CHAPTER

11

On not throwing the baby out with the dishes

INTERPRETING TEST RESULTS

Sure, we've made mistakes. But let's not throw the baby out with the dishes.

—LYNDON BAINES JOHNSON

OK, you've done your testing, and you've got everyone's notes. How do you decide what to change?

Review the results right away

After each round of tests, you should make time as soon as possible for the development team to review everyone's observations and decide what to do next.

At this meeting, you should distribute copies of everyone's notes and copies of whatever screens or sketches were tested. You're doing two things at this meeting:

- **Triage**—reviewing the problems people saw and deciding which ones need to be fixed.
- **Problem solving**—figuring out how to fix them.

It might seem that this would be a difficult process. After all, these are the same team members who've been arguing about the right way to do things all along. So what's going to make this session any different?

Just this:

> The important things that you learn from usability testing usually *just make sense*. They tend to be obvious to anyone who watches the sessions.

Also, the experience of seeing your handiwork through someone else's eyes will often suggest entirely new solutions for problems, or let you see an old idea in a new light.

And remember, this is a cyclic process, so the team doesn't have to agree on the perfect solution. You just need to figure out what to try next.

Can this marriage be saved?

Naturally, the biggest question on everyone's mind, especially during the first few tests, is always "Are we basically on the right track?"

At these debriefing meetings, you're always trying to figure out whether the part of the site that you're testing can be made to work by tweaking it, or if you have to scrap it and take a whole different approach. For instance, if some users don't get the whole idea of the site, does it mean that the basic concept is flawed, or do you just have to change some of the wording?

Here's my advice:

- **Always consider tweaking first.** It's rare that a site is completely off-base, but there's a tendency to be spooked by any bad user reactions, particularly after the first round of testing. Before scrapping anything, always stop and think, "What's the *least* we could do that might fix the problems we're seeing?" If it seems like a particular tweak has a reasonable chance of working, mock it up and test it as soon as possible.
- **Focus on specifics.** Try to avoid sweeping statements and focus on the precise points where people seemed to go astray. "They didn't seem to notice the navigation when they got to the second-level pages" is a much more productive place to begin problem solving than "The navigation didn't work." One suggests that you should think about ways you could make the navigation slightly more noticeable; the other suggests you should start considering a whole new navigation scheme.
- **Tweak, but verify.** Remember, this is a cyclic process. Since you'll be doing another test soon, you can afford to try some tweaks before scrapping anything.

On the other hand:

- **If the problem is deep, bite the bullet.** I sometimes walk into situations where the site has a serious fundamental problem that nobody is talking about. The problem has usually been obvious to everyone involved for a long time, but nobody wants to be the one to mention it because (a) it seems like there's no solution, and (b) it seems like it's too late to do anything but tough it out and hope for the best. Sometimes I get called into these situations precisely because they need an outsider to announce what's already obvious to everyone.

  For instance, suppose that when you do your first test it becomes clear that the way you've chosen to organize the site isn't the way your users would organize it, and as a result they're having a very hard time figuring out where you've put things. Or perhaps you've been building your company's site assuming that your customers will want to do *x* and *y*, but when you start testing you learn that they're happy with the way they already do *x* and *y* and they're not really interested in doing them online.

  If you've got a deep-seated problem, a post-test debriefing meeting is a good place to finally face it. Once it's out on the table, it usually turns out to be more manageable than it seems when you're not talking about it.

- **Remember: It's *almost* never too late to challenge basic assumptions.** Rethinking the basics doesn't necessarily mean changing everything, and it often turns out that the solution to the problem is simpler than you've feared.

Typical problems

Here are the types of problems you're going to see most often when you test:

- **Users are unclear on the concept.** They just don't get it. They look at the site or a page and they either don't know what to make of it, or they think they do but they're wrong.
- **The words they're looking for aren't there.** This usually means that either (a) the categories you've used to organize your content aren't the ones they would use, or (b) the categories are what they expect, but you're just not using the names they expect.
- **There's too much going on.** Sometimes what they're looking for is right there on the page, but they're just not seeing it. In this case, you need to either (a) reduce the overall noise on the page, or (b) turn up the volume on the things they need to see so they "pop" out of the visual hierarchy more.

Some triage guidelines

Here's the best advice I can give you about deciding what to fix—and what not to.

- **Ignore "kayak" problems.** In any test, you're likely to see several cases where users will go astray momentarily but manage to get back on track almost immediately without any help. It's kind of like rolling over in a kayak; as long as the kayak rights itself quickly enough, it's all part of the so-called fun. In basketball terms, no harm, no foul.

  As long as (a) everyone who has the problem notices that they're no longer headed in the right direction quickly, and (b) they manage to recover without help, and (c) it doesn't seem to faze them, you can ignore the problem. In general, if the user's second guess about where to find things is always right, that's good enough.

  Of course, if there's an easy and obvious fix that won't break anything else, then by all means fix it. But kayak problems usually don't come as a surprise to the development team. They're usually there because of some ambiguity for which there is no simple resolution. For example, there are usually at least one or two oddball items that don't fit perfectly into any of the top-level categories of a site. So half the users may look for movie listings in Lifestyles first, and the other half will look for them in Arts first. Whatever you do, half of them are going to be wrong on their first guess, but everyone will get it on their second guess, which is fine.[1]

- **Resist the impulse to add things.** When it's obvious in testing that users aren't getting something, most people's first reaction is to add something, like an explanation or some instructions.

  Very often, the right solution is to take something (or things) away that are obscuring the meaning, rather than adding yet another distraction.

[1] *You may be thinking "Well, why not just put it in both categories?" In general, I think it's best for things to "live" in only one place in a hierarchy, with a prominent "see also" crosslink in any other places where people are likely to look for them.*

- **Take "new feature" requests with a grain of salt.** People will often say, "I'd like it better if it could do *x*." It always pays to be suspicious of these requests for new features. If you probe deeper, it often turns out that they already have a perfectly fine source for *x* and wouldn't be likely to switch; they're just telling you what they like.
- **Grab the low-hanging fruit.** The main thing you're looking for in each round of testing is the big, cheap wins. These fall into two categories:
    - **Head slappers.** These are the surprises that show up during testing where the problem and the solution were obvious to everyone the moment they saw the first user try to muddle through. These are like found money, and you should fix them right away.
    - **Cheap hits.** Also try to implement any changes that (a) require almost no effort, or (b) require a *little* effort but are highly visible.

And finally, there's one last piece of advice about "making changes" that deserves its own section:

Don't throw the baby out with the dishes

Like any good design, successful Web pages are usually a delicate balance, and it's important to keep in mind that even a minor change can have a major impact. Sometimes the real challenge isn't fixing the problems you find—it's fixing them *without* breaking the parts that already work.

Whenever you're making a change, think carefully about what else is going to be affected. In particular, when you're making something more prominent than it was, consider what else might end up being de-emphasized as a result.

That's all, folks

As Bob and Ray used to say, "Hang by your thumbs, and write if you get work."

I hope you'll check in at the companion Web site www.stevekrug.com from time to time. I'm going to be working on some pseudo-research (as I like to think of it) using eye tracking equipment to learn more about what we actually see when we look at Web pages, and I'll try to post some observations now and then.

But above all, be of good cheer. As I said at the beginning, building a great Web site is an enormous challenge, and anyone who gets it even half right has my admiration.

And please don't take anything I've said as being against breaking "the rules"—or at least bending them. I know there are even sites where you want the interface to make people think, to puzzle or challenge them. Just be sure you know which rules you're bending, and that you at least *think* you have a good reason for bending them.