CHAPTER
3

When Should You Test?

Some type of usability testing fits into every phase of a development lifecycle. The type of testing is distinguished by the research questions asked, the state of the completeness of the product, and the time available for implementing solutions to problems revealed in testing. This chapter outlines four types of tests that fit into the general phases that any product development cycle goes though (see Figure 3-1).

Our Types of Tests: An Overview

The literature is filled with a variety of testing methodologies, each with a slightly different purpose. Often, different terms are used to describe identical testing techniques. Needless to say, this can be extremely confusing. In deciding which tests to discuss and emphasize, the most beneficial approach might be to use the product development lifecycle as a reference point for describing several different types of tests. Associating a test with a particular phase in the lifecycle should help you understand the test's purpose and benefits.

We discuss three tests — exploratory (or formative), assessment (or summative), and validation (or verification) tests — at a high level, according to the approximate point in the product development lifecycle at which each would be administered. The fourth type of test, the comparison test, can be used as an integral part of any of the other three tests and is not associated with any specific lifecycle phase.

The basic methodology for conducting each test is roughly the same and is described in detail in Chapter 5. However, each test will vary in its emphasis on qualitative vs. quantitative measures, and by the amount of interaction

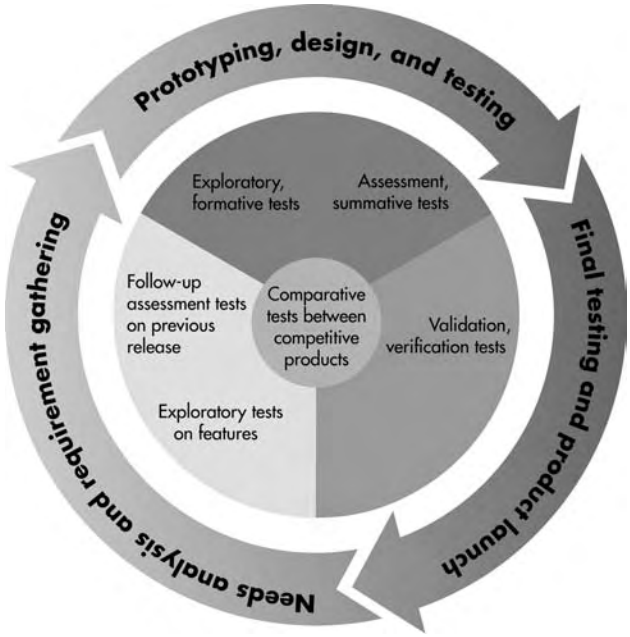

Figure 3-1 Usability testing throughout the product lifecycle

between test moderator and participant. Also, the tests expounded here are definitely biased toward an environment of tight deadlines and limited resources, and chosen with a keen eye on the bottom line.

Our other purpose for presenting the test types in terms of the product development lifecycle has to do with the power of iterative design. Usability testing is most powerful and most effective when implemented as part of an iterative product development process. That is, a cycle of design, test and measure, and redesign throughout the product development lifecycle has the greatest probability of concluding with a usable product. Even if important product flaws or deficiencies are missed during one test, another testing cycle offers the opportunity to identify these problems or issues.

An iterative design and testing approach also allows one to make steady and rapid progress on a project, to learn through empirical evidence, and to ''shape'' the product to fit the end users' abilities, expectations, and aptitude. We feel very strongly that such an approach provides the value when resources are limited, and that one will obtain the best results by conducting a series of short, precise tests that build one upon the other.

However, while the tests we are about to describe lend themselves to an iterative design process, one need not be concerned about applying the tests at *exactly* the correct moment. Rather, consider what it is that you need to understand about your product, and let that drive your test objectives and the appropriate application of a particular test method. Also, do not be put off if

you are unable to conduct multiple tests. One test is almost always better than none, and it is better to focus on what you *can* do than on what you cannot do.

The first three tests, exploratory (or formative), assessment (or summative), and validation (or verification), are shown in Figure 3-1 next to the approximate points in the lifecycle at which they are most effectively conducted. Now let's review each in turn.

Exploratory or Formative Study

When

The exploratory study is conducted quite early in the development cycle, when a product is still in the preliminary stages of being defined and designed (hence the reason it is sometimes called ''formative''). By this point in the development cycle, the user profile and usage model (or task analysis) of the product will have (or should have) been defined. The project team is probably wrestling with the functional specification and early models of the product. Or perhaps the requirements and specifications phase is completed, and the design phase is just about to begin.

Objective

The main objective of the exploratory study is to examine the effectiveness of preliminary design concepts. If one thinks of a user interface or a document as being divided into a high-level aspect and a more detailed aspect, the exploratory study is concerned with the former.

For example, designers of a Web application interface would benefit greatly knowing early on whether the user intuitively grasps the fundamental and distinguishing elements of the interface. For example, designers might want to know how well the interface:

- Supports users' tasks within a goal.
- Communicates the intended workflow.
- Allows the user to navigate from screen to screen and within a screen.

Or, using the task-oriented user guide of a software product as an example, technical writers typically might want to explore the following high-level issues:

- Overall organization of subject matter
- Whether to use a graphic or verbal approach
- How well the proposed format supports findability

- Anticipated points of assistance and messaging
- How to address reference information

The implications of these high-level issues go beyond the product, because you are also interested in verifying your assumptions about the *users*. Understanding one is necessary to define the other. Some typical user-oriented questions that an exploratory study would attempt to answer might include the following:

- What do users conceive and think about using the product?
- Does the product's basic functionality have value to the user?
- How easily and successfully can users navigate?
- How easily do users make inferences about how to use this user interface, based on their previous experience?
- What type of prerequisite information does a person need to use the product?
- Which functions of the product are ''walk up and use'' and which will probably require either help or written documentation?
- How should the table of contents be organized to accommodate both novice and experienced users?

The importance of this type of *early* analysis and research cannot be over emphasized, for this is the point in time when critical design decisions set the stage for all that will follow. If the project begins with wrong assumptions and faulty premises about the user, the product is almost guaranteed to have usability problems later. Similarly to building a house, once you lay the foundation for one type of model, you cannot simply build a totally different model without first ripping out the existing framework. The underlying structure determines all that will follow.

Overview of the Methodology

Exploratory tests usually dictate extensive interaction between the participant and test moderator to establish the efficacy of preliminary design concepts. One way to answer very fundamental questions, similar to those listed previously, is to develop preliminary versions of the product's interface and/or its support materials for evaluation by representative users. For software, this would typically involve a prototype simulation or mockup of the product that represents its basic layout, organization of functions, and high-level operations. Even prior to a working prototype, one might use static screen representations or even paper drafts of screens. For hardware representations, one might use

two-dimensional or three-dimensional foamcore, clay, or plastic models. For user support materials, one might provide very rough layouts of manuals, training materials, or help screens.

When developing a prototype, one need not represent the entire functionality of the product. Rather, one need only show enough functionality to address the particular test objective. For example, if you want to see how the user responds to the organization of your pull-down menus, you need only show the menus and one layer of options below. If the user proceeds deeper than the first layer, you might show a screen that reads, ''Not yet implemented,'' or something similar and ask what the participant was looking for or expecting next.

This type of prototype is referred to as a ''horizontal representation,'' since the user can move left or right but is limited in moving deeper. However, if your test objective requires seeing how well a user can move down several menu layers, you will need to prototype several functions ''vertically,'' so users can proceed deeper. You might achieve both objectives with a horizontal representation of *all* major functions, and a vertical representation of two of the functions.

During the test of such a prototype, the user would attempt to perform representative tasks. Or if it is too early to perform tasks, then the user can simply ''walk through'' or review the product and answer questions under the guidance of a test moderator. Or, in some cases, the user can even do both. The technique depends on the point in the development cycle and the sophistication of the mockups.

The testing process for an exploratory test is usually quite informal and almost a collaboration between participant and test moderator, with much interaction between the two. Because so much of what you need to know is cognitive in nature, an exploration of the user's thought process is vital. The test moderator and participant might explore the product together, with the test moderator conducting an almost ongoing interview or encouraging the participant to ''think aloud'' about his or her thought process as much as possible. Unlike later tests where there is much less interaction, the test moderator and participant can sit side by side as shown in Figure 3-2.

Ask participants for their ideas about how to improve confusing areas. Unlike later tests where there is more emphasis on measuring *how well* the user is able to perform by collecting quantitative data, here you strive to understand *why* the user performs as he or she does by collecting qualitative data. Regardless of whether you use a working prototype, static screens, early manuals, or whether the user performs tasks or simply ''walks through'' a product with the test moderator, the distinguishing feature of the exploratory test is its emphasis on discussion and examination of high-level concepts and thought processes, thus helping to form the final design.

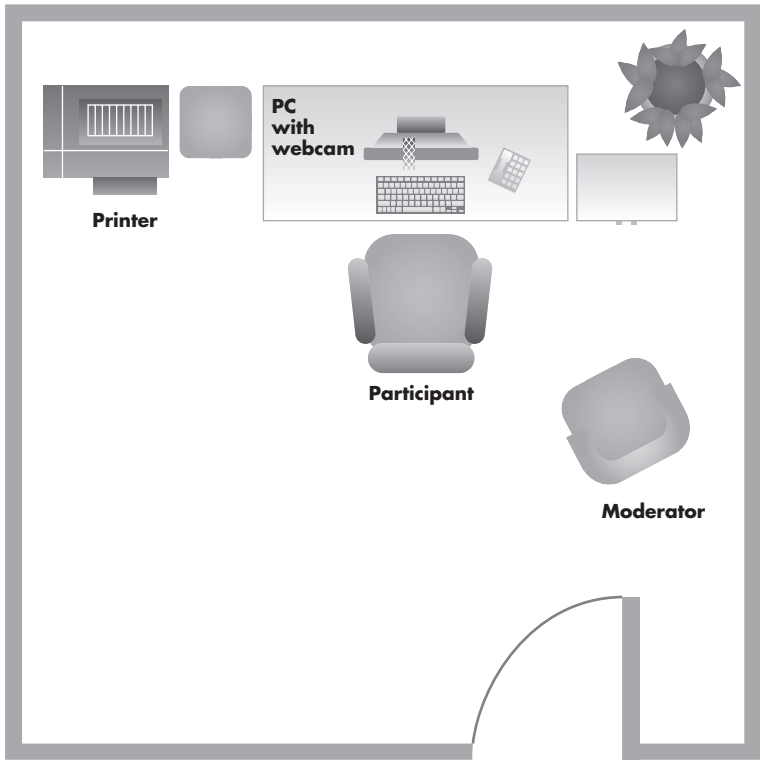

Figure 3-2 Test monitor and participant exploring the product

Example of Exploratory Study

Because the nature of the exploratory test is often somewhat abstract, let's review how a typical exploration might proceed for a product, such as a web site. Assume that you are exploring the home page of a web site, which employs options in the left navigation, each revealing further choices when the user mouses over it. Assume also that this is a very early stage of development, so the user interface simply consists of a single screen without any underlying structure or connections. However, the navigation menus function, so the user can view the menu options underneath each menu heading, as shown in Figure 3-3.

Now let's look at Figure 3-4, which contains an excerpt of a test script for conducting an exploratory test, to see how the test might proceed. You might continue in this vein, having the user attempt to accomplish realistic tasks with much discussion about assumptions and thought process. Alternatively, though, if the web page is in such a preliminary stage that the navigation does

Figure 3-3 Web page navigation interface

The purpose of our session today is to review the design for a new web site and get your opinions about it. As we review this design together, I will be asking you a series of questions about what you see and how you expect things to work. Please feel free to ask any questions and offer any observations during the session. There are no wrong answers or stupid questions. This product is in a very preliminary stage; do not be concerned ifit acts in unexpected ways.

Let's begin with a hypothetical situation. You would like to understand just what it is that this company offers.

(*User indicates how the task would be attempted, or attempts to do the task if the navigation works.*)

You would like to calculate the cost for offerings from this company. How do you start?

(*User indicates how the task would be attempted, or attempts to do the task if the navigation works.*)

Okay, you've found the pricing page. What does it tell you?

(*User discusses the information on the page, describing what is useful, clear* (*or not*)*, and where there could be more detail.*)

Figure 3-4 A Portion of an exploratory test script

not work, and you wanted to evaluate the effectiveness of the organization of the navigation, you might ask the user to simply point to the navigation label under which he or she would expect to accomplish a particular task, similarly to a paper-and-pencil evaluation. This approach would establish which tasks were harder to initiate and less intuitive.

Exploratory tests are often conducted as comparison tests, with different prototypes matched against each other. This prevents the project team from committing too early to one design, only to find out later that the design has serious flaws and liabilities. An example of this type of test is shown later in this chapter.

The important point of exploratory tests is that you can be extremely creative in simulating early versions of the product. Paper screens, prototypes with limited functionality, and so on all help to acquire important high-level information before the design is cast in concrete. It is never too early to learn how the user perceives the product and its fundamental presentation.

The benefits of using exploratory research to establish the soundness of high-level design *prior* to fleshing out all the details are innumerable. The time saved alone makes early research well worth doing. Explore very basic ideas and concepts as soon as you are able to simulate how they will work to users. Do not wait to take action until a very well thought-out, full-blown design takes shape.

Assessment or Summative Test

When

The assessment test is probably the most typical type of usability test conducted. Of all the tests, it is probably the simplest and most straightforward for the novice usability professional to design and conduct. Assessment tests are conducted either early or midway into the product development cycle, usually after the fundamental or high-level design or organization of the product has been established.

Objective

The purpose of the assessment test is to expand the findings of the exploratory test by evaluating the usability of lower-level operations and aspects of the product. If the intent of the exploratory test is to work on the skeleton of the product, the assessment test begins to work on the meat and the flesh. Assuming that the basic conceptual model of the product is sound, this test seeks to examine and evaluate how effectively the concept has been

implemented. Rather than just exploring the intuitiveness of a product, you are interested in seeing how well a user can actually perform full-blown realistic tasks and in identifying specific usability deficiencies in the product.

Overview of the Methodology

Often referred to as an information-gathering or evidence-gathering test, the methodology for an assessment test is a cross between the informal exploration of the exploratory test and the more tightly controlled measurement of the validation test. Unlike the exploratory test:

- The user will always *perform* tasks rather than simply walking through and commenting upon screens, pages, and so on.
- The test moderator will lessen his or her interaction with the participant because there is less emphasis on thought processes and more on actual behaviors.
- Quantitative measures will be collected.

Validation or Verification Test

When

The validation test, also referred to as the verification test, is usually conducted late in the development cycle and, as the name suggests, is intended to measure usability of a product against established benchmarks or, in the case of a verification test, to confirm that problems discovered earlier have been remedied and that new ones have not been introduced. Unlike the first two tests, which take place in the middle of a very active and ongoing design cycle, the validation test typically takes place much closer to the release of the product.

Objective

The objective of the validation test is to evaluate how the product compares to some predetermined usability standard or benchmark, either a project-related performance standard, an internal company or historical standard, or even a competitor's standard of performance. The intent is to establish that the product meets such a standard prior to release, and if it does not, to establish the reason(s) why. The standards usually originate from the usability objectives developed early in the project. These in turn come from previous usability tests, marketing surveys, interviews with users, or simply educated guesses by the development team.

Usability objectives are typically stated in terms of performance criteria, such as efficiency and effectiveness, or how well and how fast the user can perform various tasks and operations. Or the objectives can be stated in terms of preference criteria, such as achieving a particular ranking or rating from users. A verification test has a slightly different flavor. The objective here is to ensure that usability issues identified in earlier tests have been addressed and corrected appropriately.

It only makes sense then that the validation test itself can be used to *initiate* standards within the company for future products. Verification can accomplish the same thing. For example, if one establishes that a setup procedure for a software package works well and can be conducted within 5 minutes with no more than one error, it is important that future releases of the product perform to that standard or better. Products can then be designed with this benchmark as a target, so that usability does not degrade as more functions are added to future releases.

Another major objective of the validation test is to evaluate, sometimes for the first time, how all the components of a product work together in an end-to-end study. For example, how documentation, help, and software/hardware are integrated with each other, or all the steps in a longer process or workflow. The importance of an integrated validation test cannot be overstated. Because components are often developed in relative isolation from each other, it is not unusual that they do not work well together. It behooves an organization to discover this prior to release because, from the user's viewpoint, it is all one product and it is expected to perform that way.

Still another objective of the validation test, or really any test conducted very late in the development cycle, has become known in the trade as ''disaster or catastrophe insurance.'' At this late stage, management is most concerned with the risk of placing into the marketplace a new product that contains major flaws or that might require recall. If such a flaw is discovered, slipping the schedule may be preferable to recalling the product or having to send out ''fixes'' to every user. Even if there is no time to make changes before release, you are always at an advantage if you can anticipate a major deficiency in the product. There will be time to prepare a solution, train the support team, and even prepare public-relation responses. Even so, with all these advantages, there are companies that would rather not know about problems that exist in a product.

Overview of the Methodology

The validation test is conducted in similar fashion to the assessment test with three major exceptions.

- Prior to the test, benchmarks or standards for the tasks of the test are either developed or identified. This can be specific error or time measures, or as simple as eliminating the problems identified in earlier exploratory tests.

- Participants are given tasks to perform with either very little or no interaction with a test moderator. (And they are probably not asked to ''think aloud.'')
- The collection of quantitative data is the central focus, although reasons for substandard performance are identified.

Because you are measuring user performance against a standard, you also need to determine beforehand how adherence to the standard will be measured, and what actions will be taken if the product does not meet its standards. For example, if the standard for a task addresses ''time to complete,'' must 70 percent of participants meet the standard, or will you simply compare the standard to the average score of all participants? Under what conditions will the product's schedule be postponed? Will there be time to retest those tasks that did not meet the standard? These are all questions that should be addressed and resolved *prior* to the test.

Compared to an assessment test, a validation test requires more emphasis on experimental rigor and consistency, because you are making important quantitative judgments about the product. Make sure that members of the design team have input and buy-in into developing the standards used during the test. That way they will not feel as if the standards were overly difficult or unattainable.

Comparison Test

When

The comparison test is not associated with any specific point in the product development lifecycle. In the early stages, it can be used to compare several radically different interface styles via an exploratory test, to see which has the greatest potential with the proposed target population. Toward the middle of the lifecycle, a comparison test can be used to measure the effectiveness of a single element, such as whether pictorial buttons or textual buttons are preferred by users. Toward the end of the lifecycle, a comparison test can be used to see how the released product stacks up against a competitor's product.

Objective

The comparison test is the fourth type of test and can be used in conjunction with any of the other three tests. It is used to compare two or more designs, such as two different interface styles, or the current design of a manual with a proposed new design, or to compare your product with a competitor's. The comparison test is typically used to establish which design is easier to use or learn, or to better understand the advantages and disadvantages of different designs.

Overview of the Methodology

The basic methodology involves the side-by-side comparison of two or more clearly different designs. Performance data and preference data are collected for each alternative, and the results are compared. The comparison test can be conducted informally as an exploratory test, or it can be conducted as a tightly controlled classical experiment, with one group of participants serving as a control group and the other as the experimental group. The form used is dependent on your goals in testing. If conducted as a true experiment designed to acquire statistically valid results, the alternatives should vary along a single dimension — for example, keeping the content and functionality constant, but altering the visual design or the navigation scheme — and the expected results of the test should be formulated as a hypothesis.

If conducted less formally as a more observational, qualitative study, the alternatives may vary on many dimensions. One needs to ascertain why one alternative is favored over another, and which aspects of each design are favorable and unfavorable. Inevitably, when comparing one or more alternatives in this fashion, one discovers that there is no ''winning'' design per se. *Rather, the best design turns out to be a combination of the alternatives, with the best aspects of each design used to form a hybrid design*.

For exploratory comparison tests, experience has shown that the best results and the most creative solutions are obtained by including wildly differing alternatives, rather than very similar alternatives. This seems to work because:

- The design team is forced to stretch its conceptions of what will work rather than just continuing along in a predictable pattern. With the necessity for developing very different alternatives, the design team is forced to move away from predictable ways of thinking about the problem. Typically, this involves revisiting fundamental premises about an interface or documentation format that have been around for years. The result is often a design that redefines and improves the product in fundamental ways.
- During the test, the participant is forced to really consider and contemplate why one design is better and which aspects make it so. It is easier to compare alternatives that are very similar, but harder to compare very different ones. Why? Similar alternatives share the same framework and conceptual model, with only the lower-level operations working differently. Very different alternatives, however, are often based on different conceptual models of how each works and may challenge the user, especially one experienced with the product, to take stock of how the tasks are actually performed.

Iterative Testing: Test Types through the Lifecycle

Now, having reviewed the basics of each type of test, let us explore how a series of tests might in fact work. Let's suppose that your company is developing a web-based software application and its associated documentation. The software is a personal information manager, consisting of calendar, contact, and task management functionality. You intend to conduct three usability tests at three different times in the product development lifecycle. Following is a hypothetical series of tests on this product throughout the lifecycle, complete with hypothetical outcomes at the end of each test. Understand the details have been greatly simplified to provide an overview of iterative design in action.

Test 1: Exploratory/Comparison Test

The situation

Two early prototypes of the interface have been developed (see Figures 3-5 and 3-6). The interfaces use the same underlying architecture, programming languages, and functionality, although the layout of their navigation is considerably different from each other.

The prototypes have very limited working functionality (e.g., about 30 to 40 percent of the proposed functions work). There is no documentation, but

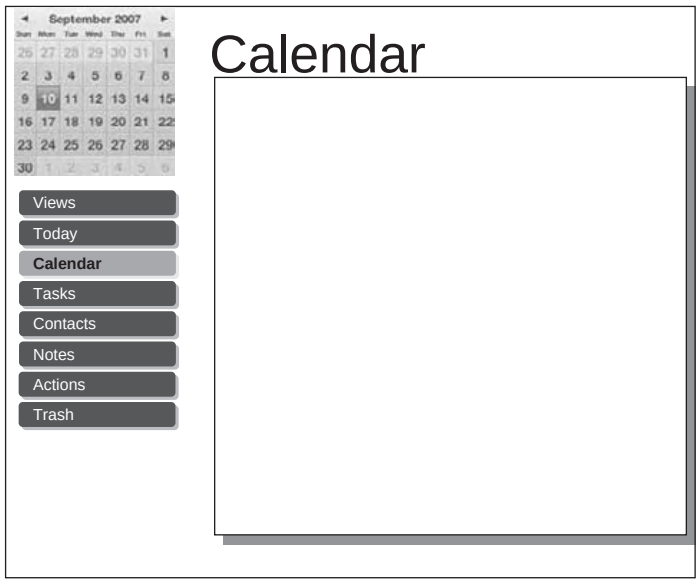

Figure 3-5 Left navigation interface

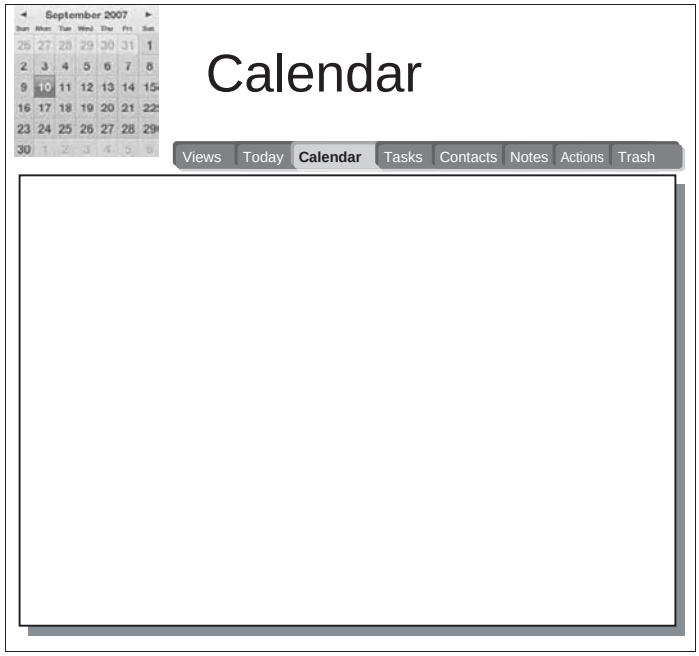

Figure 3-6 Top navigation interface

during the test, a technical expert will be available to reveal limited but crucial information needed to use the product. (See the gradual disclosure technique in Chapter 13 for an explanation of how to use a technical expert in this way.) Primitive help topics, available on paper only, will be provided to the participant on demand; that is, when the participant clicks the appropriate prompt or asks a question, the test moderator will provide what would normally be embedded assistance, instruction prompts, or messages on paper as they would appear on the screen.

Main Research Questions

- Which of the two interface styles/concepts is the most effective? In which is the user better able to remain oriented within the program?
- What are the best and worst features of each approach?
- What are the main stumbling blocks for the user?
- After some period of initial learning, which style has the greatest potential for the power user?
- For which tasks will users need help, further instructions, or supporting documentation?

- What types of written information will be required?
  - Prerequisite
  - Theoretical or conceptual
  - Procedural
  - Examples
  - Training

Brief Summary of Outcome

The test was conducted. As is typical of comparison tests at this point, there was no ''winner'' per se. Rather, the result was an interface with the best attributes of both prototypes. The navigation schema employing the navigation on the left was most efficient and effective, but some of the options available did not seem to belong with the others and so will remain in a navigation bar across the top of the main work area. Apparently, the options to remain in the top navigation are done less frequently.

There were many advanced features for use in a corporate setting that users needed additional information about. Because this personal information manager will be used throughout a large company, some functionality was added to support work group collaboration, which added complexity to the product. To remedy the complexity issue, the first line of defense is to develop a documentation set that includes, at minimum, a guide for setting up preferences, some self-paced training on interface operations, and a procedural user guide for more advanced, less frequent tasks.

Test 2: Assessment Test

The Situation

Time has passed. A single prototype has now been expanded to approximately 60 to 80 percent of its eventual functionality. There are comprehensive help topics for working functions in a separate section of the web site. A first draft, of simplified documentation, on 8 1/2" by 11" bond paper is available for the test, with a table of contents, but no index.

Main Test Objectives

- Confirm whether the findings of the original test adequately match interface operations with the user's workflow.
- Expose all major usability deficiencies and their causes for the most common tasks.

- Determine if there is a seamless connection of help topics, embedded assistance, and messaging with the functionality and user interface. Does the software give support at the right moments? Is the help center organized in a way that answers participants' questions?
- Is the documentation being utilized as designed? Is it accessible? Are graphics understood and at the appropriate level of detail? Are certain sections not read at all? Are additional sections required? Is all terminology clear? Are there areas that require more explanation?
- Where do participants still have questions? What are their questions?

Brief Summary of Test Outcome

Many difficulties in operations were identified, but the users' workflow matched that employed by the design team for the product's interface operations. Essentially, the high-level interface ''works,'' and the lower-level details remain to be implemented and refined. The help information was accurate and helpful, but users rarely invoked it unless prompted. There was a strong preference for trial and error with this particular user audience. When users were prompted to try the help, it was found that the organization of the help topics needs to be extensively revamped and made more task-oriented. Even more theoretical, contextual information needs to be included for the most advanced users. This last issue turned out to be very controversial because designers felt it was not their responsibility to force particular operational approaches on corporate working groups. It is possible that an interactive primer for users may be required for infrequent but important tasks.

Test 3: Verification Test

The Situation

Some weeks have passed. For this last test, a fully functional product with comprehensive help topics has been prepared. All sections of the documentation have been through one draft, with half of the sections undergoing a second draft. The documentation has a rough index for the test. A small ''tour'' for users about quarterly and semi-annual tasks was developed. For the major tasks of the product, specified measurable time and accuracy criteria have been developed. For example, one criterion reads:

Using the setup guide, a user will be able to correctly implement View and Network preferences within 10 minutes, with no more than two attempts required.

Unbelievably, and for only the first time in the recorded history of software development, there actually *will* be time to make minor modifications before release.

Test Objectives

- Verify that 70 percent of participants can meet established successful completion criteria for each major task scenario.

  (The 70 percent benchmark is something that Jeff has personally evolved toward over time, and that Dana has used effectively. It provides a reasonably challenging test while still leaving the design team some work to do before product release to move that number toward a more acceptable and traditional 95 percent benchmark. A benchmark of 100 percent is probably not realistic except for tasks involving danger or damage to the system or possible loss of life, and should never be used lightly. In the 1960s NASA found that achieving 100 percent performance cost as much as 50 times the cost of achieving 95 percent performance. It is likely that such costs have gone down over 40 years, but the point is that you should only use the higher benchmark if you are willing to pay the piper.)
- Identify any tasks and areas of the product that risk dire consequences (e.g., are unusable, contain destructive bugs) if the product is released as is.
- Identify all usability deficiencies and sources of those problems. Determine which deficiencies must be repaired before release and which, if there is not time within the schedule, can be implemented in the next release.

Brief Summary of Test Outcome

Every major task passed the 70 percent successful completion criteria with the exception of two. The team felt that the problems associated with those tasks could be corrected prior to release, and wanted to schedule a very quick test to confirm. Twenty recommendations from the test were identified for implementation prior to release, and at least fifteen recommendations were diverted to future releases.

Providing a ''tour'' of advanced features prior to the test proved to be a stroke of genius. Participants loved it, and some even insisted on taking it back

to their current jobs. One user suggested the company market it or a longer virtual seminar as a separate product for customers, and that is already in the works.

The revamped organization of the user guide was much more in tune with users' expectations than the previous set, although the index proved difficult to use. More task-oriented items must be added to the index to improve accessibility.

As you can tell from this condensed series of tests, the product evolved over time and reflected each test's findings. We strongly advocate such an iterative approach, but again, do not be discouraged if you can manage only one test to begin. Now let's talk about what it takes to be a good test moderator.