Chapter 11

Data: Capturing, Prioritizing, and Communicating

So far this book has been about the paper prototype and the people involved in creating and testing it. But the heart of a usability study is its data—all the stuff that you learn. This chapter looks at some ways of capturing the data from paper prototype usability tests and what to do with it afterward. The ultimate purpose of collecting usability data is to make the interface better, so a good method is one that accomplishes this goal quickly and effectively. Naturally, every company is different, so a technique that is ideal for one product team may be unsuitable for another.

Capturing the Data (Note-Taking)

I ask all usability test observers to take notes. Not only does this ensure that we capture as much information as possible for the debriefing meeting (which I discuss in this chapter), but it also helps the observers focus on what the users are doing.

What Observers Should Look For

For starters, give each observer a copy of the tasks, especially if you've filled in the Notes section of the template as described in Chapter 6. This information will help observers understand the purpose of each task and some of the specific things you want to look for. If all the observers had a direct hand in creating the paper prototype, they'll already have plenty of their own questions, so you might not have to brief them on what to look for. However, if there are observers who were less involved, you might want to fill them in on the issues that arose during your walkthroughs.

But in general, what should observers look for? Cases where users are confused, don't understand a term, or misconstrue a concept are quite common and worth writing down. User quotes are often useful. Anything that surprises you is likely to be important because it indicates that users are behaving differently than you expected—if you expected the users to zig and they zagged instead, there's something going on at that decision point that you should understand better. And last, but not least, it's important to note the things that worked well in the interface so that you don't waste time redesigning something that doesn't really need it.

I occasionally find it helpful to give some observers specific assignments: "Gina, you watch whether they use context-sensitive help. Ali, you write down every error message the Computer gives them out loud." I tend to do this when the observers are unfamiliar with the interface (it gives them a productive way to become involved) or when the team has a lot of specific questions.

Observation, Inference, and Opinion

It sounds like a tautology to say that observers should write down observations, but in my experience some people don't understand what that means. It's important for observers to record data in a form that won't be subject to contentious debate later. It's natural for us to filter information through our own set of ideas and prejudices, but this subjectivity means you're now dealing with opinion rather than data—and you probably had all the opinions before you got started. So let's look at the differences between an observation, an inference, and an opinion.

Observations are objective, factual statements of something the user said or did. Observations are the "what," "when," and "how" of the action, for example, "User chose Landscape orientation," or "User said, 'I can't tell what it did.'" Observations by their nature are not subject to debate—any two people who watch the same test should theoretically walk away with the same set of observations. (In practice, they won't, because it's not possible to write down everything, so each observer chooses a subset of observations based on perceived importance.) Most of your usability test notes should consist of observations—what did the users type, click, and say? But observations do not describe the why, in other words, the causes, of problems or anything that goes on inside users' heads. Those are inferences.

Inferences are conclusions based on some observations and possibly some assumptions as well. Examples of inferences are "User is lost" or "This dialog box is confusing." Unlike observations, inferences are not fact; two observers can and do draw opposite inferences from exactly the same observations. As a way of checking whether something is an observation or an inference, ask yourself, "How do I know that?" If it could be verified from a videotape exactly as you've described it, it's probably an observation. But if it contains an assumption about what was going on inside the user's head, it is probably an inference. For example, the statement "He likes this version better" is an inference unless the user actually *said,* "I like this version better," in which case then it is indeed an observation.

An **opinion** is something you think should be done about your inferences or observations, such as, "Search results should by sorted by increasing price" or "We need to show an example." (The words *should* and *need* are often clues that you've written an opinion in your notes.) Ideas that occur to you during usability tests are not necessarily bad, but you don't want to confuse them with things that users have asked for.

Note: Sometimes I do deliberately include inferences or opinions in my notes, but I flag them (for example, with "???" or "idea") so that I know these are just my thoughts, not necessarily reality.

If you're not aware of these distinctions, especially between observations and inferences, it's easy to confuse them when looking back over your notes. The problem with inferences and opinions is that two observers can see the same thing but put a different spin on it. Here's an example of several possible inferences that could be drawn from one observation:

Observation: User paused for several seconds before entering "My IRA" in the Description field for the account.

Inference 1: He didn't know whether the description was required.

Inference 2: He wasn't sure whether spaces were okay.

Inference 3: He wondered whether this description would appear in his portfolio instead of the account name.

Inference 4: He wasn't confused; he was just deciding what to call it.

And so on. As you can see, the people making these inferences have drawn very different conclusions about what the problem is (or whether there is one) and will go on to formulate different opinions about how to solve it. Thus, I think of inferences and opinions as "argument seeds" planted in one's notes, because they're likely to sprout into disagreement later.

A good test facilitator encourages users to articulate what they're thinking and doing, thus turning what would otherwise be inferences into observations. (This is the essence of the sportscaster role.) For example, if the user paused for more than a few seconds, the facilitator might ask, "What's going through your mind right now?" The user's answer should reveal which of these situations is happening.

Inferences and opinions in your notes are "argument seeds," likely to sprout into disagreement later.

At some point you probably do want to start determining the possible causes of problems, but it's best to wait until the debriefing meeting to do this. Let's assume that everyone wrote down the observation that the user paused. Some possible reasons for this might be supported by additional observations; for example, if the user later said, "Good, it took it" then that observation might support inference 2 about whether the description could include spaces. Sometimes there will still be disagreements, but if everyone offers their inferences as fact, it becomes much harder to figure out what really happened.

I'm not advocating that we should overanalyze everything to ridiculous lengths. I deliberately chose a rather trivial example of a momentary pause to illustrate the point that some problems are more important than others. As I explain later in this chapter, by first prioritizing the issues that were found in the usability tests, the team will avoid dissecting the minor ones.

Overanalyzing details to ridiculous lengths reminds me of the joke about a scientist who takes a train trip with a friend. Gazing at the passing scenery, the friend remarks, "Look, that flock of sheep has just been sheared." The scientist scrutinizes the flock and cautiously assents, "So they appear to have been . . . on this side."

Taking Notes

There is no one right way to take notes—each observer will have his or her own conventions and shorthand. This is fine unless you plan to read each other's notes in their original format, in which case it can be confusing unless you know what each other's conventions are. Table 11.1 shows some notes I took from a usability test of a travel Web site. Other than spell checking and some minor editing for clarity, these are typical of the notes I take. As shown in the Discussion column, subtle wording can mask some of the differences between observations and inferences, and further underscores the risks of interpreting someone else's notes.

Table 11.1 Example Test Observations from a Travel Web Site

Observation (from my notes)	*Discussion*
Next trip he's taking? Going to Chicago in September for a wedding in Barrington. A friend had told him a particular place—Amerisuites. What other factors would he consider? Says he prefers a pool. Wants a restaurant since he's not going to have a car. Shuttle service to the airport would also be good. I ask about price: "I like to be under $100, unless I'm on vacation."	This is the sort of speech I tend to paraphrase—the factors the user is listing give us some background on how he'll approach this task, but in this case his exact words aren't crucial.
Clicks NW suburbs. "It gave me a list of 43 properties, hotels in the northwest region of Illinois. Since I know I'm going to the Amerisuites, I'll probably do a search off the first page." He backs up to try that approach.	Saying that the user backs up is an observation; the reason why is an inference. In this case the user's subsequent actions happened to confirm it, but before the fact it can only be an inference.
Types barrington in city, Hotel name—amerisuites. "That would be a problem if I don't know how to spell it." He knows the exact dates of his trip, enters them in the search.	The phrase "he knows the exact dates" appears to be an inference. However, earlier in the test the user had said when his trip was, so in the larger context of the test this is actually an observation. If I were rewriting this for a report, I would clarify by phrasing it as two observations: "User had mentioned exact dates earlier; he entered those dates in the search."
No matches found. Says he thinks he spelled amerisuites wrong . . . or maybe it's not in Barrington. Gets rid of the hotel name and it returns one property in Barrington.	If the user didn't say what he was thinking, I might write "(Not sure of spelling???)" to indicate that I'm making an inference. If I had asked him, my notes would say, "I ask about spelling."

Laptop versus Hand-Writing

I tell observers to take notes in whatever form is most comfortable for them, and most people bring a notebook and pen. Some teams write their observations directly onto sticky notes, which as I'll explain can give you a leg up on analyzing the results. I prefer to use a laptop for note taking because I type quickly and can capture far more information. The notes I take by hand don't have as much detail, and thus it's easier to blur the difference between observations and inferences.

One potential concern about a laptop is that it can be intimidating or distracting to users, but I haven't found this to be the case. It helps if you:

- **Type quietly.** Some keyboards are noisier than others, and fingernails can make additional noise. The quieter the typing is, the less distracting it is.
- **Type continuously.** If the room is quiet and suddenly I start banging on the keyboard, that could alert the users that they've just done something interesting (which they will probably assume is a mistake). But if I keep up a fairly constant stream of typing, it becomes part of the background noise and thus is less salient.

I've found it useful to learn to type without looking at my screen—not too hard once you accept that you'll make a liberal number of typos; that way you can keep your eyes on the action. (As a note-taking facilitator, I have learned to type one thing while saying another, which is much more difficult and takes practice.) Whether your notes are handwritten or typed, if you follow my suggestion to leave at least 2 hours between usability tests, you should have time to go through your notes. Fill in details and repair any especially unintelligible passages while the events are still fresh in your mind.

Although it is uncommon (it's never happened to me), if a user ever asks to see your notes you should allow this because users have a right to know what data you've collected about them. The implication is that you should never put anything in your notes that you'd be uncomfortable letting users read.

Capturing User Quotes

I tell myself that the user's voice speaks 100 times louder than my own. One or two user quotes are usually worth whole paragraphs of my own opinions about something—it's hard to argue with a good user quote. But even though I type fairly fast, I can't keep up with speech, so sometimes my quotes turn into paraphrases. (For those who don't type fast or want to be certain they've gotten a quote right, a tape recorder might be a useful backup.)

As with observations and inferences, you want to be able to distinguish after the fact which is which, so my convention is to use quotation marks to set off wording that I'm certain I've captured verbatim. When I fall behind and have to start paraphrasing, I'll end the quote but continue with the gist of what the user

The user's voice speaks 100 times louder than my own.

said. Or, sometimes I use ellipsis (. . .) to leave out the words I didn't capture exactly, then pick up the quote farther on. One of the risks of paraphrasing is that it can be hard to tell afterward how accurate you were. One trick I've learned is to use "says" in my notes when the user is talking and I'm fairly certain I'm capturing the meaning but not the exact words—in other words, I believe a videotape would confirm what I'm writing but I wouldn't bet a year's salary on it. If I'm less certain (or am certain that I missed something), I use "Something about . . ." which is a reminder for me later to ask the other observers if they can fill in what I missed.

Including Context

This has happened to me more times than I care to admit: I'll have some awesome quote in my notes such as, "Oh, wow, it's like they read my mind" and 2 days later, *I have no idea what the user was talking about.* In the heat of the moment, between asking questions and listening to answers and all that, I forgot to include enough contextual cues in my notes to reconstruct what happened. In particular, context means keeping track of what screen the users are looking it and what data they enter. You may also want to write down what the facilitator says, to help you distinguish between a spontaneous comment and one that was prompted by a question—the former usually carry more weight.

Debriefing Meeting: Prioritizing the Issues

Usability testing is like getting a drink from a fire hydrant—there's a whole bunch of data, coming at you at high speed. So what do you do with it all? It's important to get the product team together at the end of a usability study and sort out what you learned. Even if you've been listing issues and making changes after each test, it's still useful to step back and look at the larger picture. This happens at the debriefing meeting.

There are two main outputs from the debriefing process: prioritization of the issues and an action plan for addressing them. Some teams do both things at one meeting; if some observers aren't involved in implementing changes, they can participate in the prioritization but leave when you're ready to create the action plan.

Because it's usually not practical for everyone to observe every test, people may arrive at the debriefing meeting with different ideas about what happened.

So the first order of business is to share everyone's observations and agree on priorities. Then and only then should you start discussing what to do about the issues.

Affinity Diagram

A technique called an *affinity diagram* is useful for identifying the patterns in qualitative data. Figure 11.1 shows what one looks like, and Figure 11.2 describes the process (a handout is available at *www.paperprototyping.com*). Here's a brief overview: Everyone picks their top observations from their notes and writes them on individual sticky notes or blank index cards. (At this point I remind people that their inferences and opinions are best saved for later, although inevitably a few of them will creep in.) Tape all the cards to a wall. Everyone reads all the observations, and then you sort them into groups, name each group, and then vote on which group(s) have the greatest impact on the success of the next release.

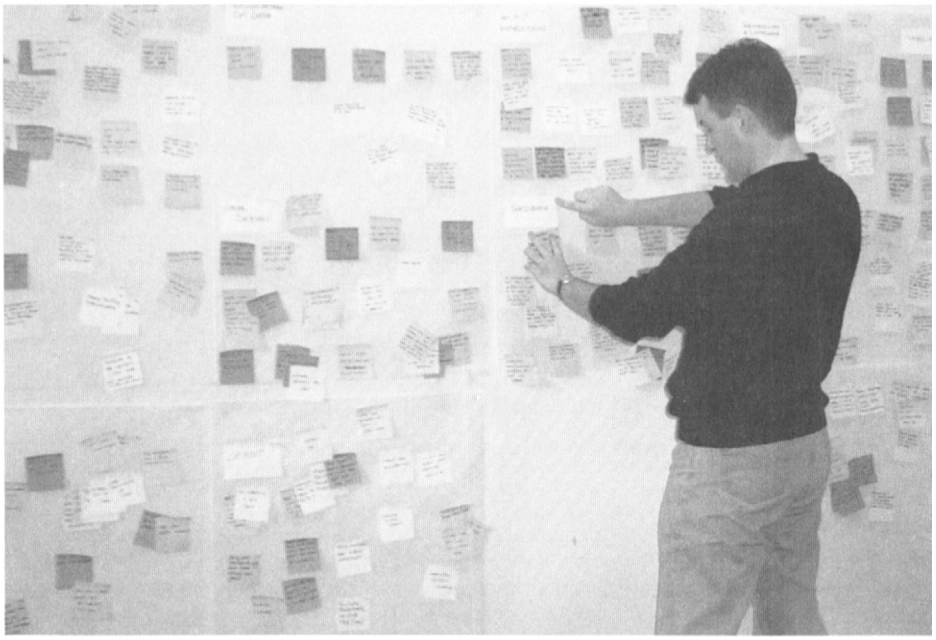

Figure 11.1 In an affinity diagram, the team sorts individual observations into related groups. Putting flip chart paper on the walls first as shown here makes the whole thing easily portable to another wall.

1. Observers go through their notes and identify the unsolved issues that they believe are most important to the success of the next release. They write those issues, *one per card,* onto index cards (or sticky notes). You may want to have a rule that the number of tests affects the number of cards from each person—perhaps about 5 per test.
2. Tape all the cards to one wall in random order.
3. Everyone reads all the cards. Don't worry about duplicates or issues that were solved by subsequent prototype changes—keep those issues in the process. (Variation: A person who discovers a card that covers the same issue as one of their own is allowed to remove *his or her own* card, but not someone else's.) If people think of additional issues they're allowed to add cards.
4. Sort the cards into groups, *without discussion.* (Discussion doesn't necessarily improve the quality of the end result but it's almost guaranteed to make the process take longer.) Keep the groups far enough apart that it's clear what is grouped with what. If someone disagrees with the way a group has been set up, he or she should simply move the cards. In particular, look for large groups that could be subdivided and small groups that have the same theme. This step ends when all the cards have been placed in a group (a solo card or two is okay) and no one is making further changes to the groups.
5. Using sticky notes (I'll assume yellow ones), name each group. The name should reflect the theme of the group. Each participant has the opportunity to name each group, and each group can have any number of yellow stickies. But if you get to a group and it already has a name that you agree with, there is no need to create a duplicate.
6. Everyone reads all the group names. On a piece of scratch paper, everyone writes down the three groups that they believe have *the greatest impact on the success of the next release.* Ask yourself, "If we had time to address only three of the groups, which three would I pick?" Choose your top three regardless of whether the work must be done by you or others—these priorities are for the project, not individual to-do lists.
7. Voting: Look at your three choices and rank them in order, with 1 being most important. On the yellow stickies, put an X to indicate your third choice, XX for second, and XXX for most important.
8. Find all the yellow stickies containing X's. The number of X's indicates the group's consensus about the priority of that category of issues. If you find duplicate categories, combine them. (If there is disagreement that two categories should be combined—as when one group is a subset of another—it may be more useful to keep them separate.)
9. Reality-check the results by asking, "Does everyone agree that these priorities make sense?" Discuss any dissenting views.
10. Start at the top of the priority list. Discuss each category in turn: the observations it contains, the insights you learned, and (if appropriate given those present) how to solve remaining issues.

Figure 11.2 Affinity Diagram Process

Note: There is no one "right" way to use this method. The steps in Figure 11.2 describe the way I conduct this process with my clients, but I think it's fine to experiment with any variation that also accomplishes the goal of reaching a group consensus. For example, you can use different color stickies to represent different kinds of information, give those on the core team more votes than outsiders, and so on.

An affinity diagram isn't specific to paper prototyping—it's a technique that works well whenever there's a group of people with a lot of information to sort and prioritize. (A client once told me that she uses the affinity diagram technique described in this chapter to make family decisions with her four children!) There are also methods of prioritizing issues other than the affinity diagram; see the following From the Field box.

From the Field: Prioritizing Issues with Sticky Notes

"I ask everyone to write issues on sticky notes as they observe the tests. Afterward, we go around the room. Each person reads through their stack of sticky notes and we agree whether each one is an issue, a bug, or an anomaly. Duplicates are discarded. At the end of this process we have one stack of stickies that's much smaller than the piles we started with. When an issue comes up and it's clear who should solve it, that person gets the sticky note. Sometimes everyone walks away with their set of things to solve, and there's nothing to write up afterward. Other times, we take the single stack of issues and prioritize those by an affinity diagram. The prioritized issues go into a spreadsheet—in a later meeting we'll read through the issues and assign them. This variation is helpful when a lot of data is obtained because it postpones the discussion of how to solve the issues until priorities are agreed upon."

Mary Beth Rettger, The MathWorks

Why a Group Method?

You may wonder why you'd want to do the prioritizing process as a group, especially if there are only a couple of people who'll be implementing the changes. Here are some reasons to consider.

- **Pooling observations.** No one person can take complete and perfect notes. Because each observer captures a different subset of observations, there's less chance that an issue will slip through the cracks.
- **Recognizing patterns.** Most usability test observations are qualitative in nature—they're not something you can measure with a yardstick or a voltmeter. The key to analyzing qualitative data is to identify the patterns over several users' worth of tests. Although I don't know much about pattern recognition as a science, I do know that it's a skill and some people are better at it than others. Similar to how a group stands a better chance of wilderness survival than any of its individual members, having several people look at the big picture usually means a greater chance of correctly identifying the most important issues.
- **Choosing optimal solutions.** It's easy to get focused on your particular area of responsibility and lose sight of the larger picture. When this happens, problems may not get solved in the most effective way. For example, a tech writer's first reaction to an inscrutable error message might be, "Gee, I'd better document that." (Or perhaps, "Let me improve the wording of that error message.") But a developer sitting in the same room might suddenly realize, "You know, I could put a check in the code to prevent that problem from happening at all." Conversely, the team might decide that documenting an issue really is the only feasible way to address it. By having the team discuss the options, there's a better chance of finding an efficient solution. An additional benefit is that the left hand will know what the right hand is doing—if the developer says that the error message will be eliminated, the tech writer won't waste time documenting it.
- **Disseminating results.** The more members of the product team who attend the debriefing, the less need there may be for formal documentation of the results via reports, highlight tapes, or other time-consuming activities.

A Note about Granularity

Some people record their issues directly onto the index cards or sticky notes while they observe the usability tests. I think this is okay, but be aware that it can have an effect on the granularity of the observations you collect. Not all usability issues occur at the level of a confusing field label—some unfold as a series of events over several minutes or longer. For example, perhaps you initially thought that the users made a mistake but later realize that they had a different and valid way of going about the task. I find that I sometimes have to look over a few pages of my notes to piece together an issue that is subtle and/or happened in the context of several screens. If you use stickies to record your notes, it might be a good idea to spread out all your stickies in order, see if any larger issues jump out at you, and if so then summarize them on stickies of their own.

Success Rates and Statistics

You may have noticed that so far I've said little about calculating success rates, error rates, and the like. Although I'm not opposed to metrics and statistics, I have not found them to be especially useful in analyzing the results of paper prototype tests. For example, it's hard to understand the meaning of the statement "Five out of seven users completed task 3 successfully" if the interface was different each time.

But sometimes simple statistics can be helpful in determining the importance of an issue or how to solve it. For example, I might be curious to know how many users looked for a link to the return policy on the shopping cart page, so I might go through my notes and count how many times it happened. Although it's often difficult to know ahead of time what issues you'll want to count (the most interesting ones tend to be the ones you didn't anticipate), if your notes are sufficiently detailed you may be able to extract some useful information. However, I have done this less often in paper prototype studies than when I'm testing a working version of the interface.

Action Plan

I don't believe it's an overstatement to say that you'll never fix all the problems you find from a usability study. It's just not practical to make an interface work perfectly for everyone, all the time; eventually you reach a point of diminishing returns. Your project also has constraints on the time and people resources that can be devoted to making changes.

I'm assuming that you already have some kind of process for managing your product development. I don't want to turn this into a book on project management, but here are a few thoughts to keep in mind as you discuss your action plan with your fellow team members.

- **Consider practicality.** One drawback of paper prototyping is that you can sometimes fix an issue in the prototype in a way that's not practical to implement. It's worth discussing the successful changes to the interface to ensure they are doable. If not, since usability testing gave you a better understanding of the problem, you'll at least be in a position to come up with an alternative idea.

- **Funnel usability issues into your regular process.** As a rule, I don't believe there's much benefit in setting up a separate system for managing usability issues if the people who fix them also do other development activities. Usability issues should be managed the same way as the rest of the work for the project. For example, if you've got the equivalent of a bug database (one company I worked for preferred the more delicate term *incident*), you'll probably want to funnel the unsolved usability issues into it.
- **Don't start tracking too early.** Because paper prototyping is a fast-moving, iterative activity, it can be premature to record every issue in a database—some problems will be fixed before you've even entered them. One company I know of doesn't bother trying to capture the issues until the design has shown signs of stabilizing, at which point they begin entering usability issues in their bug-tracking database.
- **Divide issues into "now" and "later" piles.** If your next release is quickly approaching, you may have to draw firm lines. One useful question is, "Which of these issues are important to the success of the next release?" Anything that doesn't pass that test goes into the bug database at a lower priority.
- **Consider allocating time for usability issues.** One valid concern about usability testing is that it's hard to know ahead of time how long it will take to fix the issues you find. One way to manage this is to set aside a predetermined amount of time to focus on the issues from usability testing—give each person a prioritized list and have them fix as many as they can in a certain amount of time. The interface may still go out the door with known usability problems, but at least you've given it your best shot. (Caveat: This approach only works for non-critical issues, not showstoppers such as "No one can successfully install the product.")

Communicating and Documenting the Results

It usually isn't practical to write a detailed report of paper prototype test results—in 10 years as a usability consultant, I've never done a detailed report for a paper prototype, although I have occasionally written such reports for more formal usability studies. However, it's important to communicate your findings to people who couldn't be there and/or will need the information later. Once you're ready to move into the implementation phase, the paper prototype contains a lot of information that you may need to document in some way. There's additional information in the Computer's head about how the interface behaves, and you don't want to lose that either.

This section contains several ideas for communicating the results of a usability study. They differ in their formality, the situations in which they're useful, and the effort needed to do them. You'll need to consider the particulars of your company culture in deciding which one(s) are right for you.

Top-10 List

The first thing I do after the debriefing meeting is to take the prioritized list of issues and create a short summary of the top issues. Typically I'll write one to two paragraphs about each issue that summarize our discussion from the meeting. The resulting report is usually no more than two to three pages; when I do it in email it's even shorter. For some product teams this is all the reporting that's needed.

Methodology Report

You might find it helpful gather all your methodology-related documents (user profile, screener, tasks, and so on) into one document to make things easier for next time. Because "methodology report" sounds overly stodgy, I sometimes I call this the "paper trail" report. Its primary purpose is to keep you from reinventing wheels because user profiles and tasks can often be reused in subsequent usability studies. Alternatively, you might simply collect all these documents into one (real or virtual) folder for safekeeping.

Highlight Tape

A highlight tape uses clips from the paper prototype tests. I rarely make highlight tapes because they're time-consuming, and they're usually unnecessary if you can get people to observe the tests. So consider this a technique to use only on rare occasions (for example, to accompany a presentation you're giving to 30 coworkers in another city) and make sure you've obtained the users' consent for that purpose.

Walkthrough Video

A walkthrough video can be an efficient means of summarizing what you learned from a paper prototype usability study. A walkthrough video is especially useful when the development team is in multiple locations. I've worked on several projects where the design part of the team was in Boston but the developers were in California, Russia, or Israel. Unlike a highlight tape, a walkthrough video is scripted and it shows a member of the development team instead of real users.

To make a walkthrough video, first jot down what you want to cover—how users reacted to various aspects of the interface, what you changed, how successful those changes were, and so on. Then turn on the video camera, focus it on the prototype, and start doing a show-and-tell explanation of what you learned. (If the development team is located in another country, consider making the tape in their native language.) There's no need to do everything in one take, so stop the camera whenever you need to gather your thoughts. A typical walkthrough video is 10 to 15 minutes long and can be created in an hour or so.

Interface Specification

To some extent, a paper prototype itself is a kind of an interface specification because it is a collection of screens. But you might need to tie the images together in a way that shows the sequence, with explanations of behaviors and underlying processing. A quick-and-dirty way to do this is to scan the prototype pieces and use them in the spec. The image quality will leave something to be desired, but this method is fast. Alternatively, some companies render the prototype pieces in a way that shows the final appearance—this takes more time, but sometimes high-fidelity visuals are needed, especially if an outside group is doing the implementation. Companies differ in the degree of detail they require in specifications, but ask yourself whether the paper prototype might be a useful starting point.

Documentation of the Interface Behavior

Paper prototypes show the elements of an interface. But information about how those elements interact—as well as the behavior of hard-to-prototype elements such as cursors—is often found only in the head of the Computer. A team at The MathWorks created a method for capturing and managing this information. During prototyping and design sessions, team decisions about how the prototype would behave were recorded on index cards ("When the user clicks, the selection rectangle recenters."). During usability testing, the team looked for places where hard-to-prototype behavior was confusing or hadn't been clarified, such as how to make a control point visible on top of a complex image. They created additional index cards for these.

The original intent of the index cards was as a way to track problems to be solved, as well as to help translate the users' actions into specific functions in the technical design document. (As it turned out, they made a good "cheat sheet" for the Computer as well.) But the quality engineer and the documentation writer for the project—both of whom were involved in the prototyping exercise—found additional ways to use the index cards. By the time the developers created a working version of the software, the quality engineer had used the cards to create a test plan and the documentation writer had a first draft of the tool's documentation.

Summary

The seven chapters in this part have walked you through the process of planning a usability study, creating tasks and the prototype, conducting usability tests, and using the results. In a nutshell, this is what I do as a usability consultant. The nuances can take years to master, but my hope is that I've given you enough information and confidence to get started. Don't forget that more resources can be found in the References section and on the Web site. Good luck!