Chapter 14

When to Use Paper

Chapter 12 looked at the kinds of questions that paper prototypes can (and can't) answer about your interface. In this chapter we look at circumstances of your project that can nudge you toward or away from using paper. In this chapter I assume that you've decided paper is a viable technique for what you want to learn but you're still debating whether testing a paper prototype would make sense for your project as opposed to testing some other version of your interface. If you've already made your decision about using a paper prototype, you can skip this chapter.

War Stories

In 10 years of usability testing, I've collected a number of "war stories" of usability tests (and sometimes whole usability studies) gone horribly awry, with consequences ranging from funny to frustrating. Here's a sample that will give you a sense of what can happen. All of these stories are true, although details have been omitted to protect the embarrassed.

War Story 1: Honey, I Shrunk the Functionality

This case involved a revision to an existing software product, so there was a large body of underlying code already in place. The team had been working very hard on some new functionality, and it was a focus of the usability test tasks we'd created. But 2 days before the first usability test, the latest build was so unstable that we had to drop back to a previous build that lacked the new functionality. That version was buggy as well, causing delays in the usability tests when the system

behaved strangely or crashed. Because customers were coming from out of town to participate in the tests, we didn't want to inconvenience them by rescheduling. The tests still provided useful information, but the team was conscious of how much more they would have learned if they had had all their new functionality ready in time.

War Story 2: Ready, Fire, Aim

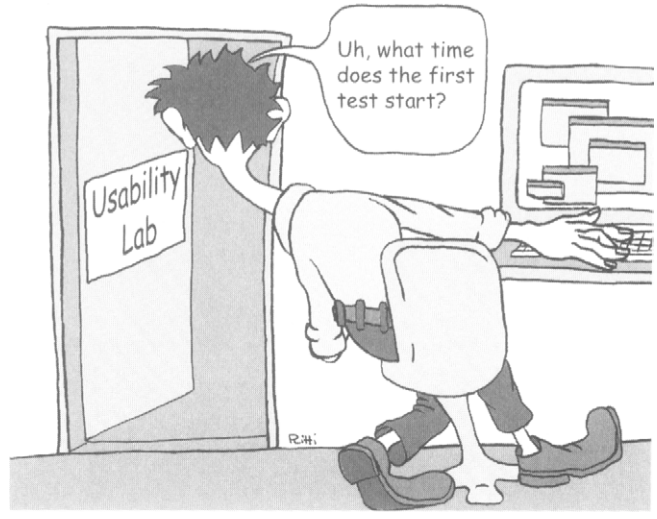
We'd successfully conducted three usability tests of an e-commerce site running off the company's development server. The planned launch was a month away, and development was proceeding at a feverish pace. Right before the fourth test, someone accidentally uploaded the wrong thing and wiped out the most recent 3 days' worth of work from the entire team. That was the end of that usability study—with so little time available and a snafu on top of it, no further usability tests could be scheduled. We were just glad we'd managed to conduct three tests before it happened.

War Story 3: It's Coming (and so Is Christmas)

Another prelaunch Web site. Development was being done by an outside company who didn't like to announce every little schedule slip, especially because they'd been working long hours to meet an aggressive launch date. At 6 pm the evening before the first day of testing, the team finally admitted that they weren't going to be ready. Testing was postponed from a Thursday to the following Tuesday. Six participants had to be rescheduled. Not all of them could attend the new date, so replacements had to be recruited on a tight schedule. (And even when things can be pulled together in time, it can be harrowing—my record for the closest I've gotten to usability testing before seeing the interface for the first time is 2 hours. Hearing a developer ask, "What time does the first test start?" is never a good sign!)

War Story 4: Network = Not Work

In this case, the problem wasn't with the Web site we were testing but with the network connection in the room we were using as a temporary usability lab. Right before the first test, the network became a "not-work." I spent the first half hour of our test session chatting with the user while several people wrestled with intransi-



This is never a good sign! (Illustration by Rene Rittiner.)

gent cables and hardware. When they were ultimately unsuccessful, we moved the test to the development manager's office. Only three observers could fit in the office—half the number that had wanted to attend the test. The really frustrating thing about this case was that, being somewhat paranoid about hardware, I had insisted on a thorough test run the day before and everything had worked flawlessly.

War Story 5: Accidental Pornography

I was testing the e-commerce site of a reputable company. The user was completing an order form and through a rather bizarre sequence of actions managed to access a porn site by accident. I was very thankful that she and I were the only ones present and that she recovered her composure (mine was pretty shaken as well) after I figured out what had happened and explained it to her.* In addition to being embarrassing, there was some legal exposure (pardon the pun) here as well—earlier in that same study, a user had arrived at the test facility without having signed the consent form. She first wanted to ask me about what kinds of sites we would be

* The sequence of events: After entering her last name, the user hit the Tab key. Instead of going to the address field, the cursor went to the URL field in the browser. The user typed the first 2 digits of the street address—19—before noticing that the cursor wasn't in the address field. She clicked in the field and resumed typing. A couple fields later, she mistakenly hit the Enter key instead of Tab, which activated the URL field, and we visited 19.com, a porn site.

visiting because she was uncomfortable with some of the material available on the Internet. In an alarmingly prophetic choice of words, I reassured her that if these sites were movies they'd all be rated G. If the porn accident had happened to this user, she could have sued me.

Paper Prototype War Stories

You may have noticed that none of these war stories feature a paper prototype. In case you're wondering where those war stories are, so am I! Honestly, I don't have any—in 10 years I have yet to experience a situation where a paper prototype prevented a usability test from taking place. (I've had to cancel paper prototype tests when users didn't show up, but that can happen in any usability test.) I've also worked with development teams who were concerned that they weren't going to be ready in time, but in the end they always were. Hmm.

My experience suggests that paper prototype tests are less vulnerable to some kinds of problems than usability tests that rely on machines and other capricious entities. But equally important, I can think of many cases where product teams tested prerelease interfaces and *didn't* have a disaster or where the benefits of paper just weren't that compelling. So let's look at various aspects of your project—the people involved, the stability of the technology, the kinds of tasks you want to test—so that you can be proactive in identifying factors that might bollix up the usability study that you're planning. If a factor doesn't seem to apply to your situation, feel free to ignore it. At the end of the chapter, I summarize these discussions in a checklist.

People and Logistics

First let's consider the people—who's on your development team, where they're located in relation to users, and how difficult your target user population is to (re)schedule.

Composition of Development Team

Paper prototyping seems to work well when the product team is large and/or interdisciplinary. Many people who have useful ideas lack the technical ability to

implement them, such as customer support reps, Marketing, QA, and writers. With a paper prototype there is no technical barrier to prevent ideas from coming from a variety of sources. Although it's possible for the designer to go to these people to get their input, it may be more efficient to gather the people together and let them contribute their expertise directly to the prototype. (Naturally, you'll want to take a look at the individuals who make up your team to decide whether this is a good thing or not!)

The converse is somewhat true—if the development team is small and very proficient with their tools, the benefits from using paper may not be as great. I've seen teams with two or three developers who could make substantial changes fairly rapidly. At one company, after each usability test the room would immediately empty as the developers rushed back to their desks to make changes before the next test. (Unfortunately, this was the same company featured in the "Ready, Fire, Aim" war story, who also proved their capability to make substantial *mistakes* on short notice!)

Location of Development Team and Users

Usability tests involve a whole cast of characters—users, the facilitator, the Computer, and observers. In a paper prototype test, you have to get the whole cast together in the same room. Unless everyone is already at one location, it can be inconvenient to reschedule a usability test. I've known cases where customers traveled considerable distances to participate in usability tests, like a guy who flew across the United States for a 2-hour usability test and went back home the same day. It's also not unusual for development team members from another city to come in for a couple days in to observe usability tests. The more people who assemble for the test, the farther away they're coming from, and the more influential they are, the more painful it is when something forces you to reschedule. But if you're developing an intranet and your users are right down the hall, postponing a usability test is no big deal.

Remote Usability Testing

It's difficult to usability test a paper prototype unless the Computer and users are in the same room. However, some of my colleagues have experimented with ways to test prototypes remotely, by preparing materials in either paper or electronic form and then walking through them with the user on the phone. See the From the Field box on p. 324 for examples.

From the Field: Remote Usability Testing

"I once did paper prototype testing over the telephone with excellent results by sending the participants a paper prototype booklet that I asked them to open only after our phone conversation began. The booklet supported the use scenario, so the participant could say 'Now I'd press the OK button,' and I could say, 'All right, now you see the screen shown on page D.' I was surprised at how well it actually worked, and it enabled us to include many more participants at next to no cost. (Note that the application we were testing was quite simple; this technique might become unwieldy with an interface containing more screens.)"

Betsy Comstock, PolyCom

"Rather than sending the papers, I created a set of prototype images and stored them on a Web server. Each image URL is encrypted so the user won't be able to guess what the next image would be by tweaking the URL on his/her browser. The users were connected via ICQ for text communication and we even managed to connect to a participant in Russia! Every time I asked the user to work on a page, I sent him the image URL by ICQ and gave verbal instructions over the phone. In that way I could guarantee that we were in sync."

Shawn Zhang, usability specialist

Remote usability testing is an intriguing idea, but it may not yield the richness of data that you get by being in the same room with users. There may be interaction drawbacks if the user has to flip pages or take other special action to get to the next screen. On the other hand, if travel isn't feasible, remote testing is much better than nothing, and it may allow the product team to get feedback from a larger number of users than they'd be able to see in person.

Rescheduling Costs

Rescheduling users can cost money. If you pay an outside agency to do user recruitment, they may charge you a fee for rescheduling the same users to a different day. They almost certainly will charge for each qualified person they recruit,

including those that couldn't be rescheduled. Some companies don't blink at wasting far larger sums than the few hundred dollars needed to reschedule a usability study, but other companies have tight budgets and throwing away money is painful.

Development Context

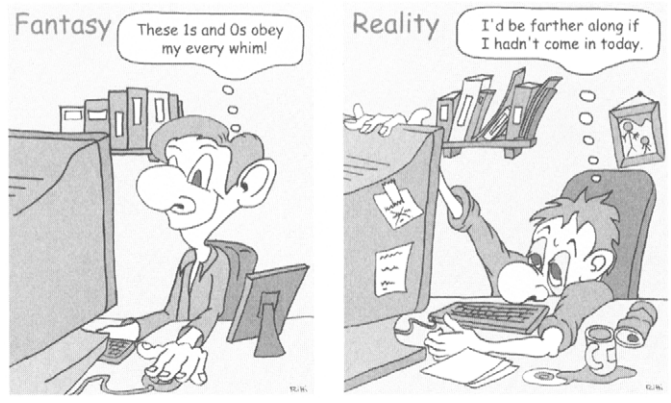
As you can tell from the war stories, a lot of things can prevent you from conducting usability tests. The factors discussed in this section will help you estimate the likelihood of a technical snafu interfering with your testing plans.

Dependence on Technology

Prerelease stuff is buggy. As Murphy's Law says, anything that can go wrong, will go wrong. This state of affairs is normal during product development, but it can wreak havoc with your usability testing plans. The more modules or applications that have to play nicely together, the more vulnerable you are to technical problems—and the effect is multiplicative. Say you have a task that utilizes four software modules. Each performs reliably 90% of the time and crashes the other 10%. Put these four modules together, and your chance of getting through the task without crashing is $0.9 \times 0.9 \times 0.9 \times 0.9$, or only about 66%. It's kind of like Murphy's Law with multiple Murphys. In contrast, paper prototypes don't rely on databases, networks, servers, or any other form of technology, which eliminates most of the things that can go wrong. I've even done paper prototyping during a power failure!

You programmers out there, I know what you're thinking: "But this is my application. I'm the one writing the code." Yes, but until you're well into the debugging stage, your code (or code written by others that yours depends on) may not be stable enough to allow users to do meaningful work without crashing. Bugs happen, and software development is sometimes a process of taking two steps forward and one back.

It is possible to conduct usability tests with buggy software, and I have done this many times. If the development team knows where the land mines are located, the facilitator can usually steer users safely around them. If the users find a land mine we didn't know about, perhaps it's not difficult to restart the system and get back to where they left off. But these situations can be nerve-wracking, not only for the users but also for the product team. As a facilitator, I sometimes find



We often aren't quite as in control of software development as we would like. (Illustration by Rene Rittiner.)

myself mentally chanting, "Stop him before he clicks Submit, stop him before . . ." Which is not so bad—it's part of the facilitator's job—but the real issue is that the observers have the same chant going through *their* heads. I've seen developers who were so focused on the survival of their interface that they didn't get much value from watching the usability tests. If you're concerned that a buggy interface will be too distracting to the product team (or upsetting to users, who may disregard assurances that it isn't their fault), consider using a paper prototype instead.

Usability Testing versus Development

I deliberately put the "versus" in the heading because sometimes that's how it seems to me—it's possible for usability testing to interfere with the development process and vice versa, especially in its later stages. The degree to which a project is vulnerable to this depends on whether there's a separate environment that can be used for testing. If the usability tests rely on the latest build on the development server, development must come to a carefully orchestrated pause so that someone submitting a "minor" change doesn't accidentally break the very thing you're trying to test. This can have a detrimental effect on schedules, unless the entire development team plans to attend the usability tests or wouldn't be submitting any changes during that time anyway.

But if you use a paper prototype, usability testing and development are completely separate and can proceed in parallel without risk of hurting each other. I've known a few development teams who chose to conduct paper prototype tests late

in the project—when a reliable version of the interface was available—simply to alleviate any risk that the development schedule would be impacted.

You might respond that a good project manager is capable of handling all these circumstances, especially with decent version control software. That's true, but not all project managers are that good—when I was a project manager my sanity was fragile enough as it was, and the need to maintain a stable environment for usability testing would have pushed me right over the edge. If your development process runs like a Swiss watch, paper prototyping may not offer you any advantages. But for those who are already logistically challenged, paper prototyping can make life easier.

Test Environment—Hardware and Facility

One way to avoid conflicts with ongoing development is to set up a separate test environment with all hardware, software, network connections, and so on needed to do the tasks. Maybe you're lucky enough to have a fully outfitted usability lab. Or your QA department or training facility may have something like this, although if they use it every day you may not be able to borrow it.

You may be thinking, "What's all the fuss about a test environment? I'll just run it on my laptop." If your interface runs in a simple and self-contained environment like that, then you're right, you don't really have this problem. But some test environments require lots of technology—think servers, multiple computers, hardware devices, and the like. The harder it is to create a good, realistic test environment, the stronger the argument in favor of using a paper prototype to avoid going to all that trouble (and possibly expense because hardware isn't free).

Test Environment—Accounts and Databases

For some testing situations, it's easy to set up dummy accounts and databases for users to use when performing tasks. Other times this isn't as feasible. For example, one financial services Web site didn't have a test database for usability testing (the developers had one but it wasn't possible to share it), so for testing purposes we had to use the real Web site, which naturally accessed the real customer database. For tasks that assumed the user already had an account, we needed a real account. There was a VP at the company who allowed his account to be used for this purpose (provided we didn't buy or sell anything!), but this guy was in no way a typical user—he owned dozens of stocks. Thus, users had to wade through a large amount

of unfamiliar data, which made some of the tasks artificially complex. And since we couldn't transact, we were limited in what we could test.

Tasks, Data, and Test Scenarios

The usability tasks that you want users to do can have a bearing on whether paper will offer advantages over testing the real thing. Paper prototypes can sometimes give you greater flexibility with your tasks than you'd get with the real interface. In other cases they're a poor substitute for reality.

Control of Content

You don't just need control of the technology—for usability testing purposes, you often need some degree of control over what the user sees, such as information that comes from a database. For live Web sites, don't forget content such as ads that may change without warning (not to mention the risk of accidental pornography). If some of the content isn't under your control, think about whether this could have a detrimental effect on your usability tests.

Between-Test Reset Procedures

In setting up a series of usability tests, consider what you need to do between tests to *reset* the test environment. For a Web site, this may be as simple as clearing the link history in the browser and deleting cookies. For software applications, it may be easy enough to delete all the files the user modified during the test session and replace them with fresh copies. But not always. One Web application had an underlying work flow involving several users and databases—when Person A was done handling a request, it resulted in database changes and work being sent to Person B via email. To simulate the work flows needed for our tasks, one of the developers created some elaborate procedures that we had to run between tests, and in one case even between tasks. (And because he couldn't attend all the tests, he also had to train others on those procedures.) I asked him afterward to estimate how long he'd spent on all that setting up and resetting; he said about 15 hours. Because this interface was fairly simple, we could have created a paper prototype in perhaps half that time.

Installation and Configuration Tasks

Tasks involving installation or configuration are a special case of reset procedures—they can be a pain. Say you have a test scenario where the user installs a software application. Perhaps in task 1 you'd test the ideal case where everything goes smoothly, and in task 2 you want to throw the user a curve in the form of a compatibility problem they have to resolve. To do task 2, you'd have to uninstall the software and set up the conditions that would cause the incompatibility. Or you could save an image of the pristine test machine on a server and restore it, although in my experience this can result in several minutes of downtime (which may not be a drawback if you can hold a discussion with the users in the meantime). With a paper prototype, you could simply start task 2 by telling the user, "Assume you're doing the same thing on a different machine."

Real-World Consequences

I once visited a prospective client that was developing a Web application for booking corporate travel. They knew that their interface lacked feedback, and rather shamefacedly admitted that on a couple of occasions while testing the interface they had accidentally bought plane tickets . . . nonrefundable ones, naturally. That's an example of a real-world consequence. Similarly, some software applications control physical equipment. If you're testing an interface for medical equipment or some other potentially dangerous device, you may have a situation in which users can blithely get themselves into real-world trouble. I once taught paper prototyping to some engineers who worked on process control for heavy machinery used in processing steel. One of them remarked that you got a whole new appreciation for the effects of a bug when it could result in sheets of steel spewing across the factory floor at 200 mph. Obviously, paper prototypes avoid such problems.

Lengthy Processing Delays

It's not a big deal in a usability test if the user occasionally has to wait a few seconds for a page to download or a large file to be opened. But if there is processing that takes any longer than the time needed to offer the users a refill of their coffee cups, you might be wasting too much time by testing the real interface. For example, in a usability test of software for network administrators, users were asked to

set up some tests of the network that would take several hours to execute in real life. With the paper prototype, we just told them, "It's now the next morning, and here's the report." (Sometimes with software you can do something similar, but other times it's just too much of a pain.)

User-Defined Tasks

In a paper prototype test, it's a given that you know what the tasks are because you came up with them before creating the prototype. But sometimes it's important to watch users doing their own tasks—something that they care highly about—rather than tasks you invented. When users work on their own tasks, they may uncover subtleties that you weren't aware of (or even entire *tasks* you weren't aware of). Unless the data and/or functionality of the interface is quite limited, user-defined tasks usually aren't feasible with paper because you need breadth—you can't predict where users will go. For example, you may need the entire pet products catalog rather than just the section devoted to ferret toys.

Timing and Scope

There are good times in the project cycle to use paper prototyping, and there are better times. There are even a few bad times.

Stage of Development

In the early stages of development, before implementation begins, paper prototyping may be your only viable option for doing usability tests. In the middle of development, you might have the option of testing either a paper prototype or a working version of the interface. But right before a release? Yuck. In my opinion, the most difficult time to do any kind of usability testing is in the month before launch—the development team is focused on fixing bugs and getting the product out the door, so it's difficult to get any bandwidth from them to observe usability tests. Even if a few tests can be squeezed in, there may be little opportunity to fix problems before the release, which can be demoralizing. In essence, testing done in the month before launch might really be testing done for the *next* version of the interface, so if you find yourself in this situation, you might think about waiting

until then. If you're still primarily concerned about this version, make every effort to do your testing as early as possible.

Although paper prototyping may be most common during earlier stages of development, it isn't wrong to use it in any situation in which the logistical advantages described in this chapter are sufficiently attractive.

Scope of Planned Changes

If you're designing something brand new, paper allows you to do several iterations of a design before you start implementation. The flexibility of paper as a design medium isn't a compelling advantage if you're not in a position to make substantial changes—perhaps you've already committed to a particular approach and you're just refining some of the details or maybe you've already done some usability testing and have a good handle on the issues. But if you know the current design will present you with some challenging problems to solve, paper will give you freedom to experiment.

Making Your Decision

The checklist in Table 14.1 summarizes the factors that this chapter has covered. (For a printable version, see *www.paperprototyping.com.*) This checklist isn't a definitive decision-making tool that will tell you whether you should use a paper prototype. Its goal is more modest—to ensure that you've thought about the risks particular to conducting usability tests with your interface. Hopefully this checklist will prevent your project from becoming a war story like the ones at the beginning of this chapter.

The number of check marks in each column isn't especially important, and not all the factors will apply to every situation. The main thing you should do is look for showstoppers in testing the working interface. If your software application depends on the stability of several modules being developed by an outside firm, you're moving to a new building 2 days before the first usability test, and the users are flying in from Singapore, the chances of a trouble-free usability study are vanishingly small. Go with paper and save yourself the aggravation. On the other hand, if the interface you want to test is under the control of two whiz-bang developers and the users are down the hall, there are relatively few inherent risks in testing the real thing.

Table 14.1 Checklist: Working Interface versus Paper Prototype

	Working Interface	*Paper Prototype*
People and logistics	❑ There are only a few people working on the design, perhaps 3 or 4 at most.	❑ The product team is larger.
	❑ The people directly responsible for the design are already proficient with a computer-based tool like VB, Dreamweaver, etc.	❑ Using a computer-based prototyping tool would require a learning curve before we were proficient enough to prototype what we envision.
	❑ Everyone working on the design has a technical (i.e., programming) background.	❑ There are nonprogrammers on the team whose input is important, such as tech writers, customer support reps, or Marketing.
	❑ The development team and users are all located within a reasonable commuting distance, or remote testing is a viable option.	❑ We aren't all in the same geographic area and remote testing isn't a good option—if we're going to do usability testing, someone has to travel.
	❑ The users are readily available. They won't be upset if we have to reschedule a test, and a few rescheduling fees won't break our budget.	❑ The recruiting firm charges us a fee to reschedule users, or the users are people we don't want to inconvenience.
Development context	❑ The interface is a standalone piece of software or other technology.	❑ There are many software modules required for successful completion of the tasks.
	❑ All the software modules needed for usability testing are stable, or we're confident we can test around the problems without too much difficulty.	❑ Some software modules are under development, are of uncertain reliability, or otherwise not under our direct control.
	❑ It's not a problem to pause development for a couple of days to do usability testing.	❑ People are constantly submitting changes—we either have to ask them to hold off for a few days or we'd need to set up a separate test environment.

Table 14.1 Checklist: Working Interface versus Paper Prototype—cont'd

	Working Interface	*Paper Prototype*
	❑ It's not difficult to set up a test environment, including hardware, databases, accounts, etc.	❑ It would take time and/or money to set up a test environment, or it would be hard to create one that's sufficiently realistic.
	❑ We can control everything the user sees, including product databases and ads.	❑ Some of the content comes from sponsors, manufacturers, or other outside sources, and we can't always predict or control it.
Tasks, data, and test scenarios	❑ Reset procedures are minimal—transactions can be easily faked or reversed.	❑ Resetting the test environment between tests is complicated—transactions are real, high consequence, or difficult to undo.
	❑ Installation and configuration are not being tested.	❑ Tasks include installation or configuration.
	❑ There are no long delays for the user to sit through—a minute at most, and usually less.	❑ The user may have to wait more than a minute or two for the system to do its processing.
	❑ We want users to bring their own real tasks to the test session.	❑ The set of tasks is known in advance; we've created them.
Timing and scope	❑ Usability testing can wait until the middle or end of the development process.	❑ Usability testing before or during development is desirable.
	❑ There is a previous working version of the interface.	❑ The interface doesn't exist yet or needs to be substantially redesigned.
	❑ We can't make huge changes at this point, just a few tweaks.	❑ We know there are significant problems, or we'd like to experiment with solutions.

Hybrid (Paper + Software) Testing

If you're having trouble deciding, why not take the best of both worlds? When a design is new or is being overhauled, it probably makes sense to do all your prototyping with the same method. But sometimes, especially if you've already done some usability testing, there may just be isolated parts of the design that you're working on. You may want to consider a hybrid prototype, where you use the real interface for most of the tasks but switch to paper for a screen or two as needed. (If you're up for a bit of Wizard of Oz, have the users write inputs on the paper. Then you can ask them to close their eyes while you "magically" replicate their actions in the software and then have them open their eyes and continue testing with the software.) Figure 14.1 shows an example where a single paper

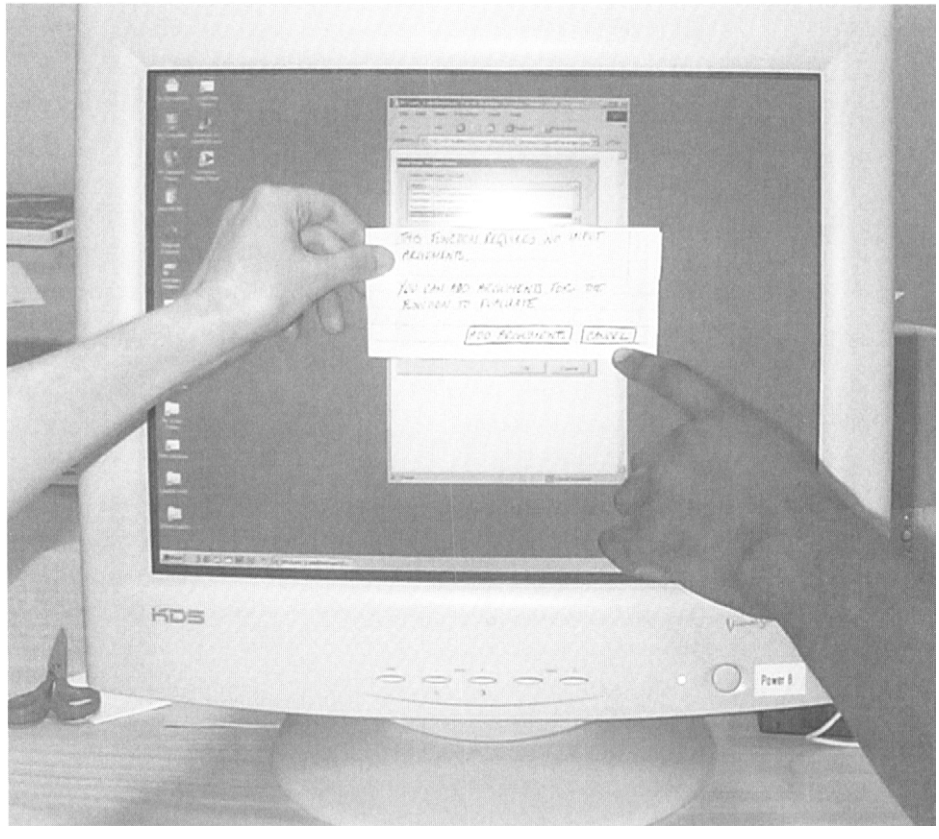


Figure 14.1 When you have only isolated screens to test, you can use a hybrid approach of software and paper.

prototype screen is held in front of a computer screen; it's also possible to put a few paper prototype screens on the table next to the user.

Hybrid prototyping may not be the best choice for companies that are just getting started with usability testing. The hybrid approach is somewhat awkward when the paper prototype consists of more than a couple screens or you do a lot of switching back and forth. I once conducted a usability study of a Web application where we alternated each task between a paper prototype and a laptop. This worked reasonably well, but the logistics were complicated—when users were working on the laptop, we also had it projected on the wall so that the rest of the team could see. When we switched to the paper prototype, however, we had to turn off the projector, move the laptop aside, and rearrange our seating. We lost some test time due to the constant shuffling of people and equipment, although the users didn't seem to mind.

I would use the hybrid approach again, but I would first make sure that it was truly needed, in other words, that there was a set of questions that called for each method. Otherwise I'd probably vote for doing the whole prototype one way or the other—in this case, the problems we found from testing on the laptop would also have shown up in a paper prototype.