A one-on-one usability test can quickly reveal an immense amount of information about how people use a prototype, whether functional, mock-up, or just paper. Usability testing is probably the fastest and easiest way to tease out showstopping usability problems before a product launches.

Usability tests are structured interviews focused on specific features in an interface prototype. The heart of the interview is a series of tasks that are performed by the interface's evaluator (typically, a person who matches the product's ideal audience). Tapes and notes from the interview are later analyzed for the evaluator's successes, misunderstandings, mistakes, and opinions. After a number of these tests have been performed, the observations are compared, and the most common issues are collected into a list of functionality and presentation problems.

Using usability tests, the development team can immediately see whether people understand their designs as they are supposed to understand them. Unfortunately, the technique has acquired the aura of a final check before the project is complete, and usability tests are often scheduled at the end of the development cycle—after the feature set has been locked, the target markets have been determined, and the product is ready for shipping. Although testing can certainly provide insight into the next revision of the product, the full power of the technique remains untapped. They can be better used much earlier, providing feedback throughout the development cycle, both to check the usability of specific features and to investigate new ideas and evaluate hunches.
When to Test

Since usability testing is best at seeing how people perform specific tasks, it should be used to examine the functionality of individual features and the way they're presented to the intended user. It is better used to highlight potential misunderstanding or errors inherent in the way features are implemented rather than to evaluate the entire user experience. During the early to middle parts of a development cycle, usability testing can play a key role in guiding the direction of functionality as features are defined and developed. Once the functionality of a feature is locked in and its interaction with other features has been determined, however, it's often too late to make any fundamental changes. Testing at that point is more an investment in the next version than in the current one.

Moreover, unlike some of the other techniques mentioned throughout this book, usability testing is almost never a one-time event in a development cycle for a product, and should not be seen as such. Every round of testing can focus on a small set of features (usually no more than five), so a series of tests is used to test a whole interface or fine-tune a specific set of features.

The first thing the development team needs to do is decide on the target audience and the feature set to examine. This means that a good time to start usability testing is when the development cycle is somewhat under way, but not so late that testing prevents the implementation of extensive changes if it points to their necessity. Occasionally, usability testing reveals problems that require a lot of work to correct, so the team should be prepared to rethink and reimplement (and, ideally, retest) features if need be. In the Web world, this generally takes a couple of weeks, which is why iterative usability testing is often done in two-week intervals.

A solid usability testing program will include iterative usability testing of every major feature, with tests scheduled throughout the development process, reinforcing and deepening knowledge about people's behavior and ensuring that designs become more effective as they develop.
Example of an Iterative Testing Process: Webmonkey 2.0 Global Navigation

Webmonkey is a cutting-edge Web development magazine that uses the technologies and techniques it covers. During a redesign cycle, they decided that they wanted to create something entirely new for the main interface. Since much of the 1.0 interface had been extensively tested and was being carried through to the new design, they wanted to concentrate their testing and development efforts on the new features.

The most ambitious and problematic of the new elements being considered was a DHTML global navigational panel that gave access to the whole site (see Figures 10.1 and 10.2), but didn’t permanently use screen real estate. Instead, it would slide on and off

![Figure 10.1 The Webmonkey 2.0 Navigation Panel design (open).](image-url)
the screen when the user needed it. Webmonkey's previous navigation scheme worked well, but analysis by the team determined that it was not used often enough to justify the amount of space it was taking up. They didn't want to add emphasis to it (it was, after all, secondary to the site's content), so they decided to minimize its use of screen real estate, instead of attempting to increase its use. Their initial design was a traditional vertical navigation bar, identical to that found in the left margin of the 1.0 site, but in its own panel. The panel was hidden most of the time, but would reveal its contents when an arrow at the top of a striped bar on the left side was clicked. The target audience of Web developers would hopefully notice the striped bar and arrow, and click on it out of curiosity.

Webmonkey developed on an iterative development cycle, so Web developers and sophisticated users were invited to a series of
tests, with each test phase being followed by a design phase to incorporate the findings of the test. Although the purpose of the test was to examine the participants’ entire user experience, the developers paid special attention to the sliding panel. In the first round of testing, none of the six evaluators opened the panel. When asked whether they had seen the bar and the arrow, most said they had, but they took the striped bar to be a graphical element and the arrow to be decoration.

Two weeks later, the visual design had not changed much, but the designers changed the panel from being closed by default to being open when the page first loaded. During testing, the evaluators naturally noticed the panel and understood what it was for, but they consistently had trouble closing it in order to see the content that it obscured. Some tried dragging it like a window; others tried to click inside it. Most had seen the arrow, but they didn’t know how it related to the panel and so they never tried clicking it. Further questioning revealed that they didn’t realize that the panel was a piece of the window that slid open and closed. Thus, there were two interrelated problems: people didn’t know how the panel functioned, and they didn’t know that the arrow was a functional element.

A third design attempted to solve the problem by providing an example of the panel’s function as the first experience on the page: a short pause after the page loaded, the panel opened and closed by itself. The designers hoped that showing the panel in action would make the panel’s function clearer. It did, and in the next round of testing, the evaluators described both its content and its function correctly. However, none were able to open the panel again. The new design still did not solve the problem with the arrow, and most people tried to click and drag in the striped bar to get at the panel. Having observed this behavior, and (after some debate) realizing that they could not technically implement a dragging mechanism for the panel, the designers made the entire colored bar clickable, so that whenever someone clicked anywhere in it, the panel slid out (or back, if it was already open).

In the end, people still didn’t know what the arrow was for, but when they clicked in the striped panel to slide it open, it did, which was sufficient to make the feature usable, and none of the people observed using it had any trouble opening or closing the panel thereafter.
How to Do It

Preparation

Although it's similar to the “friends and family” test described in Chapter 2, a full-on usability test takes significantly longer to plan, execute, and analyze (see Table 10.1). You should start preparing for a usability testing cycle at least three weeks before you expect to need the results.

Setting a Schedule

<table>
<thead>
<tr>
<th>Timing</th>
<th>Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t - 2$ weeks</td>
<td>Determine test audience; start recruiting immediately</td>
</tr>
<tr>
<td>$t - 2$ weeks</td>
<td>Determine feature set to be tested</td>
</tr>
<tr>
<td>$t - 1$ week</td>
<td>Write first version of script; construct test tasks; discuss with development team; check on recruiting</td>
</tr>
<tr>
<td>$t - 3$ days</td>
<td>Write second version of guide; review tasks; discuss with development team; recruiting should be completed</td>
</tr>
<tr>
<td>$t - 2$ days</td>
<td>Complete guide; schedule practice test; set up and check all equipment</td>
</tr>
<tr>
<td>$t - 1$ day</td>
<td>Do practice test in the morning; adjust guide and tasks as appropriate</td>
</tr>
<tr>
<td>$t$</td>
<td>Test (usually 1–2 days, depending on scheduling)</td>
</tr>
<tr>
<td>$t + 1$ day</td>
<td>Discuss with observers; collect copies of all notes</td>
</tr>
<tr>
<td>$t + 2$ days</td>
<td>Relax; take a day off and do something else</td>
</tr>
<tr>
<td>$t + 3$ days</td>
<td>Watch all tapes; take notes</td>
</tr>
<tr>
<td>$t + 1$ week</td>
<td>Combine notes; write analysis</td>
</tr>
<tr>
<td>$t + 1$ week</td>
<td>Present to development team; discuss and note directions for further research</td>
</tr>
</tbody>
</table>
Before the process can begin, you need to know whom to recruit and which features to have them evaluate. Both of these things should be decided several weeks before the testing begins.

**Recruiting**

Recruiting is the most crucial piece to start on early. It needs to be timed right and to be precise, especially if it's outsourced. You need to find the right people and to match their schedules to yours. That takes time and effort. The more time you can devote to the recruiting process, the better (although more than two weeks in advance is generally too early since people often don't know their schedules that far in advance). You also need to choose your screening criteria carefully. The initial impulse is to recruit people who fall into the product's ideal target audience, but that's almost always too broad. You need to home in on the representatives of the target audience who are going to give you the most useful feedback.

Say you're about to put up a site that sells upscale forks online. Your ideal audience consists of people who want to buy forks.

In recruiting for a usability test, that's a pretty broad range of people. Narrowing your focus helps preserve clarity since different groups can exhibit different behaviors based on the same fundamental usability problems. Age, experience, and motivation can create seemingly different user experiences that are caused by the same underlying problem. Choosing the "most representative" group can reduce the amount of research you have to do in the long run and focus your results.

The best people to invite are those who are going to need the service you are providing in the near future or who have used a competing service in the recent past. These people will have the highest level of interest and knowledge in the subject matter, so they can concentrate on how well the interface works rather than on the minutia of the information. People who have no interest in the content can still point out interaction flaws, but they are not nearly as good at pointing out problems with the information architecture or any kind of content-specific features since they have little motivation to concentrate and make it work.
Say your research of the fork market shows that there are two strong subgroups within that broad range: people who are replacing their old silverware and people who are buying wedding presents. The first group, according to your research, is mostly men in their 40s, whereas the second group is split evenly between men and women, mostly in their mid-20s and 30s.

You decide that the people who are buying sets of forks to replace those they already own represent the heart of your user community. They are likely to know about the subject matter and may have done some research already. They're motivated to use the service, which makes them more likely to use it as they would in a regular situation. So you decide to recruit men in their 40s who want to buy replacement forks in the near future or who have recently bought some. In addition, you want to filter out online newbies, and you want to get people with online purchasing experience. Including all these conditions, your final set of recruiting criteria looks as follows:

- Men or women, preferably men
- 25 years old or older, preferably 35–50
- Have Internet access at home or work
- Use the Web five or more hours a week
- Have one or more years of Internet experience
- Have bought at least three things online
- Have bought something online in the last three months
- Are interested in buying silverware online

Notice that there is some flexibility in the age and gender criteria. This is to make the recruiter's life a little easier. You may insist that the participants be all male and that they must be between 40 and 50 years old, but if a candidate comes up who matches the rest of the criteria and happens to be 33 and female, you probably don't want to disqualify her immediately. Purchasing experience, on the other hand, requires precise requirements since getting people who aren't going to be puzzled or surprised by the concept of e-commerce is key to making the test successful. Testing an ecommerce system with someone who's never bought anything online tests the concept of ecommerce as much as it's testing the specific product. You rarely want that level of detail, so it's best to avoid situations that inspire it in the first place.
For this kind of focused task-based usability testing, you should have at least five participants in each round of testing and recruit somewhere from six to ten people for the five slots. Jakob Nielsen has shown (in Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier, available from www.useit.com/papers/guerilla_hci.html) that the cost-benefit cutoff for usability testing is about five users per target audience. Larger groups still produce useful results, but the cost of recruiting and the extra effort needed to run the tests and analyze the results leads to rapidly diminishing returns. After eight or nine users, the majority of problems performing a given task will have been seen several times. To offset no-shows, however, it's a good idea to schedule a couple of extra people beyond the basic five. And to make absolutely sure you have enough people, you could double-book every time slot. This doubles your recruiting and incentive costs, but it ensures that there's minimal downtime in testing.

In addition, to check your understanding of your primary audience, you can recruit one or two people from secondary target audiences—in the fork case, for example, a younger buyer or someone who's not as Web savvy—to see whether there's a hint of a radically different perspective in those groups. This won't give you conclusive results, but if you get someone who seems to be reasonable and consistently says something contrary to the main group, it's an indicator that you should probably rethink your recruiting criteria. If the secondary audience is particularly important, it should have its own set of tests, regardless.

Having decided whom to recruit, it's time to write a screener and send it to the recruiter. Screeners and recruiting are described in Chapter 6 of this book. Make sure to discuss the screener with your recruiter and to walk through it with at least two people in-house to get a reality check.

Then pick a couple of test dates and send out invitations to the people who match your criteria. Schedule interviews at times that are convenient to both you and the participant and leave at least half an hour between them. That gives the moderator enough slop time to have people come in late, for the test to run long, and for the moderator to get a glass of water and discuss the test with the observers. With 60-minute interviews, this means that you can do four or five in a single day and sometimes as many as six. With 90-minute

**Warning** You should strive to conduct a different test for each major user market since—by definition—each user market is likely to use the product differently. User markets are defined in Chapter 7.

In addition, Jared Spool and Will Schroeder point out (in www.winwriters.com/download/ch01_spool.pdf) that when you are going to give evaluators broad goals to satisfy, rather than specific tasks to do, you need more people than just five. However, in my opinion, broad goal research is less usability testing than a kind of focused contextual inquiry (Chapter 8) and should be conducted as such.

**Warning** If you're testing for the first time, schedule fewer people and put extra time in between. Usability testing can be exhausting, especially if you're new to the technique.
interviews, you can do three or four evaluators and maybe five if you push it and skip lunch.

**Choosing Features**

The second step is to determine which features to test. These, in turn, determine the tasks you create and the order in which you present them. You should choose features with enough lead time so that the test procedure can be fine-tuned. Five features (or feature clusters) can be tested in a given 60- to 90-minute interview. Typical tests range from one to two hours. Two-hour tests are used for initial or broad-based testing, while shorter tests are most useful for in-depth research into specific features or ideas (though it's perfectly acceptable to do a 90-minute broad-based test).

Individual functions should be tested in the context of feature clusters. It's rarely useful to test elements of a set without looking at least a little at the whole set. My rule of thumb is that something is testable when it's one of the things that gets drawn on a whiteboard when making a 30-second sketch of the interface. If you would draw a blob that's labeled “nav bar” in such a situation, then think of testing the nav bar, not just the new link to the homepage.

The best way to start the process is by meeting with the development staff (at least the product manager, the interaction designers, and the information architects) and making a list of the five most important features to test. To start discussing which features to include, look at features that are

- Used often
- New
- Highly publicized
- Considered troublesome, based on feedback from earlier versions
- Potentially dangerous or have bad side effects if used incorrectly
- Considered important by users
A Feature Prioritization Exercise

This exercise is a structured way of coming up with a feature prioritization list. It's useful when the group doesn't have a lot of experience prioritizing features or if it's having trouble:

- Step 1: Have the group make a list of the most important things on the interface that are new or have been drastically changed since the last round of testing. Importance should not just be defined purely in terms of prominence; it can be relative to the corporate bottom line or managerial priority. Thus, if next quarter's profitability has been staked on the success of a new Fork of the Week section, it's important, even if it's a small part of the interface.
- Step 2: Make a column and label it "importance." Look at each feature and rate it on a scale of 1 to 5, where 5 means it's critical to the success of the product and 1 means it's not very important.

Next, make a second column and label it "doubt." Look at each feature and rate how comfortable the team is with the design, labeling the most comfortable items with a 1 and the least comfortable with a 5. This may involve some debate among the group, so you may have to treat it as a focus group of the development staff.
- Step 3: Multiply the two entries in the two columns and write the results next to them. The features with the greatest numbers next to them are the features you should test. Call these out and write a short sentence that summarizes what the group most wants to know about the functionality of the feature.

### Top Five Fork Catalog Features by Priority

<table>
<thead>
<tr>
<th>Feature</th>
<th>Importance</th>
<th>Doubt</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>The purchasing mechanism, <em>Does it work for both single items and whole sets?</em></td>
<td>5</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>The search engine, <em>Can people use it to find specific items?</em></td>
<td>6</td>
<td>5</td>
<td>26</td>
</tr>
<tr>
<td>Catalog navigation, <em>Can people navigate through it when they don't know exactly what they want?</em></td>
<td>5</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>The Fork of the Week page, <em>Do people see it?</em></td>
<td>4</td>
<td>4</td>
<td>16</td>
</tr>
<tr>
<td>The Wish List, <em>Do people know what it's for and can they use it?</em></td>
<td>3</td>
<td>5</td>
<td>15</td>
</tr>
</tbody>
</table>
Once you have your list of the features that most need testing, you're ready to create the tasks that will exercise those features.

In addition, you can include competitive usability testing. Although comparing two interfaces is more time consuming than testing a single interface, it can reveal strengths and weaknesses between products. Performing the same tasks with an existing interface and a new prototype, for example, can reveal whether the new design is more functional (or—the fear of every designer—less functional). Likewise, performing the same tasks, or conducting similar interface tours with two competing products, can reveal relative strengths between the two products. In both situations, however, it's very important not to bias the evaluator toward one interface over the other. Competitive research is covered extensively in Chapter 14.

Creating Tasks

Tasks need to be representative of typical user activities and sufficiently isolated to focus attention on a single feature (or feature cluster) of the product. Good tasks should be:

- **Reasonable.** They should be typical of the kinds of things that people will do. Someone is unlikely to want to order 90 different kinds of individual forks, each in a different pattern, and have them shipped to 37 different addresses, so that's not a typical task. Ordering a dozen forks and shipping them to a single address, however, is.

- **Described in terms of end goals.** Every product, every Web site, is a tool. It's not an end to itself. Even when people spend hours using it, they're doing something with it. So, much as actors can emote better when given their character's motivation, interface evaluators perform more realistically if they're motivated by a lifelike situation. Phrase your task as something that's related to the evaluator's life. If they're to find some information, tell them why they're trying to find it ("Your company is considering opening an office in Moscow and you'd like to get a feel for the reinsurance business climate there. You decide that the best way to do that is to check today's business headlines for information about reinsurance companies in Russia."). If they're trying to buy something, tell...
them why ("Aunt Millie's subcompact car sounds like a jet plane. She needs a new muffler"). If they're trying to create something, give them some context ("Here's a picture of Uncle Fred. You decide that as a practical joke you're going to digitally put a mustache on him and email it to your family").

- **Specific.** For consistency between evaluators and to focus the task on the parts of the product you're interested in testing, the task should have a specific end goal. So rather than saying "Go shop for some forks," say, "You saw a great Louis XIV fork design in a shop window the other day; here's a picture of it. Find that design in this catalog and buy a dozen fish forks." However, it's important to avoid using terms that exist on the interface since that tends to tip off the participant about how to perform the task.

- **Doable.** If your site has forks only, don't ask people to find knives. It's sometimes tempting to see how they use your information structure to find something impossible, but it's deceptive and frustrating and ultimately reveals little about the quality of your design.

- **In a realistic sequence.** Tasks should flow like an actual session with the product. So a shopping site could have a browsing task followed by a search task that's related to a selection task that flows into a purchasing task. This makes the session feel more realistic and can point out interactions between tasks that are useful for information architects in determining the quality of the flow through the product.

- **Domain neutral.** The ideal task is something that everyone who tests the interface knows something about, but no one knows a lot about. When one evaluator knows significantly more than the others about a task, their methods will probably be different than the rest of the group. They'll have a bigger technical vocabulary and a broader range of methods to accomplish the task. Conversely, it's not a good idea to create tasks that are completely alien to some evaluators since they may not know even how to begin. For example, when testing a general search engine, I have people search for pictures of Silkie chickens: everyone knows something about chickens, but unless you're a Bantam hen farmer, you probably won't know much about Silkie. For really important tasks where an
obvious domain-neutral solution doesn't exist, people with specific knowledge can be excluded from the recruiting (for example, asking “Do you know what a Silkie chicken is?” in the recruiting screener can eliminate people who may know too much about chickens).

- **A reasonable length.** Most features are not so complex that to use them takes more than 10 minutes. The duration of a task should be determined by three things: the total length of the interview, its structure, and the complexity of the features you’re testing. In a 90-minute task-focused interview, there are 50–70 minutes of task time, so an average task should take about 12 minutes to complete. In a 60-minute interview, there are about 40 minutes of task time, so each task should take no more than 7 minutes. Aim for 5 minutes in shorter interviews and 10 in longer ones. If you find that you have something that needs more time, then it probably needs to be broken down into subfeatures and reprioritized (though be aware of exceptions: some important tasks take a much longer time and cannot be easily broken up, but they still need to be tested).

**Estimating Task Time**

Carolyn Snyder recommends a method of estimating how long a task will take.

- Ask the development team how long it takes an expert—such as one of them—to perform the task.
- Multiply that number by 3 to 10 to get an estimate of how long it would take someone who had never used the interface to do the same thing. Use lower numbers for simpler tasks such as found on general-audience Web sites, and higher numbers for complex tasks such as found in specialized software or tasks that require data entry.

For every feature on the list, there should be at least one task that exercises it. Usually, it’s useful to have two or three alternative tasks for the most important features in case there is time to try more than one or the first task proves to be too difficult or uninformative.

People can also construct their own tasks within reason. At the beginning of a usability test, you can ask the participants to describe a recent situation they may have found themselves in that your product could address. Then, when the times comes for a task, ask...
them to try to use the product as if they were trying to resolve the situation they described at the beginning of the interview. Another way to make a task feel authentic is to use real money. For example, one ecommerce site gave each of its usability testing participants a $50 account and told them that whatever they bought with that account, they got to keep (in addition to the cash incentive they were paid to participate). This presented a much better incentive for them to find something they actually wanted than they would have had if they just had to find something in the abstract.

Although it’s fundamentally a qualitative procedure, you can also add some basic quantitative metrics (sometimes called performance metrics) to each task in order to investigate the relative efficiency of different designs or to compare competing products. Some common Web-based quantitative measurements include

- The speed with which someone completes a task
- How many errors they make
- How often they recover from their errors
- How many people complete the task successfully

Because such data collection cannot give you results that are statistically usable or generalizable beyond the testing procedure, such metrics are useful only for order-of-magnitude ideas about how long a task should take. Thus, it’s often a good idea to use a relative number scale rather than specific times.

For the fork example, you could have the following set of tasks, as matched to the features listed earlier.

<table>
<thead>
<tr>
<th>FORK TASKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>The search engine: can people use it to find specific items?</td>
</tr>
<tr>
<td>Catalog navigation: can people navigate through it when they don’t know exactly what they want?</td>
</tr>
</tbody>
</table>

(Continued)
<table>
<thead>
<tr>
<th>Feature</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>The purchasing mechanism: does it work for both single items and whole sets?</td>
<td>Say you really like one of the designs we just looked at (pick one) and you'd like to buy a dozen dinner forks in that pattern. How would you go about doing that? Now say it's a month later, you love your forks, but you managed to mangle one of them in the garbage disposal. Starting from the front door to the site, how would you buy a replacement?</td>
</tr>
<tr>
<td>The Fork of the Week page: do people see it?</td>
<td>This one is a bit more difficult. Seeing is not easily taskable, but it's possible to elicit some discussion about it by creating a situation where it may draw attention and noting if it does. It's a couple of months later, and you're looking for forks again, this time as a present. Where would be the first place you'd look to find interesting forks that are a good value? Asking people to draw or describe an interface without looking at it reveals what people found memorable, which generally correlates closely to what they looked at.</td>
</tr>
<tr>
<td>[turn off monitor] Please draw the interface we just looked at, based on what you remember about it.</td>
<td></td>
</tr>
<tr>
<td>The Wish List: do people know what it's for?</td>
<td>While you're shopping, you'd like to be able to keep a list of designs you're interested in, maybe later you'll buy one, but for now you'd like to just remember which ones are interesting. How would you do that? [If they don't find it on their own, point them to it and ask them whether they know what it means and how they would use it.]</td>
</tr>
</tbody>
</table>

When you've compiled the list, you need to time and check the tasks. Do them yourself and get someone who isn't close to the project to try them. This can be part of the pretest dry run, but it's always a good idea to run through the tasks by themselves if you can.
In addition, you should continually evaluate the quality of the tasks as the testing goes on. Use the same guidelines as you used to create the tasks and see if the tasks actually fulfill them. Between sessions think about the tasks' effectiveness and discuss them with the moderator and observers. And although it's a bad idea to drastically change tasks in the middle, it's OK to make small tweaks that improve the tasks' accuracy in between tests, keeping track of exactly what changed in each session.

**Writing a Script**

With tasks in hand, it's time to write the script. The script is sometimes called a "protocol," sometimes a "discussion guide," but it's really just a script for the moderator to follow so that the interviews are consistent and everything gets done.

This script is divided into three parts: the introduction and preliminary interview, the tasks, and the wrap-up. The one that follows is a sample from a typical 90-minute e-commerce Web site usability testing session for people who have never used the site under review. About a third of the script is dedicated to understanding the participants' interests and habits. Although those topics are typically part of a contextual inquiry process or a focus group series, it's often useful to include some investigation into them in usability testing. Another third is focused on task performance, where the most important features get exercised. A final third is administration.

**Introduction (6–7 minutes)**

The introduction is a way to break the ice and give the evaluators some context. This establishes a comfort level about the process and their role in it.

```
[Monitor off, Video off, Computer reset]

Hi, welcome, thank you for coming. How are you? (Did you find the place? OK? Any questions about the NDA? Etc.)
```
I'm ________, I'm helping _________ understand how well one of their products works for the people who are its audience. This is ________, who will be observing what we're doing today. We've brought you here to see what you think of their product: what seems to work for you, what doesn't, and so on.

This evaluation should take about an hour.

We're going to be videotaping what happens here today, but the video is for analysis only. It's primarily so I don't have to sit here and scribble notes and I can concentrate on talking to you. It will be seen by some members of the development team, a couple of other people, and me. It's strictly for research and not for public broadcast or publicity or promotion or laughing at Christmas parties.

When there's video equipment, it's always blatantly obvious and somewhat intimidating. Recognizing it helps relieve a lot of tension about it. Likewise, if there's a two-way mirror, recognizing it—and the fact that there are people behind it—also serves to alleviate most people's anxiety. Once mentioned, it shouldn't be brought up again. It fades quickly into the background, and discussing it again is a distraction.

Also note that the script is written in a conversational style. It's unnecessary to read it verbatim, but it reminds the moderator to keep the tone of the interview casual. In addition, every section has a duration associated with it so that the moderator has an idea of how much emphasis to put on each one.

Like I said, we'd like you to help us with a product we're developing. It's designed for people like you, so we'd really like to know what you think about it and what works and doesn't work for you. It's currently in an early stage of development, so not everything you're going to see will work right.

No matter what stage the product team is saying the product is in, if it's being usability tested, it's in an early stage. Telling the evaluators it's a work-in-progress helps relax them and gives them more license to make comments about the product as a whole.
The procedure we're going to do today goes like this: we're going to start out and talk for a few minutes about how you use the Web, what you like, what kinds of problems you run into, that sort of thing. Then I'm going to show you a product that ________ has been working on and have you try out a couple of things with it. Then we'll wrap up, I'll ask you a few more questions about it, and we're done.

Any questions about any of that?

Explicitly laying out the whole procedure helps the evaluators predict what's going to come next and gives them some amount of context to understand the process.

Now I'd like to read you what's called a statement of informed consent. It's a standard thing I read to everyone I interview. It sets out your rights as a person who is participating in this kind of research.

As a participant in this research:
- You may stop at any time.
- You may ask questions at any time.
- You may leave at any time.
- There is no deception involved.
- Your answers are kept confidential.

Any questions before we begin?

Let's start.

The informed consent statement tells the evaluators that their input is valuable, that they have some control over the process, and that there is nothing fishy going on.
The preliminary interview is used to establish context for the participant's later comments. It also narrows the focus of the interview into the space of the evaluator's experience by beginning with general questions and then narrowing the conversation to the topics the product is designed for. For people who have never participated in a usability test, it increases their comfort level by asking some "easy" questions that build confidence and give them an idea of the process.

In this case, the preliminary interview also features a fairly extensive investigation into people's backgrounds and habits. It's not unusual to have half as many questions and to have the initial context-setting interview last 5 minutes, rather than 10 to 15.

[Video on]

How much time do you normally spend on the Web in a given week?

How much of that is for work use, and how much of that is for personal use?

Other than email, is there any one thing you do the most online?

Do you ever shop online? What kinds of things have you bought? How often do you buy stuff online?

Do you ever do research online for things that you end up buying in stores? Are there any categories of items that this happens with more often than others? Why?

Is there anything you would never buy online? Why?

When it's applicable, it's useful to ask about people's offline habits before refocusing the discussion to the online sphere. Comparing what they say they do offline and what you observe them doing online provides insight into how people perceive the interface.
Changing gears here a bit, do you ever shop for silverware in general, not just online? How often?

Do you ever do that online? Why?

[If so] Do you have any favorite sites where you shop for silverware online?

[If so] What do you like the most about [site]? Is there anything that regularly bothers you about it?

Evaluation Instructions (3 minutes)

It's important that evaluators don't feel belittled by the product. The goal behind any product is to have it be a subservient tool, but people have been conditioned by badly designed tools and arrogant companies to place the blame on themselves. Although it's difficult to undo a lifetime of software insecurity, the evaluation instructions help get evaluators comfortable with narrating their experience, including positive and negative commentary, in its entirety.

In a minute, I'll ask you to turn on the monitor and we'll take a look at the product, but let me give you some instructions about how to approach it.

The most important thing to remember when you're using it is that you are testing the interface; the interface is not testing you. There is absolutely nothing that you can do wrong. Period. If anything seems broken or wrong or weird or, especially, confusing, it's not your fault. However, we'd like to know about it. So please tell us whenever anything isn't working for you.

Likewise, tell us if you like something: Even if it's a feature, a color, or the way something is laid out, we'd like to hear about it.

Be as candid as possible. If you think something's awful, please say so. Don't be shy; you won't hurt anyone's feelings. Since it's designed for people like you, we really want to know exactly what you think and what works and doesn't work for you.

(Continued)
Also, while you’re using the product I'd like you to say your thoughts aloud. That gives us an idea of what you're thinking when you're doing something. Just narrate what you're doing, sort of as a play-by-play, telling me what you're doing and why you're doing it.

A major component to effective usability tests is to get people to say what they're thinking as they're thinking it. The technique is introduced up front, but it should also be emphasized during the actual interview.

Does that make sense? Any questions?

Please turn on the monitor [or “open the top of the portable”]. While it's warming up, you can put the keyboard, monitor, and mouse where they're comfortable for you.

First Impressions (5–10 minutes)

First impressions of a product are incredibly important for Web sites, so testing them explicitly is always a good thing and quick to do. Asking people where they're looking and what they see points out the things in an interface that pop and provides insight into how page loading and rendering affects focus and attention.

The interview begins with the browser up, but set to a blank page. Loading order affects the order people see the elements on the page and tends to affect the emphasis they place on those elements. Knowing the focus of their attention during the loading of the page helps explain why certain elements are seen as more or less important.

Now that it's warmed up, I'd like you to select “Forks” from the “Favorites” menu.

[Rapidly] What's the first thing your eyes are drawn to? What's the next thing? What's the first thought that comes into your mind when you see this page?
[after 1–2 minutes] What is this site about?

Are you interested in it?

If this was your first time here, what would you do next? What would you click on? What would you be interested in investigating?

At this point, the script can go in two directions. Either it can be a *task-based interview*—where the user immediately begins working on tasks—or it can be a *hybrid interview* that’s half task-based and half observational interview.

The task-based interview focuses on a handful of specific tasks or features. The hybrid interview is useful for first-time tests and tests that are early in the development cycle. In hybrid interviews, the evaluator goes through an interface tour, looking at each element of the main part of the interface and quickly commenting on it, before working on tasks.

A task-based interview would look as follows.

*Tasks (20–25 minutes)*

Now I'd like you to try a couple of things with this interface. Work just as you would normally, narrating your thoughts as you go along.

Here is the list of things I'd like you to do. [hand out list]

The first scenario goes as follows:

**TASK 1 DESCRIPTION GOES HERE**

[Read the first task, hand out Task 1 description sheet]

The second thing I'd like you to do is

**TASK 2 DESCRIPTION GOES HERE**

[Read the second task, hand out Task 2 description sheet] etc.
When there is a way to remotely observe participants, it is sometimes useful to ask them to try a couple of the listed tasks on their own, without the moderator in the room. This can yield valuable information about how people solve problems without an available knowledge source. In addition, it’s a useful time for the moderator to discuss the test with the observers. When leaving the room, the moderator should reemphasize the need for the evaluator to narrate all of his or her thoughts.

Including a specific list of issues to probe helps ensure that all the important questions are answered. The moderator should feel free to ask the probe questions whenever it is appropriate in the interview.

*Probe Questions (investigate whenever appropriate)*
- Do the names of navigation elements make sense?
- Do the interface elements function as the evaluator had expected?
- Are there any interface elements that don’t make sense?
- What draws the evaluator’s attention?
- What are the most important elements in any given feature?
- Are there places where the evaluator would like additional information?
- What are their expectations for the behavior/content of any given element/screen?

A hybrid interview could look as follows. It begins with a quick general task to see how people experience the product before they’ve had a chance to examine the interface in detail.

*First Task (5 minutes)*
Now I’d like you to try something with this interface.

Work just as you would normally, narrating your thoughts as you go along.

The first scenario goes as follows:

**TASK 1 DESCRIPTION GOES HERE**

[read the first task]
Interface Tour (10 minutes)
OK, now I'd like to go through the interface, one element at a time, and talk about what you expect each thing to do.

[Go through:
  • Most of front door.
  • A sample catalog page.
  • A shopping cart page.]

[Focus on:
  • Site navigation elements.
  • Search elements.
  • Major feature labels and behaviors.
  • Ambiguous elements.
  • Expectations.]

Per element probes [ask for each significant element, when appropriate]:
• In a couple of words, what do you think this does?
• What does this [label, title] mean?
• Where do you think this would go?
• Without clicking on it, what kind of page would you expect to find on the other side? What would it contain? How would it look?

Per screen probes [ask on each screen, when appropriate]:
• What's the most important thing on this screen for you?
• Is there any information missing from here that you would need?
• After you've filled it out, what would you do next?
• How would you get to the front door of the site from here? What would you click on?
• How would you get to [some other major section]?

Tasks (10 minutes)
The second thing I’d like you to do is

TASK 2 DESCRIPTION GOES HERE.

[read the second task]

(Continued)
The last thing I'd like to try is

**TASK 3 DESCRIPTION GOES HERE**

[read the third task]

By the time all the tasks have been completed, the heart of the information collection and the interview is over. However, it's useful for the observers and analysts to get a perspective on the high points of the discussion. In addition, a blue-sky discussion of the product can provide good closure for the evaluator and can produce some good ideas (or the time can be used to ask people to draw what they remember of the interface as the moderator leaves the room and asks the observers if they have any final questions for the participant).

**Wrap-up and Blue-Sky Brainstorm (10 minutes)**

Please turn off the monitor, and we'll wrap up with a couple of questions.

*Wrap-up*

How would you describe this product in a couple of sentences to someone with a level of computer and Web experience similar to yours?

Is this an interesting service? Is this something that you would use?

Is this something you would recommend? Why/why not?

Can you summarize what we've been talking about by saying three good things and three bad things about the product?

*Blue-Sky Brainstorm*

OK, now that we've seen some of what this can do, let's talk in blue-sky terms here for a minute. Not thinking in practical terms at all; what kinds of things would you like a system like this to do that the one doesn't? Have you ever said, "I wish that some program would do X for me"? What was it?
Finally, it's useful to get some feedback about the testing and scheduling process.

As with every phase of user research, the product stakeholders should have input into the testing script content. The complete script draft should still be vetted by the stakeholders to assure that the priorities and technical presentation are accurate. The first draft should be given to the development team at least a week before testing is to begin. A second version incorporating their comments should be shown to them at least a couple of days beforehand.

**Conducting the Interview**

There are two goals in conducting user interviews: getting the most natural responses from evaluators and getting the most complete responses. Everything that goes into the environment of a user interview—from the physical space to the way questions are asked—is focused on these two goals.
The Physical Layout

The physical layout should look as little like a lab as possible and as much like the kind of space in which the product is designed to be used. If the product is to be used at work, then it should be tested in an environment that resembles a nice office, preferably with a window. If it's for home use, then it should be tested in an environment like a home office. The illusion doesn't have to be all pervasive; it's possible to achieve the appropriate feeling with just a few carefully chosen props. For the home office, for example, soft indirect lighting and a tablecloth over an office desk instantly makes it less formal.

Often, however, the usability test must be performed in a scheduled conference room or a rented lab, where extensive alteration isn't possible. In those situations, make sure that the space is quiet, uncluttered, and as much as possible, unintimidating.

Every interview should be videotaped, if possible. Ideally, a video scan converter (a device that converts computer video output to standard video) and a video mixer should be used to create a “picture-in-picture” version of the proceedings, with one image showing a picture of the person and the other of their screen (Figure 10.3). The video camera should be positioned such that the evaluator's face and hands can be seen for the initial interview and so that the screen can be seen for the task portion of the interview. The moderator does not have to appear in the shot.

Accurate, clear audio is extremely important, so the video camera should have a good built-in microphone that filters out external noise, or you should invest in a lapel microphone, which the evaluator can clip onto his or her clothing, or a small microphone that can be taped to the monitor. The downside of lapel microphones is that although they capture the evaluator's comments, they don't always catch those of the moderator. An ideal situation is to have two wireless lapel microphones and a small mixer to merge the two sound sources, or a single external microphone that is sensitive enough to capture both sides of the conversation without picking up the external noise that's the bane of built-in camera mics. But that's a lot of equipment.

If a two-way mirrored room is unavailable, closed-circuit video makes for good substitute. This is pretty easy to achieve with a long video cable and a television in an adjacent room (though the room should be sufficiently soundproof that observers can speak freely without being heard in the testing room). So the final layout of a typical round of usability testing can look like Figure 10.4.
Figure 10.3 Picture-in-picture video documentation.

Figure 10.4 A typical usability testing configuration.
Moderation

The moderator needs to make the user feel comfortable and elicit useful responses at appropriate times without drastically interrupting the flow of the user's own narration or altering his or her perspective. The nondirected interviewing style is described in depth in Chapter 6 and should be used in all user interviews.

Apart from the general interviewing style outlined in Chapter 6, there are several things that moderators should do in all interviews.

- **Probe expectations.** Before participants click on a link, check a box, or perform any action with an interface, they have some idea of what will happen. Even though their idea of what will happen next may not be completely formed, they will always have some expectation. After the users have performed an action, their perception is forever altered about that action's effect. The only way to capture their view before it happens is to stop them as they're about to perform an action and ask them for their expectations of its effect. With a hyperlink, for example, asking the evaluators to describe what they think will happen if they click on a link can reveal a lot about their mental model of the functionality of the site. Asking "Is that what you expected?" immediately after an action is also an excellent way of finding out whether the experience matches expectations.

- **Ask "why" a lot.** It's possible to learn a lot about people's attitudes, beliefs, and behaviors by asking simple, direct, unbiased questions at appropriate times. Five-year-olds do this all the time: they just ask "why" over and over again, digging deeper and deeper into a question without ever telegraphing that they think there's a correct answer. For example, when someone says "I just don't do those kinds of things," asking "why" yields better information than just knowing that he or she does or doesn't do something.

- **Suggest solutions, sometimes.** Don't design during an interview, but it is OK to probe if a particular idea (that doesn't exist in the current product) would solve their problem. This is useful as a check on the interviewer's understanding of the problem, and it can be a useful way to sanity-check potential solutions. For example, a number of people in a test said they kept their personal schedule using Microsoft Outlook and their Palm
Pilot. They weren't interested in online schedules since they felt it would involve duplicating effort even though they liked the convenience of a Web-based calendar. When the moderator suggested that their offline schedule could be synchronized with the online, they were universally excited and said that they would be much more likely to use the entire service if that feature were available.

- **Investigate mistakes.** When evaluators make mistakes, wait to see if they've realized that they've made a mistake and then immediately probe their thoughts and expectations. Why did they do something one way? What were they hoping it would do? How did they expect it to work? What happened that made them realize that it didn't work?

- **Probe nonverbal cues.** Sometimes people will react physically to an experience in a way that they wouldn't normally voice. When something is surprising or unexpected or unpleasant, someone may flinch, but not say anything. Likewise, a smile or a lean forward may signify satisfaction or interest. Watch for such actions and follow up, if appropriate. For example, “You frowned when that dialog box came up. Is there anything about it that caused you to do that?”

- **Keep the interview task centered.** People naturally tend to tangent off on certain ideas that come up. As someone is performing a task, they may be reminded of an idea or an experience that they want to explore. Allowing people to explore their experiences is important, but it's also important to stay focused on the product and the task. When someone leans back, takes his or her hands off the keyboard, stops looking at the monitor, and starts speaking in the abstract, it's generally time to introduce a new task or return to the task at hand.

- **Respect the evaluator's ideas.** When people are off topic, let them go for a bit (maybe a minute or so) and see if they can wrap up their thoughts on their own. If they're not wrapping up, steer the conversation back to the task or topic at hand. If that doesn't seem to work, then you can be more explicit: “That's interesting and maybe we'll cover it more later, but let's take a look at the Fork of the Week page.”

- **Focus on their personal experience.** People have a tendency to idealize their experience and to extrapolate it to others' needs or to their far future needs. Immediate experience, however, is
Note Throughout this chapter, I have used the words "evaluator" and "participant" to refer to the people who are evaluating the interface, rather than "subject," "tester," "guinea pig," or whatnot. This is intentional. The people who you have recruited to evaluate your interface are your colleagues in this process. They are not being examined, the product is. It's tempting to set the situation up as a psychology experiment, but it's not. It's a directed evaluation of a product, not an inquiry into human nature, and should be treated as such on all levels.

Note Some researchers claim that it's possible to have multiple observers in the same room without compromising the quality of the observations. I haven't found that to be the case, nor have I chosen to have any in-room observers most of the time. It may well be possible to have a bunch of observers in the room and still have the participant perform comfortably and naturally—stage actors do this all the time, after all. However, I try to avoid the complications that this may introduce into the interpretation of people's statements by avoiding the question entirely.

much more telling about people's actual attitudes, needs, and behaviors, and is usually much more useful than their extrapolations. When Peter says, "I think it may be useful to someone," ask him if it's useful to him. If Inga says that she understands it, but others may not, tell her that it's important to know about how she views it, not how it could be designed for others. If Tom says that something "may be useful someday," ask him if it's something that's useful to him now.

Managing Observers

Getting as many members of the development team to observe the tests is one of the fastest ways to relate the findings of the test and win them over.

Make the appropriate staff watch the usability tests in real time, if possible. There's nothing more enlightening to a developer (or even a vice president of product development) than watching their interfaces misused and their assumptions misunderstood and not being able to do anything about it.

The best way to get observers involved is through a two-way mirror or a closed-circuit video feed. Bring in plenty of food (pizza usually works). The team can then lounge in comfort and discuss the tests as they proceed (while not forgetting to watch how the participants are actually behaving). Since they know the product inside and out, they will see behaviors and attitudes that neither the moderator nor the analyst will, which is invaluable as source material for the analyst and for the team's understanding of their customers.

If neither a two-way mirror nor a closed-circuit feed is available, it's possible to have members of the team observe the tests directly. However, there should never be more than one observer per test. It's intimidating enough for the evaluator to be in a lab situation, but to have several people sitting behind them, sometimes scribbling, sometimes whispering, can be too creepy for even the most even-keeled. The observer, if he or she is in the room, should be introduced by name since this acknowledges his or her presence and gives the observer a role in the process other than "the guy sitting silently in the corner watching me."

Observers should be given instructions on acceptable behavior and to set their expectations of the process.
USABILITY TEST OBSERVER INSTRUCTIONS

1. Listen. As tempting as it is to immediately discuss what you’re observing, make sure to listen to what people are really saying. Feel free to discuss what you’re seeing, but don’t forget to listen.

2. Usability tests are not statistically representative. If three out of four people say something, that doesn’t mean that 75% of the population feels that way. It does mean that a number of people may feel that way, but it doesn’t mean anything numerically.

3. Don’t take every word as gospel. These are just the views of a couple of people. If they have good ideas, great, but trust your intuition in judging their importance, unless there’s significant evidence otherwise. So if someone says, “I hate the green,” that doesn’t mean that you change the color (though if everyone says, “I hate the green,” then it’s something to research further).

4. People are contradictory. Listen to how people are thinking about the topics and what criteria they use to come to conclusions, not necessarily the specific desires they voice. A person may not realize that two desires are impossible to have simultaneously, or he or she may not care. Be prepared to be occasionally bored or confused. People’s actions aren’t always interesting or insightful.

5. Don’t expect revolutions. If you can get one or two good ideas out of each usability test, then it has served its purpose.

6. Watch for what people don’t do or don’t notice as much as you watch what they do and notice.

For in-room observers, add the following instructions:

7. Feel free to ask questions when the moderator gives you an explicit opportunity. Ask questions that do not imply a value judgment about the product one way or another. So instead of asking, “Is this the best-of-breed product in its class?” ask “Are there other products that do what this one does? Do you have any opinions about any of them?”

8. Do not mention your direct involvement with the product. It’s easier for people to comment about the effectiveness of a product when they don’t feel that someone with a lot vested in it is in the same room.

If the observers are members of the development team, encourage them to wait until they’ve observed all the participants before generalizing and designing solutions. People naturally want to
start fixing problems as soon as they're recognized, but the context, magnitude, and prevalence of a problem should be known before energy is expended to fix it. Until the landscape of all the issues is established, solution design is generally not recommended.

Tips and Tricks

- Always do a dry run of the interview a day or two beforehand. Get everything set up as for a real test, complete with all the appropriate hardware and prototypes installed. Then get someone who is roughly the kind of person you're recruiting, but who isn't intimately involved in the development of the product, and conduct a full interview with him or her. Use this time to make sure that the script, the hardware, and the tasks are all working as designed. Go through the whole interview, and buy the evaluator lunch afterward.
- Reset the computer and the lab in between every test. Make sure every user gets the same environment by clearing the browser cache, resetting the history (so all links come up as new and cookies are erased), and restarting the browser so that it's on a blank page (you can set most browsers so that they open to a blank page by default). Clear off any notes or paperwork from the previous person and turn off the monitor.
- If possible, provide both a Macintosh and a PC for your usability test, allowing the evaluator to use whichever one he or she is more comfortable with. You can even include a question about it in the screener and know ahead of time which one the participant typically uses.
- Don't take extensive notes during the test. This allows you to focus on what the user is doing and probe particular behaviors. Also, the participants won't associate their behavior with periods of frantic scribbling, which they often interpret as an indicator that they just did something wrong.
- Take notes immediately after, writing down all interesting behaviors, errors, likes, and dislikes. Discuss the test with any observers for 10–20 minutes immediately after and take notes on their observations, too.
How to Analyze It

Although some things are going to be obvious, a formal analysis is necessary to get to underlying causes and to extract the most value from the interviews. Analyzing the output is a three-stage process: collecting observations, organizing observations, and extracting trends from the observations.

Collecting Observations

There are three sets of observations to be collected: the moderator's, the observers', and the analyst's.

Collecting the moderator's and observers' notes is pretty straightforward. Get their notes (or copies), and have them walk you through them, explaining what each one means. In addition, interview them for additional observations that were not in their notes. These are frequently large-scale perspectives on the situation that the person made in the days or hours following the last test.

The analyst's notes are the most important and time-consuming part of the data collection process. The analyst should go through at least four of the videotapes and note down all situations where there were mistakes or confusion or where the evaluators expressed an opinion about the product or its features. He or she should note which features the evaluators had problems with, under what circumstances they encountered those problems, and provide a detailed description of the problem. The majority of the usability problems in the product will likely be found during this phase, as the patterns in people's behavior and expectations emerge.

Quantitative information, although not generalizable to the whole target market at large, is often useful when summarizing and comparing behavior (however, it's fraught with potential problems as people reading reports can latch on to largely meaningless numbers as some kind of absolute truth). To collect quantitative information, first create a measurement range for each question that everyone in the analysis team agrees upon. Don't use a stopwatch, and take exact numbers. The statistical error present in the small sample of people in a usability test swamps out the accuracy of a stopwatch. The most useful metrics are the ones that are the most general. Flow Interactive, Limited (www.flow-interactive.com), a U.K.
user experience design and evaluation consulting company, uses the following range to measure how long people take to perform a task:

0—Fail
1—Succeed very slowly in a roundabout way
2—Succeed a little slowly
3—Succeed quickly

Most of the time, this is all the precision you need since an order-of-magnitude measure is all that’s necessary to be able to make critical comparisons. Each scale should have three or five steps (don’t use two, four, or six since it’s hard to find a middle value; don’t use more than five because it tends to get confusing) and a separate value for failure.

Make a grid for each participant consisting of the task metrics you’re going to collect. As the videotapes are being watched, note the severity in each cell (when appropriate, define severity using the same language and scale that is used by the development team to define how serious code bugs are). For the fork tasks, the following table would reflect one person’s performance.

<table>
<thead>
<tr>
<th>User: Marion</th>
<th>Time to Read</th>
<th>Errors</th>
<th>Time to Complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find Louis XIV</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Buy replacement</td>
<td>5</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Find similar forks</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

**Key:**

0—Don’t read
1—Read very slowly
2—Read moderately slowly
3—Read quickly

0—Fail because of errors
1—Many errors
2—Some errors
3—Few or no errors

0—Fail
1—Succeed very slowly in a roundabout way
2—Succeed a little slowly
3—Succeed quickly
Then, when compiling the final analysis, create a table for each metric that summarizes the whole user groups' experience. For the completion time metric, the table could look as follows.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Marion</th>
<th>Eva</th>
<th>Marc</th>
<th>Barb</th>
<th>Jon</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find Louis XIV</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1.2</td>
</tr>
<tr>
<td>Buy replacement</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1.8</td>
</tr>
<tr>
<td>Find similar forks</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0.4</td>
</tr>
</tbody>
</table>

The average numbers, although not meaningful in an absolute context, provide a way to compare tasks to each other and between designs.

Note down feature requests and verbatim quotations from the evaluators, especially ones that encapsulate a particular behavior ("I don't understand what "Forkopolis" means, so I wouldn't click there," for example). Feature requests are often attempts to articulate a problem that the evaluator can't express in any other way. However, they can also be innovative solutions to those same problems, so they should be captured, regardless.

**2x Video Decks Are Cool**

To make the video review process go faster, I recommend using a video deck (or a digital video player) that can play back video and audio at 1.5 or 2 times natural speed. The speech is still understandable (although silly, since people sound like chipmunks unless the voice is pitch-shifted down, as it is done on Sony's professional-grade video hardware), and it's possible to make your way through a tape much faster.

If time and budget allow, a transcription of the whole session is helpful, but it should be used only as an aid in observing the tapes because it misses the vocal inflection and behavior that can really clarify some situations. For example, a confused pause of five seconds while an evaluator passes his pointer over every single visual element on the screen looking for somewhere to click is insufficiently conveyed by his statement of "Aha! There it is."
Organizing Observations

First, read through all the notes once to get a feeling for the material. Look for repetition and things that may be caused by common underlying problems.

Then put all the observations into a pile (literally, or in a single large document). Opening a separate document in a word processor, go through each observation and group it with other similar observations in the new document. Similarity can be in terms of superficial similarity ("Term not understood"), feature cluster ("Shopping cart problems"), or in terms of underlying cause ("Confusing information architecture"). Group the observations with the most broadly sweeping, underlying causes. Pull quotations out and group them with the causes that they best illustrate.

Extracting Trends

Having grouped all the observations, go through the groups and consolidate them, separating the groups of unrelated topics. Throw away those that only have one or two individual observations. For each group, try to categorize the problem in a single short sentence, with a couple of sentences to fully describe the phenomenon. Explain the underlying cause as much as possible, separating the explanation of the phenomenon from your hypothesis of its cause. Concentrate on describing the problem, its immediate impact on the user experience, and the place where the problem occurred. Be very careful when suggesting solutions. Ultimately, the development team knows more about the technology and the assumptions that went into the product, and the responsibility for isolating underlying causes and finding solutions is theirs. Your recommendations should serve as a guide of where solutions could be found, not edicts about what must be done.

Describe the severity of the problem from the user's perspective, but don't give observations numerical severity grades. If a shorthand for the characterization of observations is desired or requested, categorize the observations in terms of the effects they have on the user experience, rather than giving them an arbitrary severity. Such a scale could be "Prevents an activity," "Causes confusion," "Does not match expectations," "Seen as unnecessary."
Once all this is done, you should have a list of observations, hypotheses for what caused the phenomena, and quotations that reinforce and summarize the observations. You're ready to present your results to the team! Effective presentations are covered in Chapter 17.

Example

This is a short report summarizing a test on another Webmonkey prototype for the site's development team. It builds on the previous testing that the site had gone through and focuses on the changes made to the front door and the renaming of various sections in the site.

Executive Summary

Five Web developers were shown the functional prototype for Webmonkey 4.0. In general, they liked it, especially the tutorials and the color scheme, but some of the organization confused them (specifically, the difference between the “Categories” and “Tutorials” sections). The new folderlike navigation metaphor made sense to everyone, and they wished that it was on every page. Everyone saw the “Cool Tools” section, but thought it was an ad and ignored it, and although they liked the “Inspiration” section, they expected it to be more than just animation in the long run.

Finally, a couple of people said that it would be cool to have Webmonkey link to good, useful external content in an unbiased way since it would be useful and would reinforce Webmonkey's street creed.

Executive summaries are very useful when communicating results. The vice president of product development may never read the report, but a couple of paragraphs giving 50,000 of the results of the usability test are likely to be read. When attaching the report in an email, the executive summary should be included in the email, while the rest of the report—including the executive summary—is included as an attachment.
Procedure

Five people who spend a significant amount of time developing Web sites were invited. They were first asked some preliminary questions about their general net usage and where they went for developer information (both on the Web and in general). They were then shown the site prototype and asked to go through it in detail, concentrating on specific details, including the folder-style navigation, and the Cool Tools section. After giving their responses to the front door, they were asked to scroll down through one of the top stories, talking about their experience with the interface and their thoughts on the content. They were then asked to look for some specific content as a way of gauging their understanding of the layout of the site. Finally, they were asked some wrap-up and blue-sky questions, and the test was concluded.

A fast description of the procedure demystifies the process and provides important context for report recipients to be able to understand the results.

Evaluator Profiles

Michael

Michael spends more than 40 hours a week on the Internet, 20 hours of which is spent making Web pages, including design, programming, and production. Of all the development sites, he likes Webmonkey because of its "broad range." He also regularly reads "Flash Zone" because it can give him tutorials that he can't get in print. For CGI work, he follows another site, "CGI Resources" (the "CGI Zone"? The specific site wasn't clear from the interview).

John

John spends 30 hours a week on the Internet, half of which is work related. He spends at least 10 hours a week making Web sites, including design, markup, and code. He uses reference books and Webmonkey for technical Web-related information. He also goes to "SGML University" and has never been to builder.com. Most of the time, he goes to these sites with specific questions. In general, he would like developer sites to be better organized by topic.
David

David spends 20–30 hours a week on the Internet, 75% of which is work related, and 5% to 10% is spent doing Web development, most of which is design. His main sources of technical information are Webmonkey and notes from school. He has never seen builder.com and goes to Webmonkey for both technology updates and to answer specific questions.

[remaining profiles omitted]

Evaluator profiles are useful both to help the report reader understand the context in which people’s statements were made and as a way to personify the participants to those who were unable to observe the tests. Like the user profiles created in Chapter 7, these profiles help personalize the abstract concept of a product’s users and make the results that much more immediate.

Observations

General Observations

1. People like tutorials above all else. All the evaluators were drawn to the tutorials, sometimes to the exclusion of other content. The tutorials section was often the first one mentioned when the evaluators were asked where they would click on next. It was also the section people preferred to go to for general information even though there was a broader range of content in the “Categories” section.

2. Almost everyone said they liked the color scheme. Without being asked about it, most of the evaluators volunteered that they really liked the color scheme on the homepage.

3. People generally come to development sites with specific questions in mind, not to see “the latest.” When asked whether they go to sites like Webmonkey to catch up on the latest technology or to get answers to specific questions, people generally said that it was to answer specific questions.

Likewise, when asked how they preferred to navigate through sites like Webmonkey, people said that they preferred searching, rather than browsing, since that brought them closer to the specific information they were looking for.
Features

1. There was confusion between the content people would find in the "Categories" section and the "Tutorials" section (and, to a lesser extent, between the "Tutorials" section and the "Guides" section). Partially because of the ambiguity of the "Categories" name and partly because of the similar—but not completely identical—labels in the two sections, people were confused about what they would find in each section.

2. The "Cool Tools" section was noticed early by nearly everyone, but treated as a big ad and, thus, ignored by most. Until it was pointed out to a number of the evaluators that there was nonadvertising content in "Cool Tools," they did not appear to notice it. Most pointed to the picture and the price as indicators of why it was considered to be advertising content.

3. A couple of people saw and liked the idea behind the "First Time Here" link.

4. People didn’t know to go to "Backend" for CGI topics and were unsure of what kinds of things would be found there. One person mentioned that he’d prefer it be called “Server Stuff” or something similar.

5. People didn’t notice the reference pop-up on the main page at first, and when they did, they weren’t sure about its relationship to the content accessible from the folders in the left-hand margin. However, most everyone found it to be a useful tool with contents that made sense (except for "ISO Entities"). A couple of people suggested that it be put in the left-hand margin along with the folders.

Whenever possible, include screenshots.
Navigation

1. Everyone understood the folder metaphor on the front door.
2. The inconsistent content and appearance of the left margin navigation was somewhat confusing. A number of people mentioned that they were surprised that the navigation in the left-hand margin changed from the front door to the subsections and the tutorials. Several mentioned that they would have preferred a continuation of the folder metaphor from the front door.
3. People generally understood the pathnamelike breadcrumb navigation at the top of the page though not everyone noticed it. The biggest disconnect came when people would jump to a “Tutorial” directly from the top page (thus expecting the path to be something like “home/tutorial/javascript”) and the path read “home/categories/javascript/tutorial,” which did not match their expectation.

Naming

1. The “Categories” name wasn’t clear. People weren’t sure what “Categories” was referring to, and one person didn’t even see it as a section to be clicked on.
2. “Guides” wasn’t clear as a section title. There was confusion in most of the evaluators between the “Guides” section and the tutorials.
3. Likewise, “eBiz” wasn’t clear. Although not everyone was asked about it, the couple of people who were didn’t know what to expect on the other side.
4. “Heard on the Street” was ambiguous. Without looking at it when the participants were asked to define what the “Heard on the Street” section was and how it was different from the other content sections, most people said that it was a “most recent” section or that it highlighted some news development.

Conclusion

The combination of the attraction of the concept of tutorials with the unclear wording of “Categories” caused people to frequently ignore the categories section entirely.

A lot of the confusion comes from the ambiguity of single-word names. “Categories” and “Guides,” although they adequately describe the sections after people have already seen them, give people little information about them
before they've seen them, since, as words, they're quite general. Thus, the
naming of sections (and maybe everything on the site in general) has to be
done with the user's context in mind. What may, in retrospect, make perfect
sense may be confusing and ambiguous before a definition is produced.

Quotations

Michael

“'You know the functionality is out there, you just want to know how to put
it together.'”

“When I started going there, it was a beginning site, it was very good
for that, but then it kind of stayed there and I moved on.” [re: Builder.com]

“I saw 'Cool Tool Pick,' and I thought that this would say GoLive
and this would be Dreamweaver and this would be something else.”

“If one of them weren't there, it might be easier to differentiate be-
tween them.” [re: “tutorials” vs. “categories”]

John

“It stands out without being the obnoxious Wired magazine look.”

“If I were coming here to find a specific answer on something, I would
go to ‘Tutorial.’”

“‘Categories’ is everything and these are just subtopics.”

“I would prefer to see the latest tutorial [rather than Inspiration] at
the top.”

[remaining quotations omitted]

A couple of sentences of evaluators' actual words often better il-
lustrate the points you're trying to convey than a paragraph of ex-
planation. The readers can then see the patterns you're trying to il-
lustrate for them. When placed next to each point, they serve to
reinforce each point as it's made. When presented all at once, they
communicate the feel of a usability test.

Usability tests are one of the workhorses of user experience re-
search. They can be done quickly and inexpensively, and provide
a lot of immediately actionable information. Too often they're used as
the only form of user feedback, but when used correctly, they're an
invaluable tool.