The Code is your Enemy

*Builder Problems · Coding leads to failure ·
Proof from your peers*

"Ugh! I could feel them monetizing
me with their eyes!"

ANDERSON

credit 1

YOU'RE A BUILDER. THAT'S GREAT. IT'S ALSO THE PROBLEM.

You're a builder, a creator—whether a back-end programmer, a Linux hacker, a Javascript ninja, a UX magician, a designer. You make stuff.

That's great of course, because in a new startup everyone needs to be either making stuff or selling stuff—there's no room for managers and executives and strategists. But **this also produces a natural weakness**, and when I look at what made me a successful entrepreneur—not just a great coder—one reason is that I discovered and overcame this weakness.

The weakness is the same as your strength as they often are:[2] Your love of creation. You love to write clean, tested, scalable, extensible, beautiful code. You love converting "JTBDs" into 960-wide artwork. You love developing an entire app in the browser against a scalable back-end.

And because you love it, you do it. You wake up in the morning thinking about what you can make, not how you can sell. You open Visual Studio before you consult your to-do list because there's something you just *need* to tweak. You launch xterm before your CRM (if you even have one, which you don't) because the server was running just a tad slower than you'd expect and you want to paw through log files.

The trouble is, this is almost certainly not the activity that would most benefit your startup.

As much as you're a minor deity when it comes to vim or Balsamiq, so are almost all the technical founders of all high-tech startups. We can all write code—at least well enough to get a product launched and through a few iterations. We can all make a functional website—at least well enough to produce orders.

MOST STARTUPS FAIL, DESPITE EXCELLENT CODING AND/OR DESIGN SKILLS.

Read that headline again; this should bother you[3] more than it does. Yet, because coding is your love, your passion, you just keep coding your way to failure.

Most startups fail because not enough people show up on the home page, or people show up but they don't try the product, or they don't pay for the product, or it's too expensive to get them to show up, or they don't tell their friends to come along, or because it's not solving a pain that people have, or it's not solving a pain that people know they have, or it's too hard to explain the pain, or a bunch of other things **that are not whether the code works or whether it looks good.** It's a Drake Equation.[4]

Customers don't open their wallets based on your unit test coverage or whether you used Bodoni instead of Times New Roman on the home page. In fact I've made millions of dollars on companies with hideously ugly websites[5] and buggy code.[6] Those things are actually not the most important things.[7]

So if these things—the raw materials and skills used in web-based start-ups—are necessary but insufficient, what are those things outside your comfort zone which nevertheless are the things that are actually valuable to your company?

Here are two:

1. Have you talked to 50 potential customers? By that I mean *fifty*, not a dozen. I once vetted an idea and after the first 10 interviews I thought I was really on to something. Suddenly things changed and future interviews weren't so clear. Turns out there was accidental bias in the people I selected, obvious only in hindsight. Another time,

everyone said it was a great idea, but it wasn't,[8] which was only clear after dozens of interviews. Do you find it hard to locate that many people? Well it will still be hard to locate them after you've built a product, but then it's unlikely you built the right product, so now you've wasted months of time. So solve that hard problem now. Don't forget to vet the price at the same time[9] and make sure they're actually going to buy it,[10] otherwise it doesn't count. Here's how to interview.[11]

2. Are people coming to your website every day? If not, solving that is *much* harder and *much* more outside your control than building software. Consider: Would you rather get hired as the CTO of a company with 1,000 daily new, unique, qualified visitors with a buggy product, or the CTO of a company with a clean, stable product and 10 uniques visits to the home page? You *know* you can fix the first case; no one knows if the second will ever be a real business. But if you stay nose-down in the code instead of working on getting attention, you're building the second case. Are you sure the market even exists?[12]

This article gives much more detail[13] about what you should be doing, and even how to find those potential customers to interview.

Here's a coder-centric way of thinking about all this more generically: When you tackle a large development project, do you tackle the high-risk, inadequately-understood modules first, or leave those to the end? First, of course, because you understand the rest and you need to solve the unknown problems while you still have time to pivot and re-plan.

This is the same thing, it's just that in a business it's the attention, marketing, positioning, selling, defining part that's high-risk and inadequately-understood, and *all of the coding and design* is low-risk and well-understood in comparison.

So force yourself out of your comfort zone. You'll also do coding and design, and that's fine of course. But force yourself to *mostly* do those *other* things that create a valuable business.

Put down the compiler and talk to customers.[14]

PROOF FROM YOUR PEERS

The above was written in 2013; in 2024 engineering founders still won't listen, as was on display with a tweet from the great Rob Walling.[15] Here's that tweet, and many successful bootstrapped founders concurring, and lamenting that still no one listens.

The Twitter thread has far more responses, all just like these.

Can you learn this lesson today, without having to "find out for yourself?"

Just this once?

I'm so naive

I'm from a software engineering background and I'm very comfortable in that space. I thought building my own SaaS would be super easy. I write code every day, it's just another project: build it, launch it, boom.

Can't believe how stupid I was.

Turns out it isn't enough to implement an idea and put it online after all. Customers aren't going to magically appear out of thin air. Who knew.

SaaS sales and marketing is a whole different thing to implementing code. Like day and night, really.

Rob Walling ✅
@robwalling

Oh developers...they're gonna develop.

I honestly feel for this person.

Still surprises me that 14 years of @startupspod and 13 years of @MicroConf have barely made a dent in stories like these.

Still much work to be done 😅

credit[16]
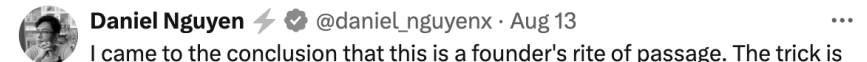
Daniel Nguyen ⚡✅ @daniel_nguyenx · Aug 13   ···
I came to the conclusion that this is a founder's rite of passage. The trick is to make this mistake very very quick I think (launch fast, fail then tweak the mindset to market first)
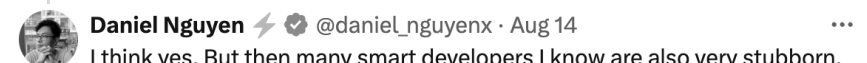
💬 1   ♻   ♡ 15   ᐧ�823 556   🔖 ↥
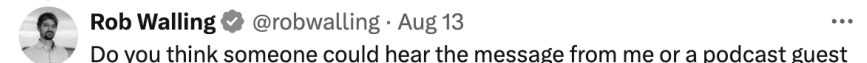
Rob Walling ✅ @robwalling · Aug 13   ···
Do you think someone could hear the message from me or a podcast guest and learn from that?

I know some folks have, so this isn't theoretical. But I'm wondering if most people don't hear this message at all, or if they hear it and don't believe it.

💬 6   ♻   ♡ 9   ᐧᐧ 516   🔖 ↥

Daniel Nguyen ⚡✅ @daniel_nguyenx · Aug 14   ···
I think yes. But then many smart developers I know are also very stubborn.

I think they do hear about this, but they don't believe it. Or don't want to step out of their comfort zone (building).

💬   ♻   ♡ 2   ᐧᐧ 147   🔖 ↥

credit[17]

Miguel Sarenas ✔ @MiguelSarenas · Aug 14
Yeah I think everyone goes through this phase and figures out the harsh reality haha

💬   🔁   ♡ 2   📊 30   🔖 ⬆️

Klaas ✔ @forgebitz · Aug 13
building is the fun part, it's also only around 25% of your job

💬   🔁   ♡ 4   📊 224   🔖 ⬆️

Eddie Forson ✔ @Ed_Forson · Aug 14
Most developers turned founders go through this phase.

I read about it before, told myself I would not make this mistake, and yet still made it 🤦.

Some lessons need to be experienced, not only read about.

💬   🔁   ♡ 3   📊 45   🔖 ⬆️

Adam Wright ✔ @adamdenverco · Aug 13
It's more of a "we all have to learn this lesson the hard way" problem.

💬   🔁   ♡ 2   📊 89   🔖 ⬆️

Aaron Edwards · 🛠 AI ✔ @UglyRobotDev · Aug 13
Been there done that with my first product @infiniteuploads !

💬   🔁   ♡   📊 31   🔖 ⬆️

Thanks to Simón Muñoz[19] for translating into Spanish.[20]

Stefan Wirth ✔
@NafetsWirth

Honestly?

I read "start small, stay small" and "e-myth" pretty early on and I thought I knew.

But I didn't.

It was always in the back of my mind but avoiding discomfort can go a long way.

Some lessons need to be paid in blood to truly understood.

12:54 PM · Aug 13, 2024 · **279** Views

credit[18]

Current version of this article:

`https://longform.asmartbear.com/code-is-your-enemy/`

More articles & socials:

`https://asmartbear.com`

© 2013 A Smart Bear Press

REFERENCES

1. https://andertoons.com/women/cartoon/6034/ugh-i-could-feel-them-monetizing-me-with-their-eyes
2. https://longform.asmartbear.com/pivot-points/
3. https://longform.asmartbear.com/avoid-blundering/
4. https://longform.asmartbear.com/startup-drake-equation/
5. https://longform.asmartbear.com/design/
6. https://longform.asmartbear.com/software-quality-mortgage/
7. https://longform.asmartbear.com/the-important-thing/

8. https://longform.asmartbear.com/good-startup-ideas/
9. https://longform.asmartbear.com/pricing-determines-your-business-model/
10. https://longform.asmartbear.com/customer-validation/
11. https://longform.asmartbear.com/customer-development/
12. https://longform.asmartbear.com/problem/
13. https://longform.asmartbear.com/product-market-fit-formula/
14. https://longform.asmartbear.com/put-down-the-compiler/
15. https://robwalling.com/
16. https://x.com/robwalling/status/1823345489560944868
17. https://x.com/daniel_nguyenx/status/1823349536414445893
18. https://x.com/NafetsWirth/status/1823417820170051624
19. https://x.com/simonvlc
20. https://www.estrategiadeproducto.com/p/el-codigo-es-tu-enemigo