Extracted from:

Competing with Unicorns

How the World's Best Companies Ship Software and Work Differently

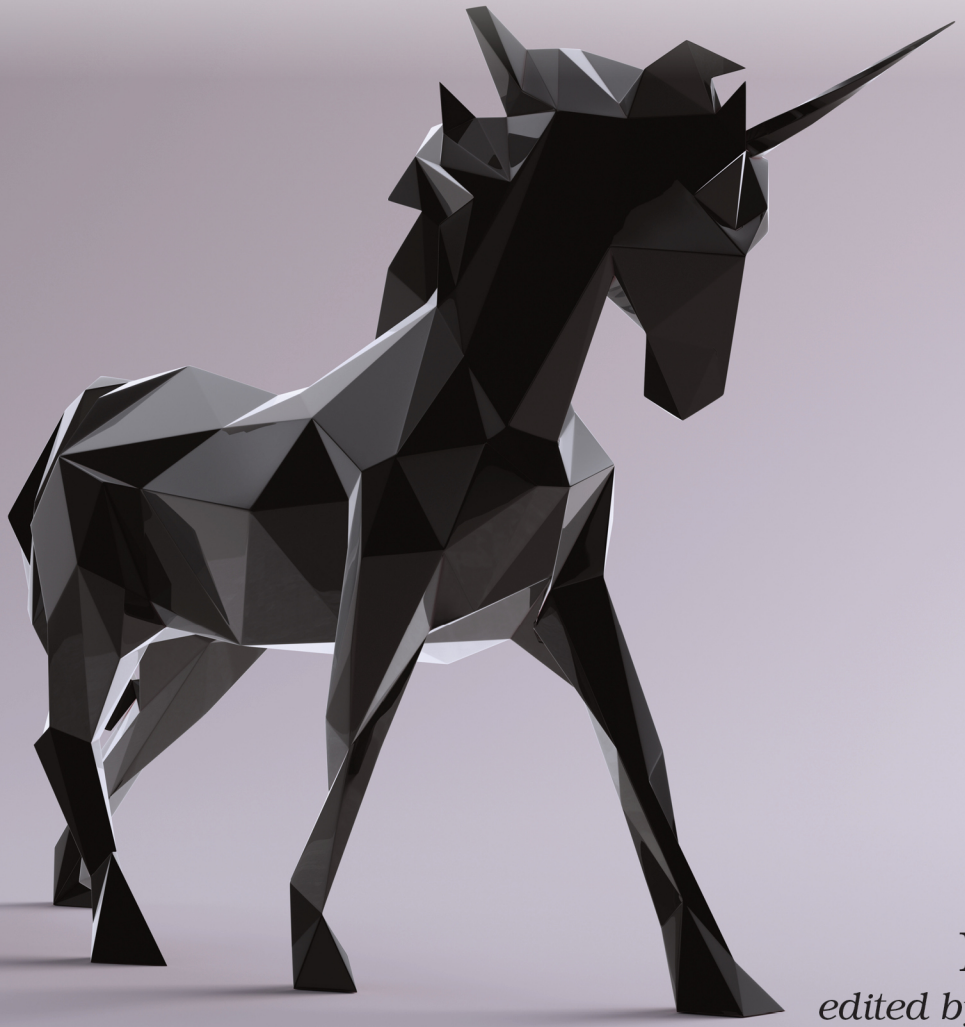
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Competing with Unicorns

How the World's Best Companies Ship Software and Work Differently



Jonathan Rasmusson

edited by Michael Swaine

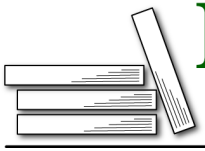
Competing with Unicorns

How the World's Best Companies Ship Software and Work Differently

Jonathan Rasmusson

The Pragmatic Bookshelf

Raleigh, North Carolina



Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Dave Rankin
Development Editor: Michael Swaine
Copy Editor: Sakhi MacMillan
Indexing: Potomac Indexing, LLC
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

ISBN-13: 978-1-68050-723-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—March 2020

CHAPTER 1

What's Different About Startups

Startups iterate, are super-focused on product, and put a premium on learning. Ask any traditional company if they value these things and they'll undoubtedly say: "Yes. Of course we do!" But if you watch their actions, you'll see they don't.

In this chapter we're going to see what makes startups different, what traditional companies need to rediscover if they want to compete on product, and why both view the purpose of software so differently.

Gaining this insight will not only enable you to see what delivery practices need to change in order to do new product development, it will kick-start the cultural and attitude changes necessary to start working this way.

Startups Are from Mars

When you first join a startup (or a unicorn that still operates like one), you immediately feel a difference. No longer is software development about time, dates, and budgets. Now it's all about the customer, impact, and learning.

To see why, let's quickly look at why startups don't value the things we do in the enterprise (things like conformance to plan) and instead put so much emphasis on exploration, discovery, and learning.

Existence Not Guaranteed

Startups are living on borrowed time. They don't have the luxury of recurring customers or revenue. There's only so much money in the bank. So they need to demonstrate traction and value quickly.

The point here isn't that there is a sense of urgency. Startups always feel the need to move fast. It's that they don't yet know what they need to build—and that they need to figure that out quickly as shown in the figure on page 2.

Live and Die by the Strength of Their Product

In the early days, startups may be able to get by purely on the basis of their idea. But as soon as they take money, they need to demonstrate value fast. Most often this is done through their product.

Product is everything to a startup. It's how they demo. It's how they attract new customers. It's how they raise money. It's how they learn. If the user goes to press play and the music doesn't stream—that's bad.

So product is everything to a startup. And as they experiment, iterate, and learn, they continuously get closer to their product market fit.

Are Searching for Product Market Fit

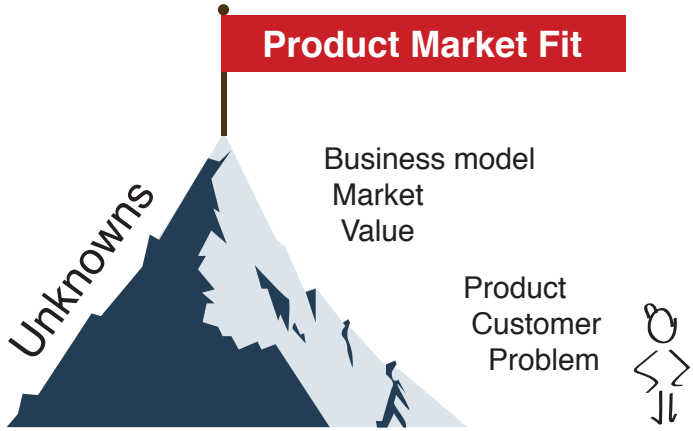
Product market fit is about finding the perfect product for the right market. This is what startups are continuously searching for because once they find it, they know they've got a winner.

You know you have perfect product market fit when:

- You can't make your product fast enough.
- Usage is growing faster than your ability to add servers.
- Money starts pouring in faster than you can spend it.
- You can't hire fast enough to keep up with demand.

Face a Lot of Unknowns

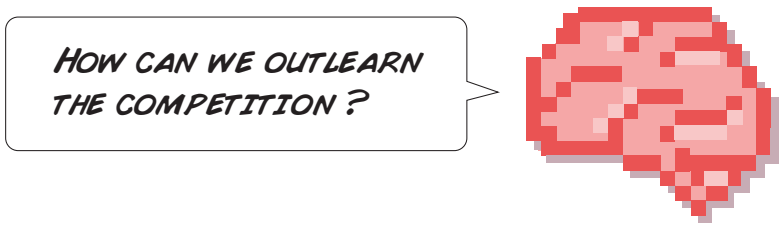
At the end of the day, startups face a lot of unknowns when trying to find their way in the world. They don't know who their customers are, what their product needs to do, much less how they're going to make money as shown in the figure on page 3.



But this is OK. It's all part of the game. A consequence, however, is it tilts startups heavily into experimentation and learning. Because at their heart, startups are really learning machines.

The Learning Machine

Startups care a lot about speed. But something they care about even more is learning.

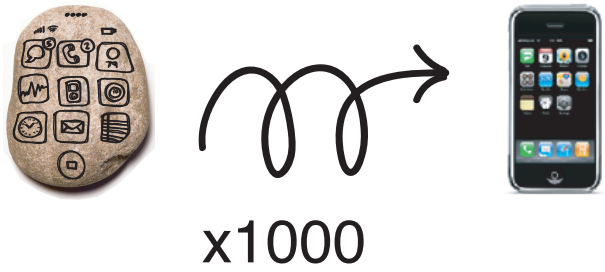


Startups are trying to outlearn the competition. If they have a great idea, chances are others will think it's a great idea too. So they're in a race. Not a race just in terms of speed. They're in a race about answering unknowns.

This causes them to do certain things. Chief among them when developing software and building product is iterating.

Startups Iterate

Startups iterate relentlessly when building product. Instead of building a product once and declaring victory, they instrument, analyze, and test and then take that knowledge and feed it back into the product over and over again as shown in the figure on page 4.



This takes an exploratory and discovery kind of mindset. You're never really done with product. You only get incrementally closer to where you want to be.

This enables them to demonstrate value. Show traction. And ultimately calibrate to something really close to what their customers want.

To do this at scale, however, they need everyone thinking critically. So one thing they do, that we in the enterprise don't, is empower.

Startups Empower

Spotify, Amazon, Google, and Facebook empower and trust people in ways most traditional companies wouldn't dare. They show them financials. They entrust them with all the data. They give developers admin access to their machines. And they are constantly asking their teams: "What can we do to help make you go faster."

It's a liberating way of work, because now instead of blaming management for a bad schedule, or crappy computers or environments, the teams are responsible for all these things themselves.

Unicorns out-trust and -empower teams when compared to traditional companies. And it shows in the quality of the work, and the quality of the product.

So the purpose of software delivery for startups is to:

- Experiment and learn
- Calibrate toward product market fit
- Demonstrate value to investors
- Prove to themselves they're on the right track

So let's now switch gears and see what the purpose of software development looks like through the eyes of the big traditional enterprise.

Enterprises Are from Venus

To understand why startups approach software delivery differently than traditional enterprises, you need to understand the different worlds each is coming from.

Startups, and tech companies, live and die by their ability to build product and serve customers. Building product means validating unknowns, getting product in front of customers, and then iterating on it twenty-seven times until they get it right. Highly iterative. Very exploratory. Outward facing.

Enterprises, on the other hand, are more inward facing. Here the focus is more on automating in-house systems all in the name of productivity and efficiency. With known requirements, in-house customers, and relatively few unknowns, the focus here is more around managing expectations, predictability, and planning.

What's changing for enterprises is that these two worlds of product and in-house enterprise development are now colliding. As startups continue to enter and disrupt traditional markets, enterprises are having to respond with new products and services at a speed and pace they're not used to. And here the startups have a real advantage.

With their focus on the customer, ability to identify gaps in markets, and aptitude to quickly adapt and learn, startups leave enterprises in the dust when it comes to execution. And what enterprises are quickly learning is that the enterprise software playbook they use to ship in-house software doesn't work when it comes to building product. It's the wrong vehicle for the job.

Which is why when enterprises do try to compete on product, they typically make two big mistakes.

1. TREAT PRODUCT DEVELOPMENT THE SAME AS ENTERPRISE DEVELOPMENT.
2. DON'T GIVE TEAMS ENOUGH AUTONOMY AND TRUST.

Because at their heart, most enterprise companies today aren't learning machines. What they are really into is meeting expectations instead.

Every Company Is Now a Software Company

Mark Andreessen once famously said software is eating the world, and he was right. Apple is becoming a bank. Uber, Airbnb, and Netflix have forever changed the world of transportation, accommodation, and entertainment, and Spotify completely disrupted music. And at the heart of it all is software.

As Microsoft CEO Satya Nadella put it: "Every company is now a software company." And whether you make cars, sell insurance, or run a bank, software is at the heart of it. And your company's ability to wield and create it will only continue to play a bigger role in your company's success.

Which is why just about every car manufacturer in the world now has an office in Silicon Valley. Not because Silicon Valley knows anything about cars. More because it knows a lot about software and how to create it.

www.satellitetoday.com/innovation/2019/02/26/microsoft-ceo-every-company-is-now-a-software-company/

The Expectation-Setting Machine

Ask any enterprise if they value learning, and the answer should be a resounding yes. But if you watch their actions, you'll see they value something even more—meeting expectations.

It starts at the top, with the CEO who begins by setting expectations about the upcoming year and what shareholders can expect from the company in the next quarter.

These expectations then get translated into annual budgets and eventually trickle down to teams in the form of projects and plans.

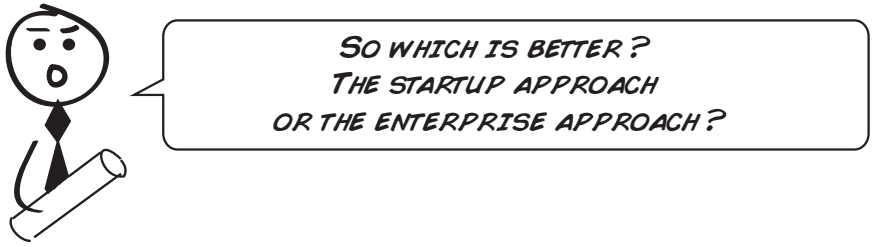
These projects and plans then become hard commitments. To ensure these expectations are met, enterprises then hire cadres of professionals called project managers who shepherd these commitments through the organization and spend a lot of time and energy ensuring everything goes according to plan. Tracking, reporting, and conformance to plan are rewarded above all else. And from a software delivery point of view, this is what success in the enterprise looks like.

When you compare these two worlds, the differences between how startups and enterprises approach delivery becomes much clearer as shown in the table on page 7.

Enterprise	Startup
Inward facing	Outward facing
Following plans	Learning
Automating existing systems	Creating new product
Few unknowns	Many unknowns
Project driven	Product driven
One shot	Iterative
Date & budget	Customer & impact
Top down	Bottom up
Low empowerment & trust	High empowerment & trust
Stick to the plan	Create the plan

Enterprise software delivery is largely an inwardly facing exercise in system automation. The customer is known. The requirements are known. And even if there are details to be worked out, there's an existing system the team can study to figure out what to automate. Discovery isn't valued here. Sticking to the plan is.

Startups are the complete opposite. Startups are looking for the plan. They're trying to discover what their customers want. There's no one sitting down the hall telling them what to build. Startups need to engage real customers and discover it—which is why startups put such a premium on learning, and why delivery takes a much more exploratory and iterative approach.



It's not a matter of which is better—they solve two completely different sets of problems.

The enterprise approach is good for planning and predictability. Companies can plan a year in advance, spec out what they would like to build, and then treat those projects as one-off initiatives. Highly predictable. Great for upfront planning.

The startup product development cycle, however, is different. Here the product can't be treated as a one-off. It simply needs to be done. And the path to getting there isn't a straight line. It's highly iterative. This requires ongoing investment, a longer time horizon, and a much greater emphasis put on discovery and learning—a very different approach and attitude toward delivery.

And this is where enterprises who are trying to stay nimble and compete struggle. Enterprises need to see software delivery not purely as a means of automating existing in-house systems but as a tool of discovery that can lead to new products, services, and capabilities never before imagined. It just requires a different way of thinking.

Ironically, all big slow-moving enterprises were once small nimble startups themselves. They've just forgotten how to flex that muscle that made them successful in the first place.

And this is really what makes these unicorns successful. They've found a way to let most of their teams operate like mini-startups, while gaining all the economies of scale that come with being an enterprise.

Build It and They Will Not Come

I once worked as a project manager for a company that wanted to pivot into a completely new vertical of work. Instead of selling electricity and power (which they were very good at), they wanted to get into the cattle industry (how hard could it be?). Turns out it was pretty hard.

Instead of building a cheap prototype, shipping something quick, and seeing if anyone would show up, they treated this new venture like any other in-house software project. They set aside a couple million dollars, spent two years building the thing, only to ship it and discover no one actually wanted it. Complete failure. CEO was fired.

The point of this story is that enterprises wanting to act and innovate like startups can't use the same playbook for in-house software development. It's a completely different game, with a completely different set of rules, and you need to unlearn what made you successful in the enterprises (following plans) and instead adopt a much more exploratory mindset.

You can't just build it and expect them to come. And many a startup has fallen prey to this line of thinking.

What This Means for You

Here's the takeaway for you: if you want to create teams that are capable of building great product, you need to change the lens through which they see the world. That means the following:

1. Redefine success

Success is no longer about conforming to plan. Success for us in product development is discovery and learning—which is why you are going to see your first product come out very fast. And then another and another quickly after that. There will be missteps. But with each release the product will get better. And we will know because we'll be looking for traction and measuring impact and value every step of the way.

2. Adopt a learning mindset

This means starting with questions, instead of going in assuming you have all the answers. You may have a hunch around what a great product might look like. But until you validate it, you don't really know. So you're going to want to focus a lot more on experimentation and learning, and not assume you've got everything figured out.

3. Find people who are comfortable dealing with unknowns

Working this way means going out there and finding answers—not sitting back and waiting for someone to hand the spec to you. No one is going to give you the requirements. You need to go out there and discover them for yourself. This makes some people uncomfortable. And many would rather just sit back and be told what to do. These are the wrong people for this type of work. You aren't looking for settlers here. What you need are explorers and pioneers.

4. Remove any stigma around failure

Failure is part of the game when building product. And if your company punishes failure, you're going to have a hard time recruiting the very people you need. So remove the stigma around failure. Let your team and the company know that we expect to fail a few times before getting things right—and that this isn't a one shot deal. The first release is just the beginning.

5. Empower and trust them to get the job done

This sounds like a cliche, but most companies don't really trust their employees (at least the way tech companies do). Great product is built by highly empowered and trusted teams. And in Chapter 3, Empower Through Squads, on page ?, you're going to learn how to empower your teams while supporting them organizationally along the way.

Look. I don't want to make any of this sound easy. You're going to face resistance from all sorts of angles in your organization. You are redefining success. Changing career paths. And moving the goal line on what it means to be

successful and get promoted. Changing the status quo is never easy, and you may well run into people who will fight this tooth and nail every step of the way.

Warning: Not everyone likes working this way

But the good news is that things are changing. As traditional companies continue to be challenged and disrupted, more and more are realizing they don't really have a choice. To survive they need to get better at building new products. They know they need to level up. And to do that, they're going to have to get out there and rediscover the very things that made them successful when they were small upcoming startups themselves.

The first step is changing perspective—which you just completed. The next step is changing how we organize and do the actual work—which is the focus of the next chapter.

Getting Away from the Mothership—How the IBM PC Got Built

Changing mindsets isn't a problem only non-tech companies face. IBM went through this when transitioning from an area of dominance, mainframe computing in the 1970s, into an area where it had little to no experience—the personal computer.

Realizing they were losing market share in the minicomputer space to the likes of DEC and Wang, IBM was determined not to be left behind in personal computers. There was just one problem.

No product ever shipped at IBM without taking 300 people three years to develop. IBM didn't have time for the old playbook. They needed something new now.

That's when Bill Lowe stepped up and said he could do it in a year, with a much smaller team, on one condition. They leave the mothership, IBM corporate promise to leave them alone, and they get complete control and say in how they build the product. IBM agreed.

They let the PC team set up in far away Boca Raton, Florida. They let the team decide how to source parts, how to allow third-party vendors to contribute software, and basically act as a completely independent business unit. And in one year, with an operating system provided from a then-budding new startup called Microsoft, the IBM PC was built, delivered ahead of schedule, in twelve months—a time faster than any other hardware product in IBM's history. And a new wave in personal computing had begun.

www.ibm.com/ibm/history/exhibits/pc25/pc25_birth.html

FOOD FOR THOUGHT

What would your software delivery process look like if you didn't have projects?

How much leeway do your teams have in product they build?

A little ☐ A lot ☐

How often do your teams go back and iterate on what they have already built?

Often ☐ Sometimes ☐ Never ☐

Who maintains the software your teams build?

The team ☐ Someone else ☐

Think Different

Product development is different from enterprise development. Unlike enterprise software development, which optimizes for following a plan, startups and tech companies are learning machines—which means they put a premium on learning, experimentation, and discovery.

To build product this way, you need to start getting comfortable with working in the unknown, having a different definition of success, and removing the stigma of failure when building new product.

With this frame of mind, we're now ready to go deeper and see why unicorns don't use the number one instrument we do in the enterprise for getting things done—the project—and instead use something that gives a lot more power and autonomy to teams. Something called a mission.