RJ10377 (A0604-015) April 10, 2006
Computer Science

IBM Research Report

Investigating Early-Stage Design of User Interfaces for Cross-Device Web Applications

James Lin
IBM Research Division
Almaden Research Center
650 Harry Road
San Jose, CA 95120-6099

James A. Landay
DUB Center
Computer Science and Engineering
University of Washington
Seattle, WA 98195

Lawrence Bergman, Guruduth Banavar, Danny Soroker, Richard J. Cardone
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Investigating Early-Stage Design of User Interfaces for Cross-Device Web Applications

James Lin [1*], James A. Landay [2], Lawrence D. Bergman [3], Guruduth Banavar [3], Danny Soroker [3], Richard J. Cardone [3]

[1] IBM Almaden Research Center
USER Group
650 Harry Rd
San Jose, CA 95120, USA
jameslin@us.ibm.com

[2] DUB Center
Computer Science and Engineering
University of Washington
Seattle, WA 98195, USA
landay@cs.washington.edu

[3] IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA
{bergmanl, banavar, soroker, richcar}@us.ibm.com

ABSTRACT

Designers increasingly need to create web applications that can run on multiple types of devices, such as desktop PCs, handhelds, and mobile phones. However, the ability of designers to explore design ideas is hampered by the lack of tools for early-stage design of user interfaces for cross-device web applications. To understand how designers currently handle such design tasks and discover what features a tool should have to support and enhance the design process, we interviewed cross-device UI designers, and we prototyped and evaluated an early-stage, cross-device UI design tool. We found that such a tool should make it easier to maintain consistency across devices; allow designers to use, capture, and reuse design patterns; give designers more control over the retargeting process; and show how UI elements across devices are related.

ACM Categories and Subject Descriptors

D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User Interfaces*, H.5.2 [**Interaction Interfaces and Presentation**]: User Interfaces – *Prototyping*

General Terms

Design, Human Factors

Keywords

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

INTRODUCTION

Designers of web applications face a computing world that is becoming increasingly complex. Users are more frequently augmenting traditional desktop computer usage with mobile devices such as handheld computers and mobile phones. This allows users to access web applications in many more locations and situations than they can with a PC, but it also increases the burden on user interface designers. Designers cannot simply deploy a desktop user interface on different types of devices; they must tailor the user interface to match the characteristics of each individual device, such as a smaller screen or a phone keypad.

Currently, designers either create a user interface for each device from scratch, which is time consuming, or they rely on programs that take an existing user interface for one device and automatically generate versions for other devices at run time, which usually produces undesirable results. For example, Google's HTML-to-WML proxy splits Travelocity's home page into 12 pages, each requiring scrolling.

We believe that a hybrid approach is the most useful: a tool that allows designers to design a user interface for one device, and then generates UI designs for other devices. We call this *retargeting*. The tool would then allow the designer to modify the generated UIs to create the finished device-specific interfaces. The main advantage of this approach is that designers can quickly design high-quality web applications for multiple devices.

We want to determine how useful such a tool would be to designers, especially in the early stages of design when ideas are most fluid. Will the designers find these generated user interfaces useful or will the generated artifacts just get in the way? How will the generated user interfaces be used? What characteristics should the generation process have to make such a tool useful?

To explore these questions, we took a two-pronged approach. By interviewing designers in their workplaces, we obtained an overall understanding of the cross-device UI design process. We also received detailed feedback about what a cross-device UI design tool should have, by building and evaluating a prototype of such a tool. These findings are informed our design of Damask, an early-stage cross-device UI tool.

The rest of the paper is organized as follows. First, we describe our study of current practices and implications for cross-device design tools. Next, we describe our tool prototype, called HopiSketch, and how designers use it. We also describe our evaluation of HopiSketch and the feedback we received from designers. Then, we discuss the

* This work was done while James Lin was a graduate student in the Group for User Interface Research, Electrical Engineering and Computer Sciences Department at the University of California, Berkeley.

Table 1. Interview participants.

Designer	Type of company	Desktop	Palm	Pocket PC	WAP phone	Hi-end phone
A1	Web portal	✓	✓	✓	✓	
A2	Enterprise software	✓	✓	✓	✓	
A3	Mobile access to corporate data	✓	✓	✓	✓	
A4	Corporate portal	✓		✓	✓	
A5	UI design firm	✓	✓	✓	✓	
A6		✓	✓	✓	✓	
A7	Startup incubator	✓		✓		✓
A8	Mobile phone carrier	✓	✓		✓	
A9	Mobile phone carrier	✓				✓

implications of these results on design tools for cross-device UIs. We also discuss how we addressed our findings in Damask, a tool for such UI design activities. Finally, we discuss related work and conclude.

STUDY OF CURRENT PRACTICES

We interviewed nine UI designers across eight companies who worked on cross-device web UI projects, seven men and two women. We interviewed six of the designers in their offices, two over the phone, and one by e-mail. All of these projects targeted desktop PCs and mobile phones, and all but one also targeted PDAs. Table 1 lists the type of companies for which our participants worked and the devices for which they designed.

We focused our questions on how the designers addressed the issue of handling multiple devices. We wanted to know whether their companies grouped designers by device (e.g., PDA vs. phone designers) or by application (e.g., e-mail vs. calendar). We asked how they maintained consistency across designs, whether the desktop and mobile versions were developed at the same time, whether the same team worked on both versions, and if not, whether the two teams discussed their designs with each other. We also asked them whether they observed recurring interaction design patterns [21] in their designs and whether they documented them.

Finally, we discussed our ideas for a cross-device UI design tool and asked them for their reactions and to speculate on how useful such a tool would be. While it is hard for someone to predict how they would use a tool, the questions were designed to learn more about the designers' concerns for such a tool, not to learn about specific features.

We will discuss our findings along the following themes: responsibilities of the designers; scope of cross-device projects; organization of project teams; managing UI consistency; tools and documentation, and patterns; and the need for real-time change across devices.

Responsibilities of the Designers

All of the designers were responsible for overall information and interaction design, and some also handled graphic design. None of them were developers. Seven of the designers did detailed UI design work. The others, Designers A2 and A9, guided the people doing the detailed design work and made sure they followed good usability principles and adhered to the companies' mobile UI style guides.

Scope of Cross-Device Projects

For most of the cross-device projects, the mobile UI offered a subset of the desktop UI's functionality. For Designers A5 and A6, mobile access and desktop access were thought of as two aspects of their projects as a whole; neither was considered a subset or superset of the other. Designer A9's projects were focused on phone interaction; the desktop was used mostly for managing aspects of the mobile experience, like storing pictures that the user took with the phone's digital camera.

Organization of Project Teams

At all but one of the companies, there were at most three people in charge of the UI design for a project. At the UI design firm, a project typically had two to six designers.

For six of the designers, the cross-device projects were targeted at multiple devices from the beginning. The designers worked on both the desktop and mobile versions at the same time.

Designers A1, A2, and A4 only worked on the mobile UIs. These applications were originally written for the desktop and were later ported to mobile devices. These designers did not consult the desktop UI designers or their design documents; they simply looked at the desktop UI directly.

When asked how the tool should support multiple designers, the designers did not suggest any elaborate features. Designer A9 said that he never saw other designers actually use collaboration features in other tools and stressed that the overall learning curve of a new tool has to be low for a designer to consider using it.

We were particularly interested in finding out how a team of designers typically split up responsibility for designing a cross-device UI project. There are several possibilities:

- *Device added later:* A UI for a device is designed long after the UI for another device is done
- *Group by feature:* One designer designs a large part of the UI for *all* devices, at the same time other designers work on other parts of the UI
- *Group by device:* One designer designs the UI for only *one* device, at the same time other designers work on other devices

The process makes a big impact on the design of a cross-device design tool. For example, Process 3 is not a good fit for a tool that takes a UI designed by a designer, and

presents a generated UI for another device to that same designer.

We found that Processes 1 and 2 were the most common. Only Designer A2 said that they followed Process 3, which was a mistake, because he and his colleagues had trouble maintaining consistency among the various device-specific UIs. For example, one application would say "e-mail" and another would say "message." Consequently, they switched to Process 2 for their next revision.

Managing UI Consistency Across Devices

All of the designers said that maintaining consistency across devices was a major issue. While the interaction obviously cannot be the same across all devices, the designers said that parts of the UI should be, such as menu order, terminology, colors and graphics. Designer A2 said that it was easier to keep device-specific UIs consistent if designers grouped themselves by application rather than device, as mentioned above.

The most common way that the designers achieved consistency was simply to check their designs manually to make sure they were being consistent, which was tedious. They did not have any specialized tools for this purpose.

Designers A5 and A6 typically created an information architecture diagram first, then designed the user interface off of that. Making sure their UI designs were consistent with the information architecture typically kept the designs consistent with each other.

Tools, Documentation, and Patterns

The tools that the designers used were similar to those used by other web and interaction designers. The most commonly mentioned tool was Microsoft Visio, which was used not only for conceptual diagramming, but also for laying out mobile phone UIs. Other tools included paper, whiteboards, Adobe Illustrator, and Microsoft FrontPage. None of the designers used computer tools specialized for handling multiple devices.

Three companies (A1, A2, and A9) developed style guides for mobile UIs. Since Designer A2's company also makes software development tools, the company's long-term goal is to incorporate the style guide standards directly into a development tool for mobile UIs.

Designer A2 and his co-workers also tried to tackle the cross-device UI design problem by developing their own cross-device application flow language. However, they found it hard to design a language that could encompass both high-level application flow and device-specific interaction. They eventually abandoned the project due to lack of time and manpower.

All of the designers said they observed recurring interaction design patterns in their work. However, only Designers A2 and A9 actually documented their patterns, incorporating them into their companies' mobile UI style guides. The others did not document their patterns because they did not have enough time or did not think they were useful enough to document.

When we told the designers about our idea of making design patterns a cornerstone of a cross-device design tool, all but one of the designers were enthusiastic; Designer A8 was not sure whether designers would be able to recognize patterns in their work often enough to be useful. The designers also thought that enabling designers to create their own patterns and add them to the tool's pattern library was very important, and many thought it was crucial.

Need for Synchronized Changes Across Devices

The designers' reactions varied on whether it was important to see the mobile phone UI change while they edited the desktop UI, and vice versa. Four of the designers did not think it was important; they were concerned that the transformation process simply would not be good enough to warrant real-time change. Two designers would like to have the option. The others did not know.

Implications for Cross-Device UI Design Tools

Presenting retargeting results. All of the designers designed the user interface for a particular feature across multiple devices. Therefore, a tool that takes a designer's UI for one device and presents that designer with UIs for other devices fits within current design practices.

Support for multiple designers. According to our interviewees, explicit support for multiple designers in a tool is not a high priority, since detailed design work for a particular feature is usually done by one designer.

Maintaining consistency of content across devices. Consistency was identified as a major burden of cross-device designers. The challenge is to keep the appropriate content consistent across devices, while letting the layout and navigation flow between screens change to fit the target device.

Support for design patterns. Using design patterns as the foundation of a cross-device UI design tool is a sound idea, but allowing designers to create their own patterns is essential for the long-term usefulness of this feature.

HOPISKETCH: A PROTOTYPE OF A CROSS-DEVICE DESIGN TOOL

While our interviews allowed us to discover general aspects of the design process that we needed to support, we wanted to get more detailed feedback about how an early-stage cross-device design tool should behave and what features it should have. It is hard for people to speculate about what such a tool should be like without interacting with one. Since there are no early-stage cross-design tools, we quickly designed and evaluated a prototype of one, called HopiSketch. It was built using DENIM [10] for the user interface and Hopi [2] for the retargeting process. Due to time constraints, this was done before the interviews in the previous section were completed; consequently, we were not able to incorporate all of the findings of those interviews into HopiSketch.

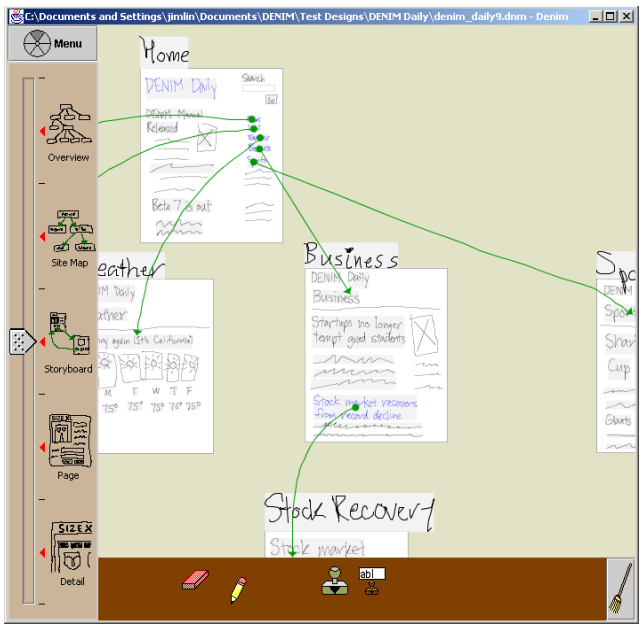

Figure 1. DENIM showing a typical design.

User Interface

We decided to use a sketch-based interface for the user interface of HopiSketch because designers usually sketch on paper during the early stages of design [14]. The user interface is based on DENIM, an existing sketch-based tool for early-stage web design.

DENIM has one window (see Figure 1) with three main areas. The center area is a canvas where the designer creates web pages, sketches the contents of those pages, and draws arrows between pages to demonstrate the behavior of hyperlinks (Figure 2). On the left is a slider that is used to set the current zoom level. The bottom area is a toolbox that holds tools for drawing, panning, erasing, and creating and inserting reusable components.

Designers test the interaction of their designs in Run mode. Opening a pie menu over a page and selecting File→Run launches a separate browser window with the page loaded. The designer can navigate through the site design exactly like in a web browser, clicking on hyperlinks and using the Back and Forward buttons.

We augmented DENIM to allow designers to insert radio buttons, check boxes, buttons, and drop-down boxes, which are commonly used in web applications, directly into their designs (see Figure 3a).

In addition, we added the ability for designers to group elements together to indicate that the elements are related. For example, a designer can group a text box and a Search button together to show that they should be treated as one unit (see Figure 3b). Groups also affect the behavior of any radio buttons: Within a group, only one radio button may be selected at a time.

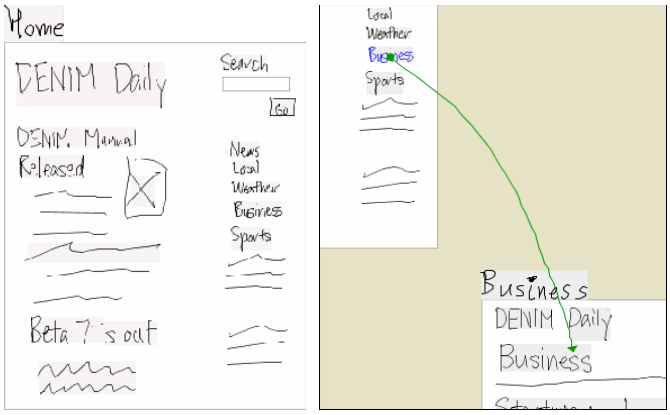

Figure 2. a) *Left* A page with the label "Home" b) *Right* An arrow, whose source is a blue hyperlink, "Business."

HopiSketch focused on design for PCs and for Palm handheld devices. To retarget a PC design to the Palm, the designer presses a Retarget button. HopiSketch takes the design, resizes the pages to fit the Palm's screen, and if needed, splits pages to minimize scrolling on the Palm. Elements within a retargeted page, such as handwriting and sketched images, are not resized or otherwise altered. Figure 4 shows a design for the PC and the results of retargeting the design to a Palm handheld.

Architecture

Figure 5 shows the overall architecture of HopiSketch. When designers press the Retarget button, the system takes the design file and feeds it to a *de-sketcher*, which translates (or *de-sketches*) it into a generic model [7]. The model is based on XHTML [22] for general content elements and XForms [23] for form elements such as radio buttons

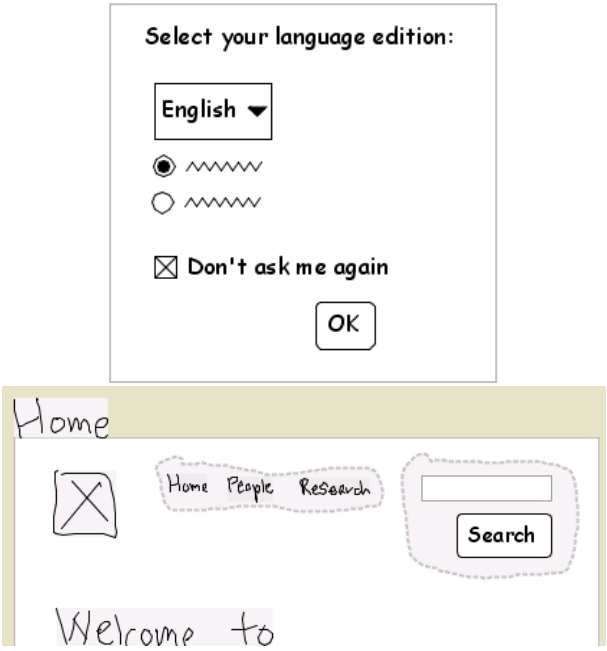

Figure 3. a) *Top:* Web form widgets within a page. b) *Bottom:* Two groups within a page.

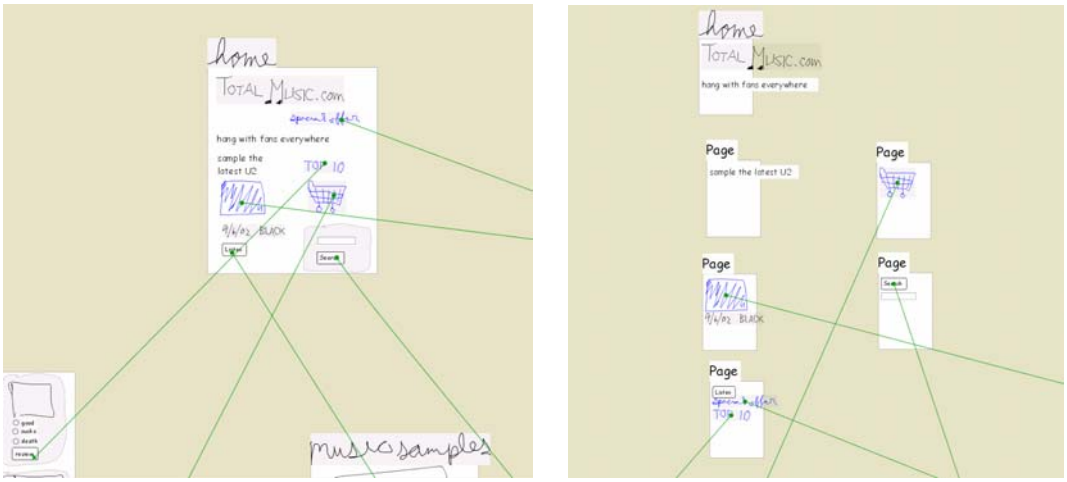

Figure 4. ***Left:*** **A UI design for the PC.** ***Right:*** **The design retargeted for the Palm handheld.**

and check boxes. One XHTML+XForms page in the model represents one page in the original DENIM file.

The model is then fed through Hopi, a system for designing cross-device web applications based on a generic model. The model first goes to Hopi's *retargeter*, which transforms it into a markup language for a target device. This process can result in one XHTML+XForms page being split up into several pages, depending on the characteristics of the target device. The retargeter creates pages that fit within a target device's screen, or are a little longer, allowing a bit of scrolling. The retargeter tries to keep elements that have been grouped together on the same page, although this is not always possible.

The resulting markup pages are then fed into Hopi's *renderer/geometry extractor*, which renders the markup using the predefined characteristics of the target device and determines the positions of elements in the markup.

Finally, a *re-sketcher* takes the markup, the extracted geometry, and handwritten elements from the original DENIM file, and creates a sketch-based version of the markup to be presented to the designer.

EVALUATION OF HOPISKETCH

To evaluate HopiSketch, we performed an informal task-based usability test. The participants were introduced to HopiSketch and then asked to create elements of a simple e-commerce site.

Participants

Six designers participated in the usability study, four men and two women. All six designers were employed at user interface design or information architecture firms, had experience designing for the desktop web, and had at least some experience designing for mobile devices. Four of the designers have worked on cross-device user interfaces, although such interfaces are not the focus of their current work. Table 2 summarizes the characteristics of the participants.

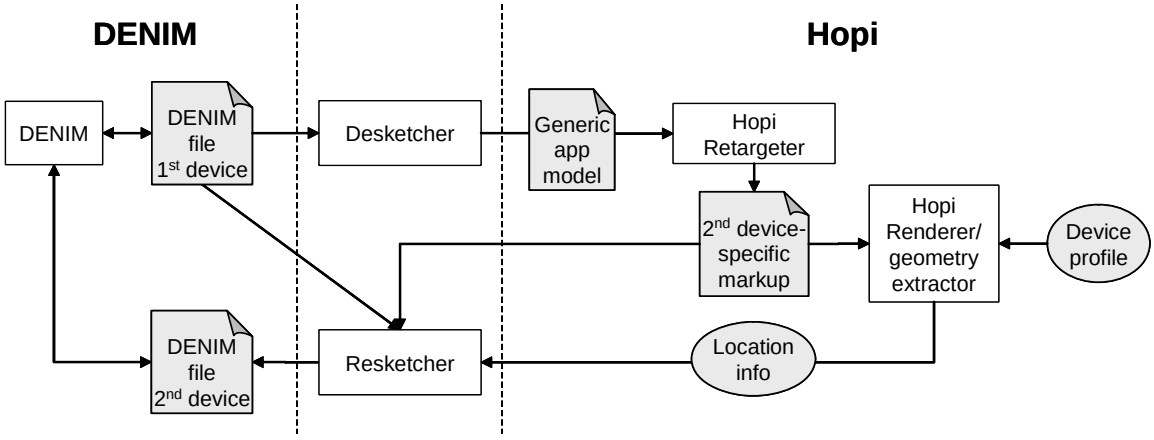

Figure 5. The architecture of HopiSketch.

Table 2. Summary of participants of our HopiSketch evaluation.

Participant	Characteristics
B1	UI designer Graphic design background Uses Photoshop and Illustrator Has worked on > 20 cross-device projects
B2	Interaction designer Liberal arts background Uses Photoshop, Fireworks, and Dreamweaver Has worked on < 5 cross-device projects
B3	Information architect Programming and business background Uses Photoshop, Visio, and Flash Has not worked on any cross-device projects
B4	Information architect Media (TV, photography) background Uses Visio and Photoshop Has worked on < 5 cross-device projects
B5	UI designer and usability engineer Computer science background Uses Illustrator and Dreamweaver Has not worked on any cross-device projects
B6	UI designer Graphic design background Uses Fireworks and Visio Has worked on < 5 cross-device projects

Methodology

The usability tests were performed on an IBM ThinkPad laptop with a Wacom Graphire tablet. First, we gave the designers a warm-up task to get used to the tablet. Next, we demonstrated HopiSketch and had the designers do some basic tasks, such as creating pages, adding elements to pages, and running the designs. Then, we asked the designers to create an online music store application for a desktop browser. We retargeted these desktop applications to Palm devices; the designers were then able to modify the generated results. About 60 minutes were available for the complete design task, including creating the desktop application and editing the Palm version.

Finally, we debriefed the designers and had them fill out a questionnaire. We were looking for comments addressing two general themes:

- Were HopiSketch and the generated user interfaces useful? Would the answer change depending on the number of devices being targeted?
- How can HopiSketch be enhanced to better support the design of cross-device applications?

Results

We found that HopiSketch had implementation flaws that made it difficult for designers to perform some tasks. In particular, the de-sketching process was not sufficiently robust and mature to handle all the designers' sketches, which led to pages being split and elements within the pages being laid out in unexpected ways.

We also found that because the designers only had about 30–40 minutes to design for the desktop, their desktop designs were not very large. Consequently, some designers said that it would have been easier to simply re-sketch their small designs from scratch instead of starting from our generated user interfaces. Some of them were also slowed down by their lack of familiarity with the Wacom tablet.

Given the maturity of HopiSketch and the time constraints of the evaluation, most designers concluded that HopiSketch was no faster than using paper and pencil for retargeting the designs that they had created. On the other hand, five of the six designers saw potential benefits of the tool within a broader context:

1. Two of the designers, Designers B4 and B5, thought that for large designs, a design tool that can retarget could potentially save them a lot of time.
2. Three of the designers also found value in the generated sketches, even though they were not ideal. Two of the designers, Designers B1 and B2, thought that the generated sketches still provided a useful starting point to design for the second device. Designer B2 said that by starting from the generated sketches, he would not forget to implement features in the PC version for the Palm version. Thus, if a feature was not present in the Palm version, it was because he explicitly deleted it from the generated design, not because he forgot to copy it from the PC version.
3. Designer B1 said that the generated sketches were useful to show to clients, to demonstrate to them how unwieldy a Palm web site would be if it had all of the functionality of the PC web site.
4. Another designer, Designer B6, said that he could imagine that a more robust version of HopiSketch would generate sketches that would help him "see potential pitfalls (or opportunities)" in the design for the target device.

When we asked the designers the minimum number of target devices that would be required for a retargeting tool such as HopiSketch to be useful, all but one of the designers said two devices. One of them said that the tool would probably be most useful if the two devices were the same general type, such as from one cell phone to another, as opposed to from desktop PC to cell phone.

However, when we asked the designers how likely they were to use a commercial-strength retargeting tool for early-stage design, the reaction was more mixed. Three designers were likely to use one, one designer was neutral, and two said they were unlikely. One of the designers who was likely to use a retargeting tool said he would do so only if it were not sketch-based. This is because he would only

use sketch-based tools for conceptual design, not for designing layouts for specific devices.

Finally, the designers gave us several suggestions that would make a retargeting tool more useful to them, which we describe in the next section.

Implications for Cross-Device UI Design Tools

The designers described a number of ways in which they believe a tool for retargeting designs could be more useful. Most of the suggestions are related to the theme of letting designers better understand, guide, and control the retargeting process. Each of the following suggestions was made by at least one designer. While these suggestions are not necessarily representative of the design community as a whole, we believe each suggestion has merit.

Control over retargeting. Four of the designers mentioned that they would like to guide the retargeting process directly. They would like to be able to explicitly tag which sections of a page should be carried over to the target-device design, and which sections should be omitted, before the retargeting process takes place. One designer said he would like to make the tags conditional on what the target device is.

Another designer said that, when targeting the Palm, the tool should not split pages automatically, since the Palm handheld has scroll buttons. Instead, the tool should create pages that would scroll and then allow designers to split the pages themselves. This shows that information about the devices' characteristics must be taken into account throughout the retargeting tool for the tool to be effective.

Iterative design. Many designers wanted to better understand the retargeting process. For example, some said they would prefer a more iterative approach than the study permitted. Due to time and tool constraints, all of the designers went through the retargeting process only once. These designers would rather design a little bit for one device, retarget, look at the results, design a bit more for the first device, and so on. One designer specifically mentioned that he would like to see the design for the target device modified in real time while he worked on the design for the initial device.

There should also be a tighter relationship between designs of the same user interface on different devices. With HopiSketch, a retargeted design has no relationship to the original design once it has been generated. Ideally, the tool should be able to propagate changes made in a generated device-specific design back to the original. However, not all changes should be propagated. A designer may want to remove an element in a mobile phone version because it is unnecessary, but keep it in the desktop version because it aids navigation. How to support such an intelligent process remains an open question.

Templates and content replication. Another theme was the ability to intelligently replicate content. For example, several designers mentioned that if they wanted a search box in the upper right-hand corner of every page, they would like to create a template that contains the search box, and apply that template to all of the pages in the site.

They also mentioned that if a page is split during retargeting, some elements in the original page, such as search or navigation aids, should be replicated on each of the resulting pages. Designers would need a way to specify which elements should be replicated, since it would be difficult to make such decisions automatically. The challenge is to provide means for specifying which elements to replicate without burdening the designer or cluttering the design.

Support for alternative design processes. A cross-device tool should be flexible enough to support a variety of design practices, especially since cross-device design is a new discipline and design practices are still evolving. For example, HopiSketch was designed to take a user interface for a large display, like a desktop PC, and retarget it to a device with a smaller display, like a Palm handheld. One designer said it was easier for him to add to a design than subtract from a design, so he would prefer to do the opposite of HopiSketch: take a Palm user interface and merge its pages to form a desktop PC version.

Improved page splitting. All of the designers said that the algorithms for rearranging and splitting up content could be improved. One designer said that any handwriting and images should be shrunk to fit the dimensions of the handheld. Similarly, one designer mentioned that since Palm handhelds can scroll, groups should never be split between two or more pages. Instead, the tool should create a scrolling page that would keep all of the items of a group together.

Sketch-based interface. Some designers found the sketch-based interface appealing. Designer B1 said it took "napkin sketching to a new experiential level without making it beautiful," and that it allows him to focus on whether his ideas are valid. Designer B2 simply said that "it's a good way to work."

Others did not find it as compelling. Designer B4 wanted additional shape and alignment capabilities, such as provided by Visio or other diagramming tools. Designer B2 liked sketching, but said he uses sketching only for conceptual design. For layout design, he would prefer to use a more structured interface.

Designer B1 suggested that the contents of the pages could contain a coarse grid similar to graph paper. This would help, but not force, designers to draw neater sketches, and would indirectly help the retargeting algorithms, since they work better when elements are aligned.

Familiar interaction. Some designers expressed reluctance to learn a new tool interface, and would have liked HopiSketch's user interface to have been more similar to

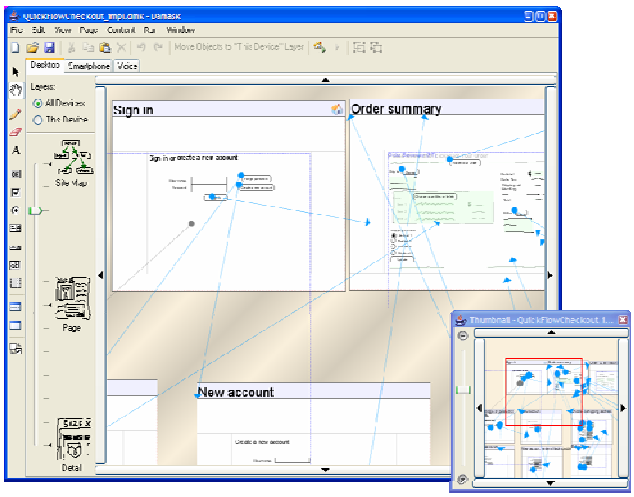

Figure 6. Damask's user interface. The tabs at the top of the canvas allow the designer to switch between the different device versions of this design. The Thumbnail window in the lower right-hand corner shows a miniature view of the design.

the tools they already use. The most commonly mentioned tools were Adobe Photoshop and Microsoft Visio.

Handling different classes of devices. There was some skepticism that our tool would be really useful for designing user interfaces to be run on different classes of devices, such as PCs and mobile phones. Designers B1, B2, and B3 said that the interaction flow is very different among different classes of devices, and that there is insufficient support in HopiSketch to handle those differences.

A cross-device design tool should be able to support the design of applications whose user interfaces have very different interaction flows depending on the device. HopiSketch does not handle such design activities because it only transforms at the page and widget level. Higher levels of abstraction within the design are needed to support disparate interaction flows, such as design patterns [9].

DAMASK

As a follow up to this work, we implemented a new cross-device UI design tool called Damask [9] (see Figure 6). It combines the advantages of designing multiple interfaces from scratch with the speed of automatically generating interfaces. With Damask, the designer designs a user interface for one device by sketching the design and using design patterns [21] from Damask's pattern library (see Figure 7). As the designer creates an interface, Damask uses the sketches and patterns to construct an abstract model [7], which captures aspects of the UI design at a high level of abstraction. Damask uses the abstract model to generate the other device-specific interfaces, which the designer can refine. Damask targets three types of interfaces: desktop web, mobile phone web, and voice prompt-and-response.

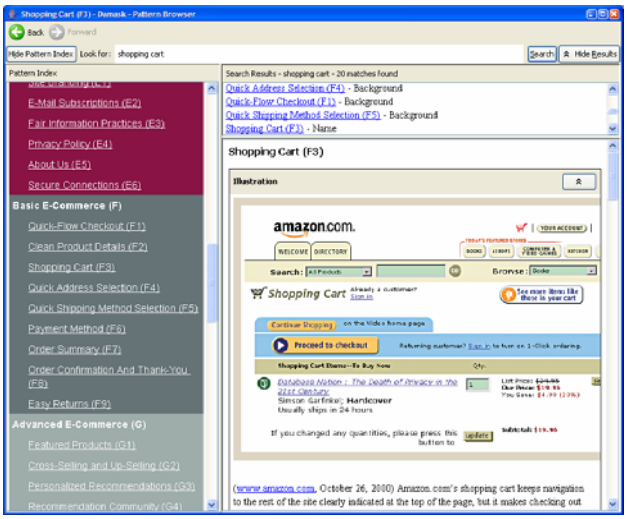

Figure 7. Damask's pattern browser. The list of available patterns is on the left, and the Shopping Cart pattern is shown on the right, under the search results.

Damask provides a Run window in which designers interact with their design sketches in a browser that simulates the devices they are targeting.

As a result of our studies, Damask incorporates many of the design considerations discussed above. For example, as the designer creates a UI for one device, the UIs for the other devices are generated synchronously, allowing a highly iterative design process. Damask's user interface allows designers to indicate which parts of a UI design should be retargeted and which should not, has better support methods of interaction besides sketching such as text entry, has a "graph paper"-like background for pages, includes the concept of page templates, and incorporates more familiar UI elements, such as more keyboard shortcuts, pull-down menus, and a toolbar, although support for sketching remains.

RELATED WORK

Here we contrast our work to model-based user interfaces and tools that transform UIs from one device or modality to another.

Model-Based User Interfaces

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 [7]. 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 as having one part where the user can select one of three items, and two other on-off selections. This level of abstraction 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.

While model-based user interfaces offer the possibility 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 user interface 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.*, [18, 20, 25]), or by specifying a task model which is then transformed into a concrete user interface (*e.g.*, [7, 17]). 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, at which most designers are not skilled, so specifying models impedes their main task of designing user interfaces.

The philosophy of most model-based user interface 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 later modified somewhat by the designer. In contrast, our tool is targeted towards prototyping. We do not expect the designer to use our system to create the final user interface, nor do we expect its generated user interfaces to be used without modification. Since we are targeting the creation of prototypes, the generated user interface does not need to be ideal—in the early stages of design, the designer is concerned more with the user's interaction flow than with the details of the interface [24].

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 handheld interfaces at run-time [4, 8, 11]. Unfortunately, shrinking interfaces from large desktop displays to small handheld displays often results in awkward interaction. Others have worked on converting graphical user interfaces to audio interfaces [13, 16], mostly to benefit the blind and visually impaired. With most of these tools, designers cannot modify the results of the interface transformation process. Since our tool is not meant for the final implementation of user interfaces, designers are free to modify the generated UI design.

Ultraman [19] 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. Our tool is targeting a different audience at a different point in the design cycle: designers with little or no programming experience, who are working at an early stage of design before any interface is completely specified.

Model-Based Transformation Tools

There are several model-based projects that specifically address the issue of creating user interfaces targeted at multiple devices. Eisenstein, Vanderdonckt, and Puerta [6] describe using MIMIC [17] to create models which describe cross-device user interfaces. Their methodology involves mapping common tasks in a task model to presentation models optimized for the task. Ali et al [1] discuss designing a cross-device user interface using three types of models: an abstract logical model, physical family models, and platform-specific user interface descriptions in UIML. In contrast, our tool avoids directly exposing models to the user interface designer.

PIMA [3], Hopi [2], and Microsoft's ASP.NET Mobile Controls [12] are tools for designing 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 (such as "choose one of many items") 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.

Calvary, Coutaz, and Thevenin [5] 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 cross-device UIs. In our tool, however, top-level abstract models are not directly exposed, so such a framework is not directly applicable.

There are several projects that specify platforms for creating universal remote controls (*e.g.*, [15, 26]). 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 (remote controls for appliances vs. web interaction), but the user interfaces that are rendered from the abstract remote control description must be appealing and useful immediately, without additional tweaking. Our work, on the other hand, is targeting a broader set of user interfaces (*e.g.*, general web-style interaction on PCs), but the interfaces that are generated will most likely be modified by the user interface designers before being released.

CONCLUSION

We found that taking a two-pronged approach to studying the design process of cross-device UIs gave us different but valuable information. We got a general sense of the design process by interviewing designers in their workplaces, and

we got detailed feedback about what a design tool should have by building and evaluating a prototype of such a tool.

Through these studies, we have found that the concept of a tool, that retargets UIs for a given designer, fits how designers currently split up responsibility of handling multiple devices among themselves. A tool that supports design patterns will allow designers to take advantage of them more systematically. The tool needs to give the designer a high degree of control over the retargeting process. Simply letting designers modify the generated interfaces is not sufficient. Designers should be able to annotate their designs so that the tool is more intelligent in its retargeting process, and the tool should be flexible enough to allow for highly iterative design and a variety of design processes. We are incorporating these lessons into Damask, our next early-stage cross-device UI design tool.

ACKNOWLEDGMENTS

We would like to the designers who participated in both studies. We would also like to thank John Karat, Noi Sukaviriya, and Tracee Wolf for helping us with the design of our study and questionnaire; Pauline Ores and Kate Swann for helping us recruit participants for our user study; and Frederique Giraud, Ashish Kundu, Yves Gaeremynck, and Vianney Chevalier for their contributions to the implementation of HopiSketch.

REFERENCES

[1] Ali, M.F., M.A. Pérez-Quiñones, M. Abrams, and E. Shell. Building Multi-Platform User Interfaces With UIML. In Proceedings of *2002 International Workshop of Computer-Aided Design of User Interfaces: CADUI'2002*. Valenciennes, France. pp. 225-236, May 15-17, 2002.

[2] Banavar, G., L.D. Bergman, Y. Gaeremynck, D. Soroker, and J. Sussman, Tooling and System Support for Authoring Multi-Device Applications. *Journal of Systems and Software (Special Issue on Ubiquitous Computing)*, 2004. **69**(3): pp. 227-242.

[3] Bergman, L.D., G. Banavar, D. Soroker, and J. Sussman. Combining Handcrafting and Automatic Generation of User-Interfaces for Pervasive Devices. In Proceedings of *2002 International Workshop of Computer-Aided Design of User Interfaces: 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: Proceedings of Human Factors in Computing Systems: 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 Proceedings of *Engineering for Human-Computer Interaction: EHCI 2001*. Toronto, ON, 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 Proceedings of *International Conference on Intelligent User Interfaces: IUI 2001*. Santa Fe, NM: ACM Press. pp. 69-76, January 14-17, 2001.

[7] 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. In Proceedings of *Design, Specification and Verification of Interactive Systems: DSV-IS'94*. Carrara, Italy. pp. 3-14, June 8-10, 1994.

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

[9] Lin, J., *Using Design Patterns and Layers to Support the Early-Stage Design and Prototyping of Cross-Device User Interfaces*, Unpublished Ph.D. Dissertation, Electrical Engineering and Computer Sciences, University of California, Berkeley, 2005. http://www.cs.berkeley.edu/~jimlin/research/damask/dissertation/jlin_dissertation.pdf

[10] 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.

[11] 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. pp. 690-694, 2001.

[12] Microsoft, *ASP.NET Mobile Controls*. Microsoft Corporation: Redmond, WA. http://msdn.microsoft.com/mobility/prodtechinfo/devtools/asp.netmc/

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

[14] Newman, M.W. and J.A. Landay. Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice. In Proceedings of *DIS 2000: Designing Interactive Systems*. New York, New York. pp. 263-274, August, 2000.

[15] Nichols, J., *et al.*, Generating Remote Control Interfaces for Complex Appliances. *CHI Letters: Proceedings of User Interfaces and Software Technology: UIST 2002*, 2002. **4**(2): pp. 161-170.

[16] 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 Proceedings of *IFIP TC.13 Conference on Human Computer Interaction: INTERACT2001*. Tokyo, Japan: IOS Press, 2001.

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

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

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

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

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

[22] W3C HTML Working Group, *XHTML™ 1.0: The Extensible HyperText Markup Language (Second Edition)*, 2002. http://www.w3.org/TR/xhtml1/

[23] W3C XForms Working Group, *XForms 1.0: W3C Working Draft*, 2002. http://www.w3.org/TR/xforms/

[24] 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. pp. 79-84, 1990.

[25] 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.

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