Chapter 13

The Politics of Paper Prototyping

Paper prototyping is not a difficult technique. The real challenge often lies in convincing others to try it. It isn't too hard for most people to grasp the benefits of testing a design without having to code it first. But even people who recognize that paper prototyping is useful often have concerns about it. When I teach classes about paper prototyping, I ask participants to write on index cards all the questions that they or their colleagues have about the technique. Then we create an affinity diagram using the method described in Chapter 11. The results come out looking something like this, in order of importance:

1. **Validity.** Does paper prototyping find real problems? Does it find the same problems as testing the real interface?
2. **Bias.** Does paper prototyping introduce false problems? Does it change users' behavior or feedback in such a way that we can't trust the results?
3. **Professionalism.** What will others think of this technique (and us for using it)? Will the prototype be perceived as sloppy or amateurish?
4. **Resources.** Do we have time for this? Is there a payoff here, or is this just extra work? Why not just wait until the real thing is ready?

Some of these fears, especially in regard to validity and professionalism, tend to diminish once people have additional information. Other issues have more depth to them and merit a closer look. The material in this chapter provides a framework for discussing these questions with your team. I end this chapter with some tips for dealing with those who remain skeptical.

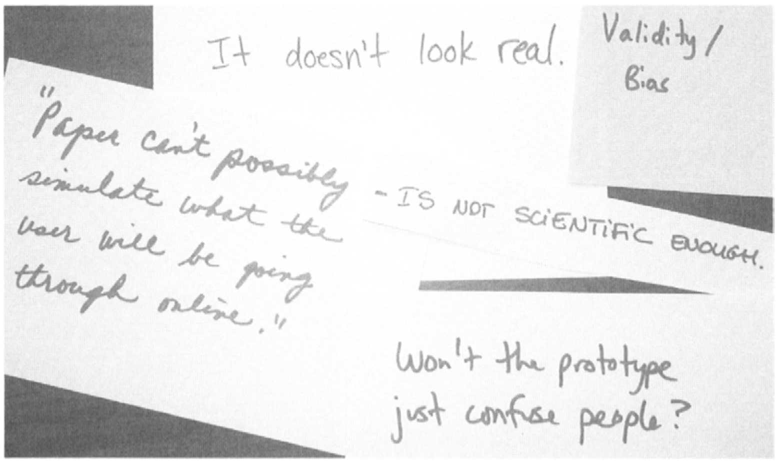
Validity

Validity is probably the biggest concern people have—they're afraid that because a paper prototype doesn't look realistic, the results from usability testing it aren't realistic either. I'm often asked whether paper prototyping finds important problems. Although this is an area where more research would be helpful, the evidence I've seen supports the generalization that paper prototypes will find the same number and types of problems as a more high-tech method. (I discussed some exceptions in the previous chapter.)

Research

Some people may be convinced of paper prototyping's legitimacy simply by virtue of the fact that there's a book on the subject. (I wish!) Other people may be swayed by the case studies and examples that I have liberally sprinkled throughout this book, especially because they comprise the experience of people at many different companies. But the more scientifically minded among your colleagues may hunger for something more substantial than anecdotal evidence. They'll identify themselves by two words: "Prove it."

I'm not a scientist, and you probably aren't either. There is a vast body of HCI (human-computer interaction) literature, some of which is relevant to paper prototyping. Following I provide information from several published papers pertaining to paper prototyping, and there are more in the References section. I'm not attempting to be exhaustive; I'm assuming that you are a practitioner who doesn't



It doesn't look real.

Validity / Bias

"Paper can't possibly simulate what the user will be going through online."

- IS NOT SCIENTIFIC ENOUGH.

Won't the prototype just confuse people?

need a lot of depth on the research, just the reassurance that there is some legitimate basis to paper prototyping.

Although I'm not a scientist, I do respect science. It's quite difficult to design a solid experiment, especially for something that's vague and hard to quantify, like "usability problems." I have just enough knowledge of experimental design to know that I'm not qualified to analyze the strengths and weaknesses of the following studies from a methodological standpoint. I can, however, attest that their findings ring true from the perspective of someone who's done a lot of paper prototyping. Without further ado, here are abstracts from some research studies that pertain to paper prototyping's validity.

Virzi, Sokolov, and Karis (1996)

Abstract: "In two experiments, each using a different product (either a CD-ROM based electronic book or an interactive voice response system), we compared the usability problems uncovered using low- and high-fidelity prototypes. One group of subjects performed a series of tasks using a paper-based low-fidelity prototype, while another performed the same tasks using either a high-fidelity prototype or the actual product. In both experiments, **substantially the same sets of usability problems were found in the low- and high-fidelity conditions** [emphasis added]. Moreover, there was a significant correlation between the proportion of subjects detecting particular problems in the low- and high-fidelity groups. In other words, individual problems were detected by a similar proportion of subjects in both the low- and high-fidelity conditions. We conclude that the use of low-fidelity prototypes can be effective throughout the product development cycle, not just during the initial stages of design."

The researchers also write: "Both techniques uncovered the same usability problems for the most part, and at about the same level of sensitivity, as witnessed by the high positive correlations in the two studies between the proportion of low- and high-fidelity subjects finding particular problems. That is, when a problem was found by a high proportion of subjects in the high-fidelity group, it also tended to be found by a high proportion in the low-fidelity group. However, some problems that were found in the high-fidelity condition were not found in the low-fidelity condition, and vice versa. This is true for both studies."

In addition to their published findings, Bob Virzi told me about some further exploratory analysis of their data. They compared the high-fi prototype to the low-fi one in several ways: the number of successfully completed tasks, the average time on a task, and the number of steps taken. These three measures seemed to suggest few differences between the two kinds of prototypes, although this conclusion requires additional scrutiny before being granted the same scientific weight as the published paper. In other words, users were able to do the same tasks, in about the same number of steps, regardless of the kind of prototype. There was a greater variability in task times, but the relative relationships were preserved—if task 6 took longer than task 1 in the hi-fi prototype, those two tasks showed the same relationship in the low-fi prototype. From a practical perspective, these results imply that if something is hard to do with the paper prototype, it's probably hard to do with the real interface as well.

Catani and Biers (1998)

Abstract: "This study investigated the effect of prototype fidelity on the information obtained from a usability test with potential users of a software product. Users engaged in a series of structured tasks using one of three prototypes which emulated a Microsoft® Windows 95™ based library search computer system and which differed only in their fidelity (low = paper, medium = screen shots, high = interactive Visual Basic prototype). **Results indicated there were no significant differences as a function of prototype in the number and severity of problems encountered nor in the subjective evaluation of the product** [emphasis added]. More importantly, there was a high degree of commonality in the specific problems uncovered by users using the three prototypes."

The authors also make an interesting conjecture: "However, in the real world, as the product moves through the development cycle, prototypes change not only in fidelity but also in content and structure. Perhaps it is this difference in content and structure which leads to the common belief that different usability problems should emerge as a function of prototype fidelity." In other words, because later-stage designs tend to be better than early-stage ones, some people might misattribute the interface problems to the prototyping method rather than the completeness of the design.

Novick (2000)

Abstract: "This paper introduces low-tech simulation as a technique for testing procedures and their documentation. The key idea is to test the interface-procedure-documentation set in the early stages of development, where changes can be still be made easily, using extremely low-cost simulation materials. Using this low-tech method, developers can test and improve the usability of documentation for user interfaces that are still being designed. An extended example, involving a new aircraft cockpit interface for text-based air-traffic control communication, presents a look at the low-tech approach in use. An evaluation of low-tech simulation shows that the approach was practical. Qualitative analysis indicates that low-tech evaluation produces results that are different from those obtained using the cognitive walkthrough for operating procedures and similar to those obtained using traditional high-tech simulation."

Although this study focused on documentation, in practice the interface and its documentation are not easily separable—Novick created a paper prototype of the interface as a means of testing its documentation. Even though only one user tested the paper prototype, Novick found that the simulation was good enough to "elicit unexpected [user] behaviors that shed light on ways to improve the procedures and their documentation." The problems found with the paper prototype were later confirmed by testing with six users on a high-tech simulator. Perhaps most interesting of all, both the paper prototype and the high-tech tests found some problems that had not been predicted by the six analysts who conducted the cognitive walkthrough. Novick did note some of the deficiencies of the paper prototype approach—its timing wasn't realistic, it didn't show color accurately, and it probably wouldn't scale up to an entire airplane cockpit—but concluded that the overall approach was practical.

Strictly speaking, these studies don't "prove" that a paper prototype will find the same set of problems as a more realistic one. Rather, they indicate that the sets of problems overlap to a considerable degree and there don't appear to be important differences in the problems that are found with one method versus another. However, these studies don't really scrutinize whether paper prototyping is less useful in finding certain kinds of problems, as I described in the previous chapter (although Novick seems to have theories similar to mine). I believe that such effects do exist, but research studies to date have not been designed to reveal them. As with any field of study, subsequent research should be able to fine-tune the questions under investigation and yield more detailed insights into exactly what paper prototyping is and isn't good for. As it stands, the evidence suggests that paper prototyping is as valid a technique for uncovering problems in an interface as usability testing the real thing.

Case Studies

The problem with finding real-life examples of paper prototyping's validity is that there's usually no point of comparison—because companies are busy trying to get products out the door, they don't have the luxury of conducting usability studies with a paper prototype and comparing the results to those obtained from testing the real interface. The From the Field box by M. David Orr (on p. 291) provides an interesting anecdotal example of someone who found himself in a position to do such a comparison.

Here's a slightly more scientific example: Chapter 4 described a study where researchers Säde, Nieminen, and Riihiaho (1998) tested a 3D mock-up of a can-recycling machine. Later in the study they tested a functional prototype with a larger number of users (50 instead of 10) and found no statistically significant differences in the severity of problems encountered in using the interface—the mock-up had accurately predicted that most users would have little or no difficulty with the "automatic" version of interface. (Note that the functional prototype had some improvements over the 3D mock-up, so this is not entirely an apples-to-apples comparison, but at least they didn't find any worms in their apple.)

This is just one example—as shown in the References section, there are many papers and articles about real-world experiences with paper prototypes. In most cases there is no point of comparison because paper prototyping was the only method used, but I have yet to find an article that concludes, "We tried paper prototyping, and it was a waste of time."

Summary: Does Paper Prototyping Find Real Problems?

As you can see, the evidence of paper prototyping's validity ranges from scientific to anecdotal, and I'll add my own experience to the latter. Over a 10-year period, I have conducted several hundred usability tests of software and Web sites, at least 100 using paper prototypes. At the end of each project, I meet with the development team to categorize and prioritize the problems, then I write a report. When I go back and reread those reports, I don't see any systematic differences in the type of findings. (A better test would be to see if someone else could find a difference.) To a large extent, I believe that I have found the same kinds of problems with paper prototyping as I've found when testing the real thing.

The real question, however, is what does it take to convince *you* (or your co-workers)? Some people don't care what research says; they need to try a method and see for themselves. Other people may need far more detail about the research than I've presented in this chapter, including a scrutiny of the methods used to conduct it. Still others may make up their minds about paper prototyping based on the opinions of people they trust. Whenever questions of paper prototyping's validity arise, the best way to answer them depends on what type of person you're dealing with; I've provided you several different approaches but you may only need to use one of them.

From the Field: Comparing Paper Prototypes and Reality—An Informal Case Study

"I'm a tech writer, instructional designer, and usability specialist who works for a consulting firm. My company had been contracted to design and develop a software system for customizing documentation. Our client manufactured 2-way radios. Each customer purchased only a subset of the radio's features, so the manufacturer needed a way to generate documentation based on the selected features.

"Because the end users were internal, a number of them were involved in the design process but the manufacturer hadn't budgeted any time to conduct usability tests. Fortunately, I teach usability courses at a local university, so I had my class conduct a usability test on a software prototype. Only one user participated, but he uncovered a handful of potentially serious problems—it was hard to know where to start, there was a lack of feedback, and an important radio button had confusing options. I estimate that the prototype had taken about 100 hours to develop, and perhaps another 20 hours were needed to fix the problems before the first version of the interface was deployed. Eventually, a rev 2 version was developed, which fixed additional problems and added features.

"About a year later, I was teaching another usability workshop and I asked my class create and test a paper prototype of the original rev 1 interface. The class conducted the testing in small groups (with a student playing the role of user) and then the groups compared their findings. Even though these 'users' had a different background than the real user population, they still were able to identify most of the same problems found by the real user, including all of the serious ones. In addition, the students had several suggestions for improving the interface that (unbeknownst to them) had actually been incorporated into the rev 2 release.

"Had the paper test been done prior to programming, the first version of the product would have been much stronger, and the second version might have been unnecessary or could have focused on enhancing the feature set."

M. David Orr
(dorr20@earthlink.net)

Bias

Another common concern is whether paper prototyping might introduce usability problems where none exist. In essence, they're talking about bias. Does paper prototyping bias the results from usability testing? You betcha . . . and so does everything else.

Usability testing is an inherently unnatural (although useful) activity *that always contains multiple forms of bias.* The trick is to recognize various forms of bias that may be present and consider their effects on the data collected from usability testing. Let's first consider several forms of bias that are often present in usability tests: users, tasks, test setting, facilitator, methodology, data analysis, observers, and bugs. Last but not least, I'll zero in on paper prototyping.

Bias: Users Who Don't Fit the Profile

If a test participant doesn't fit the profile of the target market, you can get misleading results from any kind of usability testing. Let's look at two examples: subject matter expertise and familiarity with the technology.

Subject Matter Expertise

If you test with users who don't know or care enough about the subject matter, they might not reveal important problems. For example, when I tested a Web site that sells toys, one of the tasks asked users to buy a beanbag toy for a 2-year-old. Some of the users did not have children (I was testing a variety of sites in this study), and the users without children found some issues with the site's navigation and checkout process. But it was a parent who found the showstopper: "It doesn't say whether beanbags are suitable for children under 3—no way would I buy it without knowing that." People who have in-depth knowledge of a topic are more likely to realize when functionality or content are missing, problems that are often difficult for non–subject matter experts to find.

There's a flip side to this problem: Users who lack the necessary background in the subject matter may uncover "problems" that aren't really problems for the intended audience. Many professions have jargon and concepts that are incomprehensible to an outsider. For example, when I first became involved in usability, I went on a crusade to remove the word *node* from some of the applications at my company, on the theory that it was an overly geeky term. Well, I was wrong; the users, who were engineers, had no problem at all with the term *node* and would have been confused if I'd succeeded in getting it changed to something else.

Familiarity with Technology

If you are developing an interface for people who work outside of high-tech, it's very risky to usability test with tech-savvy co-workers or friends because they know too much about the ins and outs of the type of platform you're developing for. I've seen on many occasions that users who know something about Web site design can readily use things like rollover menus and multiple browser windows that often stump a less technical audience. If your test population has more technical knowledge than your typical user, you're likely to mask out a technology-related subset of problems. On the other hand, if you're developing a product for network administrators, you don't have to worry about whether they can use a mouse.

*I once received a contemptuous email from someone who completely disbelieved an article I had written called "Seven Tricks That Web Users Don't Know" (available at **www.ibm.com/developerworks/usability/library-us-tricks**). He was a system administrator in Silicon Valley. He contended that everyone he knew, including his 10-year-old daughter, was smarter than the people I'd tested with and concluded that I should "get out more" into the real world. In my reply, I explained to him that all the usability tests I conduct are with strangers who are recruited by an outside agency in accordance with the user profile determined by the client. I refrained from touching the point about which of us should get out more.*

As discussed in Chapter 5, deciding on the profile of test participants is a necessary prerequisite for usability testing. Conducting a test with a user who doesn't fit this profile carries a significant risk of causing dissent among the product team because it's unclear to what extent the test results are credible. Worst case, the team may simply dismiss the findings (perhaps correctly so) as irrelevant.

Bias: Bad Tasks

Because I believe that task design is one of the most important and difficult aspects of usability testing, I already devoted all of Chapter 6 to this topic. Tasks can be a subtle and insidious source of bias. As I've learned from my own mistakes, it's

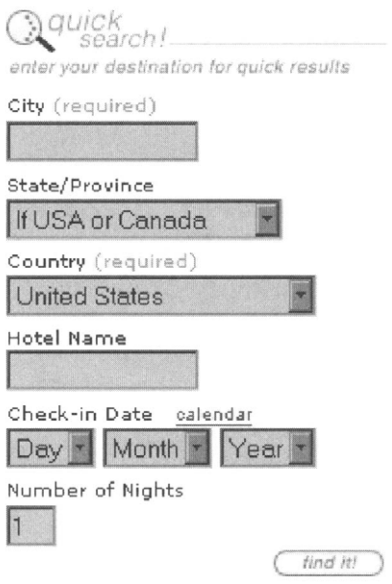


Figure 13.1 The search form used in testing a hotel reservation Web site.

easy to give unintended hints that can conceal usability problems in the interface. On the other hand, unrealistic tasks can cause users difficulty, but that difficulty may not be indicative of what they would experience in real-life use.

Sometimes the clues contained in task instructions are blatant (at least in retrospect), and other times they can be subtle. Figure 13.1 shows a search form on a travel Web site. In testing this site, we used the following task: "You and your significant other are planning a visit to Madrid, Spain. You're arriving September 9th and staying for 7 nights, departing the morning of the 16th. A friend recommended the hotel Husa Princesa. Please use the Web site to book a room." (The purpose of this task was to find out how a user would react to the error message telling them that one or more of the requested nights was unavailable.) As Table 13.1 shows, there are several subtle biases lurking in even this simple task.

This example is a fairly simple one; you can imagine how the potential for bias might increase for more complicated tasks. Unlike when you test with a non-profile user (which your team members will readily notice and complain about), bias due to tasks can lurk undetected beneath the surface of your findings. Because it's virtually impossible to eliminate bias, you should at least look at your tasks and question whether there is a bias that might interfere with the most important things that you're trying to test.

Table 13.1 Possible Sources of Bias in Task Instructions

Possible Bias	*Discussion*
The instructions reveal that Madrid is in Spain.	We did this deliberately—the team assumed users would already have done some research about their destination and we didn't want to waste test time if they didn't happen to know where Madrid was. But otherwise, this bit of information might have masked a problem with the country being required, or possibly users' need to see a map.
The instructions give the spelling of the hotel name.	Also deliberate—the team already understood the problems caused by spelling variations, and we wanted to ensure that they'd use this particular hotel because it lacked available rooms on the dates in question.
The task assumes that the user has a certain set of information when they use the site, such as exact travel dates.	An important issue—if the search requires users to enter information they don't have, they'll be stuck. But if we give them all the inputs, we won't see this problem. (In this study, we had an earlier task where we asked users to find a hotel for a vacation they were planning in real life, so we'd already gotten data about this, including the fact that people didn't always have exact dates.)
The order of the information in the task was different than the order of the fields in the search form—the dates and hotel name were reversed.	We didn't realize this. One user made an error because she followed the order in the task instructions and submitted the form after entering the dates. (Only a minor issue in this case, but it could have been important if we were testing the search form.)

Bias: Unrealistic Test Setting

Real people don't do their work in a usability lab. In the user's natural environment, there a myriad of factors that may influence how they use the interface: lighting, interruptions, other software, performance measures in their jobs that they're rewarded for, and so on. As mentioned at the end of the previous chapter,

Usability testing is an inherently unnatural activity. (Illustration by Rene Rittiner.)

there is a whole set of problems that you're unlikely to find by bringing in users; you need to go to their environment and watch them.

Here's one of my favorite examples. A few years ago, one of my colleagues consulted with a company that was working on speech recognition software. The designers were trying to improve the accuracy of the recognition and were puzzled because the beta test version would do well for users for a while but then the recognition rate would suddenly deteriorate. They couldn't reproduce the problem in their lab. When my colleague went to watch the beta users, she noticed that they closed their office doors while they were doing dictation. When interrupted by a knock or someone opening the door, users would put the system in "sleep" mode and then talk to their co-worker. In a flash of insight, my colleague realized that the speech recognition software was trying to interpret the door-knock sounds as words, and that's what made the recognition accuracy go haywire. In essence, something that was supposed to be outside the bounds of the system was interfering radically with it.

You can make your test setting more realistic (or at least understand the ways in which it's different) if you go to the effort to do so. There's a method called *contextual inquiry* where you set up site visits for the purpose of understanding what kinds of problems your users face, how they solve them, and the environment in which they do it. The insights you gain from contextual inquiry might cause you to set up your test environment differently the next time you conduct a usability study.

Note: In creating tasks and setting up the test environment, there is a balance to be struck between realism and control—to find specific things, sometimes you have to set up your test in a certain way. A paper prototype is an obviously artificial construct, but any experiment has its artificial aspects. In a good experiment, the artificial aspects are there deliberately; in a bad experiment, they're accidental.

Bias: Test Machine

Test machine bias is a subset of test environment bias. When testing on a computer, it's likely that the computer that the user is accustomed to will be different than your test machine. For example, an America Online user may navigate to Web sites using their Favorites folder, which obviously you won't have. Their method (or lack thereof) for filing bookmarks and documents will be different than yours, and so might their screen resolution, double-click speed, font size, and so on.

I've found that bias introduced by the test machine is usually less problematic than other causes, but every now and then it can bite you. In one study of a prelaunch version of a Web site, we set the default home page in the browser to the site we were testing (it was running on a development server, so we couldn't use the real URL). So this meant that the browser's Home button went to the home page of the site. We did this as a convenience for ourselves, but some users picked up on this and used the browser's Home button to get back to the home page of the site instead of the logo link intended for this purpose. In retrospect, we realized that we'd introduced a bias that made their navigation artificially easy, and we wished we'd used a bookmark instead (or typed in the URL for them, since it was too complicated to memorize easily).

You can have "test machine" kinds of problems with a paper prototype too. I've also seen users click the hand-drawn Home browser button, mistaking it for a link to the home page of the Web site. Whether paper or machine, problems of this nature are often false ones. When I facilitate usability tests, I'll step in and help users if I believe the problem they've encountered is an artifact of the test machine or paper prototype rather than the interface itself. But I'll also discuss it with the team afterward to make sure everyone agrees it was probably a false problem.

With a paper prototype, you have the option of eliminating some kinds of test machine problems, assuming they aren't relevant to what you want to test. For example, to get around the fact that the users' directory structures will all be different, you can artificially simplify your prototype so that when they go to open a file, the one you've told them to use magically appears without the need to browse for it. (You may be able to do similar things in software, but sometimes it's more trouble than it's worth.)

Bias: Facilitator

A careless facilitator can contaminate the very data he or she is collecting. Even after 10 years of experience, I sometimes catch myself asking leading questions. A facilitator's mannerisms and body language can provide clues to the user, such as nodding before the user makes a choice. (Nodding afterward, although it also constitutes a bias, is not quite as bad.) As described in Chapter 8, sometimes there are situations when a facilitator will deliberately alter users' behavior, for example, by suggesting that they look in help. To some extent these human variables are inevitable in usability testing, so it's good to be aware of how they're affecting your data. (Caveat: Self-awareness can be difficult, both cognitively and emotionally, so think of it as a process rather than an event.)

A usability test is a social setting, and that can influence users' behavior, as illustrated by the following From the Field box.

From the Field: Usability Testing in a Social Setting

"I was running people through a study on finding information on a Palm PDA. The tasks involved reading a question and looking through the text until you found the answer. One of the answers was located in the first sentence of the file. (Q: What was earnings per share? Text: Earnings per share from continuing operations were $3.05 compared to . . .) Most of the people found that number right away and made comments like 'Maybe . . . but I better scroll down just to make sure.' The times for this question were way out of line with the other tasks. In a real-world setting, the person probably would have grabbed the answer and gone with it right away. But they wanted to make sure they answered the question right. Their goal was not 'find earnings per share' but rather 'answer this question correctly for the nice experimenter,' a distinction that makes a big difference in how to interpret results."

Loel Kim and Michael Albers, the University of Memphis

(From "Web Design Issues When Searching for Information in a Small Screen Display." SIGDOC'2001. © 2001 Association for Computing Machinery, Inc. Reprinted with permission.)

In Michael's case there was some concrete evidence (the task times) that pointed to a false problem. In a paper prototype test the evidence may not be as clear, but you can still use the qualitative analysis method that I describe later in this chapter to help you decide whether you've found a valid issue or not.

Bias: Test Methodology

There are many possible ways to conduct a usability test. In the classic "think aloud" method, you ask the users to articulate what they're doing and thinking. In co-discovery, the users are working together and talking to each other. You might interact with the users a lot or very little—at one extreme, you could simply let users work uninterrupted, postponing all discussion until the end of the session. You might script every word you say or not use a script at all. You might give the users questionnaires to gather additional information. You might ask the users to review the videotape of themselves working and explain the things that confused them.

These are all different—and valid—methods of conducting usability tests. And each method carries its own risk of bias. For example, if you ask users to think aloud, it may change their behavior if it causes them to weigh their decisions more carefully. On the other hand, if you let users work uninterrupted and have them review the video later (a technique known as retrospective analysis), users may not remember perfectly what was going through their minds at the time or may even invent reasons for their behavior.

Bias: Data Analysis and Reporting

Human beings filter and prioritize information, and to some extent this is unavoidable. When we have a set of usability test observations in front of us, we continue the filtering and prioritizing process and may or may not be aware of the methods we are using to do so. For example, in the previous chapter I described several interaction problems that are hard to find with a paper prototype. Let's say that you found the "false top" problem particularly interesting because the possibility hadn't occurred to you before. Next week a friend asks you to review his Web site and you notice it has a false top. It's very likely that you will mention that problem, and perhaps even assign more importance to it than it deserves, simply because you recently heard about it and found it interesting. That's a reporting bias.

I suspect that all humans are guilty of this type of bias, and I'm dead certain that usability professionals are! The abstract of the Catani and Biers paper quoted earlier concluded by saying, "Other noteworthy results were that: (1) usability professionals (through informal heuristics analysis) and users differed in the incidence and commonality of problems identified, and (2) usability professionals did not agree in their ratings of problem severity." My colleague Rolf Molich, who coordinated some experiments known as the Comparative Usability Evaluation (CUE) studies, found much the same thing (Molich et al., 1998, 1999). It turns out that it's difficult for any two people to report the same set of usability problems or assign them the same severity. As a profession we are still working on this challenge, but in the meantime the implication is that the person conducting and/or reporting on a paper prototype study may have more effect on its findings than does the technique itself. (This is one reason why I encourage product team members to observe as many usability tests as they can—they keep me honest.)

On the other hand, a bias in analysis and reporting isn't always a bad thing. As a consultant, I sometimes deliberately investigate and report problems that I know my clients are interested in while downplaying others. For example, on one Web site I tested, the client knew that users were leaving the site after seeing the search results page. This problem was costing them money. In the usability study, I made every effort to determine the causes of this problem. Along the way I found many other problems, some of which I didn't even bother to report because fixing them would have amounted to rearranging deck chairs on the Titanic. If I had known nothing about the company's business priorities, my findings would have been less biased, but they also would have been less useful.

Bias: Observers

Awareness of being observed can change a person's behavior. I'm not a social psychologist, but I can describe some of the effects I've witnessed. Some users may be more nervous, less likely to explore, or more likely to provide favorable feedback compared with how they'd respond in real life. I've also seen users who clearly liked being "on stage" and responded in exactly the opposite manner. These are just a few of the possible effects.

As covered in Chapters 9 and 10, the things you do to prepare users for the test setting go a long way toward making them feel at ease, and proper preparation of the observers reduces the possibility that they will cause negative effects. I'm not really very concerned about the possibility of users being less willing to offer negative feedback in the presence of observers—in my experience properly briefed users understand that they provide value by speaking up when something's not working for them, and they do so.

One of the questions people sometimes have about observer bias is how aware the users are of the observers' body language—nodding, scribbling notes, frowning, and so on. Naturally, if you're sitting in a roomful of people who are blatantly reacting to everything you say or do, it's going to change your behavior. In practice, I've found that the secret is to keep the users' attention focused on the prototype. The more engaged that users are in working with the prototype (and to some extent, the Computer and facilitator), the more overt action it will take to distract their attention.

I remember one example vividly. The two users were discussing what to do next. One user said, "Maybe we should click Apply." Out of the corner of my eye, I saw all the observers (who were seated within the users' peripheral vision) nodding their heads vigorously. I had asked them not to talk, and they were following my instructions to the letter, but they were understandably rooting for their interface to do well. However, the users were completely oblivious to this clue because they were looking at the paper prototype and talking to each other. They went on to try something other than what the observers had wanted.

Bias: Bugs

Bugs are yet another form of bias. I was testing a Web application for data security where users had to create combinations of security methods, called a policy, that were then used to protect internal data resources. When users named the policy, we were curious to see whether they named it after the security methods it used (Password and Retinal Scan) or after the thing they were trying to protect (Recipe for Special Sauce). After one test, a bug somehow crept in that left us unable to delete the policy the users had created. So the users in the next test saw the policy left over from the previous test, and they gave theirs a name that was similar. This constituted a bias—we couldn't be sure whether they would have chosen the same name in the absence of seeing what the previous users did. We had to ignore the data from these users on this issue.

"Bugs" can happen in a paper prototype too, when the Computer makes an error. As with software bugs, the effect can range from benign to completely undermining a conclusion.

And Yes, the Paper Prototype Introduces Bias Too

Last but not least, let's look at how a paper prototype may change what happens in a usability test. It's possible that paper prototypes may:

1. Cause false problems.
2. Slow down the action, possibly inhibiting users from exploring the interface.
3. Affect users' impressions of the interface.

False Problems

Sometimes the paper prototype can introduce some confusion into a usability test if the users have to decipher the nature of what they're looking at. I once conducted a test of a prototype that had an expandable list where items representing different types of devices were indented in a three-level hierarchy. We had used removable tape for each line item, but due to sloppy placement it wasn't always clear what lined up with what. And each line item had a hand-drawn icon to represent the device type, but the icons looked different depending on who had done them. In testing, we saw users puzzle over the structure that the paper prototype was trying to show them. In this case, because both the alignment and icons were part of the data, it's likely that the paper prototype did introduce some degree of confusion.

Another example comes from the Pingtel case study described in Chapter 2. Hal Shubin explains: "On the paper prototype, test participants tried pressing button labels instead of the buttons. That happened in the 'soft phone' [emulator software] too. In the 3D version of the phone, there are cues not present in the paper or software versions; the screen is recessed and the buttons stand out." In both these cases, the development teams decided to downplay or dismiss these particular "problems" found in the paper prototype.

Sometimes it's unclear whether it's the medium of paper or the human Computer that's responsible for false problems. In an experiment, Uceta, Dixon, and Resnick (1998) created a hand-drawn paper prototype of an interface for ordering fast food. They scanned their sketches and made a linked PowerPoint presentation—in other words, a prototype that still looked hand-drawn but could be tested on a computer. In testing they found a few more problems with the paper version, but eventually decided the extra problems were false ones. (As you read the following explanation from their paper, note that the person they refer to as the facilitator was also the Computer.)

> "The results of the analysis indicated that the paper prototype found 55% of the usability problems with the interface while the computer based prototype found 35% of the usability problems. Given that both interfaces were exactly the same, (besides the medium of presentation), the results were surprising. However, upon further analysis, the discrepancy was attributed to the presentation environment. We suspect that the users were more frustrated at the level of fidelity and the lack of feedback with the paper-based prototype. This became evident after careful review of the videotapes, which showed that between the five participants for the paper prototype test, **there were 45 separate instances where the user was distracted in some way by the facilitator** [emphasis added]. It is ironic that this method can introduce non-existing usability problems. It seems the facilitator's presence can also create a highly artificial environment in which the user behavior can be manipulated (albeit unintentionally to the benefit or deficit of the design). Taking this condition into consideration, statistical analysis indicated that no differences were found between the sensitivity of both methods for finding usability problems."

Slower Action, Inhibiting Exploratory Behavior

Bob Virzi explained to me that he noticed some evidence of a behavioral effect in the data from the study he conducted with Sokolov and Karis. On one task that was particularly difficult, some users of the high-fidelity prototype started "thrashing"—rapidly and repeatedly taking various incorrect actions. This behavior was not in evidence with the paper prototype, where users were more deliberate. I have witnessed this effect myself on occasion, where users have said that they normally try lots of things when learning a new interface but then don't when they're testing a paper prototype. There may be a social effect here, in other words, not wanting to make the Computer do extra work. Or perhaps the Computer's slower reactions simply make rapid explorations impractical.

Whatever the cause, a paper prototype may sometimes inhibit the user from "banging on" or exploring an interface the way they normally would. However, it's difficult to predict what effect this has on usability testing results. On the one hand, if the user is working more deliberately, he or she may think through a problem more carefully and end up solving it. On the other hand, if the user is deprived of the additional information gained by experimentation, he or she may fail a task in a usability test setting but succeed in real life.

I do believe that paper prototyping can reduce the opportunities for serendipitous discovery. In tests of Web sites, I have watched users stumble upon the drop-down arrow next to the Back button in Internet Explorer (that gives access to the

10 most recent pages), comment that they've never noticed that feature before, and then go on to use it successfully. That's serendipity (and also, possibly, a bias due to the test machine if the browser they're accustomed to lacks that feature) and I don't see it when testing paper prototypes. However, I don't believe that reducing serendipity constitutes much of a problem in usability tests. In fact, if the difference between success and failure is determined by a fortunate accident, that's usually a good indication that you have plenty of bigger problems to worry about.

Liking (or Disliking) the Paper Prototype

A paper prototype can indeed affect users' perceptions of the interface, but as with timing it's hard to be certain which way the effect goes or even whether it is important. Naturally, if we ask people which prototype looks more attractive or professional, we would expect them to choose the variation that appears to be finished. In their paper, Hong, Li, Lin, and Landay (2001) confirmed that "formal representations of design were perceived to be very professional, close to finished, and less likely to change. Informal representations were perceived to be unprofessional, incomplete, and very likely to be changed."

Although it's tempting to assume that people will like the entire product better because its prototype looks more polished, that's not necessarily the case—Hong and colleagues went on to say, "Both formal and informal representations were rated similarly functional." And the Catani and Biers paper succinctly notes, "There was no significant difference on any of the 15 subjective questions as a function of prototype fidelity." Apparently, the users in these studies were able to perceive the same capabilities in an interface even when a prototype had a low-fidelity appearance.

It's also possible for people to like a paper prototype more than they will like the real product. Wiklund, Thurrott, and Dumas (1992) examined the relationship between the aesthetic refinement of a prototype and its perceived usability. They created four versions of an electronic dictionary that varied in how realistically they represented the appearance of the product. Participants rated the prototypes on a variety of scales, including ease of use and ease of learning, both before and after using them. The degree of realism didn't affect these ratings. They also had participants use the real device and provide the same ratings. But because the prototypes hadn't accurately represented the slow response times for some aspects of the real product, the usability ratings for the prototypes were more positive than those for the real thing.

There's also the question of whether you even want the users' subjective opinions. Unlike other techniques, such as focus groups or surveys, in usability testing there is more emphasis placed on what users are able to accomplish and less on their opinions. We don't just show them the interface and ask if they like it; we give them a task and watch them *do* it. Although we care what users like, we need to beware of confusing users' opinions with whether the interface actually gets the job done for them. This sort of disparity between perception and reality is quite common in usability testing. Chapter 6 described an example of a user who kept saying that he liked the Excel Function Wizard even though he wasn't able to use it without my help. This user may have been responding to the *idea* of the Function Wizard as something that would help him create formulas, but the reality was that the existing design didn't work for him.

The bottom line: Asking people if they like a paper prototype is probably not a useful exercise. It's very hard to discern exactly *what* they are liking or disliking: the appearance, the perceived capability, or the experience of using the interface. Because it is so difficult to separate these effects, I rarely ask subjective questions in usability tests, whether of paper prototypes or real products.

Examining Bias: Qualitative Analysis

The purpose of any type of usability testing is to *predict* the *possible* problems that real users will have. The question to ask yourself is, "Do we think users would have this problem in real life?" closely followed by "Why or why not?" In other words, the issue is whether one or more forms of bias may be strong enough to justify reversing the tentative conclusion that the interface has a problem.

You can't eliminate bias, so be vigilant in questioning its effects on the data you're collecting from usability testing. Although this qualitative analysis method may be overkill for most problems, it can be helpful if team members disagree about whether a legitimate problem has been found. The steps are as follows:

1. List all the **sources** of bias that might have played a role in getting a particular result from testing. Consider all the sources listed earlier, although you may decide that there are only one or two that are relevant to your situation. For example, if you found a particular problem in the only test that had a non-representative user *and* someone thinks that the paper prototype contributed to the confusion, you would have two sources to consider.

2. Determine the **direction** of each effect—does the bias act to strengthen your premise that there's a problem or weaken it? Both effects are possible. For example, if your product is intended for use by research chemists and one user who was only a chemistry student had trouble understanding some terminology, that bias acts to weaken the conclusion—more subject matter expertise might have helped the user succeed. But sometimes the presence of a bias can strengthen a conclusion rather than weaken it. For example, hand-drawn corrections to a screen shot stand out. Thus, if you make a hand-drawn correction to something you want users to notice and they still don't respond to it, you have stronger evidence that there's a problem with the interface than you'd have gotten by testing it on a computer.
3. Estimate the **magnitude** of each effect. Although this is hard to do in absolute terms (and that's why I'm calling this method a qualitative technique rather than a quantitative one), you might decide that one source of bias is relatively weak while another is strong. Try to weigh all the factors before drawing your conclusion about whether the interface has a problem.

From the Field: Bias—A Case Study

My client, Pearson Education, has a Web application called CourseCompass where college instructors create online courses to supplement their classroom activities. In a usability study, we watched users complete the registration process using a paper prototype that used screen shots printed in grayscale. In our scenario, the users were teaching an introductory psychology course and we showed them the physical textbook they'd be using. (There was also an online version of the book that accompanied the course; we had to choose the book ahead of time because its contents appeared in the paper prototype). Although the registration and course creation process were the same regardless of the subject matter, none of the users were psychology teachers—most taught English.

Once users had finished registering and defining some general information for their course, they saw a page like the one in Figure 13.2. At this point, they were supposed to access their newly defined course by clicking the Intro to Psych link to create a syllabus, assignments, and so on. Instead, users clicked the Modify button (which let them modify the information they had already entered) or the Create a Course button (which was for creating *another* course). Most users

From the Field: Bias—A Case Study—cont'd

needed a clue from the facilitator to realize that the course name, Intro to Psych, was the link to the place where they could create their course materials.

Our observation was, "Users aren't clicking the link." Our premise based on that observation was "The interface has a usability problem." In discussions with the development team, the question arose whether this problem really would have happened on a computer screen. Several interesting points arose about potential biases, some of which acted to strengthen our premise while others weakened it.

Possible Bias: Lack of color

Explanation	*Direction of Effect*	*Magnitude*
The paper prototype used screen shots printed in grayscale, so although the link was underlined, it didn't appear in blue as it would on a computer screen. This constituted a potential bias if users were less likely to notice a link that wasn't in the standard browser blue.	Weakens the premise that the interface has a problem.	Opinions differed, but the team was in agreement that the lack of color may have had some effect.

Possible Bias: Artificial data

Explanation	*Direction of Effect*	*Magnitude*
One of the developers wondered if users weren't drawn to the Intro to Psych link because they weren't psychology teachers and this wasn't the name of their real course. This was an excellent point. Artificial data may affect the way users interact with the interface. Thus, there was a potential bias due to the artificial nature of the task.	Weakens the premise that the interface has a problem.	Hard to say, but probably less than other factors. (However, the team recognized the value of using subject matter that matched what the users taught in real life and decided to do this in future tests.)

Continued

From the Field: Bias—A Case Study—cont'd

Possible Bias: Artificial motivation

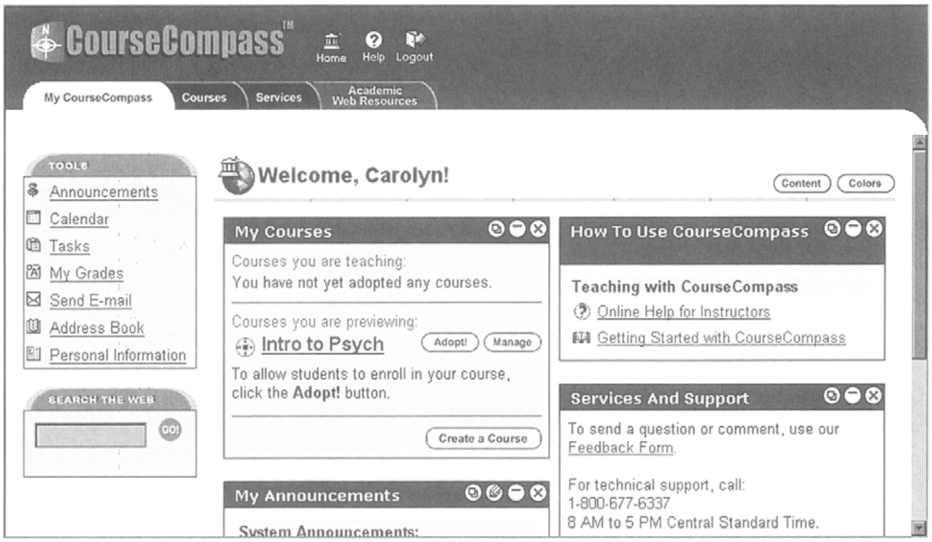
Explanation	*Direction of Effect*	*Magnitude*
The users' motivation to figure out the solution might have been stronger in real life compared with that during a usability test. Or the opposite argument—users might try harder in a usability test than in real life because they're paid to be there.	Because Pearson was concerned about adoption rates, the conservative approach was to treat any barrier to successful use as a potential problem.	Probably not as strong as other factors.

Possible Bias: Visual design

Explanation	*Direction of Effect*	*Magnitude*
We observed that users tried clicking the "Manage" and "Create a Course" buttons before clicking any of the links. The buttons seemed to draw the users' attention more strongly than the link. Some users even said that they noticed the buttons first and figured that one of them had to be the right way to proceed. (If we'd tested a hand-drawn prototype with a sloppy link and neat buttons, then that might have been yet another bias.)	Supports the premise that the interface had a problem.	Relatively strong, especially in light of what users said and did.

The team's conclusion? Yes, the lack of color in the paper prototype might have contributed to the problem, but probably not enough to reverse our conclusion that we'd found a problem with the link versus buttons. It's interesting to note that of the four sources of potential bias we discussed, only one—the lack of color—was specific to the paper prototype. The other three sources of bias—including the visual design, which the team ultimately agreed was most important—would all have been present if we'd been testing on a computer.

From the Field: Bias—A Case Study—cont'd



CourseCompass™ Home Help Logout

My CourseCompass | Courses | Services | Academic Web Resources

TOOLS
- Announcements
- Calendar
- Tasks
- My Grades
- Send E-mail
- Address Book
- Personal Information

SEARCH THE WEB GO!

Welcome, Carolyn! Content Colors

My Courses

Courses you are teaching:
You have not yet adopted any courses.

Courses you are previewing:
Intro to Psych Adopt! Manage
To allow students to enroll in your course, click the **Adopt!** button.

Create a Course

My Announcements

System Announcements:

How To Use CourseCompass

Teaching with CourseCompass
Online Help for Instructors
Getting Started with CourseCompass

Services And Support

To send a question or comment, use our Feedback Form.

For technical support, call:
1-800-677-6337
8 AM to 5 PM Central Standard Time.

Figure 13.2 There were several possible reasons why users didn't click the Intro to Psych link, and only one of them was due to the artificial look of the grayscale paper prototype.

Do you need to do this kind of detailed analysis for every problem that arises in usability testing? Of course not. In my experience, most problems found in testing paper prototypes are quite believable (sometimes blatant) and there's consensus among the product team that they're real.

But when someone asks, "But wouldn't it have been different on a computer?" my response is to ask them to explain *why* they think it might have been different. When there's a specific reason, such as "It took the Computer too long to find that screen, so the user forgot what he'd clicked," I'm inclined to agree that we might have found a false problem. But when the rationale can't be articulated beyond a vague, "I just think it might have been different," then I'm less willing to simply dismiss the problem. Similarly, an elaborate explanation of how the users were supposed to behave may indicate wishful thinking.

Whenever someone doubts the results from usability testing, it's prudent to double-check whether the user fit the profile and if there were any problems with the task or the way it was facilitated. Sometimes a seemingly free-floating anxiety about the results from a paper prototype test is valid but is caused by something other than the prototype. In other words, people can tell that something smells fishy before they've learned to identify which type of fish it is.

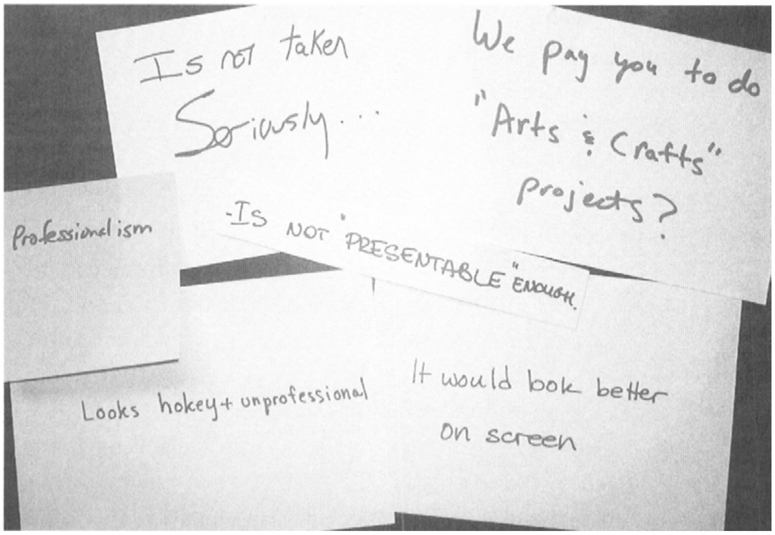
Note: By this point I may have given you so many things to worry about that you'll avoid usability testing altogether out of fear of doing it wrong! But take heart—imperfect usability studies are all we've got, and they're far better than nothing. And because good usability professionals are by nature both analytical and empathetic, help is available for the asking via professional organizations and discussion lists.

Professionalism

The thought of showing an unfinished—or even flawed—design to outsiders makes many people uncomfortable. I believe that there are some valid reasons for the fears people have, based on cultural values that many of us hold. It's important to recognize the causes of such feelings so that you can respond to them appropriately.

The Fear of Negligence

In many cultures, we face pressure for our work to be complete and error-free. As children, we were taught to cross our T's and dot our I's. In many cases there are


Is not taken Seriously...
We pay you to do "Arts & Crafts" projects?
Professionalism
-Is not "PRESENTABLE" ENOUGH.
Looks hokey + unprofessional
It would look better on screen

legal obligations to complete work, even for a task as seemingly trivial as shoveling the sidewalk in front of one's house. In a U.S. Supreme Court decision involving negligence, Justice Benjamin Cardozo wrote, "The hand, once set to the task, may not be removed with impunity."

On top of what our society already expects, many developers are perfectionists by nature and training, and their jobs reward this trait. Quality assurance concepts like "six sigma" emphasize that the tolerance for errors in a professional environment is (or at least, should be) very low.

Addressing Concerns about Professionalism

Given these pressures, it's natural for developers to be concerned that others will perceive their work as unprofessional if all they see is a sloppy and incomplete paper prototype. There are several tactics for alleviating these concerns, and all of them support the same strategy: setting expectations appropriately.

1. Position paper prototyping to the product team as a technique for gathering answers to their questions, as opposed to a way to find flaws in their work. (If usability testing is conducted in an overzealous manner, it can feel like a fault-finding crusade. Paper prototyping and usability testing should never be used to make someone look bad.) Emphasize that an essential part of usability testing is to identify what's working well, not just issues that need fixing.
2. Reassure the development team that it's absolutely appropriate for them to have a number of unanswered questions about the interface at this stage and that no one is expecting them to have a completely thought-through design. Trying things that don't work is a necessary part of innovation—before creating the first successful lightbulb, Thomas Edison claims to have discovered 10,000 ways to design a lightbulb that *didn't* work.
3. The team may be understandably concerned about seeming to promise something before knowing whether it can be delivered. Explain to both the development team and users that prototyping something does not constitute a commitment to build it—decisions and action plans come later, after the team has had a chance to digest the findings from the usability tests.
4. Set users' expectations that the design is still on the drawing board and that the final version may be considerably different than what they see. When participants arrive for a usability test, I explain that we know the design has some rough edges (sometimes literally!) but that we want to get their feedback before we've the invested the effort required to build it.

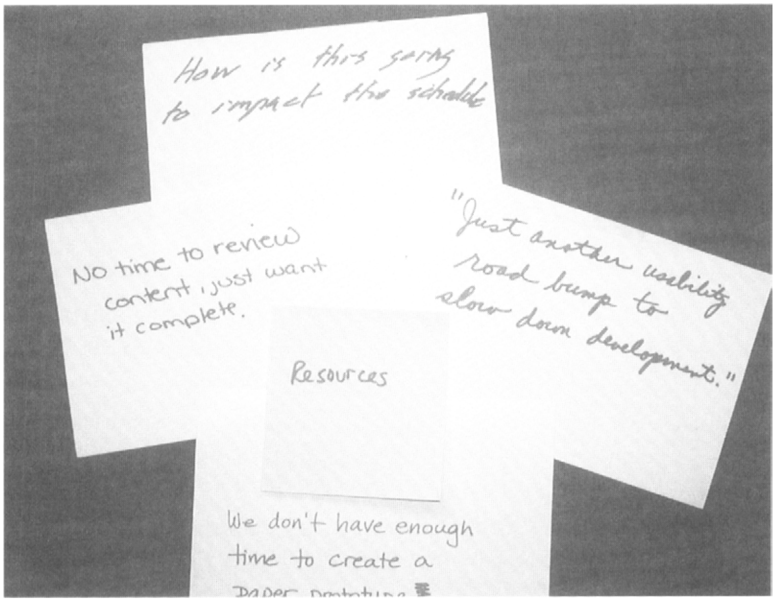
5. Last but not least, make sure the development team knows how the users will be briefed. I might tell the team, "The users will be told that the design isn't done, so they aren't expecting it to be perfect. If we find a spot where you haven't finished thinking things through, I won't let them beat you up for it. We'll just note it and move on." Remind them to resist the temptation to apologize because the unfinished nature of the design is actually a benefit and users will perceive it as such.

This may sound like a lot of bother, but it's not something you have to keep doing—once people have experienced paper prototyping for themselves, concerns about its professionalism should diminish.

Note: I'm not claiming that paper prototyping is appropriate in all professional contexts. Although there are many times when you can wear jeans, sometimes you need to don your best suit. If you're going before the board of directors to give them a progress report, I think there's some sort of unwritten law that you have to show them a PowerPoint presentation. (Although if you recall the story from Chapter 3 about Jack Zaientz and the government review committee, maybe you'll have the courage to show a few prototype screens on an overhead projector.) Likewise, if you're trying to sell something (either literally or in the sense of persuading someone), you'll want it to look slick. But for most people involved in product development, these situations are less common than the ones in which we want to collaborate with our customers and peers. For those cases, paper prototyping is sufficiently professional.

Resource Constraints

I've never worked on a project that didn't have resource constraints; every development effort has limits on the time, money, or people that can be devoted to it (usually all three). So it's natural that some people are uncomfortable about anything that sounds like additional work, even if they recognize the benefits. Chapter 5 covered the activities in a usability study and how to estimate the time you'll need. But some of your co-workers' concerns may not be addressed simply by having a schedule for all the activities. The main way to counter these kinds of concerns is to look for a point of comparison. In other words, to ask, "How will paper prototyping affect us *compared with what we're already doing?*" Let's look at three common concerns.


How is this going to impact the schedule
No time to review content, just want it complete.
"Just another usability road bump to slow down development."
Resources
We don't have enough time to create a

Won't It Result in More Work?

If you're testing to find problems, you're going to find problems, and obviously someone will have to fix them (or make a deliberate decision not to). It's hard to argue with the logic that the amount of time you'll need to spend responding to user feedback is directly proportional to the amount of feedback you ask for, which implies that paper prototyping (or any other form of usability testing) can result in additional work, at least in the short term.

The key to this question is to agree on what your time frame is. If you don't do any usability testing and problems with usability and functionality are found after release, you still have all the work associated with fixing them, but this work happens in the next release instead of the current one. (Unfortunately, this idea can be attractive to those who don't plan to remain at the company.) But you might be able to think of examples where postrelease usability problems came to light and caused havoc for technical support or were difficult to fix. In particular, look for problems of the types that paper prototypes are likely to find and make the case that it would have been possible to fix them before release. But tread carefully here—avoid sounding smug or berating others for mistakes that were "obvious" in retrospect. If possible, consider choosing an example from your own work.

How Will This Affect *Our* Development Process?

You already have a development process in place. Even if your process is an undocumented, ad hoc mess, it's still a process. *Process* simply refers to what you do, not necessarily what you'd like to do or are supposed to do! Here are some questions that'll get you thinking about how paper prototyping might alter the way in which you develop products.

- **"How long will we spend designing the interface if we don't do a paper prototype?"** One way or another, the interface has to be designed—and often redesigned. Paper prototyping may not be so much an additional activity as a different way of doing something you're already doing. (You might refer back to Chapter 3, which discussed the designing, rendering, and coding aspects of prototyping.) Don't forget to consider the time needed to document the design—at some companies, the paper prototypes themselves form the basis for interface specifications, whereas at others there's a need for more formal documentation.
- **"How long do spec reviews take?"** It can take several days to send an interface spec out for review and get people's comments back—with a paper prototype, the walkthroughs (and the usability tests) function as review activities in that they let you solicit and incorporate feedback from key stakeholders. Consider the effect that paper prototyping might have on your process for approving a design. Will it take less time for a person to approve a design if they have first attended a couple of the usability tests? (Or on the flip side, will attending tests take longer than the cursory reviews that person usually does?) Will a paper prototype help you get feedback from someone who normally doesn't read specs?
- **"At what point do we start asking for user feedback, and how long do we allocate for responding to it?"** To some extent paper prototyping may simply shift some activities that normally happen late in the project (after coding is complete) into the earlier stages (before or during the coding stage). For example, if your company plans long beta tests because that's the first time you find out what your customers *really* need, perhaps you could use a paper prototype early in the project to get customer feedback and then have a shorter beta test.

As you can see, it's hard to get an accurate answer to the question of whether paper prototyping takes extra time because there are several factors to consider. For some teams it might, whereas for others paper prototyping may partially replace activities they're already doing.

It is certainly possible to put too much time into a paper prototype—I've heard of people who spent weeks preparing their prototype and an elaborate script for usability testing. If people are concerned that paper prototyping requires an effort of this magnitude, the information in Chapter 5 about planning a usability study may be of use.

We Can Prototype Just as Well with ________

Some people may contend that a computer-based prototyping tool is just as fast and provides the same benefits. This argument is really about efficiency, although it's rarely stated that way. To consider efficiency, you have to weigh what you put into the technique with what you get out of it. Start by estimating three things:

1. **Percentage of problems you'll find with a paper prototype.** Make a list of your most important questions about the interface, then look at the previous chapter to see which questions are answerable using a paper prototype. Estimate the percentage—is it 50%? 90%?
2. **Time needed for a paper prototype.** How long would it take to create a paper prototype, to the nearest day? (Answers of 2 to 4 match my experience; answers greater than 5 are unusual and suggest that too much effort is being spent on the prototype.)
3. **Time needed for a computer-based prototype.** How long would it take to create a working mockup of the same functionality, including time to set up the test environment?

Now calculate what I call the *efficiency ratio* by dividing the percentage of problems found by the days of effort. (For simplicity, testing a working version of the interface is assumed to find 100% of the problems, although no method provides 100% accuracy. If your software prototype is incomplete in some way, you might want to estimate a lower percentage of problems found for it as well.) Compare the ratios you get—larger numbers are better because they indicate more problems found relative to the amount of effort. Finally, consider how sensitive the result is to errors in your estimates. In most cases the best course of action should be pretty clear-cut, but not always. Here are some examples.

Example 1: The cpselect tool from The MathWorks discussed in Chapter 2. Because most of their questions were about how users would work with the tool and only a minority pertained to the visuals and interaction, let's estimate that they could answer 70% of their questions with a paper prototype that would take 3 days to develop, as opposed to taking a month (20 days) to make a working version. So the calculation is as follows:

Paper 70% ÷ 3 days = 23

Working version: 100% ÷ 20 days = 5

It's easy to see why The MathWorks uses a lot of paper prototyping—it's almost a no-brainer for them given that their software is highly conceptual and complex to build. Even though I made up these estimates, you can see that the numbers would have to change a lot (for instance, building the working version in 4 days instead of 20) to yield the opposite conclusion that they'd be better off building a working version first.

Example 2: A complex information visualization interface. Say you're developing a Web site that displays real-time display of U.S. stock market data. Blocks representing companies appear in varying shades of red and green to reflect their current stock price. In this case, the paper prototype might not do a good job of portraying the subtleties of the display and interaction—assume it would answer only 60% of the team's questions. At the same time, this paper prototype might be time-consuming to develop because a display that changes would need to show many variations. Assume there's a working prototype that would need a week's worth of work to prepare for testing.

Paper 60% ÷ 4 days = 15

Working version: 100% ÷ 5 days = 20

In this case there may be a stronger argument for testing the working version, although the decision could easily go the other way if the estimates are off—if the working version ends up taking 8 days instead of 5, the team might have been better off testing with paper. The team might want to make the decision by considering the factors I discuss in the next chapter (for example, the stability of the software).

Example 3: An e-commerce site. Screens are being developed in Dreamweaver. Layout is important, so the team wants to test using screen shots. It'll take a couple of days to mock up all the necessary pages and print them out, but at that point it would only take 2 days more to link them and get the order form to work. The team's pretty comfortable that either method would answer most of their main questions; there are just a few aspects like rollover menus and some animation that won't work well in paper.

Paper 90% ÷ 2 days = 45

Working version 100% ÷ 4 days = 25

In this example the paper prototype wins at first glance, but given the small incremental amount of work to make the mocked-up site work, there might not be much risk in going that route. This is a case where probably neither decision is wrong—if no other factors argue for or against using paper, you could even flip a coin!

I find that sometimes the efficiency ratio is an illuminating tool and other times it's useless. It also ignores the benefits and drawbacks of each method (for example, paper prototypes allow for more creativity, but they don't generate code). Your answer may be quite clear in regard to one technique over the other and thus lend you confidence about the best way to proceed. But if the results aren't so clear or you have a lot of uncertainty in your estimates, the efficiency ratio may not be a helpful tool for you—feel free to ignore it. Just keep in mind that one definition of a good prototyping method is one that provides sufficient value in return for the effort you put into it.

Tips for Dealing with Skeptics

When you propose using paper prototyping in your organization, it's unlikely that everyone will embrace the technique without question. Here are some suggestions for dealing with those who remain skeptical despite your most convincing reasons.

◇ **Respect skepticism.** This one is first for a reason: Paper prototyping isn't a perfect technique, so it can actually hurt your case if you argue too vigorously in favor of it. Listen to people's concerns—try to separate the valid ones from the fears that arise from unfamiliarity with the technique. The material in this chapter will help with some of the questions and misconceptions people typically have, but it won't be enough to satisfy everyone. Skeptics have a right to hang onto their concerns until they can replace them with firsthand knowledge.

◇ **Make a sample paper prototype.** Some people resist paper prototyping simply because they can't quite picture what you're talking about. Take an existing application that is small in scope, print some screen shots, and even hand-draw a few. (This preparation should take less then an hour—if you think it'll take longer, pick something simpler.) Bring the prototype to a meeting, have a willing co-worker play user, and show people how it works.

◇ **Have influential people try it out as mock users.** Several people have told me that this tactic was successful in getting paper prototyping accepted into their

From the Field: Introducing Paper Prototyping

"Back in the mid-90s, paper prototyping wasn't widely known and there was a lot of skepticism initially. Paper prototyping wasn't very credible in the eyes of managers. So I asked some managers to participate in early test sessions and pretend to be end users. After they experienced paper prototyping as a user, they realized that it worked, and there was a lot less resistance to using it."

Timo Jokela, formerly of Nokia

organization; see, for example, the From the Field box above. (Caveat: Chapter 7 mentioned the risks in having a co-worker act as a user, so don't treat this situation the same way you would a usability test. Causing influential people to feel foolish can be a career-limiting move. One suggestion is to take more of a walkthrough approach—tell the person each step to take but have them do all the interaction with the prototype. The person will still get to see what users experience, but isn't put on the spot.)

- **Seek support from sympathetic departments.** If you're a developer or usability specialist, note that technical writers, trainers, and customer support reps are likely to be staunch supporters of usability testing—without it, usability problems become *their* problems. Invite them, and other sympathetic souls, to everything you do regarding paper prototyping and/or usability testing. Don't worry about whether you have the authority to request their presence, just make them aware of your plans and trust that they'll do their best to be involved.
- **Just do it.** Because paper prototyping can be done with as few as two people, you may not need to get a whole lot of buy-in before you start trying it. Think about whose permission you truly need, see if you can set up a couple of usability tests, and invite people to come and watch. In companies where paper prototyping is new, I've noticed that it's common for the third or fourth usability test to be much better attended than the first—once a person sees firsthand the value of the data that they get, they insist that their manager/co-workers/staff attend subsequent tests.
- **Ask for feedback.** If someone is skeptical but still agrees that paper prototyping is worth trying, promise them that afterward there will be an opportunity for everyone to give their feedback on how well the method worked, whether they want to use it again, and anything they might want to do differently. In fact, I make it a point to do this at the conclusion of every usability