Chapter 7| Usability Testing

By Mike Kuniavsky

Mike Kuniavsky is a founding partner of Adaptive Path, a user experience consulting company based in San Francisco. He's the author of Observing the User Experience: A Practitioner's Guide to User Research *(Morgan Kaufmann, 2003), from which this chapter has been adapted, and he has been developing commercial websites since 1994. He helped develop the second e-commerce site ever (launched two months before Amazon), designed the interface to the award-winning search engine HotBot, and created the Wired User Experience Laboratory, where he served as chief investigator.*

Overview

All over the world, governments and public-sector organizations are creating e-government initiatives to become more efficient, transparent, and accessible to their constituents. In some cases this has created great new services that benefit the whole community. In others, it has been a costly experiment with little payback in terms of serving citizens or organizations. In most cases, though, it's somewhere in between. In these cases, mediocre user experience makes valuable information hard to find and important services difficult to use. This may not cripple the service, but it reduces its reach into the community. The persistent few who are able to make their way through the service get what they need, while many others abandon it in frustration.

Usability testing can often easily identify problems that are not so bad as to cripple a service, but bad enough to severely limit its audience. One-on-one tests can quickly reveal an immense amount of information about how people use a prototype. Usability testing is probably the fastest and easiest way to tease out show-stopping usability problems before a product launches.

A usability test is a series of structured interviews focused on specific features in an interface prototype. The heart of each interview is a series of tasks that are performed by the interface's evaluator. The patterns that emerge from these interviews often show important functionality and presentation problems. This immediately helps the development team see whether people understand their designs as they're intended, and to prioritize where to focus their (always) limited resources.

Usability testing can't tell you whether people want or need the services provided, or whether those services actually solve real-world problems. (Many other user research techniques do that.) However, it's a valuable technique to help you understand whether people can use the solutions as intended.

Best of all, usability tests can be used throughout the development cycle to evaluate hunches, investigate new ideas, and check specific features.

The following is a description of how to do an inexpensive usability test. There are many ways to spice up usability tests to make them more in-depth or more accurate, but the basic method is so straightforward that there is really no excuse for not conducting one.

Choose an Appropriate Audience

One of the most important aspects of doing any kind of user research is finding the right audience to research. Since e-government sites are designed for everyone to use, usability research is sometimes not done to avoid excluding any audience. In doing so, the organizations creating the product are, in effect, excluding *all* audiences from the design process.

Choose Primary Audiences

Before you begin any kind of user experience, determine who your *primary user audiences* are. These audiences won't be the only people who'll use the service, but they'll be the groups most likely to use it. If a service works well for them, then it probably works for other groups as well. The criteria for what makes a good primary audience profile depends on the tasks you're trying to enable. For e-government projects, the criteria will probably be based on people's technological experience and, most importantly, on what they're trying to accomplish. Their tasks should be set out explicitly (such as "order a birth certificate," or "request an inspection") and prioritized.

Narrow Your Focus

Once you've determined your primary audiences, narrow your focus to groups of people who are likely to give the best feedback for the service you're creating. This will be a subset of all of the people in the user audience. For example, you probably don't want to test how well people know how to use computers; you just want to know how well they'll use your service. So for your *test audience* you'll probably recruit people who know how to use a computer and a web browser. If you want to test a new version of your site, you might call in a group of people who have experience with the old interface. Or, if you're interested in the experience of first-time users, you could do the opposite and choose people who have never seen your site. These considerations don't normally play a large role in determining the audience for the service as a whole, but they can be important for determining what makes a good user research audience.

Find 5 to 8 people for each audience. Having five to eight people reveals the most severe usability problems. Since you're trying to understand what the big problems are, not necessarily measure their proportion to the general population, it's unnecessary to have statistically significant samples. Working with a small audience is quite effective in uncovering many of the most severe issues that make a site appear difficult and discourage use. Moreover, if additional interviews need to be conducted because results are inconclusive or you believe that major problems still remain, it's often easier to schedule a series of small groups and adjust your focus than to try and extract all possible problems from a larger group.

Users with special needs. For any important audience that lies outside of the primary user audience, it's often unnecessary to recruit as many people as for a full test. In these cases, it may be possible to get key information by recruiting one or two people from this special audience in addition to a full primary audience group. If their experience is significantly different than the primary audience, then you know that another test—and possibly another design—for just that group is necessary.

What defines *special* depends on how you've defined the primary audience. In some cases it could mean people who are much older or younger than the primary audience; in others, it could be people who have a lot more or less computer experience. Regardless, a round of research with this sub-audience should focus on the specific needs of the group, rather than the broader picture. For example, a test for a vision-impaired audience might focus on how well a site can be read with a screen reader, rather than investigating how well information is organized or how things are named, since those are likely to be the same as for the primary audience.

Choose Appropriate Tasks

The second step is to determine which features to test. These, in turn, determine the tasks and the order in which they're presented. Five major features can be tested in a 60- to 90-minute interview and typical tests range from one to two hours.

Make a List of Features

I typically start by making a list of all the features that are either:

- Used often
- New
- Highly publicized
- Considered troublesome, based on feedback from earlier versions
- Considered important by users

In other words, what five things should people be able to do above all else?

In a light-rail information site, for example, users should obviously be able to find schedules. But they should also be able to find route maps, information on how to get to popular sites (the airport, for example), connection information, and rules for bringing bicycles on board.

Create a Task for Every Feature

For every feature on the list there should be at least one task that exercises it. It's useful to have two or three alternate tasks for the most important features, in case there's extra time or the first task proves too difficult or uninformative. A *task* is an example where someone needs each piece of information. Tasks should represent typical user activities and be sufficiently isolated to focus attention on a single feature (or feature cluster) of the site. Good tasks should be:

- **Reasonable.** They should be typical things that people do. Someone is unlikely to take 20 different trains in one day, so that's not a typical task. Two train transfers in one trip may not be common, but it's reasonable, and should probably be included.
- **Described in terms of user goals.** Every website is a tool. It's not an end to itself. Even when people spend hours using it, they're doing something with it. Phrase your task as something that's related to the evaluator's life. For example, rather than asking, "What are the stations and weekend train schedules for the J and L trains?" phrase it in terms of a story that people can relate to. "You live in the Excelsior neighborhood and want to take your kids to the zoo on Saturday afternoon, but don't want to drive. How would you get from your house to the zoo on Saturday by light rail?"

- **Specific.** For consistency between evaluators and to focus on the parts of the product you want to test, the task should have a specific end goal. So, rather than saying, "Find some train schedules," say, "You want to go to San Jose from the Haight-Ashbury district of San Francisco on light rail. Starting from the home page of www.transitinfo.org, find out how you would do that."
- **Doable.** If your site only has trains, don't ask people to find bus schedules. It can be 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 service. So, for example, first ask people what their local light-rail station is, then ask them to find a weekday schedule for that station, then how to transfer trains, and then rules about bicycles. This resembles the sequence in which people are likely to explore the site in real life and you can tell a rich story about it.
- **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. When recruiting for a light-rail site usability test, for example, it might be useful to limit recruiting to people who've used the rail system at least once in the last year, but who've never used the website tool to get a schedule.
- **A reasonable length.** Most features are not complex enough to take more than 10 minutes to use. If an experienced user needs more than 3-4 minutes to complete a task, it probably needs to be broken down into sub-features and reprioritized.

Each task should be described in a sentence or two, and written from the perspective of the user. So, for example:

You live near downtown Oakland and you'd like to get to 3COM Park for a 49ers game on Sunday on the train. Can you find information about that on this site?

Once you have a list of people and tasks, it's time to do the interviews.

Conduct Usability Testing Interviews

Usability testing interviews are structured conversations, rather than one-to-one surveys. A *discussion guide* (sometimes called a *protocol*) gives you a consistent way of asking questions without constraining the answers that people give, while the interview acts as more of an interpretation of the guide, rather than a rigid point-by-point examination. The following instructions are for a task-focused interview, but you can also conduct an interview that's more focused on understanding people's background and expectations.

Write a Discussion Guide

First, write a script that you and your invited evaluators will follow. Put a short service description at the top of a page. This will be all that the evaluators will be told about the service. Don't tell them anything else. In the real world, a short description and a link is often all that someone may know. For example:

Transitinfo.org brings together all of the public transportation information for the Bay Area, providing routes, schedules, and maps for all of the major public transportation services.

Next, write the tasks on separate sheets, one per page.

Set Up the Testing Environment

Invite your evaluators, one at a time, to sit at a computer in a quiet room where you won't be distracted. Out-of-the-way conference rooms work well.

Set up the computer for the tasks. If you're working on a website, set up the browser in the most generic configuration possible, removing custom toolbars, custom colors, display options, and extraneous bookmarks. Bookmark the start pages that people will need for each scenario that you've written.

Explain the Process

As each evaluator arrives, prepare him or her for what's going to happen. Make the evaluators feel comfortable. Introduce the process by saying:

- They've been invited to help you understand which parts of the service work for them and which are confusing.
- Even though it's called a test, they're not the ones being tested. They're evaluating how well the product works, so there's nothing they can do wrong. Emphasize that it's not their fault if they can't get something to work and that they won't hurt anyone's feelings if they say something bad about the site.
- It's important that they speak all of their thoughts aloud. Suggest that they give a play-by-play narration of what they're doing and why they're doing it.
- You'll stay in the same room and quietly listen to them while taking notes, but they should ignore you, focusing on the tasks and their play-by-play descriptions.

Conduct the Interview

Once the participants are comfortable and you've given them the initial instructions, read the product description and hand them the sheets with the task descriptions. Ask them to do the tasks as best as they can, though if they can't figure one out in a couple of minutes, they should feel free to move on to the next task. Reinforce that they should speak aloud the whole time.

Then, let them talk. Sit back and watch, quietly taking notes. If they get stuck, don't tell them where to click or what to look at. No matter what, don't tell them how to do something. If they seem to be particularly frustrated, tell them that it's not their fault if something seems impossible, and that they should move on to the next task.

Once all the tasks have been completed, or the test time is over, it's time to stop. Ask the evaluators to tell you their general impression and whether they would use the service in real life. Then give them a gift for their time (always reward participants, ideally with something of value, even if it's a gift certificate to a local restaurant), thank them, and send them on their way.

Finally, reset the computer for the next evaluator, clearing the cache and history, and setting it to a blank page.

Analyze the Results

As soon as the usability test is over, review your notes and ask yourself the following questions:

- What worked well?
- Did the users consistently misunderstand anything? If so, what?
- Were there any mistakes consistently made? If so, what?

- Did they do the things that you had expected them to do? If not, what did they do?
- Did they do things in the order in which you had expected? If not, what order did they do them in?
- What did they find interesting?
- What did you expect them to find interesting that they did not find interesting?
- Did they know what the site is for? Did they miss any big ideas?
- How many of the tasks were they able to do? Which ones did they have the most trouble with?
- When did they look frustrated? What were they doing?
- Do you know what their expectations were?
- Did the site meet their expectations? If not, where did it fail them?
- Were they ever confused? What were they doing when they were confused?

At this point, you should have some ideas of where your service has problems. You've probably seen several things come up again and again. Maybe people don't understand the name you've given to a certain section. Maybe they don't see a critical piece of information. Maybe they aren't interested in what's being offered. Maybe they love it and it fulfills everything they want, but they've never heard of it. All these things are good to know since they tell you where you're having problems and, equally important, where you're not.

What Next?

Simple usability tests like this reveal a lot of information about how well your web service meets the goals you've set, but it's only the tip of the iceberg of user research methods. There are dozens of ways to get feedback from people about their capabilities, needs, and desires. Working with the end users of your service lets them represent themselves in the development process and create the e-government services that are most useful and valuable to them. It provides information early in development, minimizing development time and maximizing resources by reducing backtracking and bug fixing. And, most importantly, it ensures that the greatest number of people who can benefit from your service will be able to use it.

References

Adaptive Path, 2001-2003. *Publications*

Center for Applied Special Technology, 1999-2002, *Bobby* (an accessibility testing tool)

Instone, Keith. 1995-2003. *Usable Web: User Testing*

Kuniavsky, Mike. 2003. *Observing the User Experience: A practitioners guide to user research*. San Francisco: Morgan Kaufmann. ISBN: 1558609237

Nielsen, Jakob. 1994. *Guerrilla HCI: Using Discount Usability Engineering to Penetrate the Intimidation Barrier*

Perfetti, Christine and Landesman, Lori. Eight is Not Enough. User Interface Engineering