Features in isolation

https://web.eecs.utk.edu/~azh/blog/featurestheywanted.html

For this specific case (product discovery interviews) take a look at *Interviewing Users* by Steve Portigal, and *Validating Product Ideas* by Tomer Sharon.

"Don't Make Me Think" by Steve Krug

TELL your users that the new thing now exists! Write a good update. Whether it's the email or change log. Let your customers know. This is called feedback loop.
Responsibility lies on the customer success team. Because by virtue of the title "customer success" it's your job to ensure the customer is successful with your product.

Keep your users in the loop, always. Do not go build in isolation

As the author found out, most of the work in building features is often not in the "raw functionality" of the feature, but rather in making something that performs its functions while *also* not costing the user significant extra mental or physical effort to use.

This is relevant perhaps 10% of the time. The other 90%, the devs have absolutely no idea how to actually perform the task the user of the software is trying to do, don't talk to customers, don't dogfood it, often don't even understand the industry, and the result is a horrible UI which forces people to severely adjust their existing processes to fit the software. Then if course the devs complain about "the stupid users." To this day 99% of address forms expect me to find my state from a drop down list of 50 entries plus DC, Puerto Rico, Guam, etc. instead of simply typing in the two letter abbreviation.

- Keep your users in the loop, always. Do not go build in isolation.
- Don't underestimate engineering challenges that you only have an external view of.
- Voice your concerns to your team regularly and often. They might be to solve them far more quickly than you or they might be able to identify what will turn into a major roadblock.
- Be ready to pivot.
- Users say things for a reason, but there may be more to it than face value.
- If you make assumptions about your users, they will find a way to surprise you.
- Features will go unused if they aren't easy to use, no matter how great they are.
- A user's workflow is everything. (I keep relearning this lesson...)
- Users are far more clever than you think.

Theres often more to what a human says than face value. The key is to always be asking questions, always ask why, multiple times. Ask questions until you feel like you're on the edge of pissing someone off.
Often, a good bit of time up front can get to a few outcomes which save you a massive headache.
1) You understand the feature so well, you're ready to go and know it will be used. 2) The REAL ask was something totally different, and now you know what you need to do (nor not in some cases, sometimes its training, using the product properly/as intended etc).
You're also going to be giving the requester the space to fully explain themselves, and sometimes they can talk themselves round as well. At the

end of the day everyones a winner when we ask more questions and listen.

You know what's worse? When you have no metrics and have NO IDEA whether users are actually using the features you ask for.
I realized this a while back due to the fact we had a feature on customer request, which was in release for some time. It had a bug in it, and nobody ever told us about it - which told us nobody ever used it.
And the only reason I know nobody used it is that we collect no metrics.

As the author eventually understands— users almost always want the features they ask for, but they may not the interpretation of them you've created.
I recently started working in design after being a developer for more than a decade. Back then, planning how things *should work for users* felt frivolous compared to coding. Beyond that, getting working code in the editor just felt so good that I never wanted to put it off. These are the excuses I'd make to avoid really thinking about the users:

- I can clean up the interface later

- I'll figure out what features I can support when I figure out how it's going to work

- I've got a good intuitive sense of what would be most useful and how people want things to work

Unless the tech requirements are extreme, knowing what features are helpful and how users will use them should inform development— not the other way around. Also, overestimating your understanding of how people want something to work almost always leads to suboptimal results.
Unfortunately, this mindset led to clunky, miserable interfaces that many people rejected. Worse, core users would learn to love it because they had no choice, and they'd develop bad UI Stockholm syndrome. Nothing carves a lousy user experience into stone for all future users like reliable core users demanding nothing changes.
Obviously, solo intern projects are an edge case, but this case study perfectly illustrates the necessity of user-focused design is in software projects. All the better if you can find someone who specializes in it.
UX expertise brings:

- deep experience reasoning about the way people use things

- knowledge of many workflows, working styles, and personalities

- understanding of how much deviation users will tolerate

- familiarity with user research techniques and their shortcomings

- ability to distinguish between research signal and noise

- knowing where to dig deeper or seek more data

- understanding what needs to be prototyped vs. mocked up vs. textually described for tests, and ideally the ability to do all three

That perspective built into this project from the beginning would have completely changed its trajectory. Users would be less irritated, and the developer could have used wasted rewrite time making a more valuable end product.

I moved from engineering to product management precisely because I realized that the problem of "what should be built" is often harder than building the thing.
Asking users what features they want is pretty much not fair - because most people don't have the skills to think through and answer that question, nor is it their job to do it. It's like asking a novel reader what they'd like to see in the next chapter. It's a cop-out with guaranteed suboptimal outcome.

Obviously product management is half art half science but in a nutshell, much better than asking an individual what features they want would be asking them - especially on a senior level - what problems they have and what keeps them up at night. Getting an understanding of that and *then* thinking about how those problems/risks can be addressed by your system is a much better starting point.

The reason I say that it's important to partner with your users on the senior level is the difference in perspective on the class of problems they want to tackle.

For example I used to work on a trading platform. If I went to my heaviest users and asked them what their problems were, they would probably say "I do 1000 trades a day, and I have to tweak each one manually - can you make that process faster" and maybe even have an idea of what that could look like. So let's say I did that and shaved a second off each trade handling so my user is happy cuz I saved them 16 minutes a day. Sounds like a job well done.

But if instead (or in addition) I talked to their boss, I might hear a very different story. Eg: "we do 1000 trades here every day, but 900 of them are straight forward. I wish I could automate those so trader can focus on the 100 complicated ones. But instead, he's so busy doing the 1000 trades that a lot of them get fucked up especially the complex ones."

That's a major change of perspective, from optimizing an existing workflow to optimizing the entire operation. The implementation is quite different - one is a UI optimization and the other is establishing some sort of automation capability that can be extended to all my clients over time. One may be doable with the team you have, one may require standing up a new group, etc.

At the end, the end user is happy (they get to do 10% of their previous load but this is the high value 10% they really want to focus on.) The key point is that neither the user nor their boss could tell me exactly what to build, but the high level perspective allowed me to understand their problem in a deep way and figure out a scalable solution for them and other clients.

Anyway that's just an example but yeah, your users can't tell you what to build and yeah you need a good product manager - or someone who can play that role well.

Automating a workflow *is* the best optimization from the users point of view.

Newer Older

Sunday, 29th May 2022
Gun control and abortions

2023-02-21
Product Design is not UX Design

Tags RSS feed