. Therefore, we will talk to employees at about six companies about this topic. We will ask them, for a given application targeted at multiple devices, whether the user interfaces were all designed at the same time or at different times, whether it was the same team or different groups of people who designed them, how much communication there was among the designers, and when the designers tested the user interfaces. We will also present our ideas about Damask and ask them for their reactions.

So far, we have talked to four designers and one developer at three companies: a web portal company, an enterprise software company, and a PC software company. At the web portal and PC software companies, we talked to mobile UI designers. We found the desktop versions of a UI were created before the mobile project was started. No team designed both the mobile and desktop versions of a user interface, and the mobile designers typically did not talk to the desktop UI designers about the UI. Instead, the mobile designers looked at the desktop UIs themselves to get some ideas about what tasks they should support and what the general flow of the UI should be, although they did not rely on them. This tells us that we need to be aware of the potential of several people using Damask to design one UI, possibly at the same time.

At the enterprise software company, we talked with one developer. He told us that his manager designed the user interface for both the PDA and desktop versions of his product, but afterwards, different teams worked on each device-specific application. The developer mentioned that because the domain of the application was so narrow, the user interface design task was constrained. The user interfaces typically consisted of tables of data processed from a database, and interacting with the UI was mainly navigating among those tables and filling forms.

When we presented our ideas about using patterns to design multi-device UIs to the interviewees, they were all enthusiastic about the approach, although they did not have many specific suggestions or recommendations. This encouraged us to continue with our pattern-based approach.

We will also look for web sites that have been implemented for both the desktop and for mobile devices, like the PDA or cell phone, and examine them for common design patterns so that they can be incorporated into Damask. We have already identified several web sites to examine, such as Amazon, Expedia, Google, and MSN.

4.2 Prototyping and Building Damask

After incorporating our findings from the previous section into our UI design of Damask, we will build a low-fidelity prototype of Damask and test it with UI designers in industry. Their feedback will inform the next prototype, which will be a high-fidelity prototype written in Java. We will use SATIN [12], a toolkit for creating sketch-based applications in Java, and the code base from DENIM [16], a sketch-based tool for web design.

While designing and building Damask, we will address the research issues discussed above. They include:

- When taking customizations made to one pattern instance and applying them to another, which customizations should be applied?
- How do we maintain consistency between device-specific UIs? How do we handle inconsistencies?
- How do we show the designer what parts of one device-specific UI correspond to parts of the other device-specific UIs?
- How do we support multiple designers accessing the same design?

4.3 Evaluation

To evaluate Damask, we will recruit UI designers from industry who have worked on multi-device projects and ask them to design a multi-device UI. The form of the experiment will depend on the results of our survey of existing multi-device UI design practices. For example, we may ask designers to design a UI for two devices or take an existing UI and retarget it for another device. Among some of the factors we may evaluate include how satisfied the designers were using Damask, how "good" other UI designers judge the designs, and how often and effectively patterns were used.

5 Summary

We believe that combining the concepts of model-based UIs and design patterns is a promising approach for creating multi-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.

6 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. Proc. *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 670-671, April 20-25, 2002.

[3] Beck, K. and W. Cunningham, *Using Pattern Languages for Object-Oriented Programs*. Technical Report CR-87-43, Tektronix, Inc. 1987.

[4] Borchers, J., *A Pattern Approach to Interaction Design*. Chicester, England: John Wiley & Sons. 268 pp., 2001.

[5] Budinsky, F.J., M.A. Finnie, J.M. Vlissides, and P.S. Yu, Automatic Code Generation from Design Patterns. *IBM Systems Journal*, 1996. **35**(2): pp. 151-171.

[6] Buyukkokten, O., H. Garcia-Molina, A. Paepcke, and T. Winograd, Power Browser: Efficient Web Browsing for PDAs. *CHI Letters: Proceedings of Human Factors in Computing Systems: CHI 2000*, 2000. **2**(1): pp. 430-437.

[7] Calvary, G., J. Coutaz, and D. Thevenin. A Unifying Reference Framework for the Development of Plastic User Interfaces. Proc. *Engineering for Human-Computer Interaction: EHCI 2001*. Toronto, ON, Canada: Springer-Verlag. pp. 173-192, May 11-13, 2001.

[8] Eisenstein, J., J. Vanderdonckt, and A. Puerta. Applying Model-Based Techniques to the Development of UIs for Mobile Computers. Proc. *International Conference on Intelligent User Interfaces: IUI 2001*. Santa Fe, NM: ACM Press. pp. 69-76, January 14-17, 2001.

[9] Foley, J.D. and P.N. Sukaviriya. History, Results and Bibliography of the User Interface Design Environment (UIDE), an Early Model-Based System for User Interface Design and Implementation. Proc. *Design, Specification and Verification of Interactive Systems: DSV-IS'94*. Carrara, Italy. pp. 3-14, June 8-10, 1994.

[10] 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. Proc. *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing: Middleware '98*. Lake District, UK, September 15-18, 1998.

[11] Gamma, E., R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series. Reading, MA: Addison-Wesley. 395 pp., 1995.

[12] Hong, J.I. and J.A. Landay, SATIN: A Toolkit for Informal Ink-based Applications. *CHI Letters: Proceedings of User Interfaces and Software Technology: UIST 2000*, 2000. **2**(2): pp. 63-72.

[13] Hussey, A. and D. Carrington, *Using Patterns in Model-based Design*. Technical Report 99-15, Software Verification Research Centre, School of Information Technology, University of Queensland, Queensland, Australia, March 1999.

[14] 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: Proceedings of User Interfaces and Software Technology: UIST 2000*, 2000. **2**(2): pp. 1-10.

[15] Lewis, C. and J. Rieman, *Task-Centered User Interface Design: A Practical Introduction*. Boulder, CO: University of Colorado, 1993. ftp://ftp.cs.colorado.edu/pub/cs/distribs/clewis/HCI-Design-Book/

[16] Lin, J., M.W. Newman, J.I. Hong, and J.A. Landay, DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design. *CHI Letters: Proceedings of Human Factors in Computing Systems: CHI 2000*, 2000. **2**(1): pp. 510-517.

[17] Lin, J., M. Thomsen, and J.A. Landay, A Visual Language for Sketching Large and Complex Interactive Designs. *CHI Letters: Proceedings of Human Factors in Computing Systems: CHI 2002*, 2002. **4**(1): pp. 307-314.

[18] Lopez, J.F. and P. Szekely, Web Page Adaptation for Universal Access, in *Universal Access in HCI: Towards and Information Society for All (Proceedings of 1st International Conference on Universal Access in Human-Computer Interaction, New Orleans, LA, August 8-10, 2001)*, C. Stephanidis, Editor. Lawrence Erlbaum Associates: Mahwah, NJ. p. 690-694, 2001.

[19] Meijler, T.D., S. Demeyer, and R. Engel. Making Design Patterns Explicit in FACE, a Framework Adaptive Composition Environment. Proc. *European Software Engineering Conference and ACM SIGSOFT Symposium on the Foundations of Software Engineering: ESEC/FSE '97*: Springer-Verlag LNCS 1301. pp. 94-110, 1997.

[20] Menkhaus, G. and W. Pree. User Interface Tailoring for Multi-Platform Service Access. Proc. *International Conference on Intelligent User Interfaces: IUI 2002*. San Francisco, CA. pp. 208-209, January 13-16, 2002.

[21] Mynatt, E.D. and W.K. Edwards. An Architecture for Transforming Graphical Interfaces. Proc. *ACM Symposium on User Interface Software and Technology: UIST '94*. Marina del Rey, California. pp. 39-47, November 2-4, 1994.

[22] Nichols, J. Informing Automatic Generation of Remote Control Interfaces with Human Designs. Proc. *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 864-865, April 20-25, 2002.

[23] Olsen, D.R., S.E. Hudson, R.C.-M. Tam, G. Conaty, M. Phelps, and J.M. Heiner. Speech Interaction with Graphical User Interfaces. Proc. *IFIP TC.13 Conference on Human Computer Interaction: INTERACT2001*. Tokyo, Japan: IOS Press, 2001.

[24] Paternó, F., *Model-Based Design and Evaluation of Interactive Applications*. Applied Computing, ed. R. Paul, P. Thomas, and J. Kuljis. London: Springer-Verlag. 192 pp., 2000.

[25] Puerta, A. The Mecano Project: Comprehensive and Integrated Support for Model-Based Interface Development. Proc. *1996 International Workshop of Computer-Aided Design of User Interfaces: CADUI '96*. Namur, Belgium: Namur University Press. pp. 19-36, June 5-7, 1996.

[26] Rational, *Rational XDE Professional*. Rational Software Corp.: Cupertino, CA and Lexington, MA. http://www.rational.com/xde/

[27] Schreiber, S. Specification and Generation of User Interfaces with the BOSS-System. Proc. *East-West International Conference on Human-Computer Interaction: EWHCI'94*. St. Petersburg, Russia: Springer-Verlag. pp. 107-120, August 2-6, 1994.

[28] Smith, I., *Support for Multi-Viewed Interfaces*, Unpublished Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA, 1998.

[29] Szekely, P., P. Luo, and R. Neches. Beyond Interface Builders: Model-Based Interface Tools. Proc. *Human Factors in Computing Systems: INTERCHI '93*. Amsterdam, The Netherlands: ACM Press. pp. 383-390, April 24-29, 1993.

[30] Tidwell, J., *Common Ground: A Pattern Language for Human-Computer Interface Design*, 1999. http://www.mit.edu/~jtidwell/common_ground.html

[31] van Duyne, D.K., J.A. Landay, and J.I. Hong, *The Design of Sites*: Addison-Wesley, 2002.

[32] van Welie, M. and H. Trætteberg. Interaction Patterns in User Interfaces. Proc. *Seventh Pattern Languages of Programs Conference: PLoP 2000*. Monticello, Illinois, August 13-16, 2000. http://jerry.cs.uiuc.edu/~plop/plop2k/proceedings/Welie/Welie.pdf

[33] Wagner, A., Prototyping: A Day in the Life of an Interface Designer, in *The Art of Human-Computer Interface Design*, B. Laurel, Editor. Addison-Wesley: Reading, MA. p. 79-84, 1990.

[34] Wiecha, C., W. Bennett, S. Boies, J. Gould, and S. Greene, ITS: A Tool for Rapidly Developing Interactive Applications. *ACM Transactions on Information Systems*, 1990. **8**(3): pp. 204-236.

[35] Zimmermann, G., G. Vanderheiden, and A. Gilman. Prototype Implementations for a Universal Remote Console Specification. Proc. *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 510-511, April 20-25, 2002.