The lone developer problem

by Evan Hahn, posted Feb 27, 2023

In short: in my experience, if a single programmer builds something, it's often hard for others to maintain later. There are several possible reasons why. Even great programmers fall into this trap!

This post is anecdotal from my own experience, so it might not be right or apply to you. But here goes:

A lot of software is built by one person. It might be an entire product built by a lone developer or a significant piece of a system.

When this happens, I've observed that **code written by a single developer is usually hard for others to work with**. This code must've made sense to the author, who I think is very smart, but it doesn't make any sense to me!

I'm not an amazing programmer but I see this happen to myself all the time. I look at code I wrote a year ago and often find it confusing. *What was I thinking when I wrote this?*

I've started calling this the "lone developer problem".

Why does this happen?

I don’t know, but I suspect this happens for a few reasons:

- When a single programmer understands all the pieces, they can tie them together in ways that make sense to them. The author sees a well-integrated system but a future reader sees spaghetti code.
- One-person projects often have different requirements. Small projects might be scrambling for their first ten users; large teams might be trying to capture their next million. Code quality is often less important during the “scramble” phase.
- Code review can be a useful way to catch bugs and design flaws, but the lone programmer can’t easily do this.
- Relatedly, it’s it’s easier to have bad ideas if you don’t have to explain them. And it’s impossible to get someone else’s good ideas when there’s no one else around!

There are probably other reasons, and these are all just guesses.

Solutions?

The obvious mitigation strategy: bring in another developer. Have them review your code or pair with you.

If you can’t do that, open-sourcing your code might pressure you to avoid some of these traps.

I’ll add that you may not really *want* to solve this problem. You don’t always need to write good code! For example, you probably shouldn’t prioritize code quality during a hackathon. (Of course, you probably want good code sooner than you think.)

Again, this post is entirely from my own perspective, and is not based on hard evidence. Your experience might be totally different! But maybe you’ve also run into the lone developer problem.