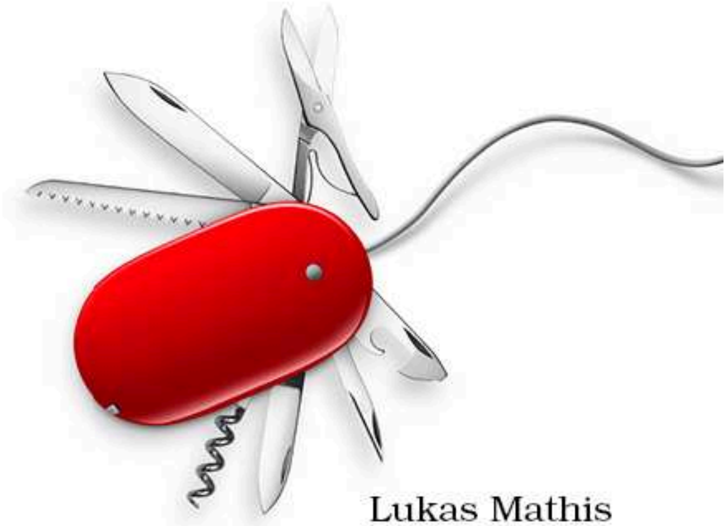
Designed for Use

Usable Interfaces for Applications and the Web

The Pragmatic Programmers

Designed for Use

Create Usable Interfaces for Applications and the Web

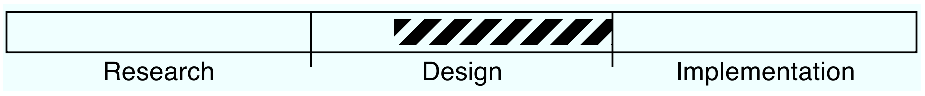


Lukas Mathis

edited by Jill Steinberg

Chapter 11

Paper Prototype Testing



What's the Technique?

In the previous chapter, I explained how to sketch your product. I said that you should sketch before you commit to code because sketches are easier to change. But I kind of skirted around the issue of how you know what changes to make.

In some cases, issues with a design—such as features that are available at the wrong time or overcrowded screens—become obvious once you see it sketched out before you.

But in other cases, it's not obvious. When is a design good? If you have more than one idea of how to do something, which idea will work best? Where are the issues with your current idea? This chapter will help you find answers to these questions.

Your sketches are basically primitive forms—prototypes, if you like—of your product. As such, you can run usability tests with them and see whether your designs work the way you expect them to work. The easiest way to do that is to show them to people and see whether they understand them.

If you're working alone or in a small team, feel free to skip the part of this chapter that explains how to run full usability tests with paper prototypes.

Why Is This a Good Idea?

The earlier you find issues with your design, the easier it is to fix them. Every problem you fix with a pencil on paper is a problem you don't have to fix in code.

Are There Any Prerequisites?

You should have started doing sketches.

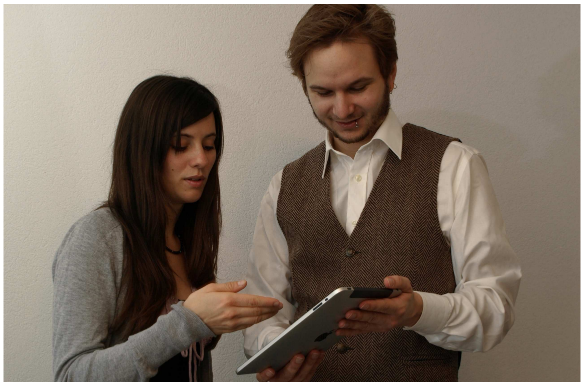
11.1 Guerilla Paper Prototype Testing

At this point in the process, you have static representations of your product. Whether it's simple, crude sketches, wireframes, or detailed mock-ups, your product exists in the form of a series of pictures.

At its most basic, paper prototype testing simply means you get a real person to "interact" with those pictures in order to gauge whether your planned user interface is understandable to your users. This could be as simple as showing somebody a drawing of a user interface and asking them something like "If you wanted to change the font size of the text document on this screen, where would you click?"

You need a reasonably detailed sketch, wireframe, or mock-up of one of your product's screens to do paper prototype tests. Then, find somebody willing to spend five minutes with you. Show them the screen, and ask a generic question: *"What do you see on this screen? What do you suppose this product could be used for?"*

Alternatively, you can ask a simple task-based question: *"If you had to use this application to add a picture to a document, what would you do?"*



Paper?

This chapter is about "paper prototypes," but the "paper" part isn't meant too literally. You don't necessarily need to do sketches on paper to do these kinds of tests. You can also do mock-ups on a computer and print them out. Or, instead of printing them, put them on a tablet computer, and show that to people. You could even create a simple interactive prototype of your product by putting sketches into an application like PowerPoint or Keynote.

I guess this chapter should really be called "how to do usability tests with (mostly but sometimes not entirely) static sketches or mock-ups of your final product," but that didn't fit into the book's layout.

This kind of test will give you a general idea of whether people understand your design and what parts people don't get. That way, you know where you have to make changes and where you're on the right track.

Simply showing people a mock-up of one screen of your product helps you understand whether people understand that screen. This is useful, but we can go a step further and test interactions. To do this, prepare more than one screen in advance. If you need pop-up elements or other elements that have to be added or removed from the screen, also prepare them in advance; draw them on Post-it notes so you can easily add them to the prototype.

Prepared in this way, you can run actual task-based usability tests. Of course, you can't let people veer too far off track, but if they do, you already know that there's a problem with your design, because people don't follow the path you expected them to take.

You can do this type of simple test with pretty much anyone. It takes only a few minutes, and it's quite easy to explain:

> I'm working on a new Twitter app. You know what Twitter is? OK, great. I'm currently working on the design of my app, and I'm trying to figure out whether people understand what I've done. You know, when you work on something for weeks, you lose any objectivity, and it's hard for me to tell whether what I'm doing makes sense to anyone else. I was wondering

> whether I could show you some of these designs and ask you a few questions about them to see whether people can figure them out? It won't take more than five minutes.

You can start doing this with friends and family members to get used to running these tests. Then, ask random people. A good place to start is a café. People usually have a few minutes to spare.

Once you see how people interpret your designs, it will be easy to figure out where your designs work and where potential problems lie.

11.2 Running Full Usability Tests with Paper Prototypes

The simple tests I've described so far will allow you to do some early usability testing on your designs. There is, of course, a much more elaborate way of doing prototype testing. I don't recommend this process for most smaller teams, but if you have the time and the people required to do such tests, they can give great feedback even before a single line of code is written.

A prototype typically represents only a subset of your finished product. Not every screen of your final product will be part of your prototype. Not every feature will be represented. For this reason, prototype usability tests are almost always based on specific tasks: you will give a person a specific task and ask that person to execute that task on your prototype. That way, you have a pretty good idea of what screens and UI elements you will need to prepare.

So, the first thing you want to do is define tasks you want to test.

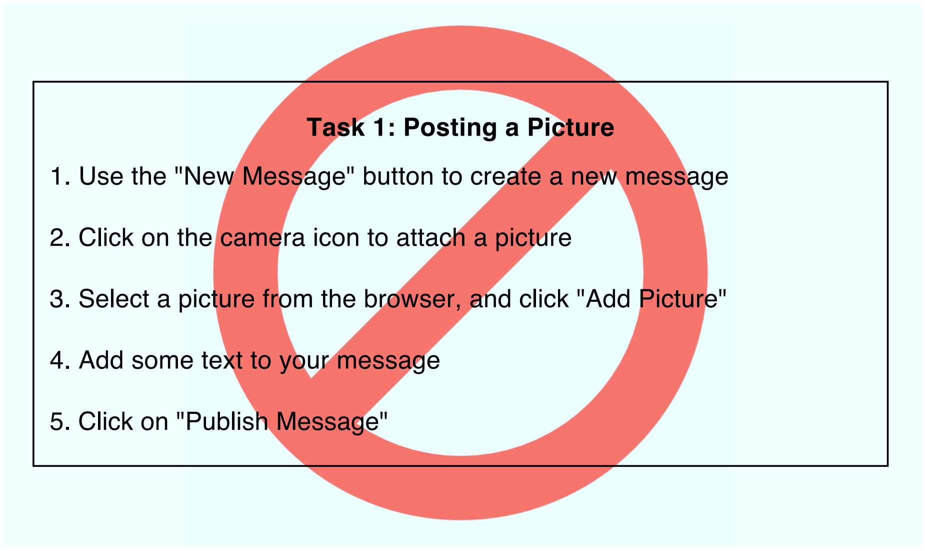
Defining Tasks

Now that you've created a storyboard and mock-ups, you probably have a pretty good idea of where potential user interaction problems might be found. If it was hard to come up with a user interface for a feature, this is probably something you want to test early. If it is a critical, central feature of your product, it's definitely something you want to test early. Pick the features that are important to your product, that you think might be hard to use, or that you think might cause problems.

Next, come up with tasks that use these areas of your product. You can draw upon the user research you did in the first part of the book, if you need.

Remember that you're not looking for opinions. Usability tests are not focus groups. The goal of a usability test is to observe people as they interact with your design so that you can find flaws in the interaction design. Pick tasks that cause people to actually use the product.

It's important that the task not be too prescriptive; tell the user what goal he has to achieve, rather than what steps he has to follow. The task should *not* look like this:



Task 1: Posting a Picture

1. Use the "New Message" button to create a new message
2. Click on the camera icon to attach a picture
3. Select a picture from the browser, and click "Add Picture"
4. Add some text to your message
5. Click on "Publish Message"

A task like this tells you only whether people are capable of following instructions. It doesn't tell you whether your product is usable. Instead, the task description should state the goals and leave the individual steps up to the user. It should look something like this:

You are at one of your company's customer events. You want to take a picture of the room and publish it on your company's Twitter account.

This is a scenario one of your users might find himself in. It will show you how somebody in this situation might interact with your product.

Perhaps counterintuitively, tasks are one area where you should avoid using your official terminology. If you use words in the task description that people can find in the user interface, you are essentially giving them a hint about how to perform the task.

It's a good idea to prepare at least five or six tasks; more is better. Each task should take between two and ten minutes. It's likely that you won't make it through all of the tasks, but it's hard to tell before doing the actual test, so it's better to be prepared if people blow through your tasks quickly.

Creating the Paper Prototype

Based on the tasks you've come up with, it's pretty easy to tell which screens you will need to show to people (but be liberal when deciding which screens to include in your prototype; people may not always take the most obvious steps toward the goals you've set).

You can use existing sketches for paper prototypes, if they're not *too* crude and simple. Using very simple sketches can cause problems if they are not easily readable by people who are not already familiar with your product.

To turn a sketch or mock-up into something suitable for a usability test, you need to add the user interface elements surrounding your own user interface. For example, if you're testing a website, add relevant parts of the browser user interface to the prototype (forward and back buttons, the address bar, the title bar, and so on). Similarly, if you're testing an application, you should add the menu bar and maybe even parts of the desktop if it is an application running on a desktop operating system.

After you've created the screens, you should think about state changes on individual screens.

What happens if somebody clicks a drop-down menu? You should prepare any pop-up windows in advance. Some people like to use Post-it notes for them, which makes it easy to stick them to the paper prototype once the user activates them.

What happens if the user has to type data into a field? There are several ways of handling this. If you are not going to reuse the paper prototype, you can simply give people a pencil and an eraser and let them change the data on the screen using these tools. If you are going to reuse the prototype (say, you're doing more than one test), you could use transparencies. Put them over your prototype, and let people draw on the transparency.

Finally, you should print each task on its own piece of paper. Since you don't know how many tasks a user will get through, using individual pieces of paper for each task allows you to hand out tasks as you go along or switch the order of the tasks on the fly if you need to do so.

At this point, you should run through each task. Make sure that the individual goals are feasible and that you've included every screen and pop-up element you're likely to need.

Paper Prototype Creation Checklist

- ☐ Prepare all the screens people are likely to progress through while doing your tasks. Don't make them too crude.
- ☐ Add the surrounding user interface elements (such as browser windows around sketches of a website) where necessary.
- ☐ Create the pop-up elements you might need.
- ☐ Prepare some way for people to draw on top of your prototype.
- ☐ Print out the tasks on individual pieces of paper.
- ☐ Do a test run of each task, and make sure you've prepared all of the screens and pop-up elements you're likely to need.

Preparing for the Test

To run a usability test with your paper prototype, you need at least one additional person: the person who is going to do the tasks. This person is sometimes called the test subject, but since you're not testing this person, I usually call her the tester. She's the one who is testing your design, after all.

I won't go into great depth on how to recruit testers in this book. There are many good resources on the topic,[1] but generally, pretty much anyone outside of your company will do. It's best to avoid recruiting testers from within your company. They are too familiar with your product and your company's jargon; this familiarity may mask usability problems with your product. It sometimes makes sense to recruit testers from your product's target audience, but generally, it doesn't matter too much. Almost anybody will do, including friends and family.

How many tests should you run with any version of the paper prototype? Paper prototypes take a bit of time to create, so it makes sense to test each prototype with more than just one person. Additionally, with a paper prototype, it's easily possible to make small adjustments

1. This Nielsen Norman report does a great job: http://www.nngroup.com/reports/tips/recruiting/.

The Original Computers

It seems funny to us that we would call a human *the computer*. Historically, though, computers actually *were* humans. The term was first used around 1600 and referred to people who carried out calculations.

It was only during World War II that machines started to take over this task, when people like Konrad Zuse and Alan Turing created the first computing machines. As these machines became more prevalent, the term *computer* eventually changed its meaning.

between tests. A good solution is to invite three or four testers and schedule them two or three hours apart. That way, you have enough time between tests to quickly go through the issues you've found and, if possible, make changes to the paper prototype accordingly.

While you can run a paper prototype usability test on your own, it's helpful to have at least one additional person there to assist you. Since you have created the paper prototype and since you know how the storyboard of your application looks, you are the perfect person to "play the computer" during the test. This means you will have to switch screens, show pop-ups, and simulate the user interface. This leaves you with little time to interact with the tester. Although you can do both if you have to, it makes sense to find a second person who can help you with this. This person is called the *facilitator*. Figure 11.1, on page 116 shows a typical setup for paper prototyping.

The facilitator is the person who guides the usability test by introducing the tester to how the test works by giving out tasks and by answering any questions the tester might have.

Computer	Controls the paper prototype, reacts to user input
Facilitator	Introduces the tester to the process, gives tasks, answers questions
Tester	Interacts with the paper prototype according to the given tasks

The facilitator should take notes during the test and (if you record the session) add the recording's timecode to the note so that each note can be matched to a video of what occurred. Taking good notes during the test can keep you from having to sift through hours of recorded usability tests.

Recording the session can be a good idea. It allows you to go through the session after it's over. Viewing a session after the fact allows you to focus specifically on what the tester is doing and will provide a ton of insight into the kinds of problems she encountered; often these are small things you might not have noticed during the test. Additionally, if you're not the one making the relevant decisions, a movie that shows people repeatedly failing at the same task can often quickly persuade people who otherwise don't believe that there even *is* a problem with the user interface.

I tend not to go into legal issues in this book. But remember that in addition to having testers sign a consent form,[2] the facilitator should also explicitly get permission from the tester to record the session if a recording will be made.

Test Preparation Checklist

- ☐ Recruit three to five testers, and schedule them about two hours apart.
- ☐ Find and train somebody to act as the facilitator.
- ☐ If you want to, prepare a way to record the test.
- ☐ Prepare a consent form for the tester.

Preparing the Tester

Have the tester sit across from the person playing the computer so that the computer can change the prototype in front of the tester. The facilitator should sit next to the tester, preferably slightly behind her so as not to be in the way.

Once everybody is ready, the first thing the facilitator has to do is to explain to the tester that it is not she who is being tested, but the user interface. This is basically the standard introduction you give at every usability test; you can read more about it in Chapter 28, *Usability Testing*, on page 242. A short version of the introduction might go:

2. If you don't already have a consent form for usability tests, I suggest you write one with the help of a lawyer familiar with the legal requirements in your location. Just make sure that your testers will be able to understand any legal provisions in the consent form.

Observers

The three roles I've mentioned so far are *computer* (the person who controls the paper prototype), *tester* (the person who interacts with the paper prototype), and *facilitator* (the person who leads the test). For most smaller companies, these are often all the people you will have in a usability test for a paper prototype. However, in larger companies, it makes sense to have more people participate and observe the tests. In my experience, programmers and people from the management team can profit tremendously from seeing how users interact with their products.

We call the people who observe tests *observers.*

For a usability test of a paper prototype, observers can either sit in the same room as the other three people, observe from a different room, or watch the video after the fact. If they sit in the same room, they need to know that they should not interfere or influence the tester in any way. Generally, I think it is a bad idea to have observers in the same room as the tester, but since tests of paper prototypes already require a computer and a facilitator, adding two or three observers might not make much of a difference.

Tell your observers to take notes during the test. Afterward, you should spend an hour or so going through the issues found during the test. The more people who observe the test, the more issues they will find. You can use affinity diagrams to prioritize issues coming from several observers. This is explained in Carolyn Snider's excellent book *Paper Prototyping: The Fast and Easy Way to Define and Refine User Interfaces* (Sni03).

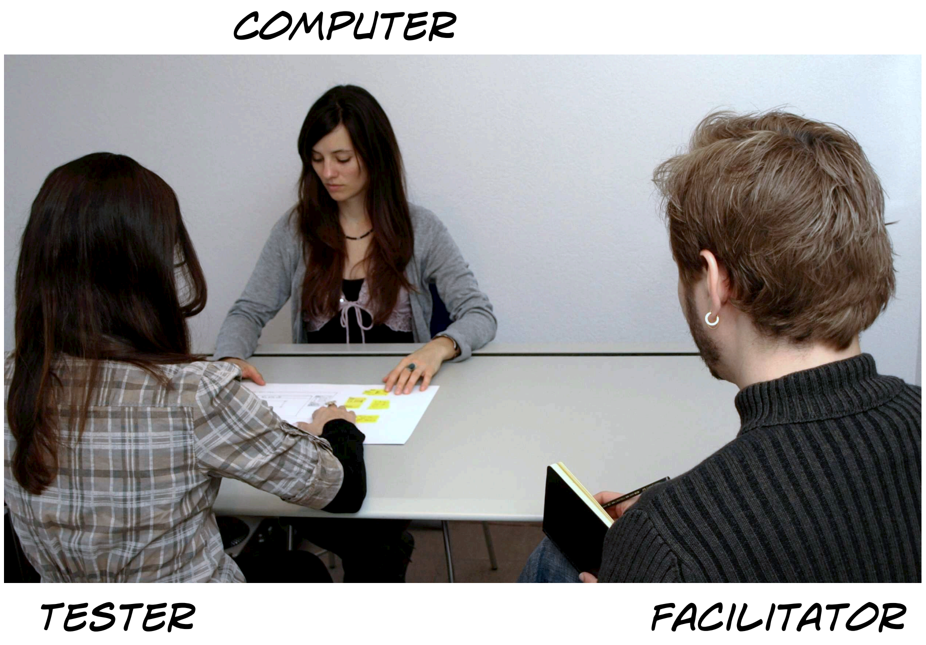


Figure 11.1: The roles in paper prototyping

Hi, I'm Michael. I work here as a software designer. This is my friend Sandra. She also works in the design department, and she will help me with this test.

First of all, I want to thank you again for taking the time to help us with this.

Today, we are testing a new design for our product to see whether it works the way we intended. I want to make it very clear that we are testing the design, not you. This new design has never been used outside of our design team, so we are hoping to find problems in the design by observing people interact with it. So, don't worry if you get stuck or if something doesn't work as you expect it to; this is exactly the kind of feedback we are looking for!

As you can see, we haven't implemented the design yet; it's still on paper. We want to iron out any problems before we start writing code. So today, my friend Sandra will play the part of the computer—she'll do all the things the computer

> would do. Since she's playing the computer, she won't say much today, but she will change what you see according to your input, like a real computer. While you interact with our design, please feel free to tell me whatever is on your mind and ask questions. Since we are trying to see how people interact with this product when we are not around, I may not always be able to answer your questions immediately, but they will still help us understand what is going through your mind.
>
> With your permission, I will record this session. This is just so we can go back and figure out how to improve our design. We will never publish this recording in any way.
>
> Do you have any questions about how this will work?

Then, have the tester sign the consent form.

It's a good idea to make some notes or keep a checklist about what to say beforehand, just to make sure that the facilitator mentions every relevant point. There's a lot of ground to cover, and it's easy to forget something.

Next, the facilitator should quickly introduce the tester to the paper prototype. Explain what she is looking at: "This is the home page of our new website." Next, explain how to interact with it:

> You can use this just like you would use a regular computer. To click something, point with your finger. You can also drag things as you would with a mouse. To type something, use a pencil to write directly onto the prototype—don't worry, we have extra copies. To delete something, use the eraser. If something you do changes the screen, our human computer will take care of this and replace the screen or add pop-ups or menus to this screen. Again, feel free to talk out loud. The computer will only react to clicking and typing, though.

You can use an example sketch to demonstrate things such as pointing, dragging, writing, and erasing while you explain this.

The facilitator should point out that the tester should talk only to him or her, not to the computer. If the tester starts talking to the computer during the test, make sure that the facilitator is always the one who responds. And again, take care to point out that it's the design that's being tested, not the tester.

Running the Test

Finally, introduce the first task:

> OK, now that we have that out of the way, let's start testing the design. I have written a task on this piece of paper. Take your time to read it. Once you're ready, you can start interacting with the design.

There are a few things the facilitator should keep in mind during the test. First of all, it's important to avoid influencing the tester. This is especially relevant when the tester gets stuck and starts asking questions. Once it becomes obvious that the tester won't be able to solve the problem by herself, it's OK to give a hint (or move on to the next task), but generally, the facilitator must take care not to lead the tester.

At most, the facilitator can ask questions but, again, should make sure not to accidentally influence the tester by offering hints on how to use the design. Avoid terminology seen in the design. The facilitator's questions should be as generic as possible, along the lines of "What are you thinking right now?" or, if the tester is stuck, "Tell me what you see on that screen." For more common mistakes made during usability tests, see Chapter 31, *How Not to Test: Common Mistakes*, on page 270.

Generally, as long as the tester seems occupied by the interface, it's best to remain silent. After all, people using your product at home don't have one of your employees looking over their shoulder, constantly asking them how they feel about your product.

The facilitator should avoid doing anything that may make the tester feel uncomfortable. Having people observe your errors can already be stressful; don't add to the stress. If the facilitator notices that the tester is getting frustrated, it's OK to intervene and offer a bit of help, encouragement, or even a quick break. Indicators that intervention is necessary are the tester asking for help, the tester starting to blame herself for problems, or the tester getting stuck. Make sure to avoid sounding condescending when you offer encouragement or a break.

The computer's job is to update the prototype based on the tester's input. If everything goes as planned, with paper prototype testing this mainly consists of replacing the current screen with a different screen or adding and removing Post-it notes of interface elements as needed. In some cases (for example, when the tester uses the search function), the computer can also erase things from screens or write text onto screens. For data-heavy applications, especially if they include a prominent search feature, it may make sense to prepare some screens that you've already filled in with data for the path you expect the tester to take.

Sometimes, the tester takes a route nobody expected. In those cases, the computer can create a quick mock-up of the screen, or, if the tester goes too far off track, the facilitator can intervene and either stop the test or bring the tester back on track.

After the tester has finished the first task, both the facilitator and the computer can take a minute to ask any questions they might have and didn't want to (or could not) ask during the test—for example, "You hesitated before clicking the Buy button; do you remember what went through your head at that point?" Personally, I prefer it if the facilitator doesn't ask too many questions during the test and instead waits until the task is finished. This makes it less likely that the facilitator influences the tester's behavior. The disadvantage is that the testers often don't clearly remember exactly what they did during the test or why they did something.

Once this is done, give the tester the next task, and have her continue with the test.

When the last task is over or you're about to run out of time, you can finish up by asking the tester for some opinions, if you want. Opinions are generally not what we're looking for, but people sometimes come up with interesting thoughts after a test. For example, you could ask what the tester *didn't* like about the design, whether she would use your product, and what she would use it for.

It's a good idea to ask what the tester thought of the whole experience. As a sort of meta-test, the things people point out after such a test can often be useful input for improving future tests.

Finally, thank the tester again for her time, and make sure she understands that the results of the test were helpful to you.

Analyzing the Results

Avoid any kind of statistical or formal evaluation. Usability tests provide qualitative data, not quantitative data. It would be misguided to do formal analyses of the results of such a test. It would also be unnecessary. In fact, even from merely looking at a video of a usability test, it is usually quite obvious where the big problems are.

Instead of investing a lot of time into analyzing the results, identify the problems, prioritize them, let each team member pick the ones they want to solve, let them come up with possible solutions, make the changes to the design, and test again as soon as possible (with another set of testers).

Knowing When to Stop Testing

So, how do you know when your design is good enough and you can stop doing usability tests?

Basically, you don't stop testing. Testing is a constant part of the development process. During this process, as you iron out the problems with your paper prototypes and move toward writing real code, you will also gradually move from testing paper prototypes toward testing running code. However, paper prototyping will remain a valuable tool every time you work on a major design change. It's always easier to test such changes on paper first and implement them in code only once you've settled on a workable, tested design.

Takeaway Points

- To know how to improve sketches and mock-ups, test them with real people.
- Tests don't have to be complex. Show sketches to people, and ask them simple questions.
- For more extensive tests, define tasks that touch critical areas of your product. Prepare at least five or six tasks. Each should take between two and ten minutes. Tasks should not be prescriptive.
- Print out the tasks on individual pieces of paper.
- Recruit three to five testers, and schedule them about two hours apart.
- You can run the whole test on your own, but it's best if you focus on simulating the computer and have another person act as the facilitator.
- Prepare all the screens people are likely to progress through while doing your tasks. Screens should not be too crude. People unfamiliar with your product should be capable of reading them.

- Add the surrounding user interface elements (such as browser windows around sketches of a website) where necessary. Create the pop-up elements you might need.
- Do a test run of each task to make sure you've prepared all of the screens and pop-up elements you're likely to need.
- When the test starts, explain to the tester what is going on. Make sure to emphasize that you are testing the design, not the tester.
- Make a checklist of everything you want to say, and check off the points as you make them during the introduction. Don't forget to have the tester sign the consent form.
- Explain how the tester should interact with the "computer." Allow him or her to draw on top of your prototype.
- During the test, avoid influencing the tester. Don't make the tester feel uncomfortable. Intervene when you feel that the tester is getting frustrated.
- After the test, do a short debriefing, and thank the tester for her help.

Further Reading

Carolyn Snider has written the definitive book on paper prototyping. It's called *Paper Prototyping: The Fast and Easy Way to Define and Refine User Interfaces* [Sni03]. If you're serious about paper prototyping, you need to read it.

Userfocus has a neat article about paper prototyping with links to more resources.[3]

3. At http://www.userfocus.co.uk/articles/paperprototyping.html.