JAKE WOBBROCK
4/18/00

A Practical Guide to Usability Testing

Joseph S. Dumas
Janice C. Redish

intellect™
EXETER, ENGLAND
PORTLAND OR, USA

2
Introducing Usability Testing

We begin this chapter by defining usability testing, examining both what is and is not required for a usability test and reiterating the important point that you should test early and often—not just once at the end of the development process. We then look at how our definition of usability testing applies in a wide variety of situations. We also add two more benefits to the list we began in Chapter 1; these two are benefits that come more from usability testing than from other techniques in the usability engineering approach. Finally, we end the chapter by comparing usability testing with two quite different techniques with which it is often confused: beta (or field) testing and research studies.

What is Usability Testing?

While there can be wide variations in where and how you conduct a usability test, every usability test shares these five characteristics:

1. The primary goal is to improve the usability of a product. For each test, you also have more specific goals and concerns that you articulate when planning the test.
2. The participants represent real users.
3. The participants do real tasks.
4. You observe and record what participants do and say.
5. You analyze the data, diagnose the real problems, and recommend changes to fix those problems.

The Goal is to Improve the Usability of a Product

The primary goal of a usability test is to improve the usability of the product that is being tested. Another goal, as we will discuss in detail later, is to improve the *process* by which products are designed and developed, so that you avoid having the same problems again in other products.

This characteristic distinguishes a usability test from a research study, in which the goal is to investigate the existence of some phenomenon. Although the same facility might be used for both, they have different purposes. (See the section, "Comparing Usability Tests to Research Studies," later in this chapter.)

This characteristic also distinguishes a usability test from a quality assurance or function test, which has a goal of assessing whether the product works according to its specifications.

Within the general goal of improving the product, you will have more specific goals and concerns that differ from one test to another.

- You might be particularly concerned about how easy it is for users to navigate through the menus. You could test that concern before coding the product, by creating an interactive prototype of the menus, or by giving users paper versions of each screen. (See Chapter 5 for more on these types of tests.)
- You might be particularly concerned about whether the interface that you have developed for novice users will also be easy for and acceptable to experienced users.
- For one test, you might be concerned about how easily the customer representatives who do installations will be able to install the product. For another test, you might be concerned about how easily the client's nontechnical staff will be able to operate and maintain the product.

These more specific goals and concerns help determine which users are appropriate participants for each test and which tasks are appropriate to have them do during the test. (See Chapter 8, "Defining Your Goals and Concerns," for more on this topic.)

The Participants Represent Real Users

The people who come to test the product must be members of the group of people who now use or who will use the product. A test that uses programmers when the product is intended for legal secretaries is not a usability test.

The quality assurance people who conduct function tests may also find usability problems, and the problems they find should not be ignored, but they are not conducting a usability test. They are not real users—unless it is a product about function testing. They are acting more like expert reviewers.

If the participants in the usability test do not represent the real users, you are not seeing what will happen when the product gets to the real users.

If the participants are more experienced than actual users, you may miss problems that will cause the product to fail in the marketplace. If the participants are less experienced than actual users, you may be led to make changes that aren't improvements for the real users. (See Chapter 9, "Deciding Who Should Be Participants," and Chapter 10, "Recruiting Participants," for more about making sure that the people who come to your test represent the users.)

The Participants Do Real Tasks

The tasks that you have users do in the test must be ones that they will do with the product on their jobs or in their homes. This means that you have to understand users' jobs and the tasks for which this product is relevant.

In many usability tests, particularly of functionally rich and complex software products, you can only test some of the many tasks that users will be able to do with the product. In addition to being

realistic and relevant for users, the tasks that you include in a test should relate to your goals and concerns and have a high probability of uncovering a usability problem. (Chapter 11, "Selecting and Organizing Tasks to Test," and Chapter 12, "Creating Task Scenarios," give more information about what to have users do in a test.)

Observe and Record What the Participants Do and Say

In a usability test, you usually have several people come, one at a time, to work with the product. You observe the participant, recording both performance and comments.

You also ask the participant for opinions about the product. A usability test includes both times when participants are doing tasks with the product and times when they are filling out questionnaires about the product. (For more on deciding what to observe and record while users are working with the product, see Chapter 13, "Deciding How to Measure Usability." For more on questionnaires, see Chapter 14, "Preparing Test Materials.")

Observing and recording individual participant's behaviors distinguishes a usability test from focus groups, surveys, and beta testing.

A typical focus group is a discussion among 8 to 10 real users, led by a professional moderator. Focus groups provide information about users' opinions, attitudes, preferences, and their self-report about their performance, but focus groups do not usually let you see how users actually behave with the product. (We discuss focus groups in more detail in Chapter 3, "Uncovering Usability Needs Before You Design.")

Surveys, by telephone or mail, let you collect information about users' opinions, attitudes, preferences, and their self-report of behavior, but you cannot use a survey to observe and record what users actually do with a product.

A typical beta test (field test, clinical trial, user acceptance test) is an early release of a product to a few users. A beta test has ecological validity, that is, real people are using the product in real environments to do real tasks. However, beta testing seldom yields any useful information about usability. Most companies have found beta testing to be too little, too unsystematic, and *much too late* to be the primary test of usability. (Because some people still believe that a beta test substitutes for a usability engineering approach and for usability testing during the process, we discuss beta testing in some detail later in this chapter.)

Analyze the Data, Diagnose the Real Problems, and Recommend Changes to Fix Those Problems

Collecting the data is necessary, but not sufficient, for a usability test. After the test itself, you still need to analyze the data. You consider

the quantitative and qualitative data from the participants together with your own observations and users' comments. You use all of that to diagnose and document the product's usability problems and to recommend solutions to those problems.

As we will discuss in Chapter 20, "Tabulating and Analyzing Data," this is not a trivial task. Usability testing is distinguished from beta testing by both the quality and quantity of data that you have. The data are systematic, comparable across the participants that you saw, and very rich.

The Results Are Used to Change the Product – and the Process

We would also add another point. It may not be part of the definition of the usability test itself, as the previous five points were, but it is crucial, nonetheless.

Someone must use the results of the usability test.

A usability test is not successful if it is used only to mark off a milestone on the development schedule. A usability test is successful only if it helps to improve the product that was tested and the process by which it was developed. As we discuss in Chapter 24, "Changing the Product and the Process," part of your task as a usability tester is doing what you can to make sure that the results of the test are used appropriately.

What Is Not Required for a Usability Test?

Our definition leaves out some features you may have been expecting to see, such as:

- a laboratory with one-way mirror
- data-logging software
- videotape
- a formal test report

Each of these is useful, but not necessary, for a successful usability test. For example, a memorandum of findings and recommendations or a meeting about the test results, rather than a formal test report, may be appropriate in your situation.

Each of these features has advantages in usability testing that we discuss in detail later, but none is an absolute requirement. Throughout the book, we discuss methods that you can use when you have only a shoestring budget, limited staff, and limited testing equipment.

When is a Usability Test Appropriate?

Nothing in our definition of a usability test limits it to a single, summative test at the end of a project. The five points in our definition are

relevant no matter where you are in the design and development process. They apply to both informal and formal testing. When testing a prototype, you may have fewer participants and fewer tasks, take fewer measures, and have a less formal reporting procedure than in a later test, but the critical factors we outline here and the general process we describe in this book still apply. Usability testing is appropriate *iteratively* from predesign (test a similar product or earlier version), through early design (test prototypes), and throughout development (test different aspects, retest changes).

Questions that Remain in Defining Usability Testing

We recognize that our definition of usability testing still has some fuzzy edges.

- Would a test with only one participant be called a usability test? Probably not. You probably need at least two or three people representing a subgroup of users to feel comfortable that you are not seeing idiosyncratic behavior.
- Would a test in which there were no quantitative measures qualify as a usability test? Probably not. To substantiate the problems that you report, we assume that you will take at least some basic measures, such as number of participants who had the problem, or number of wrong choices, or time to complete a task. The actual measures will depend on your specific concerns and the stage of design or development at which you are testing. The measures could come from observations, from recording with a data-logging program, or from a review of the videotape after the test. The issue is not which measures or how you collect them, but whether you need to have some quantitative data to have a usability test.

Usability testing is still a relatively new development; its definition is still emerging. You may have other questions about what counts as a usability test. Our discussion of usability testing and of other usability engineering methods, in this chapter and the next three chapters, may help clarify your own thinking about how to define usability testing.

Testing Applies to All Types of Products

If you read the literature on usability testing, you might think that it is only about testing software for personal computers. Not so. Usability testing works for all types of products. In the last several years, we've been involved in usability testing of all these products:

Consumer products

Regular TVs
High-definition TVs
VCRs
Remote controls
Cordless telephones
Telephone/answering machines
Business telephones

Medical products

Bedside terminal
Patient monitor
Anesthesiologist's workstation
Blood gas analyzer
Integrated communication system for wards
Nurse's workstation for intensive care units

Engineering devices

Digital oscilloscope
Network protocol analyzer (for maintaining computer networks)

Application software for microcomputers, minicomputers, and mainframes

Electronic mail
Spreadsheets
Database management software
Time management software
Compilers and debuggers for programming languages
Operating system software

Other

Voice response systems (menus on the telephone)
Automobile navigation systems (in-car information about how to get where you want to go)

The procedures for the test may vary somewhat depending on what you are testing and the questions you are asking. We give you hints and tips, where appropriate, on special concerns when you are focusing the testing on hardware or documentation; but, in general, we don't find that you need to change the approach much at all.

Most of the examples in this book are about testing some type of hardware or software and the documentation that goes with it. In some cases, the hardware used to be just a machine and is now a special purpose computer. For usability testing, however, the product doesn't even have to involve any hardware or software. You can use the techniques in this book to develop usable

- application or reporting forms
- instructions for noncomputer products, like bicycles
- interviewing techniques
- nonautomated procedures
- questionnaires

Testing All Types of Interfaces

Any product that people have to use, whether it is computer-based or not, has a user interface. Norman in his marvelous book, *The Design of Everyday Things* (1988) points out problems with doors, showers, light switches, coffee pots, and many other objects that we come into contact with in our daily lives. With creativity, you can plan a test of any type of interface.

Consider an elevator. The buttons in the elevator are an interface— the way that you, the user, talk to the computer that now drives the machine. Have you ever been frustrated by the way the buttons in an elevator are arranged? Do you search for the one you want? Do you press the wrong one by mistake?

With the configuration in Figure 2-1, impatient people might reach out to close the doors and make the alarm bell ring instead.

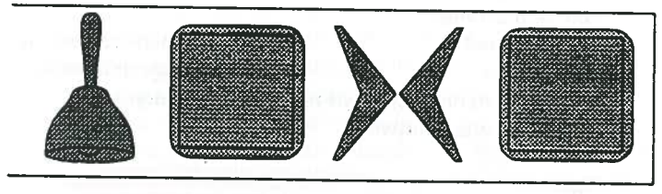

Figure 2-1. Part of the user interface to an elevator

You might ask: How could you test the interface to an elevator in a usability laboratory? How could the developers find the problems with an elevator interface before building the elevator—at which point it would be too expensive to change?

Any product or part of a product that people will use should be tested for usability.

In fact, an elevator interface could be tested before it is built. You could create a simulation of the proposed control panel on a touch-screen computer (a prototype). You could even program the computer to make the alarm sound and to make the doors seem to open and close, based on which buttons users touch. Then you could bring in users one at a time, give them realistic situations, and have them use the touchscreen as they would the panel in the elevator.

Our point is that, with a little ingenuity, you can prototype any interface and have users try it out in a usability test.

Testing All Parts of the Product

Depending on where in the development process you are and what you are particularly concerned about, you may want to focus the usability test on a specific part of the product, such as

- installing hardware
- operating hardware
- cleaning and maintaining hardware
- understanding messages about the hardware
- installing software
- navigating through menus
- filling out fields
- recovering from errors
- learning from online or printed tutorials
- finding and following instructions in a user's guide
- finding and following instructions in the online help

You must also, however, test how the parts work together and support each other. Even if you are focusing on one aspect of the product, be alert to what you see and hear about other aspects.

When we first started doing usability testing, clients sometimes asked us to "just test the documentation," because another group "owned" the software. We learned very quickly that in any usability test, even if you are focusing on the documentation, you learn a tremendous amount about the software (and vice versa). We couldn't *not* report what we had learned—and our documentation clients also realized that their software colleagues had to know what we had learned about the interface. They also realized how futile it was to think of the documentation and software separately. Good changes to both products and processes came out of those tests.

Testing Different Aspects of the Documentation

When you include documentation in the test, you have to decide if you are more interested in *whether users* go to the documentation or in how well the documentation works for them *when they do go* to it. It is difficult to get answers to both of those concerns at the same time.

If you want to find out how much people learn from a tutorial *when they use it*, you can set up a test in which you ask people to go through the tutorial. Your test paticipants will do as you ask, and you will get useful information about the design, content, organization, and language of the tutorial.

You will, however, not have any indication of whether anyone will actually open the tutorial when they get the product. To test that, you have to set up your test differently.

Instead of instructing people to use the tutorial, you have to give them tasks and let them know the tutorial is available. In this second type of test, you will find out which types of users are likely to try the

tutorial, but if few participants use it, you won't get much useful information for revising the tutorial.

Giving people instructions that encourage them to use the manual or tutorial may be unrealistic in terms of what happens in the world outside the test laboratory, but it is necessary if your concern is the usability of the documentation. At some point in the process of developing the product, you should be testing the usability of the various types of documentation that users will get with the product.

At other points, however, you should be testing the usability of the product in the situation in which most people will receive it. Here's an example:

> A major company was planning to put a new software product on its internal network. The product has online help and a printed manual, but, in reality, few users will get a copy of the manual.
>
> The company planned to maintain a help desk, and a major concern for the usability test was that if people don't get the manual, they would have to use the online help, call the help desk, or ask a co-worker. The company wanted to keep calls to the help desk to a minimum, and the testers knew that when one worker asks another for help, two people are being unproductive for the company.
>
> When they tested the product, therefore, this test team did not include the manual. Participants were told that the product includes online help, and they were given the phone number of the help desk to call if they were really stuck. The test team focused on where people got stuck, how helpful the online help was, and at what points people called the help desk.
>
> This test gave the product team a lot of information to improve the interface and the online help to satisfy the concern that drove the test. However, this test yielded no information to improve the printed manual. That would require a different test.

Testing with Different Techniques

In most usability tests, you have one participant at a time working with the product. You usually leave that person alone and observe from a corner of the room or from behind a one-way mirror. You intervene only when the person "calls the help desk," which you record as a need for assistance.

You do it this way because you want to simulate what will happen when individual users get the products in their offices or homes. They'll be working on their own, and you won't be right there in their rooms to help them.

Sometimes, however, you may want to change these techniques. Two ideas that many teams have found useful are:

- co-discovery, having two participants work together
- active intervention, taking a more active role in the test

Co-discovery

Co-discovery is a technique in which you have two participants work together to perform the tasks (Kennedy, 1989). You encourage the participants to talk to each other as they work.

Talking to another person is more natural than thinking out loud alone. Thus, co-discovery tests often yield more information about what the users are thinking and what strategies they are using to solve their problems than you get by asking individual participants to think out loud.

Hackman and Biers (1992) have investigated this technique. They confirmed that co-discovery participants make useful comments that provide insight into the design. They also found that having two people work together does not distort other results. Participants who worked together did not differ in their performance or preferences from participants who worked alone.

Co-discovery is more expensive than single participant testing, because you have to pay two people for each session. In addition, it may be more difficult to watch two people working with each other and the product than to watch just one person at a time. Co-discovery may be used anytime you conduct a usability test, but it is especially useful early in design because of the insights that the participants provide as they talk with each other.

Active Intervention

Active intervention is a technique in which a member of the test team sits in the room with the participant and actively probes the participant's understanding of whatever is being tested. For example, you might ask participants to explain what they would do next and why as they work through a task. When they choose a particular menu option, you might ask them to describe their understanding of the menu structure at that moment. By asking probing questions throughout the test, rather than in one interview at the end, you can get insights into participants' evolving mental model of the product. You can get a better understanding of problems that participants are having than by just watching them and hoping they'll think out loud.

Active intervention is particularly useful early in design. It is an excellent technique to use with prototypes, because it provides a wealth of diagnostic information. It is not the technique to use, however, if your primary concern is to measure time to complete tasks or to find out how often users will call the help desk.

To do a useful active intervention test, you have to define your

goals and concerns, plan the questions you will use as probes, and be careful not to bias participants by asking leading questions. (See the section on "Interacting with Participants" in Chapter 19, "Caring for the Participants," for some useful hints on how to ask questions.)

Additional Benefits of Usability Testing

Usability testing contributes to all the benefits of focusing on usability that we gave in Chapter 1. In addition, the process of usability testing has two specific benefits that may not be as strong or obvious from other usability techniques. Usability testing helps

- change people's attitudes about users
- change the design and development process

Changing People's Attitudes About Users

Watching users is both inspiring and humbling. Even after watching hundreds of people participate in usability tests, we are still amazed at the insights they give us about the assumptions we make.

When designers, developers, writers, and managers attend a usability test or watch videotapes from a usability test for the first time, there is often a dramatic transformation in the way that they view users and usability issues. Watching just a few people struggle with a product has a much greater impact on attitudes than many hours of discussion about the importance of usability or of understanding users.

As Figure 2-2 shows, after an initial refusal to believe that the users in the test really do represent the people for whom the product is meant, many observers become instant converts to usability. They become interested not only in changing this product, but in improving all future products, and in bringing this and other products back for more testing.

Changing the Design and Development Process

In addition to helping to improve a specific product, usability testing can help improve the process that an organization uses to design and develop products (Dumas, 1989). The specific instances that you see in a usability test are most often symptoms of broader and deeper global problems with both the product and the process.

When we discuss analyzing the data in Chapter 20, we urge you to think about the breadth and depth of the problems. First, we urge you to ask how what you see affects the entire product.

To consider a simple example, let's say you've just completed a usability test in which participants sometimes got messages from the

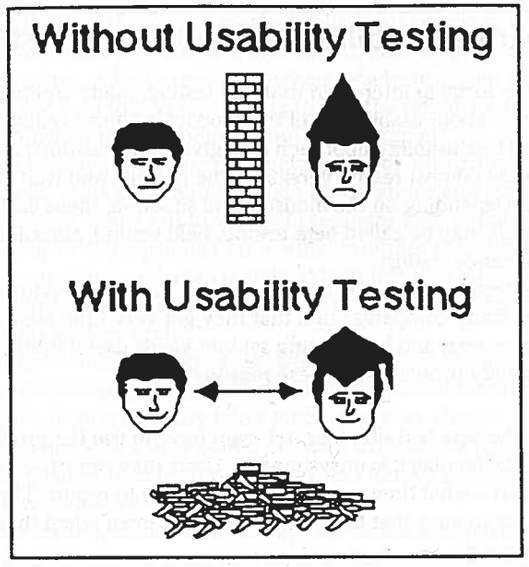

Figure 2-2. Usability testing can break down the wall between those who create the product and those who use it.

program, and they did not understand those messages. If you list just the messages that happened to come up during the test, the product team may be tempted to rewrite those specific messages.

A usability test, however, only samples parts of the product. You must ask, "What's wrong with the messages that users did not understand? What other messages are there in this program? Are the other messages like the ones that users saw and did not understand?" You would probably recognize a global problem with messages and recommend that all the product's messages be reviewed and rewritten.

Thinking globally about the product isn't enough, however. We also urge you to ask how the problems that you see came about. To continue with our example, if the messages aren't clear, you must ask, "How did they get this way? Who wrote them? If developers wrote them, what training did they have in writing for users? If there are technical writers in the company, why didn't they write the messages? If there aren't technical writers in the company, why not?"

"How" and "why" questions lead to discussions of process and ways to improve the process. Improving the process has much greater long-term impact on usability than improving the product. In Chapter 23, "Changing the Product and the Process," we suggest ways for you to stimulate changes in your organization to improve the process as well as products.

Comparing Usability Testing to Beta Testing

Despite the surge in interest in usability testing, many companies still do not think about usability until the product is almost ready to be released. Their usability approach is to give some customers an early-release (almost ready) version of the product and wait for feedback. Depending on the industry and situation, these early-release trials may be called beta testing, field testing, clinical trials, or user acceptance testing.

In beta testing, real users do real tasks in their real environments. However, many companies find that they get very little feedback from beta testers, and beta testing seldom yields useful information about usability problems for these reasons:

- The beta test site does not even have to use the product.
- The feedback is unsystematic. Users may report—after the fact—what they remember and choose to report. They may get so busy that they forget to report even when things go wrong.
- In most cases, no one observes the beta test users and records their behavior. Because users are focused on doing their work, not on testing the product, they may not be able to recall the actions they took that resulted in the problems. In a usability test, you get to see the actions, hear the users talk as they do the actions, and record the actions on videotape so that you can go back later and review them, if you aren't sure what the user did.
- In a beta test, you do not choose the tasks. The tasks that get tested are whatever users happen to do in the time they are working with the product. A situation that you are concerned about may not arise. Even if it does arise, you may not hear about it. In a usability test, you choose the tasks that participants do with the product. That way, you can be sure that you get information about aspects of the product that relate to your goals and concerns. That way, you also get comparable data across participants.

If beta testers do try the product and have major problems that keep them from completing their work, they may report those problems. The unwanted by-product of that situation, however, may be embarrassment at having released a product with major problems, even to beta testers.

Even though beta testers know that they are working with an unfinished and possibly buggy product, they may be using it to do real work where problems may have serious consequences. They want to do their work easily and effectively. Your company's reputation and sales may suffer if beta testers find the product frustrating to use. A bad experience when beta testing your product may make the beta testers less willing to buy the product *and* less willing to consider other products from your company.

You can improve the chances of getting useful information from beta test sites. Some companies include observations and interviews with beta testing, going out to visit beta test sites after people have been working with the product for a while. Another idea would be to give tape recorders to selected people at beta test sites and ask them to talk on tape while they use the product or to record observations and problems as they occur.

Even these techniques, however, won't overcome the most significant disadvantage of beta testing—that it comes too late in the process. Beta testing typically takes place only very close to the end of development, with a fully coded product. Critical functional bugs may get fixed after beta testing, but time and money generally mean that usability problems can't be addressed.

Usability testing, unlike beta testing, can be done throughout the design and development process. You can observe and record users as they work with prototypes and partially developed products. People are more tolerant of the fact that the product is still under development when they come to a usability test than when they beta test it. If you follow the usability engineering approach, you can do usability testing early enough to change the product—and retest the changes.

Comparing Usability Testing to Research Studies

In our experience, many people who are new to usability testing confuse it with social science research. Managers sometimes put too much faith in the fact that usability testing yields quantitative data. They sometimes reject recommendations from an expert reviewer because the reviewer, cannot provide quantitative data. They sometimes base major decisions on just a few of the numbers from a usability test without considering the range of data that a usability test provides.

Usability testing and scientific research are both "empirical" methods, that is, they both focus on observations of actual behaviors (Dumas, 1988; Gould & Lewis, 1985; Rubenstein & Hersh, 1984; Shneiderman, 1992). They are, however, also very different. Whether you are a tester or a user of test results, understanding the differences between testing and research should help you report and apply test results appropriately.

On the surface, a usability test looks like a research study:

- You often conduct tests in a facility, called a *laboratory*, that looks very much like a research laboratory.
- You sample participants who are representative of the population of interest.
- You often take some steps, in selecting or training participants, to control variables that might otherwise make interpreting the results difficult.
- You record measures, both objective and subjective.
- You analyze the data and write a report that uses empirical data to back up findings.

But the similarities are more apparent than real.

Focusing on Different Goals

The major difference between a social science research study and a usability test is in their goals. The goal of a research study is to test whether or not some phenomenon exists. To make that decision, the test must be done with a sample size large enough to detect the phenomenon if it is present. The ways to estimate how large the sample must be to make a proper decision fill large sections of statistics texts.

The goal of a usability test is to uncover problems, not to demonstrate the existence of some specific phenomenon. Experience has shown that you can uncover most major problems in a usability test with relatively few participants (Virzi, 1992). You cannot, however, then apply the same statistical tests to the data that you can in a social science research study.

Using the Same Laboratory

Usability tests and research studies may take place in similar physical settings. We and others conduct both usability tests and research studies in the same laboratory. Use of the same physical space, however, doesn't make the methods the same.

Selecting Participants Differently

Throughout this book, we stress that one of the criteria for conducting a valid usability test is to select participants from the population that will use the product. We do not, however, talk about this selection as "scientific sampling."

In research studies, the researchers often go to elaborate lengths to ensure that they have a *random* sample of participants from some relevant population. In fact, the statistical tests that are applied to research data assume that the participants were selected through some random process. In usability testing, you usually have a *convenience* sample—people from the appropriate population whom you happen to find and who are available to you.

Controlling Fewer Variables

In social science research, much skill and creativity go into isolating the specific variable or variables that you want to study and controlling for the influence of all other variables. In a usability test, you try to exert some control over confounding variables, such as level of computer experience, that might make the results difficult to interpret, but with only a few participants from a convenience sample, you cannot exert the level of control over confounding variables that you would work toward in a research study.

In a usability test, you also cannot usually isolate specific variables. The "independent variable" you want to study is usually the interface. Trying to isolate one specific variable within the interface will not give you the information you need to see how it actually works for users.

As a consequence, it is not always clear what is causing a problem. When participants select wrong menu options, is the cause a poorly organized menu hierarchy, poorly worded options, the participants' experience with a previous version of the product, or some combination of these possible causes?

In a usability test, in fact, the problems that you see often stem from multiple causes. To identify the problems and understand the causes, you must consider several measures together, not just the quantitative data. You bring your expert knowledge of human–computer interactions and document design to bear along with your observations, participants' comments, and the quantitative data. (We describe this process in detail in Chapter 20, "Tabulating and Analyzing Data.")

Weighing Observations More

Because a usability test is an empirical method, the test team carefully records the data of interest, such as the number of errors and the length of time it takes to complete a task. The participants typically also fill out questionnaires and give their opinions.

These are the same measures that are often recorded in a social science research study. In fact, the same performance measure could be collected in a research study and a usability test. In a usability test, however, the observations of the test team and the comments of test participants are often given more weight in diagnosing problems than they are in a research study.

Analyzing and Reporting Data Without Inferential Statistics

Most usability tests include descriptive statistics, such as means (averages), medians, ranges, and frequencies. In Chapter 20, "Tabulating and Analyzing Data," we discuss the use of inferential statistics, such as t-tests. For research studies, the results of inferential tests are frequently at the heart of the discussion of the data. Inferential statistics are, however, seldom appropriate for usability tests.

In summary, the greatest similarities between a usability test and a research study are in the physical setting in which they take place and in the types of data you collect. The other similarities are more apparent than real. The major difference is that they are serving different purposes.

3

Uncovering Usability Needs Before You Design

June 2, 1993

Ms. Jane Smith
Ajax Industries
Anycity, State

Dear Ms. Smith:

RE: Confirming your schedule to evaluate an electronic mail program

Thank you for agreeing to help us evaluate a new electronic mail program. We expect you at this time and place:

DATE: Wednesday, June 16, 1993

TIME: 9 AM

PLACE: American Institutes for Research
45 North Road, Bedford

You can reach us easily from Route 95. Take the Exit marked "Route 4, Bedford" and continue down Route 4 for three miles. I have also enclosed a map.

Expect to spend about three hours with us. We'll ask you to use the product to do just the types of tasks you might do in your work. Our client wants to find out how to improve this product and is eager to get feedback from people like yourself who would typically use a product like this.

As I explained, we'll videotape the session. We'll only use the tapes to help us evaluate the product; and we won't use your name. We'll also ask you to think out loud as you work with the product. That way, you can tell us how the product is working for you.

We have only one person at a time come to work with the product, so we are counting on you to come. If you will be unable to come, please call me at least two days in advance so we can reschedule. You can leave a message on our phone any time of the day or night. We look forward to seeing you. **You will receive $75 for helping us.**

Sincerely yours,

John Doe
Usability Specialist
PHONE: 617 - 275 - 0800

Figure 10-4. Sample confirmation letter

11

Selecting and Organizing Tasks to Test

One of the essential requirements of every usability test is that the test participants attempt tasks that users of the product will want to do with it. When you test a product of even modest complexity, however, there are more tasks than there is time available to test them. As you plan, you need to decide which tasks are the most important to include in the test. In this chapter, we describe how to:

- select tasks to test
- determine the resources you need for each task
- set priorities and order tasks

Selecting Tasks

Usability testing is a sampling process. You cannot test every possible task users can do with a product. What tasks, then, should you sample?

- Tasks that probe potential usability problems
- Tasks suggested from your concerns and experience
- Tasks derived from other criteria
- Tasks that users will do with the product

Tasks That Probe Potential Usability Problems

The first and most important criterion for selecting tasks is to use tasks that probe the potential usability problems with the product.

A good task to select is one that has the potential to uncover a usability problem.

As with any testing procedure, the more problems you find in the limited time available, the more successful your test will be (Myers, 1979). The designers of the product may be surprised at this goal. They may view usability testing as a way to verify the usability of the product, that is, how easy it is to learn and use. You may need to remind them that the goal of quality assurance testing of software is to find the "bugs" before the product is released. The quality assurance process *assumes* that any complex software program has bugs in it. A good software testing procedure finds more bugs than a poor one. The goal of a usability test is similar: develop a procedure that will find the serious usability problems. Consequently, you look for tasks that will probe areas of potential usability problems.

Tasks Suggested From Concerns and Experience

Another source to use to identify tasks is the list of usability concerns you develop with designers. We have discussed this list in Chapter 8, "Defining Your Goals and Objectives." The people who develop a product always have some ideas about where there are potential problems. They know what parts of the product were difficult to design and where they disagreed about the best approach.

Designers are often correct in knowing the kinds of *major* problems the test should probe. For example, if they made a major decision to select an interaction style such as touch, they may want you to look at the types of problems that users frequently have with touch-sensitive devices, such as touching outside of the touch pad.

> For our electronic mail example, there are several potential usability problems we identified as concerns. For example, we suspect that users would have difficulty selecting a new mail message to read. When you have more than one new message to read, the software always displays the newest message first. If you do not want to read that one, you must select the one you want by using the arrow keys to move a highlight bar down the list of messages and pressing the space bar to mark it with an arrow symbol. Because the space bar is an arbitrary mapping here, we expect users to have trouble marking a message.
>
> We also suspect that users will have problems creating folders, because the procedure for creating your first folder is different from the procedure you use to create additional folders. When you have no folders and you try to save a message, the software asks you to create a folder. All of the messages you save will go into that folder until you create others. The procedure to create the additional folders requires a different process with a different menu. It seems likely that users will have trouble with this task.

While designers have useful ideas about where users will have problems, they often ignore the more basic problems users might have with the product. They may anticipate that users will have difficulty configuring software, but not that users will have difficulty starting a system and putting in a diskette.

There is a famous videotape made during a usability test at Digital Equipment Corporation. The tape shows how two naive users do not understand what a floppy diskette is and how to insert it into a disk drive. The designers had not anticipated that users would not know how to work with diskettes. This finding made it possible for designers to improve the design of the hardware so that users who have no experience with diskettes are less likely to have problems starting to install the product.

> For our electronic mail example, there are not enough status messages to tell users that they have accomplished an action. For example, when you send mail, the software displays a message indicating that the mail was sent or forwarded or replied to. The message, however, flashes so quickly on the screen that we expect users will miss it. We were concerned that some users will send mail several times when they are not sure it has been sent.

Besides probing problems that designers identify and the basic problems users often have with complex interfaces, there may be

other potential problems. Your experience as a tester is invaluable here. In a very short time, you will see the common problems in the products you test. For example, if you know that users frequently have problems manipulating windows in applications that sit within a multitasking user interface, you will want to include some tasks that require participants to open, close, move, and resize windows.

Your list of potential problems may grow longer than you could possibly test. Do not worry about the length of the list at this point. You will shorten it later.

Once you have at least a preliminary list of problems, you need to begin to create tasks that will probe each problem. For example:

- As we described above, we expect users of our electronic mail system to have difficulty identifying a new message to read. We, therefore, want to include a task that requires users to select a new mail message to read that is not the oldest message in the list of new messages.
- The designers of the interface to a new patient monitor for a critical care unit in a hospital used a dedicated key allowing users, here doctors and nurses, to temporarily silence all alarms. When users press this key, a message appears in a message area near the top of the screen that says "Alarms silenced." It is very important that nurses and doctors see this message, even when they themselves did not silence the alarms. The designers are not confident that the highlighting, size, and positioning of this message will allow users to notice it. To probe this question, you would need a task that requires participants to silence the alarms and you would need to find out if they could verify that the alarms were silenced. You also might want a second task in which the alarms have been silenced by someone else, and you ask the participant to describe the state of the alarms.
- The writer of a manual for an update to an operating system is concerned about the commands that were modified from the earlier version of the operating system. These commands change the way that computer operators will perform some of their most frequent tasks, such as system configuration. The writer wants to know whether participants can find the appropriate sections in the manual quickly and whether they can then follow the new procedures. To probe this question, you would need to include tasks that ask participants to do the tasks that require using the new commands.

Tasks Derived From Other Criteria

In addition to selecting tasks because they relate to usability problems and concerns and goals, you can use other criteria for selecting tasks

for a usability test. Good choices are tasks that are difficult to recover from if done wrong.

Tasks That Users Will Do With the Product

All of the tasks you have selected thus far will be tasks that users will do with the product you are testing. There are other tasks that users can do with a product over and above those that relate to usability problems, concerns, or goals. For example:

- new or modified
- critical to the operation of the product
- frequently done
- done under pressure

If the development team has conducted a task analysis, you can extract the tasks from it. Otherwise, it is best to have a meeting with the developers, and with users as well, to create a list of the tasks.

Here is a list of tasks to test from our electronic mail example:

Set a password for the mail account
Create a personal mailing list
Move messages between folders
Sort items in folder
Read any new mail message
Read a selected mail message
Create and send a mail message
Edit a message
File a mail message
Create a folder
Forward a message
Delete a mail message
Delete a folder
Find a mail message
Send a CC ("Carbon Copy") of a message
Reply to a mail message
Attach a file to a message
Create a distribution list
Set the useful life of messages
Archive a message/folder
Set priorities on messages
Create a folder that several people can share
Sort the messages in a folder by date

As you can see, this list is too long for a half-day usability test. We will need to pare it down, but before we do that, let's look at the resources we need to test each task.

Determining the Resources You Need for Each Task

Before you can start eliminating tasks from the list, you need to get some additional information about each task:

- how long it will take to do the task
- what hardware, software, procedures, and other resources will you need to run the task

You do not have to create this information in minute detail at this point. But, you will need to have a general idea about time and resources to make decisions about eliminating tasks. You also will be using this information to create checklists to help you keep track of all of the resources you will need to conduct the test (see Chapter 16, "Preparing the Test Team").

Estimating Task Times

There are two types of task times that you need to estimate as you prepare for a test:

1. The time it will take to run the task during the test.
2. The time that users will feel is acceptable for completing a task.

In this chapter, we are concerned about the first time, that is, how long it is likely to take for a test participant to complete a task. We need to estimate this time so we can plan how long each test session will take.

The second time is important for setting quantitative criteria for a task. As we discuss in Chapter 13, "Deciding How to Measure Usability," you want to estimate how acceptable a task time is to *users* so that you can set criteria for measuring the usability of the task.

It is important that you understand the difference between these two times. The fact that you estimate that it might take a test participant 20 minutes to complete a task does not mean that users will find 20 minutes acceptable.

Use a time estimate for the participants who are most naive and who are likely to take the longest time.

As you try to decide *how long each task will take during the test*, consider the characteristics of the participants who will be in the test.

Your estimate will only be an educated guess, but you need to make it anyway. As you develop these estimates consider

- the time it would take an experienced person to complete the task—this will provide a baseline
- the problems that a typical participant might have and the time it will take to recover from them

- any additional time it might take to set up the task or to get ready for the next task

Your estimate may take the form of a range rather than a single value. Here is an example.

> You are testing the user interface and the documentation for an upgrade to an operating system for a minicomputer. A task you want to test is to do a partial weekly backup, that is, you will ask the participants, who will be computer operators, to backup all the files that have been created over the past 7 days. How long is it likely to take to complete this task? Clearly, the minimum time is the time it would take an expert operator who knew the commands to type them, load a blank tape, wait for the files to be copied, and then rewind and remove the tape.
>
> The major unknown variable here is the number of files there are to back up. You have control over this variable, because you have to make the files for the test. If you want to test the condition in which more than one tape is used for the backup, you will need enough files to fill more than one tape. Suppose that you decide that you will have enough files to make a 10 minute backup. It will take an expert 30 seconds to type the commands, 2 minutes to select and load a tape, about 10 minutes for the files to be copied, and 2 minutes to rewind, remove, and label the tape. Thus, the minimum time to do this task is about 15 minutes.

In our experience, estimates for task times become more unreliable as the number of usability problems with the product increases. If you see that there may be many usability problems with the product, plan to test fewer tasks. You may want to wait until the pilot test before you decide how many tasks you can test.

If you have set an upper time limit for a task, you should plan to allow the test participants to work on the task for at least that amount of time.

Listing the Resources You Need for Tasks

Once you have generated an estimate for the time it takes for tasks, list all the resources you will need to test the task. This list is also critical to your planning for the test. It will provide you with the material for your testing checklist to keep the test team organized. You need to create this list now to make decisions about whether to include tasks in the test. As you develop this list, consider:

The hardware you will need. Include not only the hardware the participant will need to attempt the task, such as a personal computer with a printer, but also the hardware you will need to set up and run the task. For example, for a test of the user interface to

- electronic mail software, you will need a computer or terminal connected to at least one other computer or terminal to send and receive mail. If you are testing a remote service you also may need a modem and a dedicated telephone line into the test room
- a workstation in a network, you will need at least one other workstation or printer to serve as a remote node, as well as cables and connectors
- a digital oscilloscope, you will need signal generators, cables, and connectors to simulate the electrical signals you will need to have for participants to measure
- a business telephone, you will need at least one phone for the participants and one phone for the observation room for sending and receiving calls. These phones may need to be connected through the same switching unit.

The software and data files you will need. Again, include both the software and data the participant will use and the software and data you use to conduct the test. For example, for a test of the user interface to

- electronic mail software, you may need
  - a mail account, with appropriate privileges, set up for the test participants
  - additional mail accounts for the people participants will send messages to and receive messages from
  - messages in the participants' mail account that the participants will sort, retrieve, attach other messages to, etc.
  - mail folders to place messages in
  - group distribution lists
  - a backup copy of all messages to use after each participant completes the test
- database management software, you may need
  - a relational database with sort keys
  - tables and other views of the data
  - histograms, pie charts, and other graphics to edit
  - a backup copy of the database and files to use after each participant completes the test
- business telephone system, you may need
  - an intercom number for the participants' phone and any other phones they will call or receive calls from
  - a list of waiting messages
  - a list of other users of the phone system
  - phone messages that have been archived
  - a backup copy of waiting messages for use after each participant completes the test

Instructions and procedures you will need. To attempt the tasks in a test, users need such information as names for messages and names of people to send the messages to. In addition, you will need procedures and tools to help you set the product up the same way for each participant. For example, for a test of the user interface to

- an electronic mail system, you might need
  - a user name and password for participants to get into their mail account
  - a message for participants to type
  - a reply for participants to type
  - a procedure for you to use to put the hardware, software, and data files back to their initial state after each participant completes the tasks
- a business telephone system, you might need
  - intercom extension numbers and outside telephone numbers for participants to call
  - names of fictitious office mates and other people participants will talk with
  - scripts for test team members to use when they interact with participants
  - a procedure for you to use to put the hardware, software, and data files back to their initial state after each participant completes the tasks

Creating the Final List of Tasks to Test

To summarize what you have created thus far, here is what a list of resources might look like for a task to send a reply to a message in a test of our electronic mail example:

Time	3–12 minutes
Hardware	Two PCs connected to network
Software and data files	Electronic mail software Mail account for participants Incoming mail message to reply to Existing account to send reply to
Instructions and procedures	Text of reply (or information to create text) Procedure for restoring account to initial condition

Once you have completed your list of tasks and resources, you will usually need to eliminate some tasks, because participants will not

have time to complete all of them. For most tests it takes about 1 hour to

- start the test
- give the prebriefing
- give the participant a break halfway through the test
- have the participant fill out the posttest questionnaire
- give the debriefing

To complete the tasks, therefore, participants will have about one hour less than the total time they will be in the test. For example, if you plan to have participants available for half a day or about 3½ hours, they will only have about 2–2½ hours to work on tasks.

Scheduling participants for half a day is a common practice. For products that are even moderately complex, you need to give participants at least 2 to 3 hours to explore the product. If you are testing a prototype or only part of the product, you may be able to spend less time with each participant.

Scheduling participants for more than half a day will cause you some logistic problems. You must take participants to lunch or arrange for them to get lunch. You also have to be concerned about fatigue. As we discuss in Chapter 18, "Caring for Test Participants," usability testing is an emotional experience for participants. Many of them will be tense; they will need frequent breaks to retain their concentration if they have to perform for more than a few hours.

As you look for ways to eliminate tasks so that participants can complete the test on time, consider

- the objective of each task and whether you can achieve more than one objective with it. For example, an objective such as having participants navigate three levels down a menu hierarchy can usually be combined with an objective for performing some function that is on the third level of the hierarchy.
- whether a task that uses expensive resources is worth testing. For example, if you need an extra signal generator to simulate a complex signal to an oscilloscope, consider if the objective of that task is critical to the evaluation of the product
- whether time-consuming tasks are more important than the two or three tasks you could include instead

Putting the Tasks in Order

The final step in creating the list of tasks is putting them in the order you will have the participants attempt them. There are two important points to consider here:

1. The tasks should flow in the natural order in which users will do them. For example, users will create a mail message before they edit and send it.
2. Tasks that are important to the evaluation of the usability of the product should come early in the test rather than near the end. It is likely that some participants will not finish all the tasks. So, if you leave important tasks until late in the test, you will collect less data on them.

From the long list of tasks we generated for our electronic mail example, we selected the most important tasks and arranged them into an order that seemed to make sense from a user's perspective:

Read a selected new mail message
Create a mail message
Edit a message
Send a message
Create a folder
Store a message in a folder
Forward a message
Find a message
Delete a message
Reply to a message
Attach a file to a message
Create a distribution list
Set the useful life of a message
Archive a message
Make a shared folder
Set special priorities on a message

As you can see, the tasks near the beginning of the list are the ones that we suspect users will want to do most often, while the tasks at the end of the list are infrequently used. If test participants are not able to finish all of these tasks, they will have tried the most important ones.

In the next chapter, we will discuss turning tasks into scenarios.

12
Creating Task Scenarios

Once you have the list of tasks for the test, you have to decide how to present those tasks to the participants. One way that works well is to give the participants "scenarios"—situations in which the task is embedded in a reasonable and very short story. In this chapter, we look at the art of creating interesting and informative scenarios. We discuss these questions:

- What is a scenario?
- What makes a good scenario?
- Do you always give participants written scenarios?
- How do you divide up the tasks and scenarios for participants?
- How do you make participants stop between tasks?

What Is a Scenario?

You use scenarios to tell participants what you want them to do during the test. Scenarios describe the tasks in a way that takes some of the artificiality out of the test.

Here are two examples of scenarios:

Scenario 1:
You've just bought a new combination telephone and answering machine. The box is on the table. Take the product out of the box and set it up so that you can make and receive calls.

Scenario 2:
You can save numbers in the telephone's memory and then call those numbers without dialing all the digits each time. Your best friend's number is 212-555-1234. Put your friend's number in the telephone's memory.

Here's an example of a scenario for our usability test of the electronic mail program:

Scenario 1:
You have just arrived at your desk after a short vacation. Check to see how many mail messages you have waiting for you. If there are any messages from Mr. Green, a Vice President of your company, read them.

As you can see, a scenario makes the task more realistic. In a scenario, you give the goal and whatever information a user would actually have when coming to do this task. You do not give the steps. The point of the test is to see if a typical user can figure out the steps that this product requires.

What Makes a Good Scenario?

A good scenario is:

- short
- in the user's words, not the product's
- unambiguous—so all participants will understand it

A good scenario:

- gives participants enough information to do the task
- is directly linked to your tasks and concerns

Short

Time is precious in a test. You do not want participants to spend more time than necessary reading the scenarios.

People also read at different rates. Because you are timing the task, you do not want these different reading speeds to unduly influence the task times. The way to keep that difference small is to write short scenarios.

Depending on the software or other means you are using to time the tasks, you may or may not be able to separate the time participants spend reading the scenario from the time they spend doing the task. See Chapter 15 on "Preparing the Test Environment" for more information on measuring time.

Most scenarios can be kept very short. If you set up the entire test so that the participant has a consistent and plausible "role," you can write each scenario to fit into that role.

Here are a few more examples of scenarios:

Scenario:
There have been some staff changes in your office.
Set up a new account for E. Dickenson.

Scenario:
You have successfully completed installing the compiler.
Test it with the program DEMO.FTN

Scenario:
You need to make a presentation to your manager about the month's sales figures.
Copy the spreadsheet SALES1.WKS into the SALES table.

In the User's Words, Not the Product's

The whole point of usability testing is to predict what will happen when people use the product on their own — without a developer or usability specialist looking over their shoulders or answering their questions.

If you are testing a menu-driven product, one of the concerns you are probably testing is whether users will choose the right menu option. If you are testing a graphical user interface with icons, one of your concerns is likely to be whether users will select the right icon. If you are testing a device with labeled buttons, you are likely to be concerned about whether users will know which button does what.

Don't give that information away in your scenarios. If you do, the product may do well in testing and still fail in the marketplace.

> Suppose you are testing a product that lets users save their mail in "logs." The menu choice for this is "log." If you write a scenario that says, "Now log the mail that you just read," the participants may successfully do the task without even understanding what task they are doing. When other users get the product in their offices, they may never store their mail because they make no connections between the task as they say it to themselves and the menu choice "log."
>
> To test this task realistically, you must write a scenario that describes the task the way users will say it to themselves. That might be: "Now store the mail you just read." Or it might be: "You don't want that message to sit in your list of mail forever, but you may want to read it again later. Save it so that you could get it back to look at later." Now you have a realistic test of whether users recognize the product's word "log" as the right one for the task they represent to themselves as "storing" or "saving."

Scenarios that are worded too closely to the product may lead to a false positive test. That is, the product may fare well in the usability lab and still fail in the marketplace.

This issue of making sure the scenarios are worded well is very important. However, don't go overboard. If you have done a lot of previous usability work to be certain that you are using plain English, users' terms, don't make up new, unusual terms for the scenarios.

Developers may have a hard time seeing this problem because the words in the product are so familiar to them that they cannot imagine users not understanding the terms. A usability specialist should either write the scenarios or review them carefully, looking for words and phrases that are product words and might not be the ones users would look for.

Unambiguous – So All Participants Will Understand It

You're trying to see how easy the product is to use. You don't want misunderstandings about the scenarios to interfere with learning as much as you can about the product.

When writing scenarios, watch out for these common pitfalls:

- not being clear about when the task is over
- not giving the information people need to do the task

In this scenario, for example, where are you supposed to stop?

> *Ambiguous scenario:*
> You have a message from Jane Jones about the Fourth Quarter Budget. Read it and write a response in which you answer her question.

Should you send the reply or just compose it and wait for further instructions?

If any of the scenarios are ambiguous, you are likely to see the problem in pilot testing. This is one reason for conducting a pilot test, as we explain in Chapter 17.

Enough Information to Do the Task

In addition to telling participants the general task or goal that you want them to accomplish, you may need to give users some data to work with. If, in a real situation, they would have information about a specific case when they come to do the task, you have to supply the information for such a case.

Suppose you are testing a new touchscreen product for retailers. You've set up a general scenario for your participants so they are in the role of a salesperson in the linen department. One of the tasks you are concerned about is whether they can handle a sale that includes multiple items. You have to give them a situation in which that task will occur.

If you are giving the scenarios in writing, you might create one like this:

> *Scenario:*
> A customer comes up to your station with several items.
> She wants to buy them on her store charge card.
> She gives you her store charge card (Account #9-80-786-5).
> And she gives you the items, which are
> - two twin-bed-size sheets. Each sheet is item #347689.
> - a package of pillow cases. The package is item #346988.
> - two pillows. Each pillow is item #456897.
>
> Ring up the purchase for her.

Note: For this test, a better way to deliver the scenarios might be to have team members pretend to be customers. For this scenario, someone would walk up to the participant/salesperson with the items and a store credit card. See the section, "Do You Always Give Partici-

pants Written Scenarios?" later in this chapter. You still have to plan all this information and make sure that the database you are using for the test has that customer and those items in it. Otherwise what you are really testing is how users deal with the messages they'll get when they try to enter a nonexistent account number or incorrect item numbers.

Note that you haven't told the participants *how* to do the task. You've only given them the same information that they would have if they were doing such a task in real life. That's the key to a useful scenario: the participants should feel as if the scenario matches what they would have to do and what they would know when they are doing that task in their actual jobs.

You have to give the participants enough data to be able to do the task. Don't give them extra data unless they are likely to actually have that data when doing the task and you are specifically testing a concern about whether they will know which data to use in the task.

You don't always have to spell out all the data. For example, if you are testing an editing program and you want the participants to write a note, you don't have to give them the exact wording of the note, but you should give them some information to write about.

We find that if you give nonspecific instructions, such as "Write a note to Brett Jones at Headquarters," some people will spend a great deal of time deciding what to write about. That throws off your comparison of time for the task you are testing. We also find that if you give participants all the words to type, however, the task is boring. What works best is to tell them something like this: "You need to let Brett Jones at Headquarters know about the staff meeting next Friday at 2 p.m. in the main conference room." With this much information, you will get to observe how each participant goes about addressing and composing a note—which you want to know—and yet not find large disparities in the time they spend deciding what to say—which you do not want to know.

Directly Linked to Your Tasks and Concerns

You should be able to name the task or tasks that go with each scenario.

Each scenario tests one or more of the tasks on the list that you decided to include in the test. Each of those tasks, in turn, is directly linked to one or more of your goals or concerns.

You don't give these task names to the participants. They get only the scenarios. But the team should know exactly what each scenario is testing. Sometimes the task–scenario link is obvious from the wording of the scenario, but sometimes it is not.

The following is an example in which the link is obvious:

Concern:	Can users figure out how to use the remote control to turn off the television set?
Test setup:	Participants have been told to use the remote control for all tasks.
Task description:	Turning off the TV.
Scenario:	You've finished watching the TV for now. Turn it off.

In this second example, the link between the task and the scenario is not obvious from the wording of the scenario:

Concern:	Will users understand the messages on the small one-line screen of the photocopier?
Test setup:	The photocopier has only two sheets of paper in the bin that will be used in this task. Packages of different size paper are available on a supply table nearby, but the participant has not been told that they will be needed.
Task description:	Figuring out what is wrong; adding paper correctly.
Scenario:	Please make one copy of this five-page paper.

Do You Always Give Participants Written Scenarios?

No, you don't always give the scenarios in writing. You want to make the test as realistic as possible, and another mode of delivering the scenarios may be more realistic than having the participants read them. It might be appropriate to

- have test team members pretend to be "customers," "supervisors," or "colleagues" and walk into the test room to deliver the scenarios in person
- use the product to deliver the scenarios
- For example, in testing telephones, the scenarios would

probably include telephone calls to the participant as well as calls the participant makes.
- mix the modes; give participants scenarios in writing, but then interrupt with calls or visits

Whatever mode you use to present the scenarios, however, you must write them out as part of your test plan. Even if the participant never sees the written scenario, you must be sure that each participant gets the same scenario delivered in the same way. Therefore, if test team members are play-acting scenarios over the phone or in person, they have to use the exact same scenarios with each participant. That may mean memorizing the situation and words as if this were theater.

The following are some examples of test situations that have lent themselves to different modes for delivering the scenarios:

- To test a new part of the airlines reservation and check-in system, the test team at American Airlines set up a ticket counter in the usability laboratory. The participants were representative gate agents. They stood behind the counter as they do on their jobs. The scenarios were delivered by test team members acting as travelers.
- To test a new multifunction telephone, the test team at AIR set up the usability laboratory to look like an office. The participants were representative office workers. They were each given a set of written scenarios, but the test team knew that not all the scenarios were included in the participant's package. The test team's version of the package included some scenarios that the test participant did not see, such as: "Call the participant on the phone and say: . . . ."
- To test an objective that users be able to easily suspend what they are doing and turn to a different task, the team testing an electronic mail package decided to interrupt participants in the middle of writing a memo. The participants started what they thought from the written scenario was a task to send a memo on a particular topic to a particular person. A person on the test team had instructions to interrupt over the intercom after the participant had addressed the memo and typed about two lines of the message. The prescripted interruption gave the participants a plausible reason to want to stop working on the memo they were writing and instead immediately send a different message to a different person.
- AIR conducted a test of a monitor that nurses would use at a central nursing station on a hospital floor. One of the concerns was whether nurses would know what to do when two alarms about patients went off at the same time. The nurses got a scenario that told them to monitor their patients' status. While they were doing that, two alarms went off. Preparing for that scenario required writing software to make the alarms sound during that task.

Match the mode of presentation to the situations in which the product you are testing will be used.

How Do You Divide Up the Tasks and Scenarios for Participants?

If you have a list of 15 tasks that you want to test, you may have exactly 15 task scenarios for the participants, but you may have more or less. You do not have to have a one-to-one correspondence between tasks on your list and your scenarios.

You may have more scenarios than tasks if you want to test the same task more than once. One of your concerns may be how quickly users will learn to do a particular task. You may hypothesize that users will make a few wrong icon choices the first time they try something, but then, after that, they'll know which icon to select and won't make any errors the second time they need to use that icon. In that case, you might have two or three scenarios for the same task in your test.

You may have fewer scenarios than tasks if you combine two or three tasks and have participants do them together.

In deciding how to match up scenarios and tasks, the major issue you have to consider is whether you want separate measurements of time, errors, or other codes for particular tasks. If the software program that you are using to measure the participants' performance can automatically give you time only between the code for "start task" and the code for "stop task," you have to give each task for which you want a separate time as a separate scenario. An example may help to explain this point more clearly:

> Let's say you are testing a project management program. You are concerned that it will be accepted by the company's managers only if it is faster to use than keeping their manual records. You know that speed in doing the tasks is dependent not only on the system's response time, but also on how quickly managers find the right menu choices and the right fields to fill in or change. Time is one appropriate measure of the ease of use of this product.
>
> Your task list for this test includes these three tasks that project managers have to do:
> - get a project's file
> - make changes to the schedule
> - add a new person to the project team

You could write a scenario that had your project manager participants do all three of these at once. In that case, you would automatically know only the total time, total errors, and total frustrations for the combined set of three tasks.

However, you might want to know how much time it takes the participants to do each part, how many errors they make doing each part, and whether one of these tasks is more frustrating than the others. In that case, you would create three scenarios. Participants would stop at the end of each one. The data-lo gging software would give you the time, error count, and frustration count for each task, that is for each scenario, separately.

In the first case, the one in which you have the participants do all three tasks as part of one scenario, you would give the participants a page that might look like Figure 12-1.

Task 1

You are the project manager for the BOOK project. You've just found out that there's been a change in schedule and staffing for the project and you have to make the computer files reflect those changes.

- Get the BOOK project on your screen so that you can change it.
- Deliverable #3 is going to be two weeks late. Change the file to reflect that.
- To make even that new deadline, your boss has authorized you to add another designer, Jed Brown, full time for two weeks starting tomorrow. Jed earns the same salary as Betsy Moore. Add Jed to the project staff.

Figure 12-1. Three tasks as one scenario, timed together

In the second case, in which you want to know separate times and other counts for each task, you would give the participants the same instructions, but you would separate them into three tasks, giving each task as a separate scenario on a separate page. It might look like Figure 12-2.

Task 1

You are the project manager for the BOOK project. You've just found out that there's been a change in schedule and staffing for the project and you have to make the computer files reflect those changes.

Get the BOOK project on your screen so that you can change it.

Task 2

Deliverable #3 is going to be two weeks late.

Change the file to reflect that.

Task 3

To make even that new deadline, your boss has authorized you to add another designer, Jed Brown, full time for two weeks starting tomorrow.

Jed earns the same salary as Betsy Moore.

Add Jed to the project staff.

Figure 12-2. The same three tasks as separate scenarios, timed separately

How Do You Make Participants Stop Between Tasks?

If you want to time the tasks separately, you have to get the participants to stop between each task. In some organizations, the usability testers give participants the tasks one at a time. That way,

they always know when the participant has finished one task and when the participant begins the next task. If you run your test that way, you can also ask questions about the task, conducting a mini-interview after each one.

However, if your test consists of many short tasks, going in to the participant between each one may be disruptive. In a typical test in the AIR lab, we give the participants a booklet of the scenarios that we want them to do during the test. Each scenario that we want to time separately is on a separate page. Because "task" is an easier and clearer word for participants than "scenario," we label the pages as "Task 1.," "Task 2.," and so on.

To remind participants to stop at the end of each task, each page includes two sentences near the bottom:

> Please tell us when you have finished this task.
> Please wait for us to tell you to turn the page.

At the beginning of the test, the person who interacts with the participant, the "briefer," also reminds the participant:

- to wait for the briefer to say when to begin the first task
- to say out loud when the task is done
- to wait again between each task until the briefer says to go ahead

Putting the scenario for each task that you want to time separately on a separate page is one way to get participants to stop between each task. The entire test team will appreciate having a few moments between each of the participant's tasks to finish their own work and be ready to observe again.

When we want to get participants' reactions to each task, we include a short posttask questionnaire in the booklet after each task. You can see an example of a posttask questionnaire in Chapter 14 on "Preparing Test Materials." A posttask questionnaire, like the mini-interview that some teams do between tasks, allows you to get the participant's immediate reaction to each task. Answering a few questions after each task also gives the participant something to do while the test team finishes logging their own comments.

Now that you have turned your task list into the specific scenarios that you'll give to the participants, we'll turn to the question of what you want to observe and measure while participants are working with the product.

13

Deciding How to Measure Usability