Identifying problems

slide 1: Introduction

Hi guys! Welcome to [Fundamentals of User Experience Design ], a Tuts+ premium course. I’m [Sarah Kahn] and today we’re going to talk about prioritizing usability problems.

slide 2

Here’s what we’re going to cover:

* + if 15 things aren’t working well, how do we decide what to fix first?
  + assigning levels to pain points
  + fix it now vs fix it later
  + band aids vs overhauls
  + incremental fixes
  + Assignment

slide 3

[scared person]

So we’ve spent a lot of time talking about how to find out what your problems are. What you might find out pretty quickly as you work through this process is that a) users are a pretty unforgiving group and this is humbling and b) you’ve got your work cut out for you. This is ok! Don’t be discouraged.

The first mistake i always made as a rookie was leaping to extremes. It’s so easy to think, oh we have all these problems, we need to scrap this completely and start from the beginning!

And that would be lovely, if we were made of time and money. But the reality many of are facing is that we have limited time and resources, and we have users who need to use our sites right now. So what do we do? How do we decide where to begin?

slide 4

[4 section chart]

This chart, is, in my opinion, a lifesaver, and one of the great prioritization tools that I we have been given by the gods of ux.

The axis range on one side from high impact to low impact- meaning, users are feeling a LOT of pain for high impact problems, or they are features or areas that have a lot of visibility. Low impact is something that isn’t used as much, or is mildly painful for not too terrible. Or, another way to put it is, a high impact problem is making users feel angry, frustrated, helpless, or just generally ragey when they encounter it. Low impact problems make them feel meh, or it’s only making a very small percentage of users feel ragey.

slide 5

[ragey]

slide 6

[quad chart again]

The other axis is ranging from hard to fix to easy to fix. Hard to fix may be technically difficult, or it might mean that something big needs to be cleaned up, or that someone high up doesn’t believe that it needs to happen, whatever the case may be, it’s something that can’t happen quickly and easily.

Easy to fix is pretty self-explanatory- easy to implement, doesn’t require you to move heaven and earth to make it happen.

What I like to do is to, once i’ve analyzed my test results or my metrics, and i have a fairly succinct list of problem areas, is to organize them on this chart.

slide 5

[example problems]

Let’s say I’ve conducted a study, and I now know that I have the following problems:

-users can’t find the login link to my application from the homepage of my website

-users are having trouble completing a major task in my application because it involves too many steps

-the color of some links is hard for users with a certain type of colorblindness to read

-i make users select their currency type my app instead of auto-converting things for them

now we’re going to step through the different quadrants of the prioritization grid with this list of problems, and hopefully that will help illustrate what we’re doing with this thing.

slide 6

[filled in quadrants]

so here i’ve placed these problems according to what I think is hard and easy to implement as well as how much impact these problems are having on users. According to this analysis, the first one we’d always go after would be the

slide 7

[Quick wins]

Since these are Things that are easy to fix and will have a high impact on users are the things that you should go after them first. Adding a login link for my application from my homepage is going to be highly visible, have a fairly high impact, and will take a minimum of development resources. So clearly it’s a quick win.

slide 7

[long term goals]

In this example, i’ve placed the problem that clearly needs some work but that doesn’t necessarily have one clear answer in the long term goals section, or, high impact difficult implementation.

In order to solve this problem, I’m going to have to spend some time brainstorming with my colleagues, talking to users, and probably will propose at least 3 or 4 different solutions before we settle on one to implement.

An important thing to remember is that often there is no clear answer to a problem that turns up. It can be challenging and a little nervewracking, and that’s ok. When you have a nebulous problem, the answer is always to do more research, and continue to break it down into smaller and smaller pieces until you get to the root of the problem.

slide 8

[low impact, easy fixes]

I’ve placed link color in low-impact but easy to fix because my users have reported it as mildly annoying, or not many users have seemed to have noticed it. Problems that are only noticed by a small subset of your users, but that won’t take a lot of resources to fix, go here. Fix them as you have time, but don’t sweat it. Users will appreciate your throughtfullness if you are able to get to these, but if you have bigger problems to take care of, by all means get to those first.

slide 9

[put on the back burner]

We all know one of those, ‘never gonna happen’s’ when we see them. Some things are just too costly in terms of time, resources, or just generally such a pain in the neck that they’d better have an amazing payoff if you’re going to take them on. And some problems just never get more important than the ‘nice to have’ or ‘one or two people mentioned it’. in those cases, it’s ok to put it on the back burner. Do keep a list, you never know when these will end up being able to be fixed as part of something else, but don’t waste time or energy fixing these exclusively, unless you have a lot of time or available resources.

slide 10

[bandaid or shotgun]

so you’ve prioritized your problems, how do you know when you need to really overhaul something, versus doing a quick fix?

Well, the prioritization chart will help here too. Your long term goal sorts of problems are likely symptoms of needing some major overhaul. They wouldn’t be difficult to implement or hard to determine a solution for if there were any easy fixes. Get out the shotgun for these.

Bandaids, however, are fine for your lightweight easy to fix problems. If you can find any easy, low-tech way to make it happen, do it.

slide 11

[iterative fixes]

most problems you will encounter won’t be so cut and dried. if it’s a clear case of back to the drawing board, or quick bandaid, that would just be too easy!

slide 12

[the 99% of problems]

Most problems can be solved with a more moderate approach. I often go at it this way- find your ideal solution, the one that takes tons of dev time and that you probably couldn’t get rolled out for another month under the best of circumstances. Break it down, and figure out which pieces you can implement right now to ease the pain.

slide 13

[ragey]

Rinse and repeat. In this way, you can gradually fix the problem, while easing the burden on your users right now. This approach works well with Agile teams also. Because if your users are ragey right this moment, it’s hurting your business, and it’s hurting their blood pressure.

slide 12

[assignment]

Ok so it’s time for our assignment. If you’ve been following along with the other assignments, you should have at least a very short list in mind, but in case you haven’t, take a look at your site and pick 4 problems. Draw your own prioritization chart, and place your problems according to where you think they should fall. Then, recruit a client or colleague and ask them to look. Explain your logic, and see if they agree!

This is not an exact science, it’s just a series of tools that help you eyeball things, as it were, to get a rough idea of where things stand and how to keep from being overwhelmed by the deluge of user feedback.

Next time on [Fundamentals of User Experience Design ], will be [Lesson 8: Designing a Test ]. This is [Sarah Kahn], and from all of us here at Tuts+, thanks for listening!