Internal tools should be sold or killed

Series: practices

January 25, 2016

I work for a software development shop; we run on billable hours. Because of the unavoidable inefficiencies of a consulting pipeline, we find ourselves with small chunks of time between projects.

We use many tools and applications to help us do our jobs better. Because our staff has the ability to build software tools, we can solve our own problems in just the way we like.

The light bulb goes off. We can use our unbilled time to build tools that provide value to our company, minimizing the "lost value" of idle engineers!

I propose a somewhat radical line of thinking for evaluating whether to build an internal project: if we are to build it, we will sell it as an external product -- or we will not build it all and kill it.

Does our company need this tool?

If No: Kill it (why are we building something we don't need?)

If Yes: Often, this is where we stop. We need it, it has value, we have people "on the bench", therefore let's build it! But if we keep following "sell or kill", we should next be asking:

Do other companies (that are similar to us) need this tool?

If No: It's time to take a deep look at our perceived need. There are certainly exceptions, but we are not usually special snowflakes. Why don't other companies have this need? What do they do instead? Why isn't this a nagging problem for them like it (allegedly) is for us?

The solution might be that we need to look at the problem a different way.

For example, one internal tool we built is an internal LinkedIn that helps track the projects we've done, who worked on them, and what technologies we gained experience in. We use it to help staff projects (who knows Angular? iOS?), create marketing materials (look at how much work we did in the medical industry!), and support business development (you're building a .NET API -- here are three other projects that have used the same tools).

But what if we structured the company into specialized teams so it was obvious who has iOS experience (the people on the iOS squad, duh)? What if we moved away from metric-based marketing focused on billed hours? What if engineers "paired" with marketing and sales so we could collaborate and answer technical questions directly?

Could those work? Maybe, maybe not. But it is worth stepping back and thinking about before undertaking an internal project.

If we *really* have this need and others don't seem to, proceed to the next question. If not, kill it and take a different approach or re-evaluate the process.

If Yes: If multiple companies have a need, there are existing products or tools out there that attempt to address this need. Proceed to the next question, please!

Do existing products solve this need?

If Yes: Awesome, we should buy one of those then. Remember that even one week of time from an engineer to build this internally has a (theoretical) cost of several thousand dollars. Six weeks to build a tool at $100/hr is comparable to a $2000/month plan for a SaaS product. Spending hundreds a month for services seems crazy expensive to individuals, but it's a drop in the bucket for most companies.

Kill the internal project and buy whenever possible.

If No: We need to critically analyze the existing options and determine why they don't meet our needs. It is very tempting to dismiss these options because we can write our own software. We can make the interface just the way we want. We can integrate with our other systems and have control over all aspects of the tool. Are we focused on solving the actual problem or more interested in coming up with our own solution?

But it is possible that nothing on the market is right for us. Proceed onwards!

Will our internal tool have a killer feature?

If Yes: Awesome, skip ahead.

If No: If we cannot describe why the existing tools aren't right for us and we have no alternative approach, the best we can hope to do is essentially rebuild an existing option. Maybe it has a different UI or minor feature differences, but it is now our responsibility to keep it running, updated, secure, and supported -- with a revolving door of developers with small chunks of time to contribute.

Without a killer feature, usage will decline and the project will become neglected and the maintenance cost will rise. If there is not a clear value proposition, kill it.

Our company needs this, other companies need this, we identified a killer feature…so let’s invest in this product and sell it to others. Right?

If No (or "Well...but..."): This tool solves a real need for us, is valuable to others, and has a unique value proposition that would cause people to give us their money. And yet, we don't want to invest and build this as an external product?

This is the ultimate gut check. Either we've identified a potentially viable product (and should staff it like a billable project) or we've lied to ourselves on the previous questions. Let's give it a real go or kill it; skip the half-hearted, in-between efforts.

If Yes: Sell it! Treat it like a paying client project. Lock in a team for six months and don't cave two weeks later when billable work comes in. Aim to build a product that can actively pull in people during downtime when it advances the product, not that we have to push people onto to kill time.

Reframe unbillable time as an investment. By doing this, we can follow Nassim Taleb's "barbell strategy": 90% of staff in safe investments (billable work) and 10% of staff in highly speculated bets (product development).

We built it! But it didn’t sell :(What now?

We've still got the same options: sell it or kill it.

Is it selling, but not enough for it to be worth continued time investment? We could literally sell it -- find someone else that wants to run the app if we still need to use it. Maybe an engineer wants to run it as a side-gig? Sell them the rights for a dollar and a lifetime plan for the company. Maybe one of the existing customers is really depending on this tool and wants to take it over? Maybe you can list it on Flippa?

If it's not selling, just kill it. We tried something and learned a bunch, but in the end, it didn't work -- celebrate the failure. Free up time, bandwidth, and energy for engineers to work on something new or to spend that time on other valuable activities. Bring the empathy we've gained to our client work and help them build better products for their customers.

And if we don’t have a killer product to build (or we don’t want to build products), what should we do with downtime then? Twiddle our thumbs?

Learn new technologies (by building Breakable Toys, not creating burdensome legacy projects), learn new skills (training, deliberate practice), share what we've learned (writing, speaking, networking), think of ways we can delight our existing customers (prototyping, proactive proposals, trying new tools on our own time), improve each other (mentoring, pairing, coaching), give back (open source, volunteer, teach), work on validating the next internal project idea...

Hi. I'm **Matt Swanson**.

I do things and sometimes write about it here. I'm a Software Engineer from Indiana and I ship code at SEP.

~~Stalk~~ Follow me:

Further reading

Internal tools should be sold or killed

Code Neutral

Juking the Stats