Put down the compiler until you learn why they're not buying

New companies rarely have a problem writing code.

The problem they have is: **We don't have enough sales.**

Some actual quotes (*sound familiar?)*:

> "We have 300 downloads and no sales."
> "People tell me I have a great idea, but none of them bought my software."
> "My sales/download conversion ratio is 1%. It should be 8%."
> "Folks are signing up for an account but they don't come back."

Of course everyone wants "more sales," but I'm specifically talking about the early stage of your company, when your v1.0 is shaky but has enough features that it should be more viable than it is. When your website copy is good enough that people are willing to sign up or download, but the sales aren't coming in like they ought.

"Things went from bad to worse, but we're hopeful now that we're doing badly again."

credit [1]

This problem is solved only one way: **You need feedback from lost sales.**

Empirical data, not your own ideas about why people might not be buying.

You need to talk with the people who were interested enough to find your website, read your marketing copy, download your product, and then *give up without even an email*. That's the low-hanging fruit; those are the people who are *in your grasp*, who should be buying *today*, but aren't.

They've self-identified as your ICP,[2] yet your product didn't fulfill the promises you made.

As Steve Johnson[3] says, "**All the answers are outside the building.**" (Watch his one-hour presentation on the subject at the Business of Software 2008 Video Archive.[4])

Or as Eric Ries says,[5] "**Not listening is the cardinal sin** … Any other mistake can be overcome: shipping bad product, removing key features, erroneously banning community members, even kicking out a whole segment of customers."

But I find that entrepreneurs—especially technical ones—fight me on this tooth and nail. And I'm not surprised because, as usual, **I too used to**

hold the I-already-know-why, I-know-my-customers-better-than-they-do attitude.

So once and for all, I'd like to dispense with the usual arguments against getting feedback:

Existing customers are telling us to do X, so we should do X.
Customer requests are important and you *must* follow their lead, especially in the beginning. But what about the 98% of trial users who *didn't buy*? It is *they* who hold the keys to more sales! Existing customers bought *in spite of* barriers to sale, so they're no help in identifying the barriers. Listen to them to increase your product's value, but listening to them to increase sales is classic survivor bias.[6]

What we need is New Feature X, then people will buy.
This is almost never true. The world is filled with successful v1.0 products that lacked obvious features; in fact I challenge you to find an exception. Ben Yoskovitz wrote a great post about this fallacy[7] (with 27 concurring comments). Even Nintendo says[8] "the most important feature is the one no one asks for."

We need to clean up the software before we can get real feedback.
At Smart Bear, the first incarnation of our code review product was so hard to decipher, I can't understand how we got customers. They used it *in spite* of the problems, not *because* of them. If you're solving a genuine pain, people will try the software, complain about it, ask for features, and generally be engaged; if that's not happening, you're not solving the right problem or not making that obvious, and *that* is critical to getting revenue.

Have you ever worked on a software project for many years and then lived through a face-lift? After you're used to the new look, you're just embarrassed when you see the old version. It's the natural order of things. Polish isn't important if you don't have enough revenue.

I'm a user myself, so I know what's missing.
That's great, but all that means is that you have 100 ideas for new features, but "more features" is almost certainly not the problem. It means is you have a "vision" which is almost certainly not how your company is going to unfold.[9]

Often the real impediment to sales is as mundane as "New users are presented with a blank screen, so they don't know what to do next, so they abandon the trial," or "The installer doesn't work properly under Vista, so people give up." The fact that you're a user yourself is the *worst* position for you to be in because you can't be objective about the new user experience, and you can't put yourself in the shoes of a user possessing below-average intelligence. Which half of them possess.

There, I said it. Most of your users are dumb; almost all are dumber than you are. You are not your typical user.

Apple just knows what's cool. So do we.
This is a common misconception, easy to believe because Apple does keep product development close to the vest. However, it's completely untrue. Steve Jobs specifically talks about[10] getting feedback from customers.

We can't afford to delay the v#.# release.
If you have no real evidence that revenue will suddenly improve with the next release, why do you think it's important to release it? Just because it has "more stuff?" The only reason to be excited is because it's different, and since the status quo isn't working, you've *got* to try something different. But is that "stuff" why people are downloading but then abandoning? Until you can answer that question with empirical data, there's no reason to believe the new stuff will be more compelling than the last stuff.

Getting revenue is a marketing/sales function; I need to be heads-down in the code.
In a startup, it's *everyone's* job to get revenue. Sure, the usual day-to-day activities should be divvied up between founders; not everyone needs to write letters to bloggers and be glued to Twitter live-search. But if you don't know why people aren't buying, that's the #1 bug and the #1 feature you need to be working on. There's lots of ways (see below) to change the product or website *in under a day* that will begin fixing the problem. Saying "it's marketing's job" really means "I'm not going to help get revenue." Unacceptable.

Hopefully by now you're convinced to get more feedback from lost sales, but how do you go about doing it?

Here I've posted eleven specific ways to get more feedback,[11] almost all of which take less than a day to implement.

And here's I've described my system for creating questions and conducting customer interviews[12] that I've used to build two unicorns.

So you have no excuse.

Current version of this article:
https://longform.asmartbear.com/put-down-the-compiler/

More articles & socials:
https://asmartbear.com

REFERENCES

1. https://andertoons.com/chart/cartoon/3709/things-went-from-bad-to-worse-but-were-hopeful-now-that-were-doing-badly-again
2. https://longform.asmartbear.com/icp-ideal-customer-persona/
3. https://www.pragmaticmarketing.com/
4. https://businessofsoftware.org/2008/01/steve-johnson-at-business-of-software-2008-why-software-is-not-a-business/
5. https://www.startuplessonslearned.com/2009/09/cardinal-sin-of-community-management.html
6. https://longform.asmartbear.com/survivor-bias/
7. https://www.instigatorblog.com/false-promise-one-more-feature/2009/08/25/
8. https://web.archive.org/web/20090228020253/http://www.time.com/time/magazine/article/0,9171,1191861-1,00.html
9. https://longform.asmartbear.com/predict-the-future/
10. https://venturehacks.com/articles/jobs-customer-development
11. https://longform.asmartbear.com/more-sales-customer-feedback/
12. https://longform.asmartbear.com/customer-development/