Chapter 12:
PRODUCT DISCOVERY

Defining The Right Product

Software projects can be thought of as having two distinct stages: figuring out what to build (build the *right product*), and building it (building the *product right*). The first stage is dominated by product discovery, and the second stage is all about execution.

When in product discovery, you welcome and explore new ideas, talk with scores of users and customers, learn how you can apply new technologies, flesh out your product concepts and test them out, and spend a lot of time thinking about the overall product direction, both immediate and longer term. It is all about discovering that mix of form and function that results in a winning product.

However, once you've spec'd out this product, and your engineering team begins the process of building it, a very profound and important shift needs to take place for the product team. Now the game is all about execution—getting the product built, tested, and delivered to market. In this stage, you spend your time keeping everyone focused, chasing down the countless issues that inevitably arise, and getting these issues resolved immediately. Acquisitions, competitors, organizational and management changes—these are all distractions, and your job is to keep the team on track so this product can get out there when it needs to be.

In countless product teams, this shift in mindset doesn't actually happen—or at least it doesn't happen until much later, often as late as entering QA. Instead, product managers continue to explore new ideas, and company execs continue to view the product spec as malleable. What results is euphemistically referred to as "churn," where the product spec continues to change in significant ways, impacting engineering and the rest of the product team. As a result, the release dates get pushed out, or features get cut, or the quality gets compromised. Or all the above.

If you're lucky enough to have a great project manager, then you probably have help keeping everything on track during execution. But even if you do, as a product manager you'll need to be cognizant of this necessary change in mindset; otherwise, it is all too easy for the product manager to be the source of the product's inability to get to market.

However, I think it's important to recognize that we all have our own unique preferences and skills. If you're naturally a discovery kind of person—preferring the freedom and creativity of the invention process—then you'll have to work extra hard to contain those urges during execution. On the other hand, if you're more naturally the project manager type who loves getting things out the door, then you'll need to work on your strategic thinking and discovery skills—remembering that what matters most is creating a product that your customers love.

One technique I have found very useful is to always keep two versions of a product going in parallel. In other words, as soon as you start the engineering for release 1.0 and switch into execution mode for that project, then you start up the discovery for release 2.0 in parallel. Always keep that innovation engine working—once a given release goes to engineering, redirect your creative urges to the next release.

One note of warning: You do need to be careful that this approach doesn't detract from the execution work for the current project.

Overall, I've found that having this outlet is a good thing. The next time a company exec drops by with a big, new requirement, rather than impacting the product you have in the oven, you already have the next release in the discovery stage and you can start the work of exploring the new requirement there.

I don't mean to make this all sound overly simple, but I do believe that with discipline it can be managed. It's essential that you develop both your discovery skills (to ensure you're coming up with winning products) as well as your execution skills (to ensure that these great ideas actually make it to your customers).

? Can You Schedule Discovery?

Have you experienced this situation before? Your company gets all excited about a product idea, and as product manager you are asked to define it. You are told that the engineers will be finished with their current project in four weeks, so that means you can take all the time you need—so long as you are ready in four weeks.

No problem, you say (after all, sometimes you're only given days, so four weeks sounds great). You'll start with an opportunity assessment to understand the problem to be solved, then you'll spend quality time interviewing real users, and identify a preliminary set of requirements. By the start of the second week, you should be able to work with an interaction designer on a prototype, in the third week, you'll do user testing with the prototype and, in the fourth week, you'll flesh out the details of the use cases and review the prototype and spec with engineering.

These are all great practices. But what happens isn't usually so great. During your initial user discussions, you find that users aren't as excited about the idea as your management is, and/or you struggle to come up with a prototype that users can figure out, and/or the users aren't excited about the ideas in the prototype when they try it.

But time is up, the engineers are ready, so you give them what you have.

The result? During the next three to six months, engineering proceeds to build that same unusable and unexciting product that you saw in your prototype, you ship it, and then your management is of course disappointed with the results.

The problem isn't the reliability of the software, so the engineering team isn't to blame—after all, they just built what you asked them to. So whose fault is it? Everyone knows it's your fault—you're the product manager.

It doesn't help to talk to users, create prototypes, and test with users, if you don't adjust your course based on what you learn.

This notion of requirements and design as a sequential, predictable and scheduled phase in a product development process is so ingrained in our industry that it's often one of the most difficult habits for product organizations to break. But we all need to get past this. Product organizations need to come to terms with the fact that the product invention process is fundamentally a creative process. It is more art than science.

This is why I prefer to think of this phase as "product discovery" more than "requirements and design." I think this nomenclature emphasizes two all-important points:

First, you need to discover whether there are real users out there who want this product. In other words, you need to identify your market and validate the opportunity with your customers.

Second, you need to discover a product solution to this problem that is valuable, usable, and feasible. In other words, you need to design your solution and validate it with your customers and your engineering team.

Sometimes the product discovery phase is straightforward. Other times it is extremely difficult. In my experience, it's not so hard to discover and validate the market opportunity, but it's often very challenging to discover the solution. Even with the help of great designers and great engineers, some problems are just truly hard (at least many of the ones worth pursuing and that haven't been solved already).

The pharmaceutical drug industry provides an extreme example. The market discovery process is usually not very difficult—there are no shortage of good medical problems worth solving (like saving your life, extending your life, or improving the quality of your life). The hard part of course is discovering a product solution. Drug companies go into this discovery phase fully aware that there's no guarantee they'll come up with anything or, if they do, how long it may take. As an industry, they have come to terms with this element of uncertainty (and this risk is priced into their products).

Yet with software, even though we all know it is very hard—and we know that the majority of software releases fail to meet their objectives—we still insist on scheduling the discovery phase like we're planning the construction of a house.

Management especially struggles with this notion of product discovery. I think there are two underlying reasons for this:

First, the discovery process isn't as predictable as we wish it was. Management fears you may spend months chasing a solution and end up with nothing to show for it. At least if they go ahead and build, they can say that they shipped something. It's the same reason why many managers are uncomfortable with *Agile* methods like *Scrum*. They want to know how many 30-day sprints it will take before they're done.

Second, the most highly constrained and expensive resource in just about every software product organization is the engineers, and the thought that an engineering team might be sitting around with nothing to do but play Foosball just drives management nuts.

Ironically, it is precisely this reasoning that leads directly to wasted engineering resources.

Realize that almost every company executes the discovery process I've described here, only instead of using one prototyper for a few weeks, they use the full engineering team for full release cycles to build the software that is then QA'd and deployed into production systems. They are using the engineering organization to build a very, very expensive prototype, and they use their live customers as unwitting test subjects. This is also why it typically takes companies three or more releases over one to two years to get something they can make money from.

This is also a big reason why so many startups fail. Most startups simply don't have the funding to go two years before they gain traction in the marketplace. So they hire engineers, take their best shot, and see what happens. Ready, fire, aim.

Startups especially must focus their energies on this much faster product discovery process. And once they discover the solution that works—one that is valuable, usable and feasible—then it's all about execution. Until that point, they don't have to hire too many engineers right away—the engineers they already have can and should actively participate in the product discovery process. And, to a degree, the engineers can prepare the infrastructure while this discovery is going on.

You can help your management understand the nature of the product discovery process. If you consistently emphasize your obligation to ensure that what engineering builds must be valuable and usable, as well as enlist their efforts to discover a successful solution, you'll start to move their focus to this—the most critical stage of the product development process.