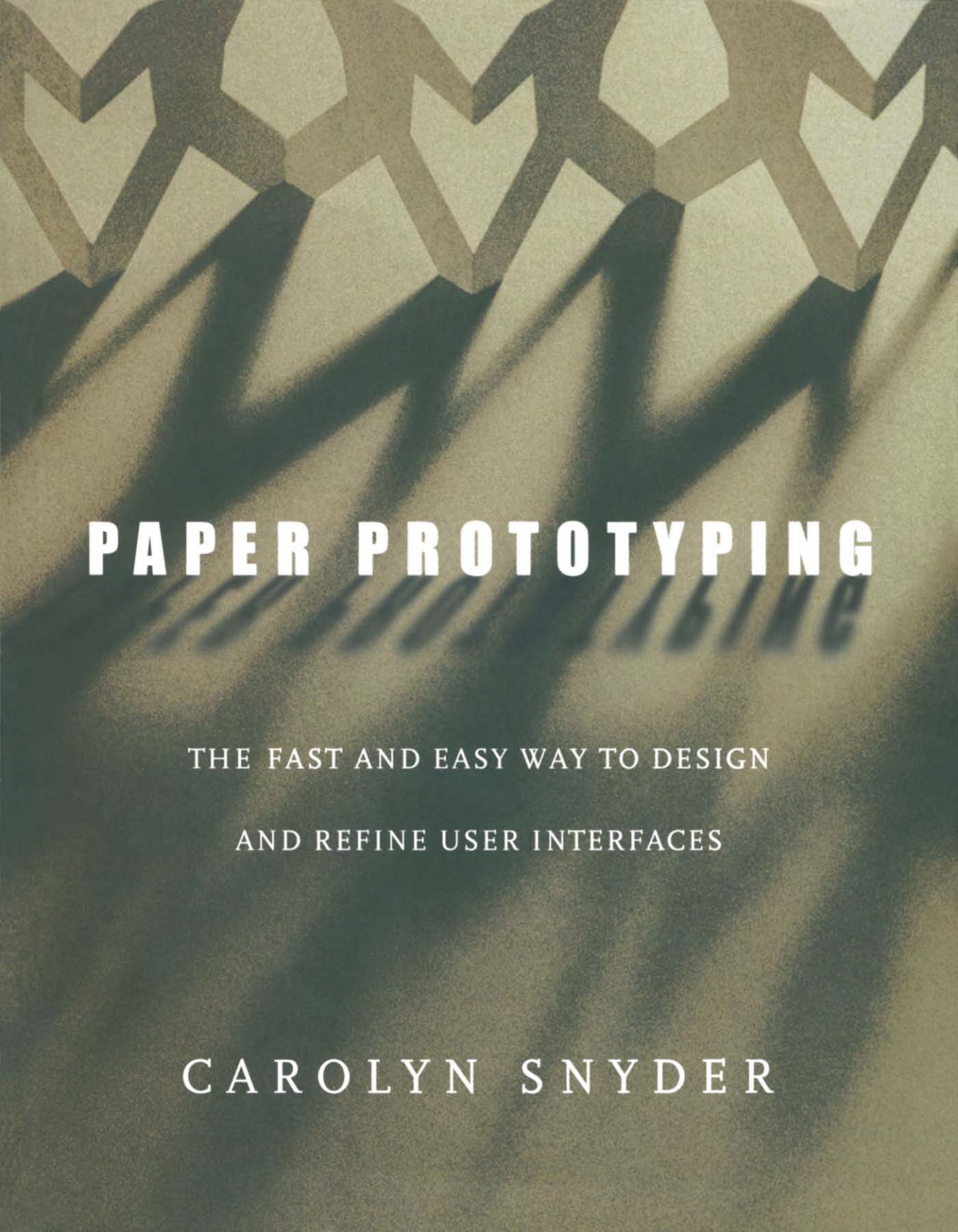

PAPER PROTOTYPING
THE FAST AND EASY WAY TO DESIGN
AND REFINE USER INTERFACES
CAROLYN SNYDER

Chapter 4

Making a Paper Prototype

I'm going to plunge into the topic of creating paper prototype widgets because most people like to see lots of examples while they're learning about paper prototyping. But in real life, before creating the paper prototype you should first define the users and tasks and then build your prototype around them. The next chapter returns to the beginning of the process and discusses all the steps in their proper order. This chapter is meant to familiarize you with how many common interface widgets (and their behaviors) can be prototyped in paper.

Paper Prototyping Materials

Although it's possible to create a paper prototype with nothing but a pen and paper, several other materials come in handy. Most of the supplies used in paper prototyping (Figure 4.1) are available from your favorite office supply store. A few items (such as the removable tape) may not be stocked in stores and are easiest to find online. See *www.paperprototyping.com* for links to the supplies mentioned in Table 4.1.

For lugging these supplies to meetings, old conference bags are ideal unless you're already using them for your groceries. You can also buy a plastic toolbox—one of my colleagues uses bright-colored ones and she notes that they attract a lot of positive attention at her company.

Figure 4.1 Paper prototyping makes use of common office supplies—those that are harder to find in stores are readily available online.

Figure 4.2 Removable tape, such as the Post-It brand shown here, is useful in paper prototyping. It comes in several widths.

Table 4.1 Office Supplies Used in Paper Prototyping

What	*Used for*	*Notes*
White poster board, about 11 × 14 inches	A fixed background upon which other paper prototype elements are placed.	Paper prototypes are usually somewhat larger than life size. I buy 11 × 14 when I can find it or cut a 14 × 22 piece in half.
Blank paper	For drawing larger prototype pieces, jotting down notes, etc.	It's okay to use a lot of paper while creating a prototype, and keeping a stack of paper on hand reminds people of that. (Bring a recycle bin too.)
Unlined index cards, 5 × 8 and 4 × 6	Useful for smaller prototype pieces: dialog boxes, pop-up messages, drop-down menus, etc.	Card stock is sturdier than regular paper and holds up better under repeated use.
Markers, pens (black and/or colored)	Hand-drawing the prototype. Choose a thick enough point so that you'll draw a bit larger than life size—regular pens may be too fine, flip chart markers are too thick, Sharpie pens are about right.	My local discount store sells sets of art markers for much less than I've found online.
Highlighter	Used with transparency and removable tape to make a highlight element.	Light-colored translucent plastic would also work.
Scissors	Used to cut screen shots into pieces, as explained in the text.	Don't run with them!
Transparent tape (Scotch tape, invisible* tape)	For attaching prototype pieces permanently, such as creating a dialog box out of two index cards. For a less permanent attachment, use removable glue.	A matte finish reduces glare, although this usually isn't a problem unless you're videotaping.

Continued

Table 4.1 Office Supplies Used in Paper Prototyping—cont'd

What	*Used For*	*Notes*
Restickable glue	Like the glue on sticky notes, it keeps elements of the prototype in place until you're ready to move them. Useful in experimenting with different layouts or if your prototype has elements that change individually, such as a Web site that uses frames.	Don't confuse it with glue marked "washable," which is not restickable. Difficult to find in stores.
Removable tape (Post-it is available in 2-line and 6-line widths)	It's opaque so you can write on it. See Figure 4.2 for an example. Use the 2-line width for edit fields (especially if the data appears elsewhere in the interface), small amounts of text that change, status line messages, list elements. The 6-line size is good for disabled buttons and quick fixes to the prototype.	A paper prototyping essential—I use enough of this stuff that I'm tempted to buy stock in 3M. Turning a corner under makes it easier to lift the tape off the paper when you want to move it elsewhere.
Transparency (overheads, acetate)	Placed over the prototype, it allows the user to "type" (hand write) data without altering the prototype. Figure 4.3 shows an example of removable tape. I use transparency when there are more than a half dozen fields to complete, otherwise I use removable tape.	Get write-on transparency rather than the stuff intended for laser printers, which is much more expensive. If you're testing in a lab with an overhead camera, transparency can cause glare—use copies of the paper forms instead.
Transparency pens, wet erase	For writing "typed" input on a piece of transparency laid on top of the prototype. Use damp paper towel or cotton swabs as an "eraser."	Permanent transparency pens work too, but since you can't erase them you'll use more sheets of transparency.

Table 4.1 Office Supplies Used in Paper Prototyping—cont'd

What	*Used For*	*Notes*
Correction fluid (Wite-Out)	For small changes to the prototype, such as a field label.	You have to let correction fluid dry before writing on it. In a usability test, I prefer to use removable tape to make quick fixes.
Fome-Cor board	For making 3D prototypes. It's polystyrene form sandwiched between two sheets of thick paper.	You'll sometimes see it spelled "foam core," although Fome-Cor is actually a brand name. Other companies make similar products.

* For those who enjoy visual humor, buy a roll of "invisible" tape, remove the tape, and hang the backing card on your bulletin board.

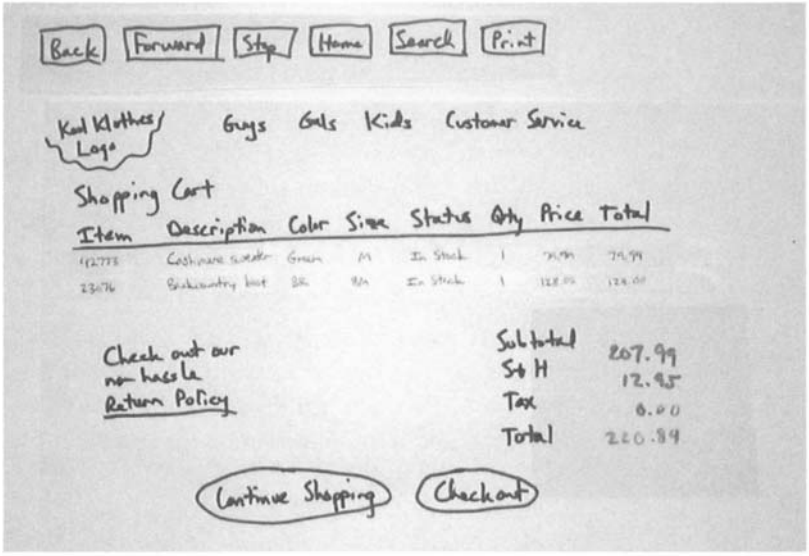


Figure 4.3 This prototype shows the use of both removable tape (line items) and transparency (the total). In this example the Computer wrote these elements on the fly to show the results of the user's actions; on screens with data entry fields the users would write on the tape or transparency themselves.

Supplies I Don't Use

There are a few items I generally don't use in the paper prototype itself, although I sometimes use them in related activities with the product team.

- **Sticky (e.g., Post-It) notes.** They don't lie flat after they've been moved a few times, so I avoid using them in prototypes. I prefer card stock and restickable glue, which essentially lets you turn anything into a sticky note. Sticky notes are often useful in group exercises (such as prioritizing the results from usability testing), however, and if they help you be creative, by all means use them.
- **Flip chart paper.** Although I suggest making paper prototypes larger than life, flip chart paper is probably too big. A colleague of mine tried it and reported that users couldn't see the whole interface without moving their heads, which introduced artificial confusion. On the other hand, flip chart paper is great for scribbling design ideas, making to-do lists for the team, and so on.
- **A ruler.** Straight lines usually aren't important for a hand-drawn paper prototype. When I was at User Interface Engineering, we used to include rulers in our paper prototyping supply kits, but we found that they encouraged people to waste time by making their prototypes overly neat. I advise using a ruler only if exact alignment is important, and if alignment is truly that important, the interface should probably be rendered using software instead of being hand-drawn.
- **Fine-tip pencils or pens.** Fine lines are difficult for observers to read, so don't use fine-tip writing instruments to create your prototype. If you feel uncomfortable working in pen because it can't be erased, pencil is okay for your earliest drafts, but then draw them in marker before testing them with users.
- **Laminator.** I've heard of people who laminated their prototype pieces to make them sturdier and/or so users could write on them with a wet-erase pen. A fine idea if you happen to have a laminator handy, but I wouldn't buy one just for this purpose. One drawback is that you can't alter a laminated piece as easily as paper—removable tape works, but correction fluid doesn't.

Creating a Background

It can be helpful to create a background to go underneath your prototype pieces. I usually use a piece of 11- × 14-inch poster board for the background because it's both sturdier and larger than a piece of regular paper. The background stays on

the table, and you put the other pieces of the prototype on top of it. Why might you use a background?

- ◈ It helps orient the users that they're looking at a representation of a computer screen or other electronic display. This isn't always necessary, but it may be helpful with less tech-savvy users.
- ◈ If controls appear on every screen, you can draw them on the background and omit them from the individual screens. (And if you change those controls, you only have to change them in one place.)
- ◈ When you're usability testing, you'll have prototype parts spread out all over the table—the background helps you keep track of what is currently visible to the user versus what you've set aside.
- ◈ It provides a reality check for screen real estate. In paper prototypes of software or Web sites I normally don't worry about exact proportions. But if an interface is composed of multiple windows and the prototype pieces spill way over the boundaries or users repeatedly move pieces on top to see the ones beneath, it's time to start worrying about screen real estate.
- ◈ If you're videotaping, you can tape the background to the table so that the prototype will stay in the camera's view.

A background is not always necessary. If you're testing a Web site and you've made screen shots that include the browser buttons, you don't need a separate background. On the other hand, if you're prototyping the display for a small-screen device, you'll probably want a background or blinder (as described later in this chapter).

Software Application Backgrounds

First, decide whether the background should represent the underlying operating system or just your application. If your application is the only thing you'll be testing, you can use its main window as your background. You might want to start from the operating system if you're testing multiple applications at once and/or want to verify that the user can launch the application. In this case, draw enough of the familiar operating system elements to orient users. For example, for a Windows desktop, I'll draw the Start button and clock at the bottom and a few common desktop icons like My Computer. Figure 4.4 shows an example of a Windows application background where the desktop elements were not deemed necessary.

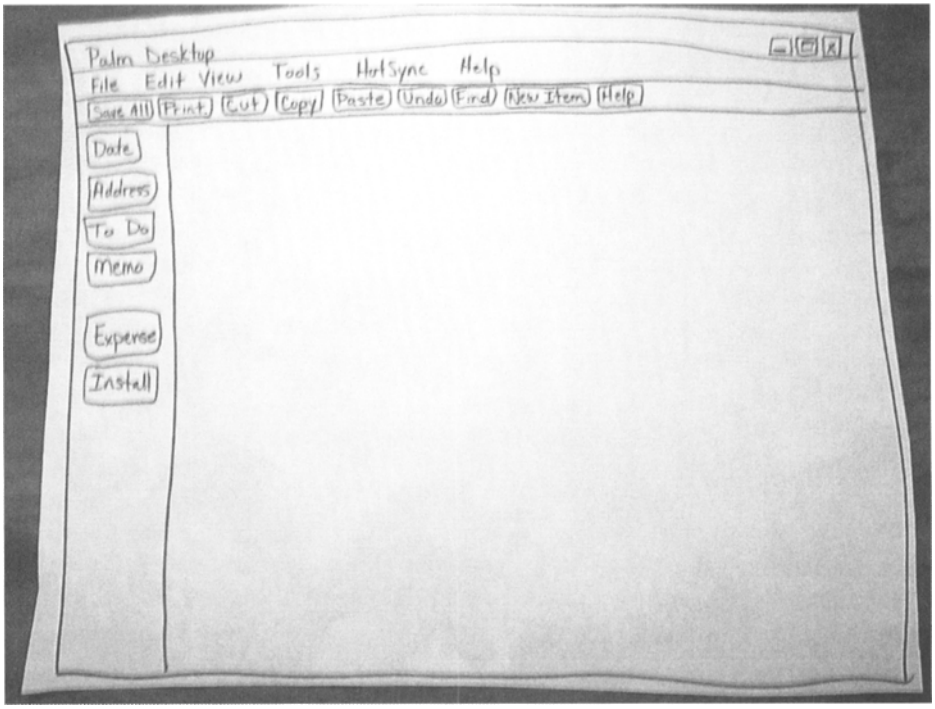


Figure 4.4 A background for a Windows application (the Palm Desktop) drawn on a large piece of paper. The menu bar, toolbar, and left-hand buttons appear on every screen. All other prototype pieces are placed on top of this background. Note that the toolbar icons have been replaced by their corresponding words.

Browser Backgrounds

For a Web browser, the version or brand isn't relevant in paper prototype testing. As a rule, you need only the most common browser buttons: Back, Forward, Home, and maybe Print or Search. In my experience, users understand these buttons just fine when they're drawn by hand; a screen shot of real browser buttons is often harder to read. Figure 4.5 shows an example of a browser background.

I usually omit the buttons for Stop and Reload—these browser controls aren't needed for paper prototype usability tests because there is nothing to "download." Similar logic applies for omitting bookmarks and the URL field—if you're testing a specific Web site you've probably made the assumption that the user got there by some means that's outside of what you're trying to test. I typically start the usability test by telling users, "You've opened your favorite Web browser and typed in *www.whatever.com.*" In my experience most users don't navigate within a

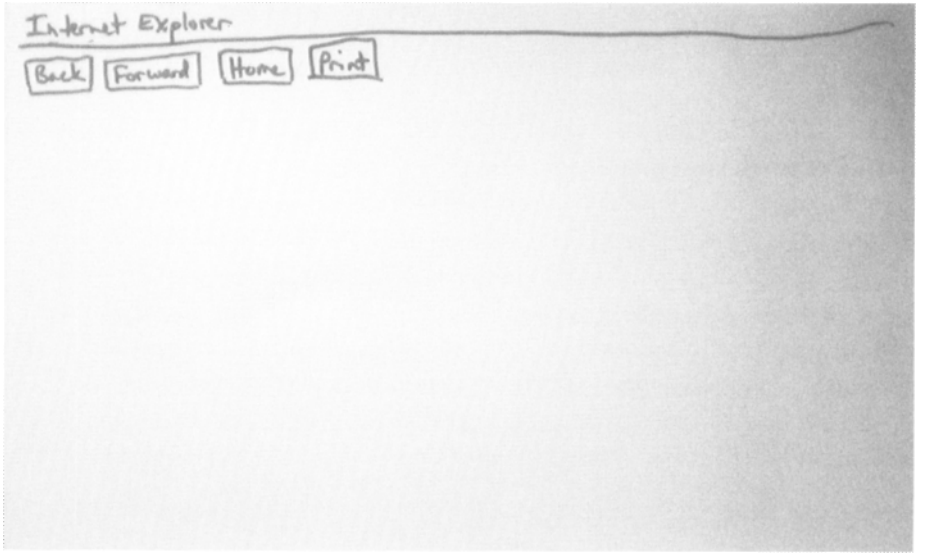


Figure 4.5 A background for testing a Web site can consist of just the most common browser buttons. The browser brand or version isn't important—this background says "Internet Explorer" simply to inform users they're looking at a Web browser.

Web site by hacking the URL field (and if we're trying to test the site's navigation, we don't want them to), so I feel pretty safe leaving it out.

> *Beware of making assumptions about how users will access your site because your assumptions may be wrong. It can be an eye-opener to start a usability test by showing users the ad or direct mail piece used to promote the site and asking them, "How would you investigate this?" One of my former clients was a start-up company that had just mailed 100,000 brochures for the purpose of driving traffic to their site and then brought me in to do some usability testing. Unfortunately, the URL was so hard to pick out of the brochure that most users (who were quite interested in the concept of the site) couldn't tell me what they'd type in to get there. Start-up companies can't afford many errors like this one; last time I visited the site, the fine print explained that the original company "ran through millions of dollars and went out of business."*

A note for both software applications and Web sites: If you're placing several prototype pieces on a background, users sometimes get confused as to whether they're looking at several different windows or one window that happens to be

made up of several pieces of paper. When I see this happen, I simply tell users, "This is all one screen," which usually alleviates the confusion.

Small-Screen Interfaces

In prototyping a small-screen device such as a personal digital assistant (PDA) or wireless phone, sometimes pixels count. When screen real estate is scarce, you may want to incorporate display constraints even for your initial prototyping efforts. It depends on what you're trying to learn from usability tests; in the early stages when you're still trying to understand users' needs or nail down the functionality, size may not be as critical as it is in the later stages of the project. Following are examples of two people who chose to do things differently, and why:

From the Field: The Importance of Screen Real Estate

"My company develops applications for PDAs such as Pocket PC and RIM, and also for wireless usage. I've found that sometimes it's important to have the look and feel of a specific environment—in our case the developers have experience with other platforms but they're not always familiar with how various widgets appear on the Palm vs. Pocket PC, etc. By creating platform-specific widgets in a graphics application, it helped them understand exactly what they needed to implement."

Phillip Hash, HiddenMind

"We used a paper-only interface back when we were still in the early stages of designing a browser/phone combination. We wanted to understand what tasks worked well in this type of interface, and to determine the appropriate set of buttons that would work for both navigation and phone functions. It would have been premature to worry about screen size too much while we were still in the exploratory phase."

Timo Jokela, formerly of Nokia

If you decide you need to worry about size constraints, one way to do it is in a graphics program: Start with a photo or screen shot of the device, create a file for each individual screen, and then print them out for testing. For example, here's

how Phillip Hash created the prototype just described: "My approach for handheld devices is to first grab screen shots of applications running on those devices, such as my Pocket PC. Then I'll open those images in Fireworks and overlay widgets on top of them." The paper prototype of the xpressa interface shown in Chapter 2 was created in a similar way. Hal Shubin started by downloading a Palm Operating System Emulator (even though he wasn't prototyping a Palm interface, it gave him something of about the right dimensions to work with). A graphic designer created a mock-up of the entire telephone and gave him back a set of GIF files. Hal used PhotoShop to create the paper prototype—the phone image was the background and he created overlays for each different screen.

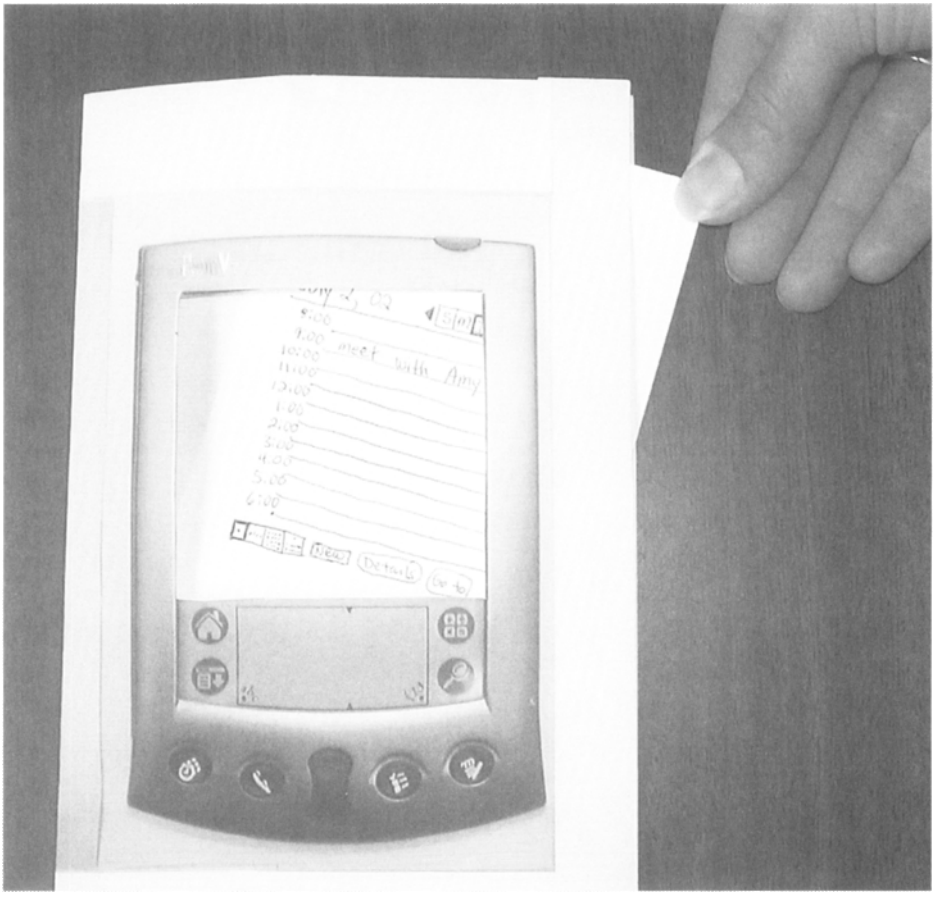


Figure 4.6 For a small-screen device where size constraints are important, you can make a blinder using a photograph of the device, somewhat larger than life. Use hand-drawn content on an appropriately sized grid for the display.

It's possible to avoid using a graphics package altogether but still keep screen constraints in mind. In his book *Handheld Usability,* Scott Weiss describes how to make a "blinder" for a small-screen device: "The key component of a prototype of a handheld device is the *blinder,* which is a sheet of card with a drawing of the hardware device with a cutout where the display would be. The size of the cutout is important because it models the amount of data that can be displayed without requiring scrolling. In order to support scrolling, the card must be larger than the drawing of the hardware device" (Weiss, 2002, p. 139) (see Figure 4.6). By using a grid (such as graph paper) for the display area, it's possible to accurately represent the number of characters that are visible but still draw them by hand. (Or, if it's faster, figure out an appropriate size font and type the data instead.)

How to Prototype Interface Widgets

Once you've created a background, the next step is to create each screen that will be placed on top of it. Figures 4.7 through 4.15 demonstrate how you can prototype the most common interface widgets.

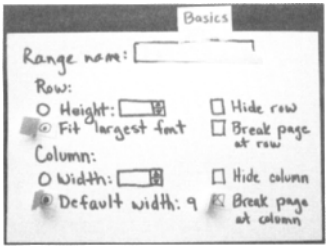


Figure 4.7 Radio buttons/checkboxes.

Buttons and checkboxes

Removable tape works well for radio buttons and checkboxes—the user touches the desired option, and the Computer moves or adds the piece of tape. Sometimes users will catch on and simply move the radio buttons themselves.

Figure 4.8 Tabbed dialog boxes.

Tabbed dialog boxes

Because a tabbed dialog box is a metaphor for a stack of index cards, prototyping them with a stack of index cards works quite well. Draw each dialog box on a separate index card, then stack them on top of each other, using removable tape to make the tabs. When the user clicks one of the tabs, simply move that card to the top of the stack. Multiple rows of tabs can get a bit cumbersome—you can still use this approach, but you'll spend more time aligning your stack of cards.

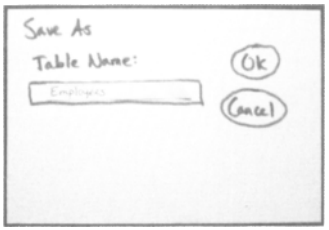


Figure 4.9 Text fields.

Text fields

Removable tape works well for text fields. The user writes on the tape, which the Computer can reuse elsewhere in the interface. In this example, the user is naming a table, so this name might appear in the list of defined tables. *Note:* For forms, it may be easier to place a piece of transparency over the entire form and have users write on that or to let them write directly on a paper copy of the form.

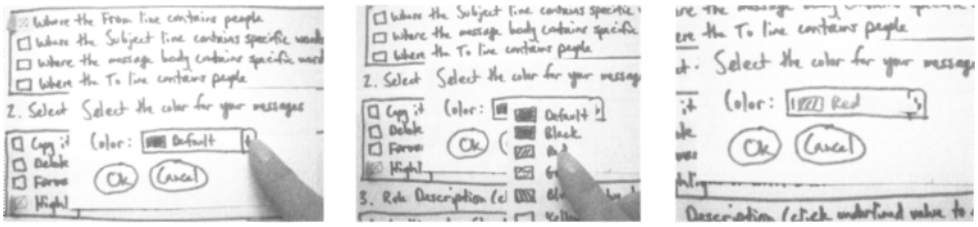
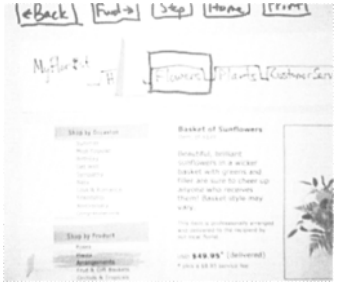


Figure 4.10 Drop-down lists.

Drop-down lists

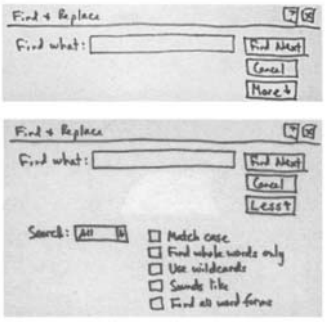
Write the default selection on the paper prototype (for example, "choose one") and put the list on a separate piece of paper. When the user clicks the down arrow, the Computer shows the list. Once the user makes a selection, the Computer writes it on a piece of removable tape and sticks it on top of the default. (The Computer may want to prepare the options that the user is likely to select ahead of time, or the Computer can create them on the fly.)



Selection bar/highlight

To show which element in a list is highlighted, make a highlight from a piece of transparency that you've colored with a light-color marker. (Some markers don't work well on transparency—the ink will puddle—but all you need is a hint of color.) In addition to the highlight at the lower left to indicate the Arrangements section, this image also shows a dark color rectangle that is used to highlight the Flowers tab.

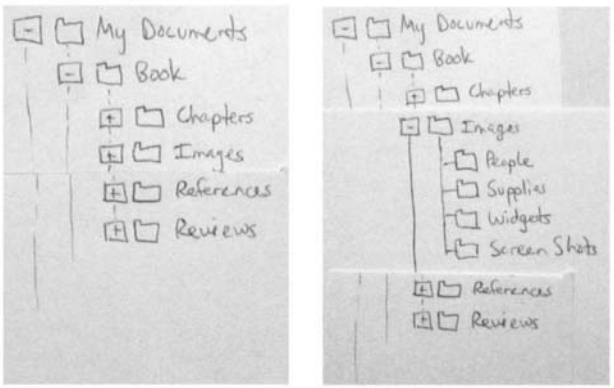
Figure 4.11 Selection bar/highlight.



Expandable dialog boxes

For a dialog box that has a More button or is otherwise expandable, you can cover the expanded part with blank piece of paper containing the button that causes it to expand. (Or you can fold the screen so that the additional options are not initially visible—a dab of restickable glue helps it lie flat.) When the user clicks the button, remove the blank paper or unfold the screen to reveal the expanded part, and change the button, in this case to Less.

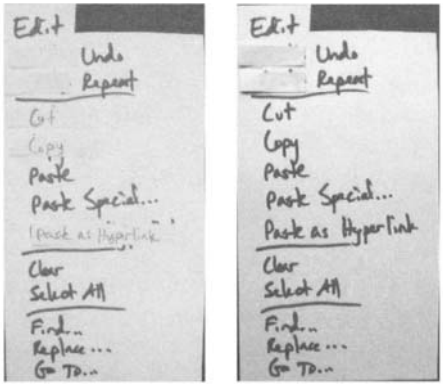
Figure 4.12 Expandable dialog boxes.



Expandable lists

Cut the list into pieces and use removable tape (or glue) so that you can separate parts of the list and add the expanded portion. You don't necessarily have to support the expansion of the entire list; after you've created the tasks and walked through them a couple times, you should have some idea of which items the user may wish to expand.

Figure 4.13 Expandable lists.



Disabled controls

If a menu option, button, or other control is initially disabled until the user does something, make a version on removable tape with a gray marker and place it over the same element in black. Once the user has done what's necessary to make the functionality available, remove the grayed version to reveal the enabled version underneath. (In the example shown, there is still tape covering the word "Can't" before Undo and Repeat.)

Figure 4.14 Disabled ("grayed-out") controls.

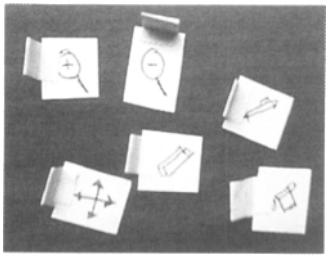


Figure 4.15 Cursors.

Cursors

I don't bother to prototype the standard arrow or I-beam cursors—this level of detail isn't needed for most paper prototype tests because the user's finger (or pen) shows where they're clicking or typing. If your application uses cursors to convey information (for example, an image editing application that displays a different cursor depending on the mode), draw them on small pieces of paper or transparency and place the current one somewhere on the interface as a visual cue for the user.

You'll probably want to have an hourglass cursor for those times when the user is waiting for the Computer. If you draw the hourglass larger than life on an index card, users will laugh and say, "Hey, just like my real computer!"

Representing the Users' Choices

You might be wondering how important it is to accurately represent the state of each radio button, list, selection, and so on. Usually it's pretty important because otherwise you're asking the users (not to mention the Computer and the observers) to remember all their choices. This cognitive effort makes the task harder and can result in artificial confusion. Sometimes it's also possible to miss subtle problems unless you have responded to all the user's actions in the exact order they happened.

Figure 4.16 shows a prototype of a screen used to create a rule for filtering email. As the user selects each Condition and Action, the Computer writes it on removable tape and places it at the bottom. Note that the removable tape initially says, "where the from line contains people." After the user clicks on the link and selects a name from the address book, the Computer places another piece of tape on top of the word people to show the selected name. In this manner the user sees the rule being built one component at a time, much as it would appear on a computer. Because many actions are possible on this screen and they can be done in any order, the pieces of removable tape at the bottom help everyone keep track of exactly what the user has done.

You might also be wondering how hard it is for the Computer to remember each correct response. The good news is that I believe it's probably easier to make and use a paper prototype than to read about how to do it. The Computer is usually someone directly involved in the design and thus knows a lot about it. As

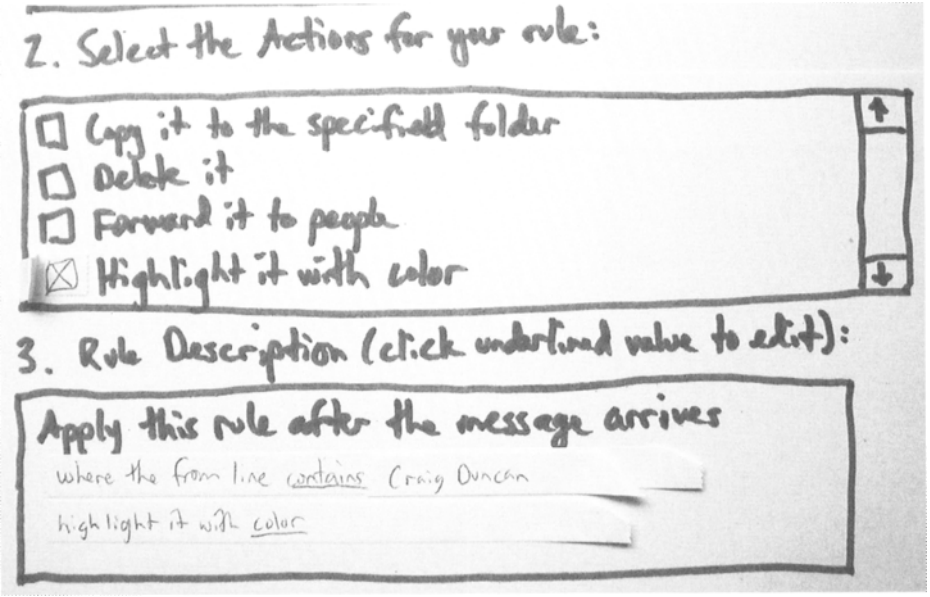


Figure 4.16 As the user specifies each component of the mail filtering rule, the Computer writes it on pieces of removable tape so that all the user's actions, in the sequence they happened, are shown in the interface.

explained in Chapter 7, the Computer practices the tasks before the first usability test, and this also helps. (As a consultant, I have helped product teams make paper prototypes of many interfaces that I initially knew nothing about. After watching a few run-throughs, I usually learn enough about how the interface works that I could be Computer if necessary.)

Hand-Drawing versus Screen Shots

Of all the examples of paper prototypes in this book so far, you may have noticed that most of them are hand-drawn and rather messy. That's deliberate; I wanted to emphasize that you don't need much artistic ability to create a paper prototype. Paper prototypes are very good at unearthing problems with concepts, terminology, workflow, content, and so forth. These types of problems are often readily apparent even without an exact visual representation of the interface. Although you may ultimately need artistic ability to create a good interface, you don't necessarily need it to create a paper prototype.

Similarly, it's usually appropriate to draw in monochrome and to fake most of the images and art—instead of the company logo, draw a box with the word "logo" in it, use a word instead of an icon, and so on. The only time I recommend against faking graphical content is when it conveys information needed for the tasks. For example, on a clothing Web site I'd use pictures of the products (perhaps cut out of a catalog) because it's important to users to see pictures of the merchandise.

If you're wondering whether it's okay to use screen shots in a paper prototype, the answer is yes. Chapter 7 provides more information about whether to use screen shots or hand-drawn paper prototypes.

Note: When I say that paper prototypes can be hand-drawn, messy, and monochrome, some people get the mistaken idea that I don't believe in the value of professional graphic design. Nothing could be further from the truth. Graphic design is both a skill and an art, and I have a great deal of respect for people who do it well. I've found that good graphic designers often embrace paper prototyping because it gives them valuable input for creating the layout and look of the interface. Usability testing will show them which design elements need to be emphasized or downplayed through the use of color, font size, white space, images, and so on. So I'm not anti-designer—in fact, people who see graphic designers as the ones who make an interface pretty are the ones who trivialize the skill.

Simulating Interaction

The paper prototyping motto is: "With a little imagination, you can simulate almost anything." There are many aspects of human-computer interaction that a human being can simulate well enough that usability testing provides useful feedback. But complex or subtle interaction usually can't be simulated perfectly; as Chapter 12 discusses in detail, this is a drawback of paper prototyping.

- **Tooltips/mouseovers.** Tell users at the start of the test that the real interface will have tooltips (a.k.a. "those little yellow boxes that pop up") to explain the icons. Tell them that if they want to see the tooltip for an icon, they can point to it and ask, "What is this?" and you'll tell them what the tooltip would say.
- **Rollover/pop-up menus.** These are conceptually similar to tooltips but are harder to simulate orally because a whole menu pops up instead of just a word or three. On a computer, I've found that what most users do is click to make the menu drop down, and then click again to make the selection. This works well

enough to show you what option the user would select, but it will also mask some of the subtle problems that can occur with rollover menus.

- **Beeps.** Simply say "beep" whenever the computer would, for example, when the user clicks outside a modal dialog box.
- **Drag & drop.** This interaction is a bit difficult to simulate perfectly. Keep in mind that many users never even try to use drag & drop in an unfamiliar interface—they use the menus instead—so it may not be something you need to worry about. But if drag & drop interaction is an integral part of your interface, ask users to specify what they're dragging and where they're dropping it. The Computer then talks through the visual changes that occur during this process. ("As soon as you move into this area, the cursor changes to this, and when you release the mouse you see . . .")
- **Right mouse menus.** Tell users at the start of the test that right mouse menus exist, and if they want to see one they should tell you that they're clicking the right mouse button, and then you'll display the menu. As with drag & drop, don't be surprised if no one tries to use right mouse menus in an unfamiliar interface.
- **Sliders, progress indicators.** I don't usually bother to make widgets for progress indicators because they can be simulated verbally, for example, by telling the user, "A progress indicator comes up. It says 20% . . . 60% . . . done." (Or "1% . . . 2% . . .") If you need a slider for some other purpose (such as a user input device), cut two slits in a piece of paper and use a strip of paper for the slider.
- **Animation and video.** Rapidly changing images can be hard to simulate. Sometimes it's easiest to describe to the users what they'd see on the screen; other times a still picture (or a series of them) will suffice. For short video clips, consider using a video player to show the video, as one product team did when they were testing their multimedia Web site.
- **Web site links.** When I first started paper prototyping Web sites, I'd take a highlighter and highlight everything that was clickable—text links, buttons, image maps, and so on. As you might imagine, the pages looked pretty garish and all that highlighting proved to be more distracting than useful. It was also extra work, so I dropped that idea. Now I tell users at the start of the test that they if they're not sure whether something is a link or just a picture, they can point to it and ask, "Is this a link?" and we'll tell them yes or no. (Or, "It wasn't originally, but apparently it should be.")

◈ **Scrolling.** On some Web pages, there is important information "below the fold" and you're interested in knowing whether users can find it. Although scrolling is a bit cumbersome to simulate with a paper prototype, there are a couple of ways to do it. The first is to fold the paper so the user initially sees only part of the page, and unfold it if they tell you they'd scroll down. This method is good enough for gathering gross data about whether users scroll on a particular page or not. If you need to do something fancier, make a cutout in a large piece of cardboard that's the size of the monitor display. Put the page underneath and slide it up or down as the user "scrolls." (Figure 4.17 shows an example.) But many development teams make a deliberate decision not to include scrolling in their tests, leaving this question to later tests with the actual design.

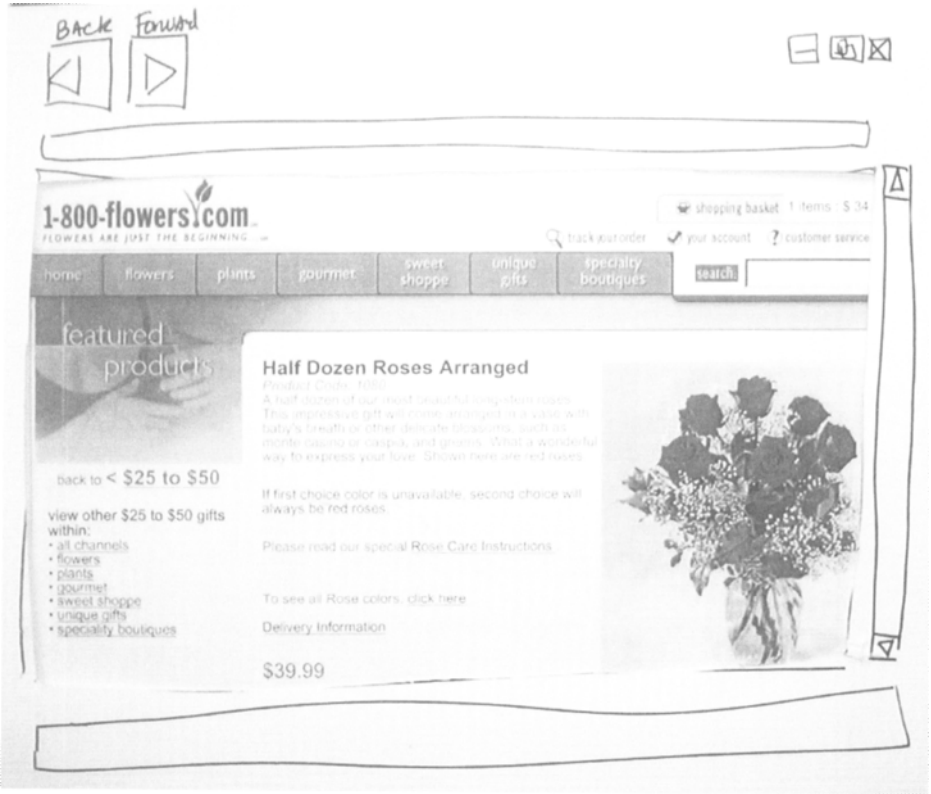


Figure 4.17 It's usually not necessary to get fancy about simulating scrolling, but if you need to do it you can use a cutout and move the scrolled piece beneath it.

Beyond the Computer Screen—Incorporating Other Elements

A user interface isn't just what appears on the computer screen—in the broad sense it includes everything that users interact with during the process of accomplishing their goals. That could include printed manuals, online help, other reference materials, hardware devices, and even human beings. And some interfaces have buttons and knobs instead of graphic user interface widgets. So let's look at how you incorporate these elements into paper prototypes.

Hardware Props

Sometimes the interface includes hardware devices in conjunction with software or a Web application. For example, portable gadgets (such as PDAs, digital cameras, and audio players) have a cable connecting them to the computer, and these devices have interfaces of their own. If a hardware device is an integral part of a product, you may want to include it in your paper prototype tests (see Figure 4.18). Obviously, users will be able to see that the device isn't connected to anything, but you can still learn a lot about how they'll interact with it. Here are some examples:

- **Tape backup system.** The development team wanted to know the point at which the user inserted or removed a tape, and when and how they labeled it. So we put a tape drive on the table next to the prototype, along with some blank tapes and a pen, and asked users to use these items as needed. One of the developers wrote "Whirrrr!" on a card to represent when the drive was making noise—funny, but also useful feedback for the users during testing.
- **Portable MP3 player.** We wanted to know when users would connect the physical device during the process of downloading music and whether they'd use it or the Web application to perform functions such as deleting individual songs.
- **Touch screen control panel.** This example is ironic because we used a computer as a *prop* for a paper prototype test. To correctly install a touch screen, the users (computer technicians) had to plug it into the same port they had chosen in the installation software we were prototyping. We had a computer and cable in the test room, and we asked users to physically plug in the cable. We noted which port they used to determine whether they'd done it correctly. In another task, we wanted users to troubleshoot a hardware problem—a

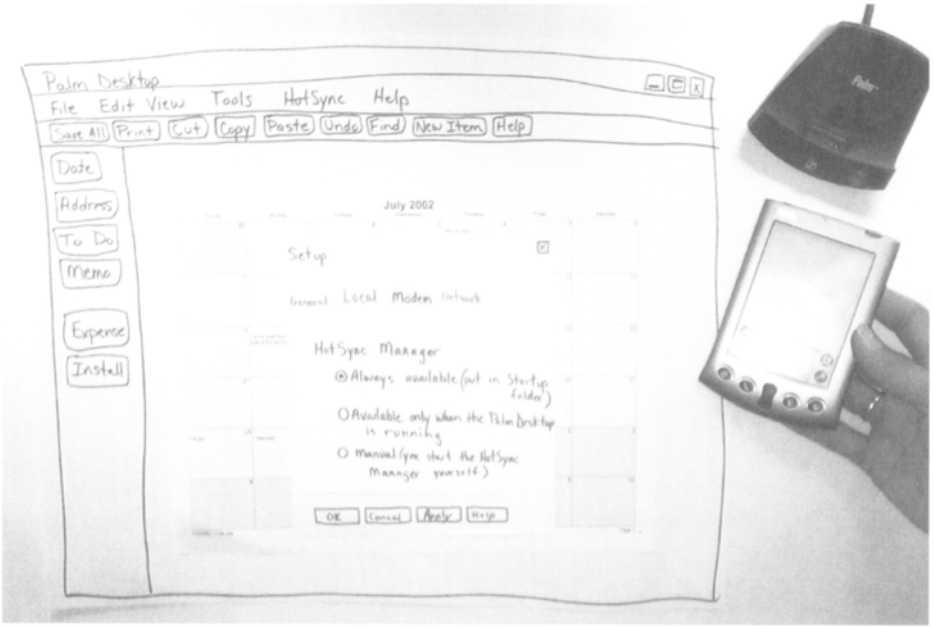


Figure 4.18 If a software interface works in conjunction with a hardware device, you can set it on the table and have users interact with it as part of the prototype.

jiggling cursor caused by a frequency conflict—that was hard to explain but easy to show. Again, we used the computer to demonstrate what the cursor problem looked like but had them solve it using the paper prototype.

One hardware prop that I don't use is a keyboard for typed input. I've never found it necessary; it's adequate to simulate most data entry by having the user write it with a pen. Sometimes it's important to know what key a user pressed—for example, Tab versus Enter to get to the next field in a form. To determine this, the test facilitator can simply ask, "What key did you press to get there?" (But once the user has given a correct answer, there's nothing further to be learned from asking this question for every field.) Unless your interface relies heavily on function keys or key combinations, trying to use a physical keyboard will only slow you down, without adding much useful information. As explained in Chapter 12, if you truly have a need to study keystroke (or mouse) level interactions, a paper prototype is probably not the best choice.

Hardware Devices

Sometimes there's no computer monitor—the user is interacting only with a piece of equipment and its set of buttons, knobs, lights, and so forth. In this case, the definition of "paper" prototyping stretches to include cardboard, Fome-Cor, or other materials used to build 3D prototypes. This technique is useful for a variety of hardware devices, including instrument panels, medical equipment, handheld devices, and consumer appliances.

In one project, researchers Säde, Nieminen, and Riihiaho (1998) were asked by a manufacturer to help design a can-recycling machine for consumers. There were two main ideas for the design—a "manual" version where the user had to insert the can horizontally a certain way so that the machine could read its bar code and an "automatic" version where the machine could read a vertically inserted can regardless of its orientation. The interface was very limited (no text, just a diagram and three indicator lights) and the design goal ambitious: All supermarket customers must be able to instantly use the machine.

After making a mock-up of each design (shown in Figure 4.19), the researchers tested both versions at a supermarket. Volunteer shoppers were given a bag full of cans and asked to recycle them. One of the researchers was obviously visible behind the machine to accept or reject the cans while another stuck colored bits of paper on the front to simulate the indicator lights. Although crude, this method of testing was sufficient for the researchers to conclude that there was a significant difference in usability: The automatic version worked well, but about half the participants would not have been able to recycle their cans with the manual concept. They also discovered some unexpected user actions, such as placing rejected cans on top of the machine. The manufacturer built a prototype of the automatic version that incorporated the findings from the mock-up, and it also performed well in usability testing. Eventually this machine went into production and is still being used today.

"Incredibly Intelligent Help"

My mentor, Jared Spool, taught me one of my favorite paper prototyping tactics, called *incredibly intelligent help.* It can be used before the online help or print documentation has been developed. This tactic is used not only to refine the help or manual but in many cases to improve the interface itself.

When users gets stuck, the facilitator prompts them to ask a question about what's confusing them. One of the product team members (designated ahead of

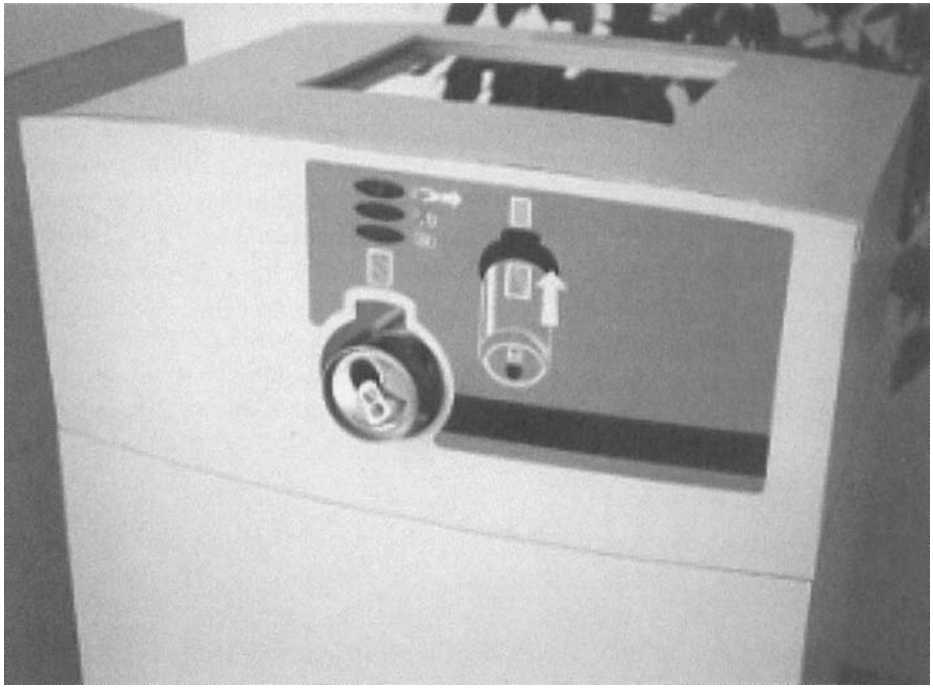


Figure 4.19 The mock-up of the manual version of the can recycling machine. In testing, a visible researcher stood behind the machine to accept or reject each can while another used colored pieces of paper to simulate the indicator lights.

time as the "Help System") gives a terse answer. The Help System should resist the temptation to answer questions that users haven't asked yet—wait and see whether the short answer solves the problem. If the users are still confused, they can ask another question and the Help System will provide a bit more detail, and so on.

The purpose of incredibly intelligent help is to find the piece of information that makes the lightbulb go on in the user's head.

The purpose of incredibly intelligent help is to find the piece of information that makes the lightbulb go on in the user's head—sometimes users only need a clue or two to get them back on track, not a long explanation. Write down the questions users ask and the explanations given. After the test, see whether the information users needed can be incorporated directly into the interface (thus avoiding the question in the first place), or if that is not practical then it should be included in the help or documentation.

Human Actors

Sometimes you may want to include humans in your paper prototype—as humans! For example, users might call an 800 number or initiate an Internet chat with an online customer service rep. To simulate this interaction, designate a member of the development team to play this role. (I don't recommend having users call the real customer support number unless you're willing to waste time on hold.) The catch is that this person shouldn't look at the interface during the conversation because in real life they wouldn't be able to see it.

One of the advantages of using human actors is that user questions emerge in a natural way, and you can determine the information required to answer them. As with incredibly intelligent help, the team should try to incorporate this information right into the interface.

Wizard of Oz Testing

Paper prototyping is similar in spirit to so-called Wizard of Oz testing. In essence, Wizard of Oz means any testing setup in which a human being acts as an intermediary between user and machine. It's often used to conduct testing with users before the underlying technology is developed. As the name implies, a human (often, although not necessarily, hidden) performs manipulations to make it appear as though something high-tech is happening. For example, several years ago I participated in a study to find out how people would give commands and dictation to a voice recognition system while editing documents. The users were not allowed to touch the computer but rather had to give all their commands orally to an experimenter sitting next to them who would then perform them literally. We got some interesting insights from these tests—for example, when scrolling up a page, the command "back up" actually means to go *down* the page!

A variation of the Wizard of Oz technique can be useful for paper prototyping. In rare cases, it's necessary to show the user something on a computer that is too difficult to simulate with paper (such as a very complicated graph) or that relies on inputs that can't be known ahead of time (such as live data). To do this, an expert sits at a computer in the same room as the users and paper prototype and enters the users' inputs from the paper prototype to show what the resulting output would be. The user still interacts exclusively with the paper prototype, but they see the results of their actions on the computer screen.

I have used this technique only rarely in paper prototyping. Because the tasks are created before the paper prototype is developed, there is usually a finite and

predictable set of results that users might see (even when you account for likely mistakes), and thus it's possible to prepare them ahead of time. But this technique may be useful for sophisticated tools where a more intuitive interface is being added on top of a hard-to-use "expert-only" system, for example, one that previously relied on command line entry.

Documentation, Help, and Training

If you're working on documentation, help, or training for an interface, testing a paper prototype will yield useful inputs to your process of deciding what really needs to be covered, and to what level of detail. In addition to the incredibly intelligent help technique, here are some tips for technical writers and trainers.

Content and Method of Access

If you happen to already have print documentation and/or online help from the previous version of the interface, sometimes you can include this material in the paper prototype test. Think of help/doc as having two aspects—*method of access* and *content.* The method of access includes whatever the user does to get to the material: click a Help button, type a term into a search engine, or open a manual to the index. The content is what they see once they get there. Depending on the stage of your development, you might have the method of access, the content, or both. (For training, it's a bit simpler. The method of access is "the person attends a class," so trainers only need to be concerned with content.)

Here's an example where we tested the help content but not the access to it. A company called Brix Networks makes Web-based tools for Internet service providers. With all the technical concepts inherent in their product, the development team anticipated that even their sophisticated target market would sometimes want clarification of terms. For example, in setting up something called a verifier group, users had to select one of five configurations from a drop-down list. One of the developers wrote a description of each configuration on a sticky note—that was the "help content" for the task. If a question came up about the options, we plopped down the sticky note on the interface (Figure 4.20). Eventually, this information was incorporated into the online help. Note that this method didn't tell us whether or how users would access the help, but it did allow us to refine the help content.

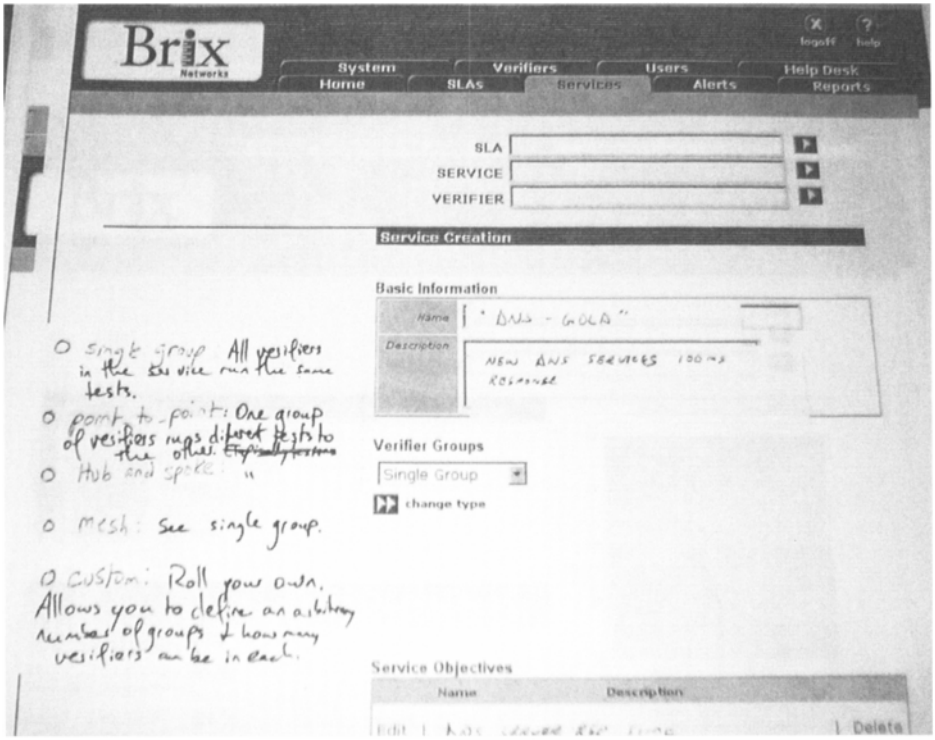


Figure 4.20 The five options for verifier groups involved technical concepts. The handwritten explanations allowed us to determine what wording would clarify the terminology. Eventually, this information was incorporated into the online help.

Preparing Material for Testing

To simulate the method of access, if you have an existing manual index that has most of the right terms in it, you (or rather, the person facilitating the usability test) could give it to the users and ask them to show you how they'd look up their question. Similarly, if users say they'd go into online help and there's no context-sensitive help topic for their current screen, the facilitator could ask them what they'd type into the search engine. In any case, you should pay attention to the users' language and the terms they naturally use—these are prime candidates for index and search terms, even if users don't use the "correct" terminology.

If you don't prepare any content ahead of time, you can use the incredibly intelligent help technique. If you do have time to draft some content, review the tasks and look for the top five or so areas where you anticipate the user running into difficulty (especially in regard to concepts as opposed to the mechanics of accomplishing a task) and start with those. There may be material from the exist-

ing interface you can scavenge, or you might want to quickly write something. You don't need to cover everything, however, and don't spend a lot of time documenting an interface you haven't tested yet. The prototype is likely to change rapidly, so wait until it has stabilized before making the documentation consistent with it.

Although I often suggest hand-drawing a paper prototype rather than creating it on a computer, I make an exception for documentation; because of the greater amount of text, it's probably faster to draft the content in a word processor and print it out. Whether handwritten or typed, avoid the temptation to do a lot of formatting—first-level headings are about the right level of detail for now.

Prioritizing the Informational Needs

If you're a writer or trainer, testing a paper prototype will show you what information users truly need, as opposed to what's nice but not necessary. You'll also get a sense of where users look for information (for example, do they notice the context-sensitive help?) Years ago I helped conduct some tests of Lotus 1-2-3 with a couple dozen spreadsheet users, and we found that no one had trouble changing font size, color, style, and so on. This was true even for people who weren't familiar with the application. Naturally, the manual had a whole section pertaining to cell styles, but not one user looked at it. On the other hand, there were some tasks that users found more difficult, and for those they did turn to the manual.

I'm of the belief that not everything needs to be documented. I like to joke that the tech writers at Lotus could have replaced all the nouns in that chapter with names of vegetables and no one would have noticed. Although that's a funny image, the serious issue is that someone had spent time documenting functionality that was self-explanatory, whereas harder-to-use functions remained undocumented. Even tech writers who believe that everything should be documented may still find it useful to know the relative priority of all the material they're responsible for. That way, they can devote more time to the things that are harder for users.

Summary

This section of the book has introduced paper prototyping—what it is and some reasons why it's useful—and provided many examples. Chapter 2 presented several case studies and their findings; this chapter has taken a widget-level view to get you thinking about how you might create a prototype of your interface. Now it's time to climb out of the details and look at the larger process of creating and testing a paper prototype.