# SketchWizard: Wizard of Oz Prototyping of Pen-Based User Interfaces

*Richard C. Davis,*[1] *T. Scott Saponas,*[2] *Michael Shilman,*[4] *and James A. Landay*[2,3]

[1]CS Division
UC Berkeley
Berkeley, CA 94720
rcdavis@eecs.berkeley.edu

[2]DUB Group
UW, CSE
Box 352350
Seattle, WA 98195

[3]Intel Research, Seattle
1100 NE 45th Street,
6th Floor
Seattle, WA 98105

[4]ChatterPop, Inc.
2035 15th St. #3
San Francisco, CA 94114
michael@shilman.net

{ssaponas, landay}@cs.washington.edu

**ABSTRACT**

SketchWizard allows designers to create Wizard of Oz prototypes of pen-based user interfaces in the early stages of design. In the past, designers have been inhibited from participating in the design of pen-based interfaces because of the inadequacy of paper prototypes and the difficulty of developing functional prototypes. In SketchWizard, designers and end users share a drawing canvas between two computers, allowing the designer to simulate the behavior of recognition or other technologies. Special editing features are provided to help designers respond quickly to end-user input. This paper describes the SketchWizard system and presents two evaluations of our approach. The first is an early feasibility study in which Wizard of Oz was used to prototype a pen-based user interface. The second is a laboratory study in which designers used SketchWizard to simulate existing pen-based interfaces. Both showed that end users gave valuable feedback in spite of delays between end-user actions and wizard updates.

**ACM Classification:** H5.2 [Information interfaces and presentation]: User Interfaces. - Graphical user interfaces.

**General terms:** Design, Experimentation, Human Factors

**Keywords:** Wizard of Oz, pen-based user interfaces, informal interfaces, sketching, mark-based user interfaces

**INTRODUCTION**

Interest in pen-based user interfaces has continued to grow over the past decade [2,12,13,17–20,24,28,32]. Unfortunately, no design standards have evolved, and adoption of such systems in the marketplace has been slow. The software shipped with pen computers is designed primarily for mouse input and usually ignores the expressive power of pen strokes and sketching. True pen-based user interfaces, such as informal interfaces [3,7,10,17,30], harness this expressive power, but these systems are notoriously difficult to design. We have addressed this situation by building SketchWizard, a tool that allows designers to build and test prototypes of pen-based user interfaces.


*UIST'07,* October 7–10, 2007, Newport, Rhode Island, USA.

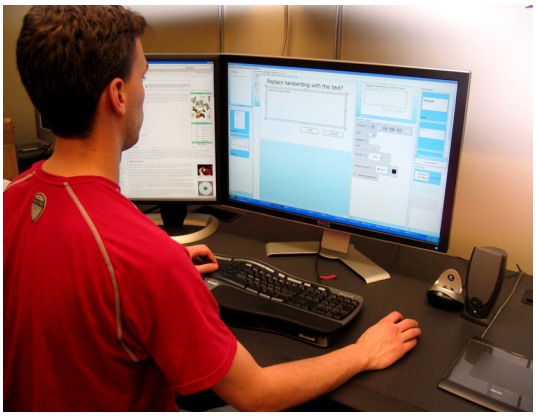

Designers work by quickly generating and evaluating numerous design ideas [9,36], but this is hard to do with pen-based interfaces because of the tight coupling between the interface and the technology behind it. These systems provide rich interaction by recognizing, and at times transforming, user sketches and pen gestures—processes that are difficult to simulate with paper prototypes [31,34]. The only current alternative is to build a working system, but this takes time and a deep understanding of technology. With SketchWizard, designers can build Wizard of Oz prototypes, which have evolved as a solution to this type of problem for other hard-to-build interface styles [6,8,11,14,15,20,22,26].

A Wizard of Oz prototype is an incomplete system that a designer can simulate "behind a curtain" (usually by taking the place of a recognizer) while observing the reactions of real end users (see Figure 1). Existing tools make it possible for designers with no programming skill to build Wizard of Oz prototypes of speech [15], location-enhanced [20,22], augmented-reality [8], and desktop [27] applications. With these early-stage Wizard of Oz prototyping tools, designers have the freedom to explore possibilities before technology details are set in stone. This flexibility helps a design team make reasonable technology decisions as the design iterates. SketchWizard gives this ability to pen-based UI designers.

Whether it is possible to build effective early-stage Wizard of Oz prototypes of pen-based UIs is an open question. In other application domains where interactions can be modeled with a small number of input/output primitives, designers can mock up fairly complete interfaces. This way, simulations can be run with minimal input from the designer (the "wizard"). Pen-based interfaces, however, have a much broader input space, processing gestures and sketches in ways that cannot be easily defined. Consequently, the wizard must quickly execute complex transformations of pen input to create an acceptable simulation. SketchWizard assists designers with this process by offering special tools for capturing and modifying end-user input.

Our work makes the following research contributions:

(a)



(b)

Figure 1: Wizard of Oz test setup. (a) A designer observing a user interacting with a pen-based user interface. (b) Another designer operating the interface behind the scenes with SketchWizard.

- A system, SketchWizard, that enables designers to create and test Wizard of Oz prototypes of pen-based UIs, capturing user actions for later analysis.
- Wizard of Oz tool features that enable fast responses to end-user actions when pen-based UIs are being simulated.
- Two sets of studies demonstrating that Wizard of Oz testing of pen-based user interfaces in the absence of an implementation is useful and feasible.

In the following section we describe the pen-based interface design process in more detail and explain SketchWizard's role. We then describe the various components of the system. Then we present two evaluations of SketchWizard: an early feasibility study that revealed the advantages and challenges of our method, and a laboratory experiment that demonstrates designers' ability to successfully simulate real interfaces with the tool. Finally, we review related work, conclusions, and ideas for future work.

## PEN-BASED INTERFACE DESIGN WITH WIZARD OF OZ

Our interest in Wizard of Oz techniques comes from the difficulty of designing successful pen-based applications. In other interface domains, designers commonly conduct many iterations with a progression of detailed prototypes before beginning software implementation. Because of the tight coupling between design and technology in pen-based interfaces, however, we found ourselves jumping quickly from rough sketches to implementation in the systems that we created [7,17,20,30,32].

We envisioned a process (similar to the one described by Dow et al. [8]) that uses Wizard of Oz prototypes to enable three additional phases of design iteration:

1. **Input language evaluation.** In the earliest stages of design, simple prototypes can be used to explore possible input languages (gestures or sketches). If a design for an input language exists, it can be tested with real users to see if it is sensible and learnable. If no design exists, then a prototype may be used to explore user intuitions about possible input languages. By recording user input, designers can analyze input languages and even use this input as test cases for training or evaluating technology options. This technique has been used to prototype interruptability systems [11] and augmented-reality, pen-and-paper air traffic control systems [24].
2. **Input and output language evaluation**. As the input language begins to take shape, Wizard of Oz prototypes can introduce responses to input, such as recognition or other transformations. Such prototypes have been used successfully to design pen-based UIs [1] and augmented-reality systems [29,35]. They can also help flesh out poorly defined aspects of system behavior [20,26].
3. **Whole-system evaluation**. After specific technologies are introduced, it is possible to evaluate the system as a whole. Wizard of Oz prototypes are still useful in such situations, but the wizard supervises the output of technology components rather than taking the place of those components [8].

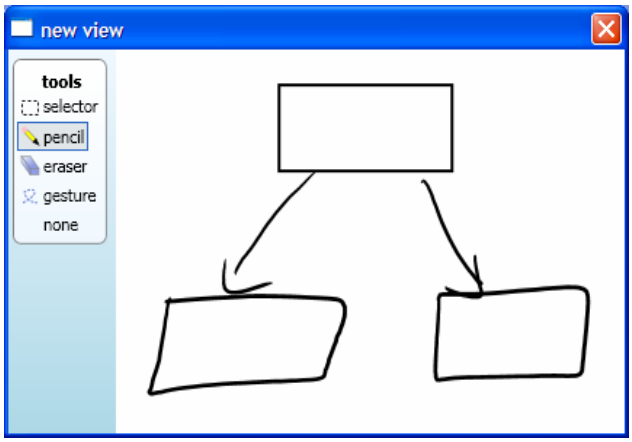
SketchWizard has grown out of our attempts to enable teams of designers and developers to use this process. The system currently supports the first two phases (designer only) and may support the third phase (designer and developer) in the future. Let's continue looking at these first two phases, while examining how SketchWizard prototypes are built and operated.

## SKETCHWIZARD DESIGN AND IMPLEMENTATION

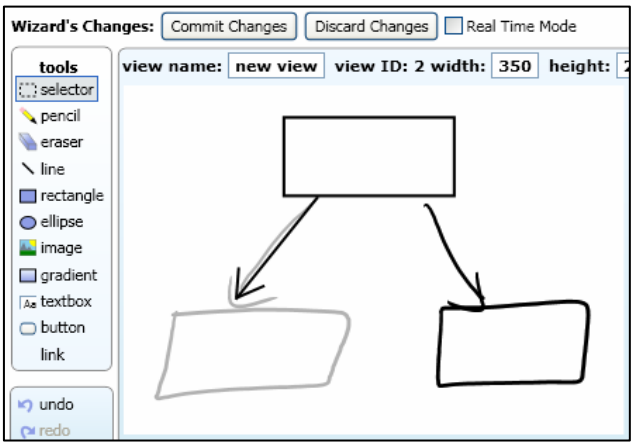
To make SketchWizard capable of simulating a wide variety of pen-based applications [3,10,12,13,16–19,28,30], we did our best to avoid constraining the appearance or behavior of prototypes. Therefore, we designed our system around a simple drawing canvas shared between an end-user interface (see Figure 2a) and a wizard interface (see Figure 2b). End users interact with objects on the canvas and have access to a limited number of other tools for drawing, erasing, selecting, and moving objects. Using a larger palette of drawing tools that includes ellipses, images, and buttons, wizards build the interfaces that users

see. With these tools, wizards can simulate rich system output, such as the "beautified" box and arrow in Figure 2.

By default, both end-user and wizard edits appear immediately in each interface. This is "Real-Time Mode." Knowing, however, that a wizard would often want to delay updating an end user's view until multiple edit operations had completed, we provided a "Manual Commit Mode." In this mode a wizard sends all pending edit operations to the end user's view by pressing a **Commit Changes** button. The wizard can also discard all pending edits. During editing, a grayed-out version of the end user's view is shown behind the wizard's canvas, enabling quick execution of sketch transformations, such as erasing a sketch and drawing a recognized version on top of it, as in Figure 2.




(a)




(b)

Figure 2: User's view (a) and wizard's view (b) of the shared drawing canvas in SketchWizard. The wizard has deleted a box and arrow and replaced the arrow with a beautified version. Updates appear to the user when the wizard presses the **Commit Changes** button.

The presence of this Manual Commit Mode introduces a subtle question: should end users be able to edit the canvas while the wizard is editing it? Our initial answer to this question was yes, but, as we shall see later, the possibility of concurrent edits was confusing for end users. In addition, the need to resolve conflicts between user edits and wizard edits introduced considerable complexity into our implementation. For these reasons, our current implementation blocks the end user from editing the canvas after the wizard has begun to make edits. When blocked, the end user's cursor changes to an hourglass to signify that the system is "working."

The full SketchWizard system consists of three separate applications: the User's View, the Wizard's Workspace, and the Session Player. The User's View appears in Figure 2a and has already been described in its entirety. In the remainder of this section we describe the Wizard's Workspace in detail, then take a brief look at the Session Player, and close by discussing implementation details and limitations of the entire system.

### Wizard's Workspace

The Wizard's Workspace (see Figure 3) allows designers to build and run simulations of pen-based user interfaces. Designers can prepare a simulation, save it, and reload it before running a test. During test sessions, wizards must operate their prototypes by making fast edit operations. Consequently, the Wizard's Workspace is optimized for fast access to commands and runs best on a large, high-resolution screen.

To the left of the main drawing canvas in the Wizard's Workspace is a palette of tools that is similar to what designers find in most vector graphics applications. This palette includes lines, rectangles, ellipses, imported images, text boxes, and gradients. Interface designs can be created in SketchWizard with these tools or copied from another Windows application that supports image, text, or XAML formats. Wizards can also create buttons that end users can press to signal their desire to execute a command.

The panel to the right of the drawing canvas is for editing the drawing properties of selected objects. Most of these properties are similar to those found in other tools (e.g., fill color, stroke color, stroke thickness, and font size). Others (e.g., the **editable** property on text boxes) control the ability of end users to manipulate these objects.

The end user's current pen mode can be set by the wizard or the end user. The wizard controls this mode with a combo box above the canvas. Checked items in this combo box will also appear in the end user's palette of tools. SketchWizard provides up to four tools to the end user: a pencil, an eraser, a selection loop, and a gesture tool that shows a trace but leaves no ink behind. This limited set of end-user tools provides all the functionality needed by a wide range of pen-based applications.

Because making real-time edits to a complex interface design can be challenging, the Wizard's Workspace has powerful tools for making fast edits: a list of views, an event stream, paste buttons, and a scratch pad.

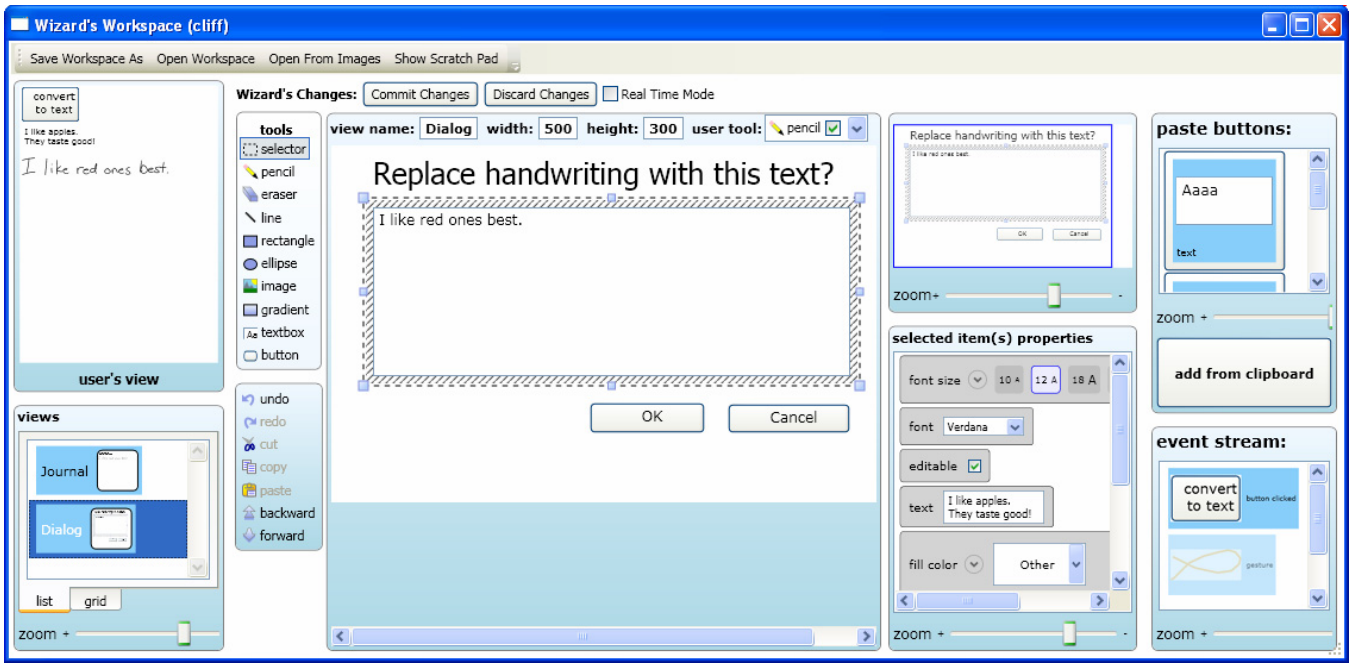



Figure 3: SketchWizard Wizard's Workspace: (Top left) Scaled-down User's View. (Bottom left) List of views that the wizard can edit and show to the user. (Center) Main drawing canvas. (Top right of center) Radar view of the main drawing canvas. (Bottom right of center) Editable properties of selected objects. (Top right) Paste buttons. (Bottom right) Event stream.

*List of Views.* The list of views (bottom left of Figure 3) allows wizards to store full screens of content and switch to them quickly during a test. This list can be used, for example, to provide multiple canvases or to create a rough simulation of a dialog box. Wizards can add any number of new views, name them, and switch to them by selecting the item in the list.

Like other wizard edits, switching to a new view updates the User's View either immediately (in Real-Time Mode) or when the wizard commits changes (in Manual Commit Mode). However, if the wizard is in Manual Commit Mode and edits a view other than the one visible to the end user, the end user is not blocked from interacting with his view. This feature allows a wizard to prepare a new view, such as a dialog box, while the end user is still working. To keep the wizard aware of the User's View at such times, a scaled-down version of the User's View appears above the list of views.

*Event Stream.* As mentioned already, end users can signal their desire to initiate actions by pressing buttons or making pen gestures. In our early use of SketchWizard, however, we found that we could become so focused on editing the end user's view that we would miss end-user actions. To keep the wizard aware of pending end-user actions such as these, we added an event stream (lower right corner of Figure 3). When an end user draws with the gesture tool or clicks on a button, an item is added to the top of this list showing the gesture or the name of the button. Events that occurred before the most recent wizard commit appear slightly transparent to distinguish them from recent events.

*Paste Buttons.* The paste buttons (upper right corner of Figure 3) are useful when wizards need to repeatedly add complex objects to the canvas. This need can arise, for example, if end-user drawings are recognized and transformed to a set of beautified or iconic forms—a common occurrence in many pen-based user interfaces. Before a simulation begins, a wizard can create a set of objects, copy it to the clipboard, and then click **Add from clipboard** to add a button to this box. Clicking on the button pastes the copied item to the canvas. If objects are selected when the button is pressed, they will be replaced with the pasted objects, giving wizards a fast way to replace objects with recognized versions.

*Scratch Pad.* Designers may wish to operate on drawings in a safe area where there is no danger of locking or updating the User's View. We provide a scratch pad in a separate window (not shown) for this purpose. This window looks much like the main Wizard's Workspace window, but it omits the paste buttons, event stream, and tools for monitoring or updating the User's View. This window can be used to build a complex transformation of a user drawing over time without interrupting the end user.

### Session Player

With some planning and practice, designers can use the Wizard's Workspace to simulate an interface accurately enough to get informative feedback on a design. This feedback may come in the form of comments from users, but the most valuable feedback comes from observing user behavior during a test. To facilitate this type of observation, the User's View session can be saved to a file and analyzed later with the Session Player.

The Session Player has a simple interface consisting of a view of the drawing canvas and the **tools** menu with a few added controls for loading and playing sessions. Session

recordings include all pen movements and changes to the drawing canvas and tool selection. These recordings can be played back at actual speed or high speed (i.e., with pauses removed). In the evaluations presented below, we shall see that the ability to review test sessions is an essential part of the SketchWizard system.

### Implementation Details and Limitations

SketchWizard is implemented in C# and XAML and runs on top of the Microsoft .NET Framework 3.0. It consists of approximately 26,500 lines of code spread across the three applications and a common library. The system is currently robust enough for designers to simulate a wide variety of pen-based user interfaces.

The system is, however, a work in progress. Although SketchWizard covers much of the pen-based interface design space, there are needs we have not addressed. Pen-based menus (e.g., marking menus [16]) are another common feature in pen-based interfaces that are not directly supported by SketchWizard. They can be simulated somewhat awkwardly, though, with multiple views.

In the later stages of design it becomes important for a pen-based interface prototype to integrate real recognition technologies. We acknowledge that the addition of recognition technologies would enhance the tool's value. Even without these capabilities, however, SketchWizard is a valuable part of the design process. The following section presents a series of evaluations that we conducted to demonstrate this.

## SKETCHWIZARD FEASIBILITY STUDY

Early in the development of SketchWizard, we conducted a feasibility study to verify that our approach to Wizard of Oz testing was sound. We hoped that real-time edits to a user's view would not introduce significant delays that would destroy a user's sense of immersion and invalidate any test results. We also hoped that this type of testing would provide feedback that went above and beyond what could be obtained from paper prototypes.

To answer these questions, we attempted to design a pen-based user interface using SketchWizard. This study had two parts, corresponding to the first two phases of Wizard of Oz testing that we described earlier ("input language evaluation" and "input and output language evaluation"). Both parts took place during the early stages of design for an interactive sketch beautification system (inspired in part by Igarashi's work [13]).

The SketchWizard implementation that we used to conduct this study was much less mature than the one that we have described here, but all the essential components were present. The Wizard's Workspace, in particular, was little more than a drawing canvas with tools for creating and positioning straight lines. It included none of the tools for fast editing shown in Figure 3, and some programming was required to tailor the system to a particular test. In addition, as mentioned earlier, users and wizards were able to edit the canvas concurrently while in Manual Commit Mode.

### Input Language Evaluation

As a first step in designing our beautification system, we wanted to see what kinds of gestures users would make spontaneously over their sketches to beautify them. We found three employees at a large software corporation who had diagrams they wished to beautify on whiteboards in their offices. We photographed these diagrams and displayed them on the SketchWizard drawing canvas. The User's View was configured with a single gesture tool (highlighter strokes that disappeared immediately after the pen was lifted). We then asked participants to gesture over their own diagrams in silence with this disappearing highlighter in such a way that we could figure out how to beautify their diagrams. Later, we analyzed their gestures with the Session Player and gave participants beautified versions of their diagrams to verify our interpretation of these gestures.

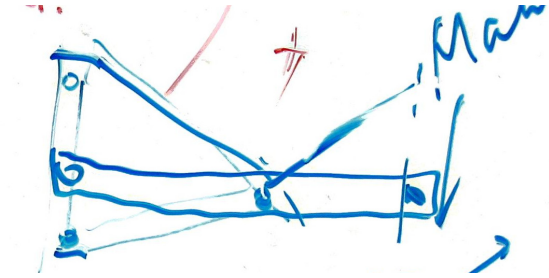
Though primitive, this study did reveal valuable information about user intuitions. Two end users worked by tracing over pieces they wished to beautify. One end user invented a particularly rich gesture set. In all, we identified 29 gestures from the three participants and classified them into 15 types. We imagined conducting this study on a larger scale to identify stronger patterns of user input, and we saved end users' gestures to train a future sketch beautification technology.

This part of our study demonstrates how a Wizard of Oz system can assist designers with input language evaluation. Even though the wizard did not need to simulate any system behavior during each test, the recordings of end-user input enabled a deeper analysis than was previously possible.

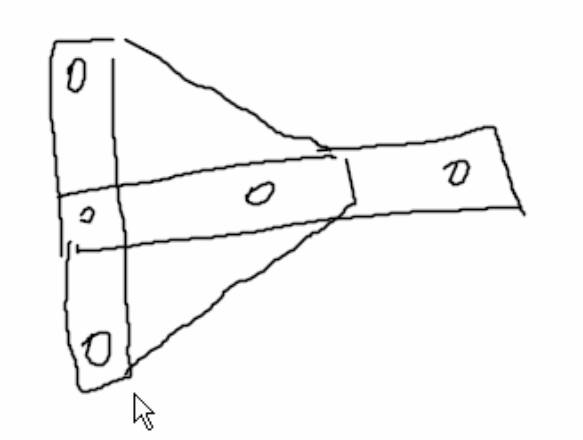
### Input and Output Language Evaluation

Using the data from our first study, we defined a simple beautification language: "tracing" to straighten individual lines, and "circling" to beautify whole sections. We then evaluated this language by conducting a second study using SketchWizard. We found seven software company employees with diagrams they wished to beautify. We then asked the participants to redraw their diagrams in the SketchWizard User's View and gesture over their diagram with a highlighter to beautify it. Figure 4 shows how one participant's drawing evolved from whiteboard drawings (part a of the figure) to drawings in our tool (part b) to beautified figures (part c). We did not tell participants that beautification was being performed by a human being. Instead, we told them that the system was unfinished, slow, and operating over a network.

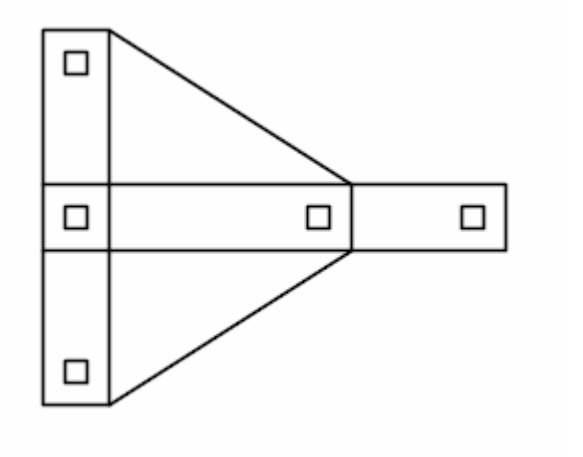
This study also produced valuable feedback on our design. All end users guessed the essentials of our input language without prompting. End users also invented two strategies for correcting beautification errors: some erased sections and redrew them; others "nudged" lines with the highlighter. We were also encouraged by the end users' positive responses to the system, though we noted that their approval correlated with beautification accuracy. We surmised that a highly accurate beautification technology was needed to make this design viable.



(a)



(b)



(c)

Figure 4: User data from input and output language evaluation. (a) Original sketch. (b) Redrawn sketch. (c) Beautified figure.

Promising as these results were, we were more interested in what they said about our Wizard of Oz system. We were surprised to find that, in spite of delays on the order of 10 to 20 seconds between end-user pen strokes and view updates, only one of our seven participants suspected that the system was run by a human being. End users were able to scale back their expectations of the system and focus their attention on the aspects of the design that we wanted to evaluate. Because concurrent edits were possible, however, many users could not tell that the system was "working" during this delay. Some users repeated pen strokes, for example, wondering if the system had missed them the first time. Our current implementation of SketchWizard removes this confusion by preventing concurrent edits.

Our experience using SketchWizard in this design process demonstrates that our Wizard of Oz testing approach can help designers discover new designs and also produce useful feedback on existing designs. The ability to capture and record user pen gestures in an appropriate context helped us discover a natural system design. Simulating system behavior with real-time edits was slow, but users still interacted with interfaces as if they were real, allowing us to observe their reactions to our chosen gesture set.

Satisfied that our approach had merit, we set about completing our implementation of the Wizard's Workspace. This application would enable designers with no programming experience to simulate more complex systems than the one we had evaluated in this study. The next section presents our final study, which evaluates designers' ability to use the complete SketchWizard system that we have described.

## SKETCHWIZARD LABORATORY EVALUATION

To assess designers' ability to work with SketchWizard, we conducted a laboratory study in which seven designers simulated two existing pen-based user interfaces. Five of our designer participants were professional interaction or interface designers, one was a student who had worked previously as an interaction designer, and one was a graphic designer. Experience ranged from up to three years to more than nine years as a designer. None of the designers had ever designed a pen-based UI, and none were familiar with the term "Wizard of Oz" before participating in the study. Designers participated in two sessions each and were compensated with a $150 gift certificate.

### Procedure

In the first session, designers were introduced to SketchWizard and the Wizard of Oz concept. They were then taught how to use SketchWizard and asked to build and simulate subsets of two existing pen-based interfaces. In the first interface, Windows Journal, designers were asked to simulate free-form note taking and conversion to recognized text. They were also asked to use a button instead of a menu to trigger recognition (see Figure 5), and they were told not to simulate the dialog boxes that appear during recognition. Though the designers could have simulated these features by adding views, we wished to focus on the more interactive aspects of the target application, so we omitted them from the designers' task.

The second interface that we asked designers to build and simulate was the DENIM web site design tool [30]. Designers were asked to simulate a mode with three responses to end-user input (see Figure 6): (1) Sketched squares were converted to page objects. (2) Handwritten text was grouped with gray boxes. (3) Sketched lines from text to pages were recognized as links, turning into straight green lines and turning the anchor text blue. These behaviors, together with those

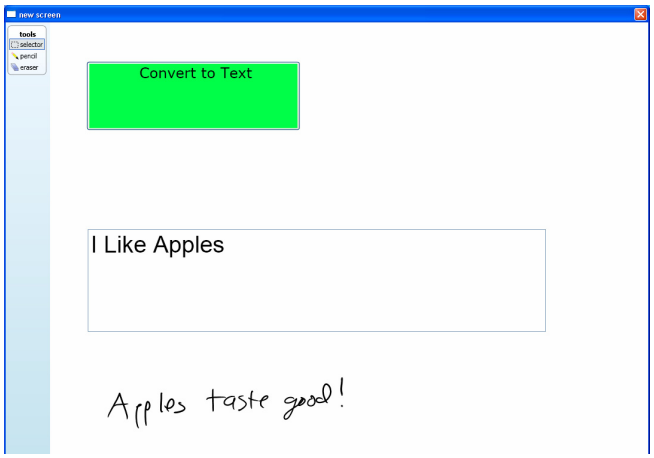



Figure 5: User's View during a study session in which a designer participant simulates Windows Journal.

of Windows Journal, cover a range of the behaviors and data types in pen-based UIs.

For each task, designers viewed a one-minute video demonstration of the interaction that they were to simulate being performed on the actual system. They then built the simulation and practiced the action of both wizard and end user. Toward the end of each task, they practiced simulating the interface as a wizard while a researcher acted as the end user. A desktop computer was used for the Wizard's Workspace, and a Tablet PC was used for the User's View. The first session lasted 75 minutes on average.

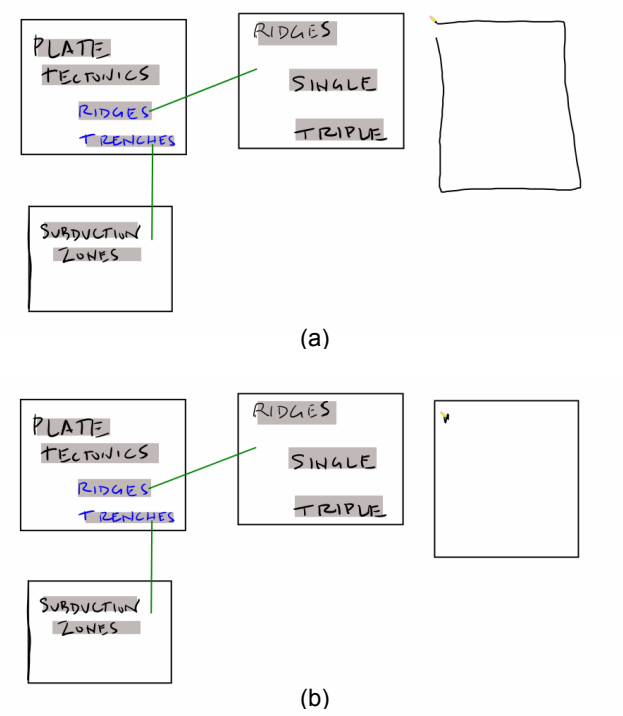



Figure 6: Recorded data from an end-user participant in our laboratory evaluation, showing before (a) and after (b) recognition of a sketched rectangle representing a web page.

The second session followed four to seven days after the first session and also lasted 75 minutes on average. Designers were again asked to run their saved simulations, but this time the end user worked on a Tablet PC in a separate room. Four of these end users were different study participants (frequent computer users from the Seattle area) who came only for this session and were compensated with a $30 gift certificate. These participants were not initially told the true nature of the study, but instead were told that they would be testing a new recognition server. The other three end users were members of our research team.

The first task given to end users was to use the (simulated) Journal interface to take notes on the Wikipedia page for "apple" (fruit) and convert them to text (see Figure 5). The second task was to prepare a site map of a simple web site about apples using the (simulated) DENIM interface (though some participants decided to create a web site on a different topic, as in Figure 6). The designers were told to do their best to simulate the recognition and interaction of the original applications.

### Results

By the end of the first session, all of the designers were able to simulate system behaviors in a reasonable amount of time (5–15 seconds). Many of our participants enjoyed the experience of being the wizard. One participant remarked, "This is like advanced paper prototyping." Some designers also stated that they would like to use SketchWizard for prototyping interfaces and asked when and where they could download the tool.

In the second session, our four end-user participants were able to comment on the interaction design of the interfaces, despite occasional annoyance with recognition delays. The test administrator noted informative end-user reactions to both the **Convert to Text** button in the Journal prototypes and the gray grouping rectangle for text in the DENIM prototypes. Continuing the trend in our feasibility study, no end user could tell that the interfaces were controlled by humans rather than computers. All imagined plausible explanations for the long delays and adapted to the situation.

When simulating Windows Journal, all of the designers watched the event stream for a signal that the **recognize** button had been pressed. When simulating DENIM, most designers chose to use paste buttons to prepare items that they knew they would need. For example, they pasted "pages" (see Figure 6) as replacements for sketched rectangles. Unfortunately, these tasks were not complex enough to require the scratch pad or multiple views.

Designers had many positive comments upon completion of the study. One designer remarked, "I've only used it a couple of times and I think I could use it in a study." Another said, "Very intuitive and easy compared to other drawing and graphic programs I've had to learn." Another saw value in SketchWizard beyond user tests, saying, "I could project this on a whiteboard during a meeting to rapidly iterate on an interface design."

Designers also saw room for improvement. Many asked for straightforward improvements, such as keyboard shortcuts, a rectangle selector, grid snapping, and configurable defaults for object line and fill styles. Several suggested that pasted items should scale to the size of the replaced objects. Some asked for tools to help them work SketchWizard into their design process, such as notes on how to run the simulation that would be saved with the design. Others complained that they felt disconnected from end users in another room. These designers wanted live audio, video, or a one-way mirror for watching the end user, or at least an instant-messenger connection to the other test administrator.

This evaluation demonstrates that designers can use SketchWizard to test pen-based interface designs with end users before any underlying technology is implemented. The event stream and paste buttons were shown to be particularly useful for simulating interfaces quickly. And once again, despite delays caused by human wizards manipulating the canvas, end users became immersed in their experience of the interface and gave valuable design feedback.

## RELATED WORK

Wizard of Oz prototyping of pen-based user interfaces is a relatively new idea. The designers of the CINCH 3D brain-imaging application used this method to prototype gestures [1], but we are aware of no other attempts to use Wizard of Oz in pen-based UI design. Note that the CINCH prototype was programmed from scratch, while SketchWizard seeks to allow prototyping of such applications by designers with no programming skill. Also note that SketchWizard does not currently allow 3D models to be imported or manipulated, but it could be extended to do so.

There is more work in interface prototyping methods that is related to SketchWizard. Next we consider how paper prototyping might be applied to pen-based user interfaces. Then we look at Wizard of Oz tools for other domains.

### Paper Prototyping

In paper prototyping [31,34], a designer prepares parts of an interface with paper, Post-its, transparency film, and other office supplies. The experimenter simulates this rough prototype with end users by moving the pieces by hand in response to end-user actions. This type of prototyping can be applied to pen-based user interfaces, but it breaks down when the end user's input is not highly constrained. If end users are allowed to take free-form notes, for example, copying and transforming these notes in real time by hand becomes prohibitively time-consuming. Even when input is constrained, as in a circuit diagram, editing can cause cascading changes that are hard to simulate with paper.

We know of only two attempts to apply paper prototyping to pen-based UIs: a handwriting recognition application and a collaborative digital whiteboard application [5]. The administrators managed the delay between end-user input and system output by assigning simulation roles to multiple experimenters, which was costly and produced awkward simulations.

SketchWizard is conceptually similar to a paper prototyping system in which the test administrator is invisible. In addition, SketchWizard has special tools for fast creation of the interface parts that must be created during a test. It also adds the important ability to capture detailed recordings of end-user interactions for later analysis.

### Wizard of Oz Prototyping Tools

A number of research tools support Wizard of Oz prototyping. NEIMO [4] is a tool for later-stage prototyping of multimodal interfaces. It allows an administrator to take the place of ink and speech recognizers in an implemented system to elicit feedback on a design before the recognition technology is finalized. Like SketchWizard, it does capture end-user interactions for analysis both by interface designers and recognition technology developers. However, NEIMO does not address the central problem of bringing designers and end users into the early stages of a pen-based UI design process.

In other application domains, several tools have addressed this problem by allowing designers to construct early-stage Wizard of Oz prototypes. SUEDE [15] is a prototyping tool for speech-based UIs that provided much of the inspiration for our work. Like SketchWizard, SUEDE provides an interface for specifying UIs that is very accessible to designers, allowing them to explore designs before a recognition technology is chosen. SUEDE also captures end-user actions during tests and provides an interface for analyzing this test data, as does SketchWizard.

Inspired by SUEDE, CrossWeaver [33] supports early-stage prototyping of multimodal interfaces. With CrossWeaver, designers create storyboards, end users can execute prototypes of these storyboards to give feedback on a design, and test data is analyzed with CrossWeaver's analysis tool. Pens are a supported input mode, and designers have the option of recognizing single-stroke pen gestures through Wizard of Oz. However, CrossWeaver does not allow end users to manipulate their own input, and it is therefore incapable of simulating the rich ink transformations that are possible in SketchWizard prototypes.

Ozlab [27] is a tool for prototyping interactions in traditional graphical user interfaces through Wizard of Oz. It was built on top of Macromedia Director to make it accessible to designers. Ozlab prototypes can allow a wizard to simulate some continuous interactions, such as dragging objects, but they cannot support user-created content as SketchWizard prototypes can. Furthermore, Ozlab does not capture end-user interactions during a test.

Topiary [20], DART [8,23], and BrickRoad [22] are all Wizard of Oz prototyping tools for location-enhanced applications. All are targeted at designers, either providing their own interface for constructing prototypes (Topiary and BrickRoad), or building on top of Director (DART). In addition, DART captures sensor data during tests in order to refine system behavior.

None of these systems provide the powerful, run-time editing capabilities that are needed to construct early-stage Wizard of Oz prototypes of pen-based user interfaces. All but Brick-Road require very detailed specifications before any simulation can be run, and all severely constrain the input language of interactions. Pen-based interfaces have such a broad input space that the input language cannot be easily defined. And as Li and colleagues point out, flexibility is particularly important in the earliest stages of prototyping [21].

## CONCLUSIONS AND FUTURE WORK

In this paper we have argued that Wizard of Oz testing of pen-based user interfaces is useful and feasible even in the absence of a working system. We have presented two sets of studies to demonstrate this: one conducted during a pen-based UI design process, and another in a laboratory setting. The system we produced, SketchWizard, enables designers with no programming skill to produce early-stage Wizard of Oz prototypes of pen-based UIs, allowing them to participate more fully in the design process. We have also presented several features of this prototyping tool that enable fast responses to end-user input during tests.

SketchWizard prototypes consist of a drawing canvas that is shared between an end user and a wizard, with simple drawing and selection tools for the end user and more extensive editing tools for the wizard. SketchWizard prototypes can elicit feedback on a design much more easily than a paper prototype can because wizards can quickly execute detailed transformations of end-user pen strokes and sketches. In addition, end-user interactions with SketchWizard prototypes can be saved for analysis by designers or developers of recognition technology.

To facilitate the process of transforming end-user input, SketchWizard has several unique features. Wizard edits can be done either in Real-Time Mode or in Manual Commit Mode. A list of views allows wizards to switch an end user's view between multiple displays of information. An event stream captures end-user actions, such as gestures or button presses, in case the wizard misses them while focusing somewhere else. Paste buttons allow complex drawings to be placed on the canvas quickly in response to end-user input. Finally, a scratch pad is provided for preparing interface elements without danger of accidentally updating the end user's view or blocking interaction.

The first study we conducted was an early feasibility study in which SketchWizard was successfully integrated into the early stages of a pen-based UI's design process. The second was a laboratory study in which real designers used SketchWizard to prototype interactions in existing pen-based UIs with end users in another room. In both evaluations, end users gave valuable feedback in spite of the delay between end-user actions and wizard updates.

In the near future, we hope to extend SketchWizard with better selection tools and keyboard shortcuts. We also hope to create new tools for faster selection of objects. In the longer term, we hope to evolve SketchWizard into a tool that will also support the later stages of pen-based interface design. Such evolution would require the ability to extend the system with additional automated behaviors, such as menus. It would also require the ability to plug in existing recognizers or other technologies to test them on end-user input. Designers could then use SketchWizard to supervise the actions of these recognizers (as suggested by Dow et al. [8]), and evolve their early designs into robust, finished systems.

## ACKNOWLEDGMENTS

This work has been supported by NSF Grants 0080562 and 0205644, and by grants from Microsoft Research and Intel Research Seattle. We also thank Patrice Simard and Mary Czerwinski for their invaluable support of this work, John Canny for his continuing support, and Yongjoon Lee for helping us execute the laboratory study.

## REFERENCES

1. Akers, D. CINCH: A cooperatively designed marking interface for 3D pathway selection. In *Proceedings of UIST '06* (October 15–18, Montreux, Switzerland), 2006, pp. 33–42.
2. Anderson, R. J., Hoyer, C., Wolfman, S. A., and Anderson, R. A study of digital ink in lecture presentation. In *Proceedings of CHI '04* (April 24–29, Vienna, Austria), 2004, pp. 567–574.
3. Bailey, B. P., and Konstan, J. A. Are informal tools better?: Comparing DEMAIS, pencil and paper, and authorware for early multimedia design. In *Proceedings of CHI '03* (April 5–10, Ft. Lauderdale, FL), 2003, pp. 313–320.
4. Balbo, S., Coutaz, J., and Salber, D. Towards automatic evaluation of multimodal user interfaces. In *Proceedings of IUI '93* (January 04–07, Orlando, FL), 1993, pp. 201–208.
5. Chandler, C. D., Lo, G., and Sinha, A. K. Multimodal theater: Extending low fidelity paper prototyping to multimodal applications. In *CHI '02 Extended Abstracts* (April 20–25, Minneapolis, MN), 2002, pp. 874–875.
6. Dahlbäck, N., Jönsson, A., and Ahrenberg, L. Wizard of Oz studies: Why and how. In *Proceedings of IUI '93* (January 4–7, Orlando, FL), 1993, pp. 193–200.
7. Davis, R. C., and Landay, J. A. Informal animation sketching: Requirements and design. In *Proceedings of 2004 AAAI Fall Symposium on Making Pen-Based Interaction Intelligent and Natural* (October 21–24, Arlington, VA), 2004, pp. 42–48.
8. Dow, S., MacIntyre, B., Lee, J., Oezbek, C., Bolter, J. D., and Gandy, M. 2005. Wizard of Oz support throughout an iterative design process. *IEEE Pervasive Computing 4*, 4 (2005), 18–26.
9. Gould, J. D., and Lewis, C. Designing for usability—Key principles and what designers think. In *Proceedings of CHI '83* (December 12–15, Boston, MA), 1983, pp. 50–53.

10. Gross, M. D., and Do, E. Y. Ambiguous intentions: A paper-like interface for creative design. In *Proceedings of UIST '96* (November 6–8, Seattle, WA), 1996, pp. 183–192.
11. Hudson, S., Fogarty, J., Atkeson, C., Avrahami, D., Forlizzi, J., Kiesler, S., Lee, J., and Yang, J. Predicting human interruptibility with sensors: A Wizard of Oz feasibility study. In *Proceedings CHI '03* (April 5–10, Ft. Lauderdale, FL), 2003, pp. 257–264.
12. Igarashi, T., Edwards, W. K., LaMarca, A., and Mynatt, E. D. An architecture for pen-based interaction on electronic whiteboards. In *Proceedings of AVI '00* (Palermo, Italy), 2000, pp. 68–75.
13. Igarashi, T., Matsuoka, S., Kawachiya, S., and Tanaka, H. Interactive beautification: A technique for rapid geometric design. In *Proceedings of UIST '97* (October 14–17, Banff, AL, Canada), 1997, pp. 105–114.
14. Kelley, J. F. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems 2*, 1 (Jan. 1984), 26–41.
15. Klemmer, S. R., Sinha, A. K., Chen, J., Landay, J. A., Aboobaker, N., and Wang, A. SUEDE: A Wizard of Oz prototyping tool for speech user interfaces. In *Proceedings of UIST '00* (November 6–8, San Diego, CA), 2000, pp. 1–10.
16. Kurtenbach, G., and Buxton, W. User learning and performance with marking menus. In *Proceedings of SIGCHI '94* (April 24–28, Boston, MA), 1994, pp. 258–264.
17. Landay, J. A., and Myers, B. A. Sketching interfaces: Toward more human interface design. *IEEE Computer 34*, 3 (March 2001), 56–64.
18. LaViola, J. J., and Zeleznik, R. C. MathPad$^2$: A system for the creation and exploration of mathematical sketches. In *ACM SIGGRAPH 2004* (August 8–12, Los Angeles, CA), 2004, pp. 432–440.
19. Li, Y., Guan, Z., Wang, H., Dai, G., and Ren, X. Structuralizing freeform notes by implicit sketch understanding. In *Proceedings of the AAAI Sketch Understanding Symposium* (March 25–27, Palo Alto, CA), 2002, pp. 91–98.
20. Li, Y., Hong, J. I., and Landay, J. A. Topiary: A tool for prototyping location-enhanced applications. In *Proceedings of UIST '04* (October 24–27, Santa Fe, NM), 2004, pp. 217–226.
21. Li, Y., Hong, J. I., and Landay, J. A. Design challenges and principles for Wizard of Oz testing of location-enhanced applications. *IEEE Pervasive Computing 6*, 2 (2007), 70–75.
22. Liu, A. L., and Li, Y. BrickRoad: A light-weight tool for spontaneous design of location-enhanced applications. *Proceedings of CHI '07* (April 28–May 3, San Jose, CA), 2007, pp. 295–298.
23. MacIntyre, B., Gandy, M., Dow, S., and Bolter, J. D. DART: A toolkit for rapid design exploration of augmented reality experiences. In *Proceedings of UIST '04* (October 24–27, Santa Fe, NM), 2004, pp. 197–206.
24. Mackay, W. E., Fayard, A., Frobert, L., and Médini, L. Reinventing the familiar: Exploring an augmented reality design space for air traffic control. In *Proceedings of CHI '98* (April 18–23, Los Angeles, CA), 1998, pp. 558–565.
25. Mankoff, J., Hudson, S. E., and Abowd, G. D. Providing integrated toolkit-level support for ambiguity in recognition-based interfaces. In *Proceedings of CHI '00* (April 1–6, The Hague, Netherlands), 2000, pp. 368–375.
26. Maulsby, D., Greenberg, S., and Mander, R. Prototyping an intelligent agent through Wizard of Oz. In *Proceedings of CHI '93* (April 24–29, Amsterdam, Netherlands), 1993, pp. 277–284.
27. Molin, L. Wizard-of-Oz prototyping for co-operative interaction design of graphical user interfaces. In *Proceedings of NordiCHI '04* (October 23–27, Tampere, Finland), 2004, pp. 425–428.
28. Moran, T. P., Chiu, P., van Melle, W., and Kurtenbach, G. Implicit structure for pen-based systems within a freeform interaction paradigm. In *Proceedings of CHI '95* (May 7–11, Denver, CO), 1995, pp. 487–494.
29. Moreno, E., MacIntyre, B., and Bolter, J. D. Alice's adventures in new media: An exploration of interactive narratives in augmented reality. In *Conference on Communication of Art, Science and Technology* (September 21–22, Bonn, Germany), 2001, pp. 149–152.
30. Newman, M. W., Lin, J., Hong, J. I., and Landay, J. A. DENIM: An informal web site design tool inspired by observations of practice. *Human-Computer Interaction*, *18*, 3 (2003), 259–324.
31. Rettig, M., Prototyping for tiny fingers. *Communications of the ACM 37*, 4 (1994), 21–27.
32. Shilman, M., Wei, Z., Raghupathy, S., Simard, P., and Jones, D. Discerning structure from freeform handwritten notes. In *Proceedings of the Seventh International IEEE Conference on Document Analysis and Recognition* (August 3–6, Washington, DC), 2003, pp. 60–65.
33. Sinha, A. K., and Landay, J. A. Capturing user tests in a multimodal, multidevice informal prototyping tool. In *Proceedings of the 5th International Conference on Multimodal Interfaces* (November 5–7, Vancouver, BC, Canada), 2003, pp. 117–124.
34. Snyder, C. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces.* Morgan Kaufmann, San Francisco, CA, 2003.
35. Voida, S., Podlaseck, M., Kjeldsen, R., and Pinhanez, C. A study on the manipulation of 2D objects in a projector/camera-based augmented reality environment. In *Proceedings of CHI '05* (April 2–7, Portland, OR), 2005, pp. 611–620.
36. 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, 1990), pp. 79–84.