RC22594 (W0210-103) October 17, 2002
Computer Science

IBM Research Report

Investigating Early-Stage Design of Multi-Device Web Applications

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

[1]Group for User Interface Research
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA

[2]IBM T.J. Watson Research Center
19 Skyline Drive
Hawthorne, NY 10532, USA

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

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g. , payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

Investigating Early-Stage Design of Multi-Device Web Applications

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

[1] Group for User Interface Research
Computer Science Division
University of California
Berkeley, CA 94720-1776, USA
jimlin@cs.berkeley.edu

[2] 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 multi-device UIs. To address this problem, we designed and prototyped an early-stage, multi-device web application design tool. Our system allows a designer to sketch a web user interface design for a single device and then generates designs for other devices. The designer can subsequently modify the generated user interface.

We informally evaluated the prototype with six user interface designers. Although our tool is in the early stages of development, we were still able to gain insight into the features that a multi-device web design tool should support. These insights include the need to give designers more control over the retargeting process and the need to support a tight relationship between designs of the same user interface targeted at different devices.

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 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.

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 on the fly, which often produces undesirable results. 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 tweak the generated UIs to create the finished device-specific interfaces. The main advantage of this approach is that designers can quickly design basic 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?

In order to explore these questions, we created a prototype tool that allows a designer to sketch a rough design for a device-specific user interface, typically for a desktop web browser. Using this initial specification, the tool generates a design sketch for a user interface specific to another device. The designer can then hand-customize the generated design.

Our informal evaluation of the prototype with UI designers suggests that a tool similar to our prototype can be useful and that a critical element in the design of any such tool is the control designers are given over retargeting. Designers want to guide the retargeting process, before the actual retargeting takes place. They would like a tight connection between device-specific UIs, so that changes in any one of the device-specific UIs are quickly, if not immediately, reflected in the others. The tool should also support different design practices, especially since multi-device design is new and design practices are still evolving.

The rest of the paper is organized as follows. First, we describe the user interface of the tool and how designers use it. Next, we discuss the tool's architecture. We then describe how we conducted our evaluation and the

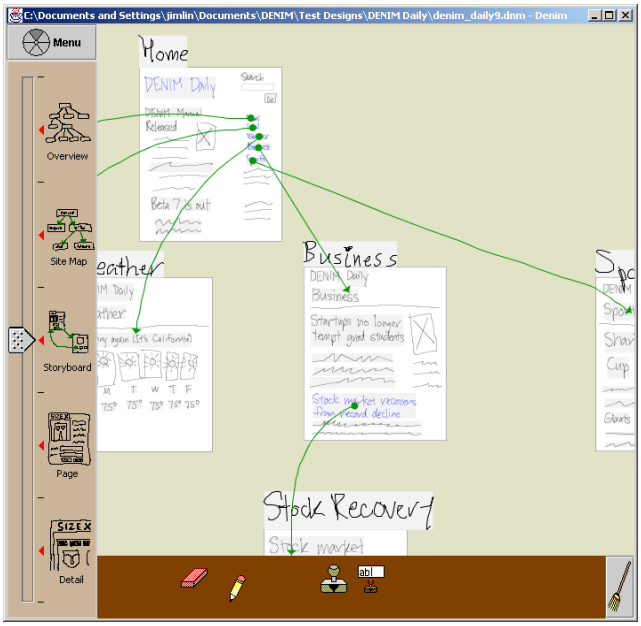

Figure 2. DENIM showing a typical design.

feedback we received from designers. Finally, we discuss related work and conclude.

USING THE PROTOTYPE

We decided to use a sketch-based interface for the user interface of our prototype because designers usually sketch on paper during the early stages of design [14]. The user interface is based on an existing sketch-based tool for early-stage web design called DENIM [10]. DENIM supports three different representations of sketched-web design input—site maps, storyboards, and pages—and it unifies these representations through zooming.

DENIM has one window (see Figure 2) with three main areas. The center area is a canvas where the user creates web pages, sketches the contents of those pages, and draws arrows between pages to represent their relationship to one another. 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.

Instead of pull-down menus, DENIM uses techniques geared towards pen interaction. For example, pie menus [4] are used for executing commands. Alternatively, pen gestures can be used for quickly executing the most common commands, such as copying, pasting, and panning.

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

DENIM designs consist of *pages* and *arrows*. Pages represent the web pages in a site. A page consists of two parts: a label describing the page, and a sketch representing the physical appearance of the web page (see Figure 3a). A designer can sketch or type in a page.

An arrow between two pages represents a relationship between those pages (see Figure 3b). To create a relationship, the designer draws an arrow between two pages. If the arrow starts from a particular item in a page, such as a word, image, or button, then the source of the arrow becomes a hyperlink, and is marked in blue. In Run mode the user can click on the hyperlink to transition to the destination page.

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 1a).

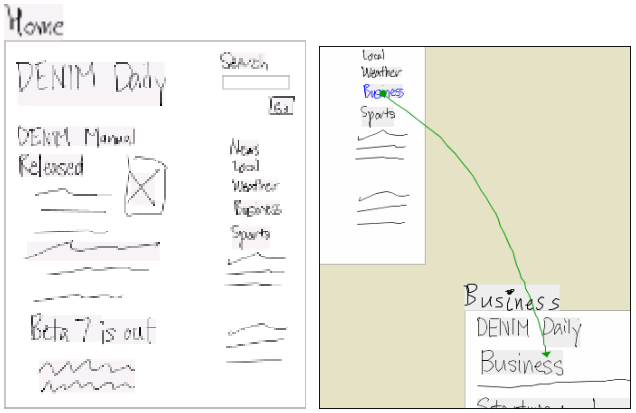

Figure 3. a) A DENIM page with the label "Home" b) An arrow, whose source is a blue hyperlink, "Business."

Figure 1. a) *Top:* Web form widgets within a DENIM page. b) *Bottom:* Two groups within a DENIM page.

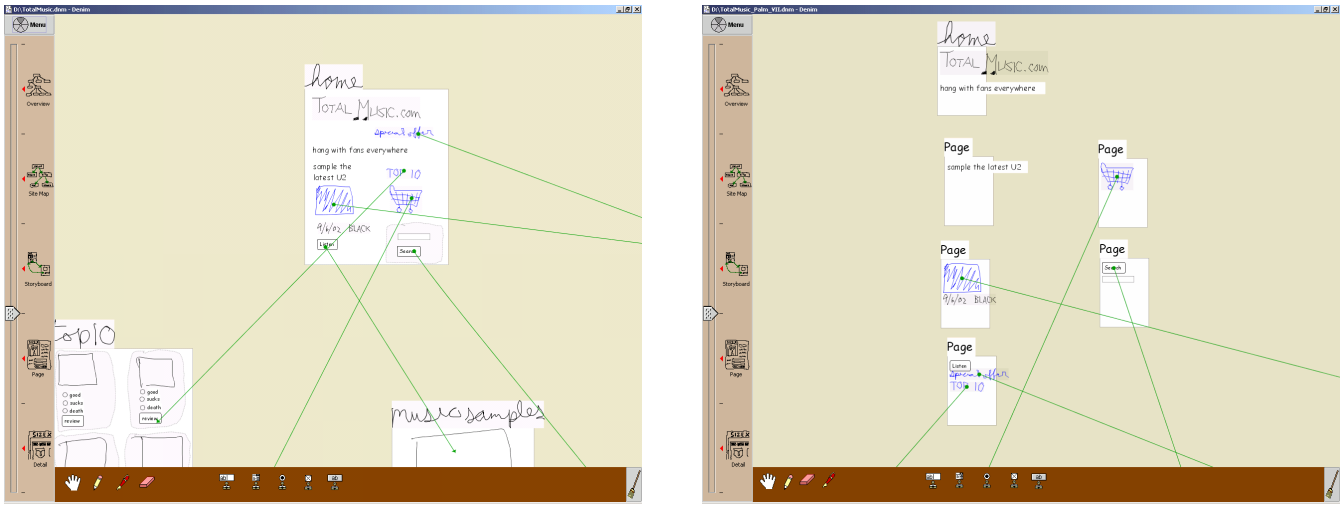

Figure 4. ***Left:*** **A DENIM design for the PC.** ***Right:*** **The design retargeted for the Palm handheld.**

We also 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 1b). Groups also affect the behavior of any radio buttons: Within a group, only one radio button may be selected at a time.

Currently, our tool focuses on design for PCs and for Palm handheld devices. To retarget a PC design to the Palm, the designer presses a Retarget button. The tool 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 OF THE TOOL

Figure 5 shows the overall architecture of our prototype. When designers press the Retarget button, the system takes the design file and feeds it to a *desketcher*, which translates (or *desketches*) it into a generic model. The model is based on XHTML [21] for general content elements and XForms [22] for form elements such as radio buttons and check boxes. One XHTML+XForms page in the model represents one page in the original DENIM file.

The model is then fed through a *specializer*, 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 specializer creates pages that fit within a target device's screen, or are a little longer, allowing a bit of scrolling. The specializer 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 a geometry extractor, which renders the markup using the characteristics of the target device and determines the positions of elements in the markup.

Finally, a *resketcher* 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.

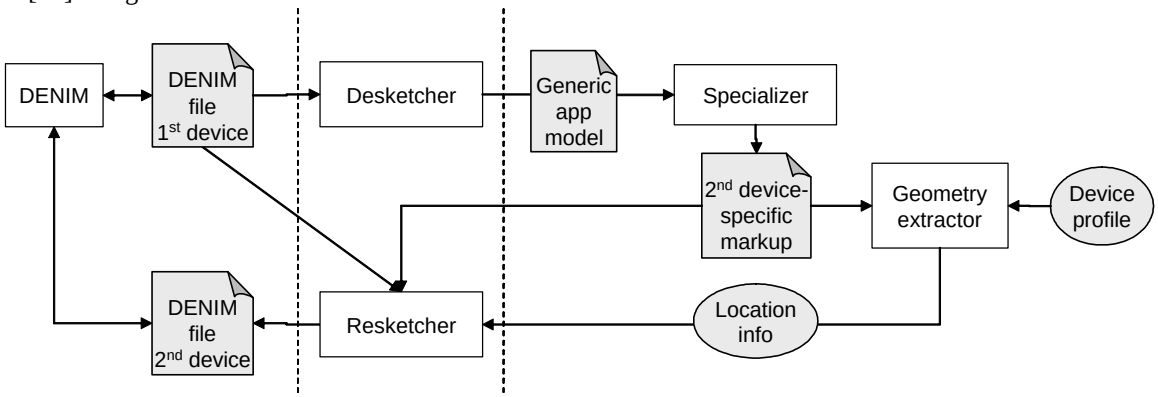

Figure 5. The architecture of the prototype.

EVALUATION

To evaluate our tool, we performed an informal task-based, usability test. The participants were introduced to our tool 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 are employed at user interface design or information architecture firms, have experience designing for desktop web, and have at least some experience designing for mobile devices. Four of the designers have worked on multi-device user interfaces, although such interfaces are not the focus of their current work. Table 1 summarizes the characteristics of the participants.

Participant	Characteristics
1	UI designer Graphic design background Uses Photoshop and Illustrator Has worked on > 20 multi-device projects
2	Interaction designer Liberal arts background Uses Photoshop, Fireworks, and Dreamweaver Has worked on < 5 multi-device projects
3	Information architect Programming and business background Uses Photoshop, Visio, and Flash Has not worked on any multi-device projects
4	Information architect Media (TV, photography) background Uses Visio and Photoshop Has worked on < 5 multi-device projects
5	UI designer and usability engineer Computer science background Uses Illustrator and Dreamweaver Has not worked on any multi-device projects
6	UI designer Graphic design background Uses Fireworks and Visio Has worked on < 5 multi-device projects

Table 1. Summary of study participants.

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 the prototype system 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 sixty 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 the tool and the generated user interfaces useful? Would the answer change depending on the number of devices being targeted?
- How can the tool be enhanced to better support the design of multi-device applications?

Results

We found that our prototype tool had implementation flaws that made it difficult for designers to perform some tasks. In particular, the retargeting 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 resketch 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 our tool and the time constraints of the evaluation, most designers concluded that using our tool 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 #4 and #5, 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 #1 and #2, thought that the generated sketches still provided a useful starting point to design for the second device. Designer #2 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 #1 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 #6, said that he could imagine that a more robust version of the tool 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 our prototype 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.

DESIGN CONSIDERATIONS FOR A SKETCH-BASED RETARGETING SYSTEM

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 of the designers. 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 that 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. Other 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 our tool, 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, a concept called *round-tripping*. 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 intelligent round-tripping 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. The tool should be flexible enough to support a variety of design practices, especially since multi-device design is a new discipline and design practices are still evolving. For example, our tool was designed to go from 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 our tool: 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 #1 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 #2 simply said that "it's a good way to work."

Others did not find it as compelling. Designer #4 wanted additional shape and alignment capabilities, such as provided by Visio or other diagramming tools. Designer #2 liked sketching, but said he uses sketching only for conceptual design. For layout design, he would prefer to use a more structured interface.

Designer #1 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 like DENIM's user interface to be more similar to 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 #1, #2, and #3 said that the interaction flow is very different among different classes of devices, and that there is insufficient support in our tool to handle those differences.

A multi-device design tool should be able to support the design of applications whose user interfaces have very different interaction flows depending on the device. Our prototype 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. Design patterns may be one such abstraction [9].

RELATED WORK

Our work is closely related to the concept of *model-based user interfaces*, designing user interfaces based on an abstract model of the interface rather than visual appearance [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, 24]), 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 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 [23].

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 [3, 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 user interface 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.

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 multi-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 multi-device user interface using four types of models: a task model, 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 [2] and Microsoft's Mobile Internet Toolkit [12] are tools for designing multi-device web applications. A designer using either of them describes the application's user interface in an abstract representation, by laying out abstract widgets linearly in a constrained Visual Basic-like form designer. The representation is then converted into concrete device-specific UIs. However, these tools are not appropriate for early-stage design, because designers tend to think about concrete user interfaces, not abstract representations.

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 multi-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, 25]). 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

When designing multi-device web applications, the level of automation can be thought of as a continuum. At one end, designers can separately design a user interface for each device. This approach results in interfaces appropriate for the target device, but is impractical if many devices are being targeted. At the other end, designers can use a tool to automatically generate user interfaces for each desired device, but the generated interfaces are often awkward to use.

We are exploring the middle ground: creating a tool that automates the mundane aspects of multi-device development, but which also lets designers modify the resulting interfaces to fit the particular devices' characteristics. Our evaluation suggests that such a tool is potentially useful, but that 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.

ACKNOWLEDGMENTS

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

REFERENCES

1. Ali, M.F. and M.A. Pérez-Quiñones. Using Task Models to Generate Multi-Platform User Interfaces while Ensuring Usability. In Proceedings of *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 670-671, April 20-25, 2002.
2. 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.
3. 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.
4. Callahan, J., D. Hopkins, M. Weiser, and B. Shneiderman. An Empirical Comparison of Pie vs. Linear Menus. In Proceedings of *Human Factors in Computing Systems*. pp. 95-100, 1988.
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. and J.A. Landay. Damask: A Tool for Early-Stage Design and Prototyping of Multi-Device User Interfaces. To be published in Proceedings of *2002 International Conference of Distributed Multimedia Systems*. Redwood City, CA, Sept. 26-28, 2002.
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. p. 690-694, 2001.
12. Microsoft, *Mobile Internet Toolkit*. Microsoft Corporation: Redmond, WA. http://msdn.microsoft.com/vstudio/device/mitdefault.asp
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. Informing Automatic Generation of Remote Control Interfaces with Human Designs. In Proceedings of *Human Factors in Computing Systems: CHI 2002 Extended Abstracts*. Minneapolis, MN. pp. 864-865, April 20-25, 2002.
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, 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. W3C HTML Working Group, *XHTML™ 1.0: The Extensible HyperText Markup Language (Second Edition)*, 2002. http://www.w3.org/TR/xhtml1/
22. W3C XForms Working Group, *XForms 1.0: W3C Working Draft*, 2002. http://www.w3.org/TR/xforms/
23. 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.
24. 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.
25. 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.