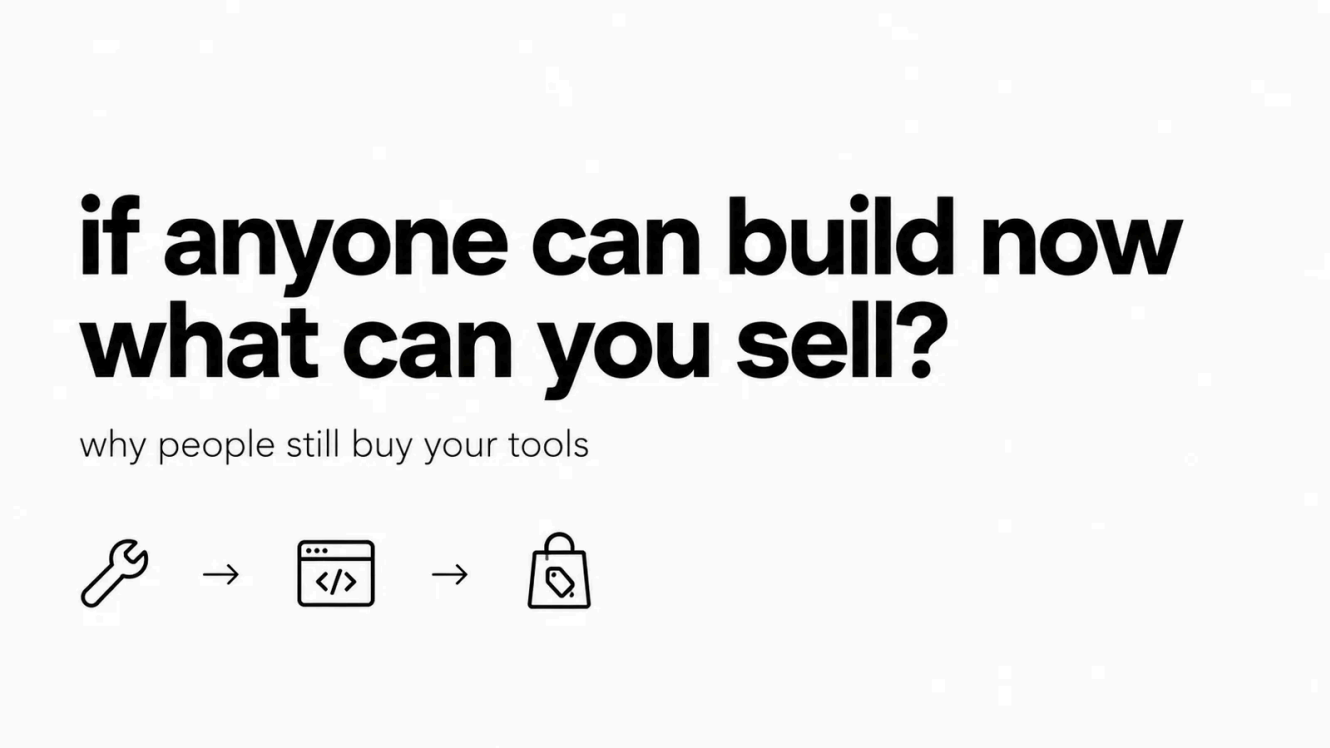
If anyone can build now, what can you actually sell?

Why people still buy your tools.



01 People still pay to NOT build things

02 Sell accumulated knowledge, not only software

03 Distribution becomes even more important

04 Sell the thing around the tool

05 Products that get better because other people use them

06 Sometimes the answer is literally: don’t sell software

07 And obviously, some tools are just screwed

Seven angles on what customers still pay for when building gets easier.

The uncomfortable question

Why would they pay every month?

This question is getting kinda uncomfortable. Say you build a small SaaS today, an internal dashboard, AI workflow, some niche analytics thing. A few years ago you could say “well, customer cannot build this themselves” and that was a pretty reasonable argument. Now... are we sure?

Somebody reasonably tech savvy can open ChatGPT, Lovable, Replit, Cursor, n8n etc and get something that does 70% of the job over a weekend. It won't be exactly your product, probably won't look as good either, but for their own use that might be enough. So why are they paying you every month? I think quite a few SaaS ideas look weaker once you actually sit with that question, especially compared to even 2 years ago.

If the whole thing is basically an interface around an API, I'd be a little nervous. You can describe it in a paragraph and get a rough version by dinner. Doesn't automatically mean there's no business there, but “I know how to make this and you don't” is doing less and less work for you.

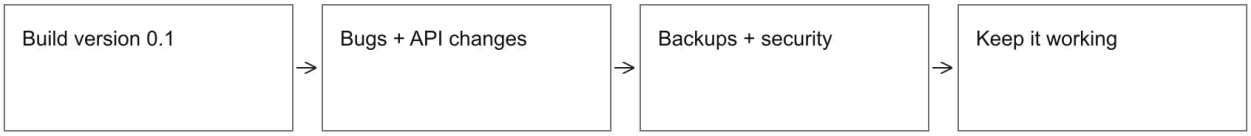
Start here

Describe your product in one paragraph. Then list the work a customer would inherit by building their own version.

01

People still pay to NOT build things

People pay to avoid owning the maintenance.



The first build is only the beginning. The customer also inherits everything needed to keep it working.

I can technically build my own CRM. Can also fix my sink, cook every meal, host an email server, sew my clothes. I don't do most of that. Being able to make something doesn't mean I want another thing to look after, and I think this gets lost when people compare a subscription price with one weekend of coding.

After the weekend there's setup, bugs, some API changing, auth randomly breaking, backups, security, that weird file somebody uploads six months later. Then your actual workflow changes and you have to go back into the thing you barely remember building. Version 0.1 is getting easy. Keeping it useful for yourself for the next two years sounds considerably less fun.

So yeah, people will still pay to avoid all that. But you have to actually make their life easier. If your product gives me more headache than my vibecoded version, well, why am I here.

02

Sell accumulated knowledge, not only software



The visible interface sits on top of the workflow and the knowledge collected from real use.

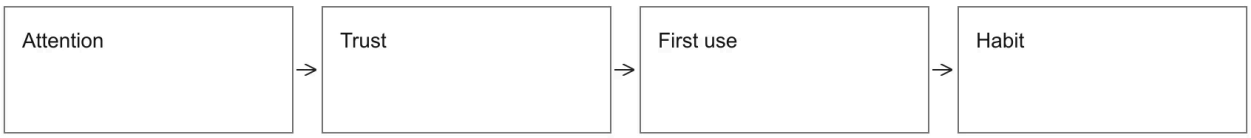
I keep thinking about the difference between a tool that technically does something and one that's been dealing with real people doing that thing for a few years. They can look almost identical. Paste something in, get an answer, nice interface. Except one already knows about the awful files people upload, the exceptions nobody mentioned in the spec, what the customer actually means when they ask for this output.

There's usually a bunch of ugly rules in there, templates somebody kept fixing, prompts that were tested against real examples, industry stuff you wouldn't know to ask about. You can copy the main workflow on Saturday. You don't get the last 3 years of “oh yeah, this breaks when they upload THIS kind of file” with it. Honestly that boring knowledge starts looking pretty valuable now.

03

Distribution becomes even more important

Cheaper code does not create a trusted product.



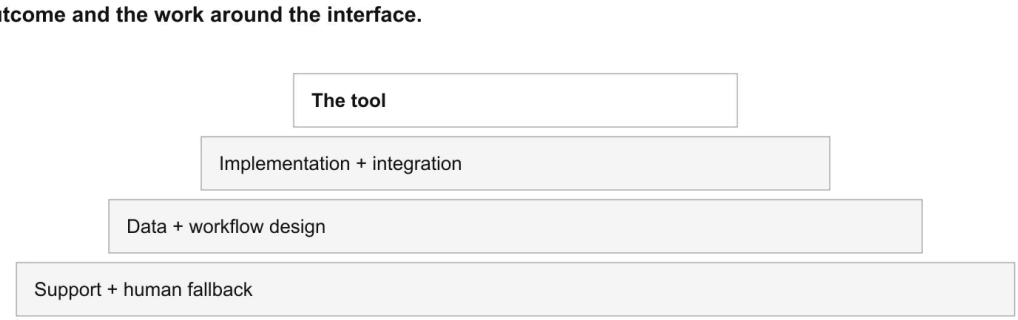
Building a product does not automatically create attention, trust, or the habit of using it.

And some fairly mediocre products are probably going to survive just because people already know them. If I need a tool today and there's one with 40k users, reviews, documentation, integrations, somebody I can message when it breaks, I might just buy it. I have other things to do. The fact that I could build an alternative doesn't have to be relevant to my decision at all.

Getting the code made is cheaper now, getting people to notice you and trust you still seems pretty hard. Then there's the habit of opening the same tool every morning, having your stuff in it, knowing where everything is. A very simple product with all of that can be worth a lot more than an impressive one nobody knows exists. Slightly depressing for engineers maybe, but yeah.

04

Sell the thing around the tool



The tool is the visible part. The work around it can carry much of the value customers pay for.

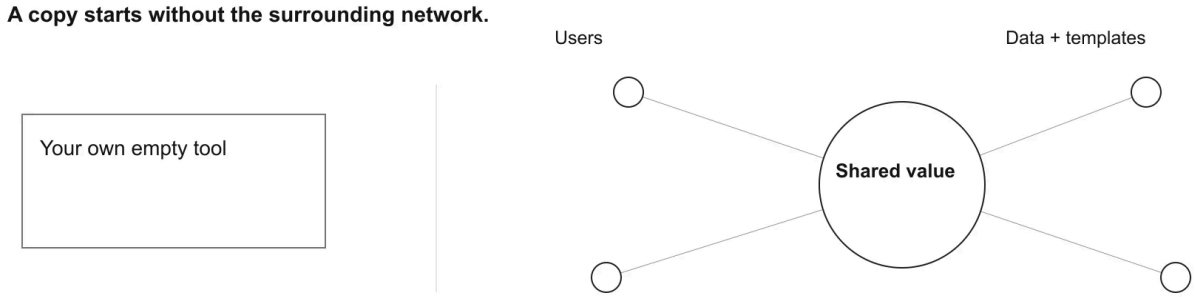
For some products I think the money is going to move towards the work around the software. A customer doesn't necessarily want an “AI tool that summarizes feedback”. They want their feedback from five different places brought together, cleaned up, mapped to the categories their team uses, and they want somebody to answer when the summaries start producing garbage.

Building the summarizer is pretty doable. Getting that whole mess working inside an actual company, then keeping it useful, is a different amount of work. You could charge for setting it up, adapting it to their process, integrations, support, expert review. Compliance or a human who takes over when the automation gets confused can matter a lot too, depending on who you're selling to.

The interface is just the part they can see. I wouldn't assume that's the part they're really paying for. Those “simple” workflows can turn into a pile of exceptions pretty quickly, and I think there's still quite a bit of money in dealing with the annoying bits for somebody else.

05

Products that get better because other people use them



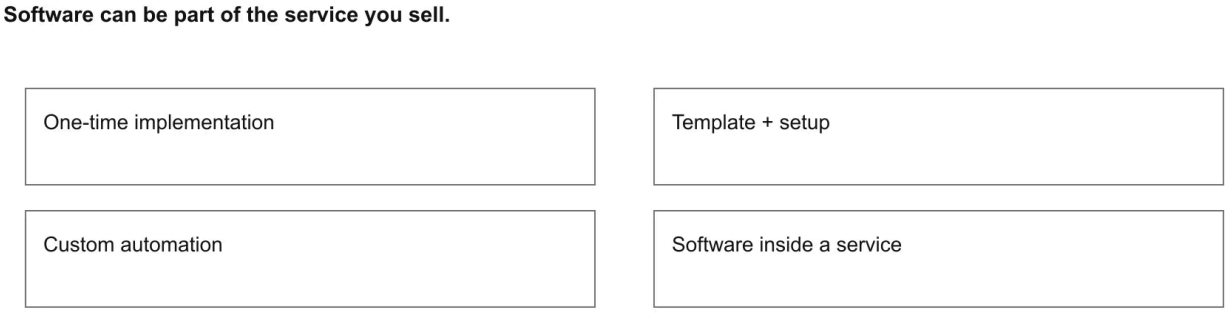
An isolated copy can reproduce the code. It does not instantly reproduce a useful network of people and contributions.

Of course building your own version gets less appealing if the useful stuff comes from other people being there. Templates users have made, data they've contributed, integrations, reputation, people to collaborate with. You can build yourself an empty marketplace too, congratulations, now you need to find everyone who was supposed to be in it.

It depends on the product, not everything has a network effect just because it has users. But where that shared stuff really matters, a personal copy is missing quite a lot. Same with some B2B tools that are already buried in how a company works. Recreating the screens might be the easy part of replacing them.

06

Sometimes the answer is literally: don’t sell software



A subscription is one option. Implementation, setup, automation, and services can also be what you sell.

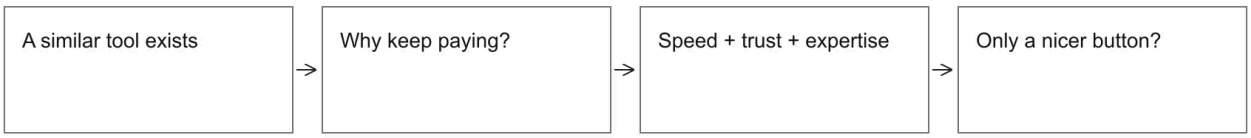
I'm also less convinced that every useful thing needs to become a SaaS subscription. If you're making automations for a particular type of company, you could get paid to build and set them up, then charge for changes when they need them. A template with some help getting it running could be enough. Or the software is just something you use while doing consulting, the client pays for the job getting done.

I think we've got a bit used to turning every idea into “how do I get thousands of people paying $19 a month”. But if building costs much less, a very specific thing for 20 customers can start making sense too. You still have to deal with them and do the work obviously, it doesn't magically become passive income. Just seems like there's more room for small businesses that wouldn't have been worth building before.

07

And obviously, some tools are just screwed

A nicer button is a weak answer to the switching question.



Test the reason to stay after a customer realizes a similar tool is buildable.

There are also tools where “why wouldn't I build this myself?” is a pretty fair question. Generic wrappers, basic dashboards, little utilities doing one transformation, very simple content generators. Everyone gets more or less the same workflow and there's not much else going on behind it. No unusual data, no years of learning the hard bits, no particular reason to trust this one over another.

People will still buy some of them because it's convenient. I'm not saying every simple tool is dead. I just wouldn't build one on the assumption that customers won't be able to make it themselves for the next 5 years. That assumption already feels pretty shaky.

The question to keep

Why would somebody keep using yours?

So when I'm looking at an idea now, “can I build it?” is becoming the less interesting bit. Usually, probably, yes. I'm more interested in what happens when a customer realizes they can make something similar themselves. Do they leave, or do they look at everything else you're taking care of and decide they'd rather keep paying?

Could be the data, people, integrations, the fact that you understand their problem. Could honestly just be that yours works and their own version keeps breaking. That's a perfectly decent reason. If all I've got is that my button looks nicer, I'd probably spend a bit more time on the idea before building it.

Write your answer

Customers keep paying because…

Read the full post ↗