NICOLE FORSGREN 2 0

LENNY'S PODCAST

BILINGUAL TRANSCRIPT

ORIGINAL BY

Lenny Rachitsky

@lennysan • x.com/lennysan

ANALYSIS BY

@Penny777 • x.com/penny777

Generated: 2026-01-16

Nicole Forsgren 2.0 - 双语对照

This is the complete bilingual transcript for Lenny’ s Podcast featuring Nicole Forsgren.

[00:00:00] Lenny Rachitsky

English:

A lot of companies are trying to measure productivity for their teams.

中文翻译:

很多公司都在尝试衡量他们团队的生产力。

[00:00:03] Nicole Forsgren

English:

Most productivity metrics are a lie. If the goal is more lines of code, I can prompt something to write the longest piece of code ever. It's just too easy to gain that system.

中文翻译:

大多数生产力指标都是谎言。如果目标是增加代码行数，我可以写个提示词（prompt）让 AI 写出史上最长的代码。钻这种系统的漏洞简直太容易了。

[00:00:12] Lenny Rachitsky

English:

How do I know if my eng team is moving fast enough, if they can move faster, if they're just not performing as well as they can?

中文翻译:

我该如何判断我的工程团队动作是否够快？他们还能不能更快？或者他们是否只是没有发挥出应有的水平？

[00:00:18] Nicole Forsgren

English:

Most teams can move faster. But faster for what? We can ship trash faster every single day. We need strategy and really smart decisions to know what to ship.

中文翻译:

大多数团队都可以跑得更快。但快是为了什么？我们每天都可以更快地交付垃圾。我们需要战略和非常聪明的决策来确定到底该交付什么。

[00:00:27] Lenny Rachitsky

English:

One of the biggest issues we're going to probably have with AI is learning how much to trust code that it generates.

中文翻译:

关于 AI，我们可能会面临的最大问题之一，就是学会该在多大程度上信任它生成的代码。

[00:00:32] Nicole Forsgren

English:

We can't just put in a command and guess something back and accept it. We really need to evaluate it. Are we seeing hallucinations? What's the reliability? Does it meet the style that we would typically write?

中文翻译:

我们不能只是输入一个命令，拿到反馈就直接接受。我们真的需要去评估它。是否存在幻觉（hallucinations）？可靠性如何？它是否符合我们通常的代码风格？

[00:00:42] Lenny Rachitsky

English:

So much of the time is now going to be spent reviewing code versus writing code.

中文翻译:

现在，大量的时间将花在审查代码上，而不是编写代码上。

[00:00:45] Nicole Forsgren

English:

There's some real opportunity there to not just rethink workflows, but rethink how we structure our days and how we structure our work. Now, we can also make a 45-minute work block useful because getting into the flow is actually kind of handed off, at least, in part to the machine or the machine can help us get back into the flow by, reminding us of context and generating diagrams of the system.

中文翻译:

这其中蕴含着真正的机会，不仅可以重新思考工作流，还可以重新思考我们如何安排一天的生活和工作结构。现在，我们甚至可以让 45 分钟的工作块变得非常高效，因为进入“心流”（flow）状态的过程实际上在某种程度上移交给了机器，或者机器可以通过提醒我们上下文、生成系统图表来帮助我们重新进入心流。

[00:01:03] Lenny Rachitsky

English:

What's just one thing that you think an eng team, a product team can do this week, next week to get more done?

中文翻译:

你认为工程团队或产品团队在本周或下周可以做哪一件事来提高产出?

[00:01:09] Nicole Forsgren

English:

Honestly, I think the best thing you can do-

中文翻译:

老实说，我认为你能做的最好的事情是——

[00:01:12] Lenny Rachitsky

English:

Today, my guest is Nicole Forsgren. With so much talk about how AI is increasing developer productivity, more and more people are asking, "How do we measure this productivity gain? And are these AI tools actually helping us or hurting how our developers work?" Nicole has been at the forefront of this space longer than anyone. She created the most used frameworks for measuring developer experience called DORA and SPACE. She wrote the most important book in the space called Accelerate and is about to publish her newest book called Frictionless, which gives you a guide to helping your team move faster and do more in this emerging AI world. Her core thesis is that AI indeed accelerates coding. But developers aren't speeding up as much as you think because they still have to deal with broken builds and unreliable tools and processes, and a bunch of new bottlenecks that are emerging.

(00:02:01):

In our conversation, we chat about her current, best and very specific advice for how to measure productivity gains from AI, signs that your team could be moving faster, what companies get wrong when trying to measure engineering productivity, how AI tools are both helping and hurting engineers, including getting into flow states, her seven-step process for setting up a developer experience team at your company, how to get buy-in and measure the impact of a team like this and a ton more. This episode is for anyone looking to improve the performance of their engineering teams. If you enjoy this podcast, don't forget to subscribe and follow it in your favorite podcasting app or YouTube. It helps tremendously. Also, to become an annual subscriber of my newsletter, you get a year free of 15 incredible products including Lovable, Replit, Bolt, n8n, Linear, Superhuman, Descript, Wispr Flow, Gamma, Perplexity, Warp, Granola, Magic Patterns, Raycast, ChatPRD, and Mobbin. Head on over to lennysnewsletter.com and click product pass. With that, I bring Nicole Forsgren.

中文翻译:

今天的嘉宾是 Nicole Forsgren。随着关于 AI 如何提高开发者生产力的讨论越来越多，越来越多的人在问：“我们如何衡量这种生产力的提升？这些 AI 工具到底是在帮助我们，还是在损害开发者的工作方式？”Nicole 在这个领域的前沿探索比任何人都久。她创建了衡量开发者体验（DevEx）最常用的框架，即 DORA 和 SPACE。她撰写了该领域最重要的著作《Accelerate》（加速），并即将出版新书《Frictionless》（无摩擦），该书为在这个新兴的 AI 世界中帮助团队跑得更快、做得更多提供了指南。她的核心论点是：AI 确实加速了编码，但开发者的整体速度并没有你想象的提升那么多，因为他们仍然需要处理失败的构建（broken builds）、不可靠的工具和流程，以及一系列新出现的瓶颈。

(00:02:01):

在我们的对话中，我们聊到了她目前关于如何衡量 AI 带来的生产力提升的最新且具体的建议、团队可以跑得更快的迹象、公司在衡量工程生产力时常犯的错误、AI 工具如何既帮助又损害工程师（包括进入心流状态）、她在公司建立开发者体验团队的七步流程、如何获得支持并衡量此类团队的影响力等等。本集节目适合任何想要提升工程团队绩效的人。如果你喜欢这个播客，请不要忘记在订阅并关注。此外，成为我时事通讯的年度订阅者，你可以免费获得 15 款不可思议的产品的一年使用权。请访问 lennysnewsletter.com 并点击 product pass。下面，让我们欢迎 Nicole Forsgren。

[00:03:01] Lenny Rachitsky (Ad)

English:

This episode is brought to you by Mercury. I've been banking with Mercury for years. And honestly, I can't imagine banking any other way at this point. I switched from Chase and, holy moly, what a difference. Sending wires, tracking spend, giving people on my team access to move money around is so freaking easy. Where most traditional banking websites and apps are clunky and hard to use, Mercury is meticulously designed to be an intuitive and simple experience. And Mercury brings all the ways that you use money into a single product, including credit cards, invoicing, bill pay, reimbursements for your teammates, and capital. Whether you're a funded tech startup looking for ways to pay contractors and earn yield on your idle cash or an agency that needs to invoice customers and keep them current, or an e-commerce brand that needs to stay on top of cash flow and access capital, Mercury can be tailored to help your business perform at its highest level. See what over 200,000 entrepreneurs love about Mercury. Visit mercury.com to apply online in 10 minutes. Mercury is a FinTech, not a bank. Banking services are provided through Mercury's FDIC-insured partner banks.

中文翻译:

本集节目由 Mercury 赞助。我已经使用 Mercury 银行服务多年了。老实说，我现在无法想象用其他方式处理银行事务。我从大通银行（Chase）转到了 Mercury，天哪，差别太大了。发送汇款、跟踪支出、授权团队成员转账都变得异常简单。大多数传统银行的网站和 App 都很笨重难用，而 Mercury 经过精心设计，提供了直观且简单的体验。Mercury 将你使用资金的所有方式整合到一个产品中，包括信用卡、发票、账单支付、团队报销和资本服务。无论你是想给承包商付钱并赚取闲置资金收益的初创公司，还是需要向客户开票的机构，亦或是需要掌控现金流的电商品牌，Mercury 都能量身定制，助你的业务高效运行。去看看超过 20 万名企业家为何喜爱 Mercury 吧。访问 mercury.com，10 分钟即可在线申请。Mercury 是一家金融科技公司，而非银行。银行服务由 Mercury 的 FDIC 承保伙伴银行提供。

[00:04:15] Lenny Rachitsky (Ad)

English:

Here's a puzzle for you. What do OpenAI, Cursor, Perplexity, Vercel, FLAN, and hundreds of other winning companies have in common? The answer is they're all powered by today's sponsor, WorkOS. If you're building software for enterprises, you've probably felt the pain of integrating single sign-on, skim, RBAC, audited logs, and other features required by big customers. WorkOS turns those deal blockers into drop-in APIs with a modern developer platform built specifically for B2B SaaS. Whether you're a seed-stage startup trying to land your first enterprise customer or a unicorn expanding globally, WorkOS is the fastest path to becoming enterprise-ready and unlocking growth. They're essentially Stripe for enterprise features. Visit workos.com to get started or just hit up their Slack support where they have real engineers in there, who answer your questions super fast. WorkOS allows you to build like the best with delightful APIs, comprehensive docs, and a smooth developer experience. Go to workos.com to make your app enterprise-ready today.

中文翻译:

给你出一个谜题：OpenAI、Cursor、Perplexity、Vercel、FLAN 以及数百家其他成功的公司有什么共同点？答案是：它们都由今天的赞助商 WorkOS 提供支持。如果你正在为企业构建软件，你可能感受过集成单点登录（SSO）、SCIM、基于角色的访问控制（RBAC）、审计日志以及大客户要求的其他功能的痛苦。WorkOS 通过专为 B2B SaaS 构建的现代开发者平台，将这些交易阻碍变成了即插即用的 API。无论你是试图签下第一个企业客户的种子期初创公司，还是正在全球扩张的独角兽，WorkOS 都是让你具备“企业就绪”能力并解锁增长的最快路径。他们本质上是企业级功能的“Stripe”。访问 workos.com 开始使用，或者直接联系他们的 Slack 支持，那里有真正的工程师快速回答你的问题。WorkOS 让你能通过愉悦的 API、详尽的文档和流畅的开发者体验，像顶尖公司一样构建产品。

[00:05:13] Lenny Rachitsky

English:

Nicole, thank you so much for being here and welcome to the podcast.

中文翻译:

Nicole，非常感谢你能来，欢迎来到本播客。

[00:05:16] Nicole Forsgren

English:

Thank you. It's so good to be here.

中文翻译:

谢谢。很高兴能来到这里。

[00:05:19] Lenny Rachitsky

English:

It's so good to have you back. I was just watching our first episode, which we did two and a half years ago. I was watching it, and I was both shocked and not shocked that we barely talked about AI. The episode was called How to Measure and Improve Developer Productivity, and we got to AI barely like an hour in and we're just like, "Hmm, I wonder what's going to happen with AI and productivity." Does that just blow your mind?

中文翻译:

很高兴你能回来。我刚才还在看我们两年半前做的第一集。看着看着，我既感到惊讶又不感到惊讶，因为我们当时几乎没谈到 AI。那一集的主题是《如何衡量和提高开发者生产力》，我们直到快一个小时的时候才勉强提到 AI，当时只是觉得：“嗯，我想知道 AI 会对生产力产生什么影响。”这有没有让你觉得不可思议？

[00:05:41] Nicole Forsgren

English:

Yeah. Because it was just hitting the scene, it was the topic of so much conversation, and at the same time, so many things don't change. So many things are still important, so many things are the same. Yeah. It's also a little wild that it's been two and a half. Where does time go? Time is a social construct?

中文翻译:

是的。因为当时 AI 才刚刚崭露头角，虽然讨论很多，但与此同时，很多事情并没有改变。很多事情依然重要，很多事情还是一样。两年半的时间确实有点疯狂。时间都去哪儿了？时间难道只是个社会建构吗？

[00:06:01] Lenny Rachitsky

English:

Yeah. Most of our conversation was just questions like, "Well, how might this impact people? How will we change the way we build product?" It was barely a thing back then. Now, it's the only thing that I imagine people want to talk about when they talk about engineering productivity. That's where we're going to be spending a lot of our time focusing on today. The reason I'm excited about this conversation, it feels like there's been so much money poured into AI tools increasing productivity. The fastest growing companies in the world are these engineering AI tools. And now, more and more people are just asking this question of just, "What gains are we getting out of this? How much is this actually helping us be more productive? How do we become more productive?"

(00:06:39):

You've been at the center of this world for longer than anyone. You've invented so many of the frameworks that people rely on now. So I'm really excited to have you back to talk about this stuff. I want to start with just this term DevEx, it's something that comes up a lot in this whole space, and we're going to hear this term a bunch in this conversation. Can you just explain what is DevEx, this term DevEx?

中文翻译:

是的。当时我们的大部分对话只是在问：“这可能会如何影响人们？我们会如何改变构建产品的方式？”那时它还几乎不算个事。而现在，我猜当人们谈论工程生产力时，AI 是唯一想聊的话题。这也是我们今天大部分时间要讨论的重点。我之所以对这次对话感到兴奋，是因为现在有巨额资金投入到提高生产力的 AI 工具中。全球增长最快的公司都是这些工程 AI 工具公司。而现在，越来越多的人在问：“我们从中得到了什么收益？这到底在多大程度上帮助我们提高了生产力？我们该如何变得更高产？”

(00:06:39):

你在这个领域的中心位置待得比谁都久。你发明了许多人们现在依赖的框架。所以我很高兴你能回来聊聊这些。我想先从“DevEx”这个词开始，这个词在这个领域经常出现，我们在接下来的对话中也会多次听到。你能解释一下什么是 DevEx 吗？

[00:07:00] Nicole Forsgren

English:

DevEx is developer experience. And when we think about developer experience, we're really talking about what it's like to build software, day to day, for a developer. So the friction that they face, the workflows that they have to go through, any support that they have. It's important because when DevEx is poor, everything else just isn't going to help. The best processes, the best tools, the best... whatever magic you have, if the DevEx is bad, everything kind of takes-

中文翻译:

DevEx 就是开发者体验（Developer Experience）。当我们思考开发者体验时，我们谈论的是开发者日常构建软件的真实感受。包括他们面临的摩擦、必须经历的工作流、以及他们获得的任何支持。它之所以重要，是因为

如果 DevEx 很糟糕，其他任何东西都帮不上忙。无论你有最好的流程、最好的工具，还是什么神奇的手段，如果开发者体验不好，一切都会大打折扣。

[00:07:34] Lenny Rachitsky

English:

Within DevEx is productivity, and I think the key insight that you had and other folks in the space of that is not just productivity, but there's also engineering happiness. We're going to get into a lot of these parts, but just maybe speak to... there's productivity and there's broader components to engineers being successful at a company.

中文翻译:

DevEx 包含生产力，而且我认为你和该领域的其他人提出的关键见解是，它不仅关乎生产力，还关乎工程师的幸福感。我们会深入探讨这些部分，但也许你可以先谈谈……除了生产力之外，工程师在公司取得成功还有哪些更广泛的组成部分。

[00:07:51] Nicole Forsgren

English:

Yeah. I love that point because productivity, first of all, is hard to define anyway. But if you're just looking at output, you can get there in a lot of different ways. But if you're getting there in ways that are high toil or high friction, then at some point, a developer is going to burn out. Or if it's super high cognitive load, if it's hard to even think about what you're doing because concentrating on the mechanics of... the plumbing of something, then you don't have the brain space left to come up with really innovative solutions and questions. So I love that it's kind of this self-reinforcing loop in terms of, "You do more work, you do better work." And it's better for people, it's better for the systems, it's better for our customers.

中文翻译:

是的。我很喜欢这一点，因为首先，生产力本身就很难定义。如果你只看产出，你可以通过很多不同的方式达到目标。但如果你是通过高强度的琐事（toil）或高摩擦的方式达到的，那么开发者迟早会精疲力竭。或者如果认知负荷（cognitive load）极高，如果你因为专注于底层细节或琐碎的衔接工作而难以思考核心任务，那么你就没有余力去提出真正创新的解决方案和问题。所以我很喜欢这种自我强化的循环：“你做得更多，你做得更好。”这对人更好，对系统更好，对我们的客户也更好。

[00:08:34] Lenny Rachitsky

English:

I was going to get to this later, but I want to actually get to this right now, this idea of flow state for engineers. I was an engineer, actually, early in my career. I went to a school for computer science. I was an engineer for 10 years. The best part of the job for me was just this flow state you enter when you're coding and building, and just things feel like so fun. It feels like AI is making that harder in a lot of ways because there's all these agents you're working with now, there's all this code that's kind of being written for you. Talk about just the importance of flow state to a developer, happiness, developer productivity, and just what you've seen AI impacting. How you've seen AI impacting that?

中文翻译:

我本来想晚点再聊这个，但现在就想切入：关于工程师的“心流状态”（flow state）。我职业生涯早期其实是一名工程师，学的是计算机科学，做了 10 年工程师。对我来说，这份工作最棒的部分就是你在编码和构建时进入的那种心流状态，感觉非常有意思。但感觉 AI 在很多方面让这变得更难了，因为现在你要和各种智能体（agents）合作，很多代码是为你写好的。请谈谈心流状态对开发者幸福感和生产力的重要性，以及你观察到的 AI 对此产生了什么影响?

[00:09:07] Nicole Forsgren

English:

Well, there are lots of different ways to talk about DevEx. One way to talk about it is kind of three key things that have components that are important of themselves, and they also kind of reinforce each other. Flow state is one of them, cognitive load is another, and then feedback loops are another. I think when you touch on this... Your question about flow state is a really good one, and I'll admit we're just a few years into this. We're still figuring out what the best flow state and cognitive requirements are for people in this because, to your point, sometimes we're getting interrupted all the time. You don't just get in the flow and lock down, and write a whole bunch of code and do the typing of a whole bunch of code as much anymore. Instead, you're kind of creating a prompt, getting some code back and reviewing the code, trying to integrate what's happening in the system, and that can really interrupt.

(00:10:02):

At the same time though, it can contribute to flow if... I've seen some senior engineers pull together some tool chains that are really incredible, where they figured out how to keep the flow going. The fast feedback loops really, really work well for them. They can kind of assign out different pieces to agents. It helps them keep in the flow in terms of... Instead of details and line-by-line writing, they're in the flow in terms of, "What's my goal? What are the pieces that I need to get there? How quickly can I get there? So then, I can step back and kind of evaluate everything, and then dive back in and fix some pieces."

中文翻译:

谈论 DevEx 有很多种方式。其中一种方式是看三个关键要素，它们本身都很重要，且相互强化：心流状态、认知负荷和反馈循环（feedback loops）。关于心流状态的问题非常好，我承认我们才进入这个阶段几年。我们仍在摸索在这种环境下人们最佳的心流状态和认知需求是什么。因为正如你所说，有时我们会不断被打断。你不再像以前那样进入心流、闭关、然后亲手敲出大量代码。相反，你是在创建提示词，获取代码，然后审查代码，尝试将其整合到系统中，这确实会产生干扰。

(00:10:02):

但与此同时，它也可以促进心流。我见过一些资深工程师搭建了非常了不起的工具链，他们找到了保持心流的方法。快速反馈循环对他们非常有效。他们可以将不同的任务分配给不同的智能体。这帮助他们在更高层级保持心流：不再纠结于逐行编写的细节，而是专注于“我的目标是什么？我需要哪些组件？我能多快达到目标？然后我可以退后一步评估全局，再深入进去修复某些部分。”

[00:10:34] Lenny Rachitsky

English:

Is there anything more you could say about this engineer that figured out this really cool workflow, about just what that looks like?

中文翻译:

关于这位想出如此酷的工作流的工程师，你还能多透露一点细节吗？具体是什么样的?

[00:10:39] Nicole Forsgren

English:

I've spoken with a handful of them, and I've kind of watched them work. I haven't built it myself yet. It's on my list. They've been able to set up this really incredible workspace and workflow where... Right now, a lot of us play around with tools and... We'll put in a prompt and we'll get a few lines back or maybe we'll put in a prompt and we'll get whole programs back. Well, what they can do is they can... Many times I'll see them say, to help prime it, "This is what I want to build. It needs to have these basic architectural components. It needs to have this kind of a stack. It needs to follow this general workflow. Help me think that through," and it'll kind of design it for it. And then for each piece, it'll assign an agent to go work on each pace in parallel, and then it'll say and upfront, "These need to be able to work together, make sure it's architected correctly. Make sure we use appropriate APIs and conventions."

(00:11:30):

Then at the end, they can let it run for a few minutes. They can think through something else that's interesting or they anticipate is going to be hairy, and they come back to something that's probably a little better than vibe coded. Because they were so systematic about it upfront, they're much closer to something that looks like production code.

中文翻译:

我和其中几个人聊过，也观察过他们工作。我自己还没亲手搭过，但在我的计划清单上。他们能建立起这种不可思议的工作空间和工作流。现在我们很多人只是在摆弄工具，输入提示词拿回几行代码，或者拿回整个程序。而他们能做的是，为了预热 AI，他们会说：“这是我想构建的东西。它需要有这些基础架构组件，需要这种技术栈，遵循这个通用工作流。帮我理清思路。”AI 会帮他设计。然后对于每个部分，他会分配一个智能体并行工作，并预先说明：“这些部分需要能够协作，确保架构正确，确保使用合适的 API 和规范。”

(00:11:30):

最后，他们可以让它运行几分钟。这段时间他们可以思考其他有趣或预感会很棘手的事情。当他们回来时，得到的东西比那种“凭感觉写的代码”（vibe coded）要好得多。因为他们预先进行了系统化的规划，产出的东西更接近生产环境级别的代码。

[00:11:51] Lenny Rachitsky

English:

So what I'm hearing is spending a little time upfront planning, what all these AI engineers are doing, versus just powering through and just figuring out as you go.

中文翻译:

所以我听到的是，这些“AI 工程师”会花一点时间预先规划，而不是直接蛮干、边做边想。

[00:12:02] Nicole Forsgren

English:

Yeah.

中文翻译:

是的。

[00:12:02] Lenny Rachitsky

English:

Okay, cool. Let me get to this quite a core question that I think on is a lot of people's minds. A lot of companies are trying to measure productivity for their teams, "Is this improving our productivity? Is this hurting our productivity?" So let me just start with this question, how are people doing this wrong currently when they try to measure their productivity gains with AI?

中文翻译:

好，太酷了。让我切入一个很多人关心的核心问题。很多公司都在尝试衡量团队的生产力：“这是否提高了我们的生产力？还是损害了生产力？”那么先问这个问题：目前人们在尝试衡量 AI 带来的生产力提升时，哪些做法是错误的？

[00:12:23] Nicole Forsgren

English:

I'll say most productivity metrics are a lie. It's really tricky because, historically... Now, look, lines of code has always been a bad metric, but many folks still use lines of code-

中文翻译:

我会说大多数生产力指标都是谎言。这非常棘手，因为从历史上看……听着，代码行数（LOC）一直是个糟糕的指标，但很多人仍然在使用它——

[00:12:37] Lenny Rachitsky

English:

[inaudible 00:12:37].

中文翻译:

（听不清）。

[00:12:37] Nicole Forsgren

English:

... yeah, as some proxy as some proxy for output or productivity or complexity or something. Well, now, for many of the systems, that they would sometimes whisper and not super talk about that uses lines of code, it's just blown out of the water because, "What do you mean by lines of code?" If the goal is more lines of code, I can prompt something to write the longest piece of code ever and add tons of comments. We know that agents and LLMs tend to be very verbose by definition, and so it's just too easy to gain that system and then introduce complexity and technical debt into all of the work that you're doing. I will say there are some things that we can kind of watch and pay attention to because... So lines of code as a productivity metric isn't great, it's pretty bad. But now, it's kind of more relevant if we can tease out which code came from people and which code came from AI because now we can answer downstream questions.

(00:13:40):

"What is the code survivability rate? What is the quality of our code? Is our code being fed back into trained systems? And for that code that's retraining systems later, especially if we're doing fine-tuning and local tuning, how much of that is machine generated? What types of loops is that creating, and what types of patterns or biases might it be inadvertently introducing?" On the one hand, it's not good as a productivity metric, but it can be useful. I'll even say the same for DORA. I have done DORA metrics, their speed metrics, their stability metrics. If that's all you're looking at, it's not going to be sufficient anymore because AI has now changed the way we think about feedback loops. They need to be much faster. Now, what DORA's meant for, kind of assessing the pipeline overall in terms of speed and stability. Still, that works. But we can't just blindly apply the existing metrics we've used before because we'll miss super important phenomenon and changes in the way people work.

中文翻译:

……是的，把它作为产出、生产力或复杂性的某种代理指标。好吧，现在对于许多私下使用代码行数指标的系统来说，这个指标已经彻底失效了。因为“代码行数”到底意味着什么？如果目标是更多的代码行，我可以写个提示词让它写出史上最长的代码并加上海量注释。我们知道智能体和大型语言模型（LLM）定义上就很啰嗦，所以钻这种系统的漏洞太容易了，这会给你的工作引入复杂性和技术债。但我会说，有些东西我们可以观察和关注。虽然代码行数作为生产力指标很糟糕，但如果我们能区分哪些代码是人写的，哪些是 AI 写的，它就变得更有意义了，因为我们可以回答下游的问题：

(00:13:40):

“代码的存活率是多少？我们的代码质量如何？我们的代码是否被反馈到训练系统中了？对于那些用于后续重新训练系统的代码（特别是我们在做微调和本地调优时），有多少是机器生成的？这会产生什么样的循环？可能会无意中引入什么样的模式或偏见？”一方面，它作为生产力指标不好，但它是有用的。我甚至对 DORA 也有同样的看法。我制定了 DORA 指标，包括速度指标和稳定性指标。如果你只看这些，现在已经不够了，因为 AI 改变了我们对反馈循环的思考方式。反馈需要快得多。DORA 旨在从速度和稳定性方面评估整体流水线，这依然有效。但我们不能盲目套用以前的指标，否则我们会错过人们工作方式中极其重要的现象和变化。

[00:14:38] Lenny Rachitsky

English:

Interesting. You invented DORA, that was kind of the main framework people used for a long time to measure productivity. And then there's SPACE, there's Core 4, there's probably others. So what I'm hearing here is all these are kind of out of date now, where AI is contributing large portions of code.

中文翻译:

很有意思。你发明了 DORA，那是很长一段时间里人们衡量生产力的主要框架。然后还有 SPACE、Core 4，可能还有其他的。所以我听到的是，在 AI 贡献了大量代码的情况下，这些框架都有点过时了。

[00:14:55] Nicole Forsgren

English:

I will say if it is a prescriptive metric, it needs to be used only in the way it was prescribed.

中文翻译:

我会说，如果它是一个规范性指标（prescriptive metric），它就必须严格按照规定的方式使用。

[00:15:00] Lenny Rachitsky

English:

So

中文翻译:

所以

[00:15:01] Nicole Forsgren

English:

DORA 4, there are four key metrics. There's two speed metrics, deployment frequency and lead time. So code commit to code deploy. There's stability metrics, MTTR and change fail rate. If those are used to assess the speed of the pipeline and the general performance of the pipeline, that's great. If you're trying to use those to understand... Because implied in that is feedback loops, right, because you used to kind of get feedback from customers. But we can't just use that blindly now when we're using AI, as an example, because we have feedback loops much earlier and not even just at the local build and test phase. We have feedback loops throughout, and even sometimes in the middle of some of the pipeline, that we really want to leverage in ways that weren't as useful before. I won't say they weren't possible, but we just didn't really focus there.

(00:15:53):

So those are prescriptive metrics. When we think about SPACE, SPACE is a framework. It doesn't tell you what metric to use. So I'll say, sometimes people get real frustrated because I didn't tell them what to measure. But now, I think that's the power of it. We're actually seeing that SPACE applies fairly well in these new emerging contexts like AI because we still want to look at... SPACE is an acronym. We still want to look at satisfaction. We still want to look at performance, what's the outcome. We still want to look at activity. Yes, in some ways, lines of code and number of PRs can be useful for something, or number of alerts or number of things, activities or counts. Seize communication and collaboration, this is also super important and useful because it's how our systems communicate with each other, and also how our people do. "What proportion of work is being offloaded to a chat bot versus talking to a senior engineer on the team?" More isn't always better and less isn't always better, it depends.

(00:16:50):

And then efficiency and flow, "Can people get in the flow? How much time does it take to do things? What is the flow like through our system?" Here, I would probably add a couple of dimensions. So chatting with some of the early authors to say trust. Not to say trust wasn't important before, but now it's very, very front of mind. Right? Before you build your code, if the compile comes back, you're fine. And that's the way it is. LLMs are non-deterministic. Right now, we can't just put in a command and guess something back and accept it. We really need to evaluate it, so, "Are we seeing hallucinations? What's the reliability? Does it meet the style that we would typically write? And if it doesn't meet, is that fine?" So it depends on... Prescriptive. You got to make sure you're using it fit for purpose. Right?

中文翻译:

DORA 4 有四个关键指标。两个速度指标：部署频率和交付周期（lead time，即从代码提交到部署的时间）。两个稳定性指标：平均修复时间（MTTR）和变更失败率。如果这些用于评估流水线的速度和总体性能，那很好。但如果你想用它们来理解……因为这其中隐含了反馈循环，对吧，以前你通常是从客户那里获得反馈。但现在当我们使用 AI 时，我们不能盲目使用这些，因为反馈循环出现得早得多，甚至不只是在本地构建和测试阶段。我们在整个过程中都有反馈循环，甚至有时在流水线的中间阶段，我们希望以以前不那么常用的方式利用这些循环。我不是说以前不可能，只是我们以前没把重点放在那儿。

(00:15:53):

所以那些是规范性指标。而当我们谈论 SPACE 时，它是一个框架。它不告诉你具体衡量什么指标。所以我常说，有时人们会感到沮丧，因为我没告诉他们具体测什么。但现在，我认为这正是它的力量所在。我们发现 SPACE 在 AI 等新兴背景下应用得相当好，因为我们仍然需要关注……SPACE 是一个缩写。我们仍然要看满意度（Satisfaction）。我们要看绩效（Performance），即产出是什么。我们要看活动（Activity），是的，在某些方面，代码行数和 PR 数量，或者告警数量、活动计数是有用的。C 代表沟通与协作（Communication and Collaboration），这也非常重要，因为它关乎系统之间以及人与人之间的沟通。“有多少工作被分流给了聊天机器人，而不是去请教团队里的资深工程师？”多并不总是更好，少也不一定，这取决于具体情况。

(00:16:50):

最后是效率与心流（Efficiency and Flow），“人们能进入心流吗？做事情需要多长时间？系统中的流转情况如何？”在这里，我可能会增加几个维度。比如和一些早期作者聊天时提到的“信任”。并不是说以前信任不重要，但现在它变得非常紧迫。以前你写代码，只要编译通过，通常就没问题。但 LLM 是非确定性的。现在我们不能只输入一个命令，拿到结果就接受。我们真的需要评估它：“是否存在幻觉？可靠性如何？它符合我们的风格吗？如果不符合，没关系吗？”所以这取决于……规范性。你必须确保你的衡量方式是“因地制宜”的。

[00:17:38] Lenny Rachitsky

English:

We're going to get to your current thinking on the best way to do this stuff. You have a book coming out that explains how to do this well, so we're going to get to that. One thing I wanted to highlight in our last chat that we had, you highlighted that one of the biggest issues we're going to probably have with AI is trust, understanding and learning how much to trust the code that it generates, and also how much... you said this, two and a half years ago, that so much of the time is now going to be spent reviewing code versus writing code. That's exactly what I'm hearing.

中文翻译:

我们稍后会聊到你目前关于如何做好这些事情的最新想法。你即将出版一本解释如何做好这些的新书，我们会谈到那个。我想强调一下我们在上次聊天中提到的：你指出 AI 面临的最大问题之一可能是信任，即理解并学会该在多大程度上信任它生成的代码。而且你在两年半前就说过，现在大量的时间将花在审查代码而不是编写代码上。这正是我现在听到的情况。

[00:18:10] Nicole Forsgren

English:

I think it'll be interesting to see how that impacts the way we structure work moving forward. We were talking about flow state and cognitive load. Now that our attention has to focus on things at certain times and it's broken up from how we used to do it, I think there's some real opportunity there to, not just rethink workflows, but rethink how we structure our days and how we structure our work.

中文翻译:

我认为观察这将如何影响我们未来的工作结构会很有趣。我们刚才谈到了心流状态和认知负荷。既然我们的注意力必须在特定时间集中在某些事情上，而且这种集中方式与以往不同，我认为这不仅是重新思考工作流的机会，也是重新思考我们如何安排每一天和工作结构的机会。

[00:18:31] Lenny Rachitsky

English:

Can you say more about that? Just what is that? What are you thinking will be happening? Where do you think things go? What are you seeing working?

中文翻译:

你能多谈谈吗？具体是指什么？你认为会发生什么？事情会朝什么方向发展？你看到哪些做法是有效的？

[00:18:37] Nicole Forsgren

English:

This is purely speculative. But for example, Gloria Mark has done some really good work on attention and deep work, and humans can get about four hours of good deep work a day. That's about it.

中文翻译:

这纯属推测。但例如，Gloria Mark 在注意力和深度工作方面做了一些非常好的研究，人类每天大约只能进行四小时的高质量深度工作。仅此而已。

[00:18:52] Lenny Rachitsky

English:

Yeah,. I feel that.

中文翻译:

是的，我深有体会。

[00:18:54] Nicole Forsgren

English:

That's kind of the upper limit-ish for the most part, and I'm sure people are going to be like, "Well, I am superhuman and I can do-

中文翻译:

这基本上是大多数人的上限，我敢肯定有人会说：“但我超凡脱俗，我可以做——”

[00:18:59] Lenny Rachitsky

English:

What if you take 20 grams of creatine?

中文翻译:

那要是吃 20 克肌酸呢？

[00:19:01] Nicole Forsgren

English:

Right. What if we microdose?

中文翻译:

对，或者微量用药（microdose）呢?

[00:19:02] Lenny Rachitsky

English:

Yeah, exact;y.

中文翻译:

没错。

[00:19:06] Nicole Forsgren

English:

Yeah. So in the context of knowing we have about four hours of good deep work... I'm sure many of us have probably hit this, right? We have good periods. Maybe it's morning, maybe it's afternoon for folks. And then you hit a time where you're like, "I'm going to clean up my inbox because that is all I can do right now. I can be functional, but I'm not going to come up with my best innovative, problem solving, authoring, code writing work." A lot of times, the way to do that and to get into it is to have these long chunks to get into flow and to get that deep work. Usually, I'm [inaudible 00:19:43] two hours-ish. An hour can be tricky because it could take time to get into that state. Okay. Well, when we think about what it used to be like, back in the old days, three years ago, three and a half years ago, we could block off four hours of time and we could probably get two or three hours of really good work done. Because we were just focused, right? There were no interruptions, minimal interruptions.

(00:20:05):

Now, the nature of writing code and systems itself is interrupt driven or full of interruptions, at least, because you start something and then it interjects. So how do we think about that? Does that mean that a four-hour word block is still useful? Probably. But does that mean that now we can also make a 45-minute work block useful? Because getting into the flow is actually kind of handed off, at least, in part to the machine or the machine can help us get back into the flow by reminding us of context and generating diagrams of the system and all the things. So I think that's a really, really interesting area that's just ripe for questions and opportunity. And please, folks, do this research and come back to me because... It might not make my list, but it's such a great question.

中文翻译:

是的。所以既然知道我们大约只有四小时的高质量深度工作时间……我相信我们很多人都经历过，对吧？我们有高效时段，可能是上午，也可能是下午。然后你会到一个点，觉得：“我要去清理收件箱了，因为我现在只能干这个。我还能维持基本功能，但我无法进行最棒的创新、解决问题、创作或写代码。”很多时候，进入这种状态的方法是拥有长块的时间来进入心流。通常需要两小时左右。一小时可能有点悬，因为进入状态需要时间。好，回想三年前、三年前半的“旧时光”，我们可以划出四小时，大概能完成两三小时的高质量工作。因为我们很专注，对吧？没有干扰，或者干扰极少。

(00:20:05):

现在，编写代码和系统本身的性质变成了中断驱动的，或者至少充满了中断，因为你刚开始做某事，AI 就插进来了。那我们该怎么看？这是否意味着四小时的工作块仍然有用？可能有用。但这是否意味着现在我们也可以

让 45 分钟的工作块变得高效？因为进入心流的过程实际上部分移交给了机器，或者机器可以通过提醒上下文、生成系统图表等方式帮助我们快速找回状态。所以我认为这是一个非常有趣的领域，充满了问题和机会。请大家去做这方面的研究并告诉我结果，因为这虽然没进我的书，但确实是个好问题。

[00:20:52] Lenny Rachitsky

English:

That is so interesting. Essentially, every engineer is turning into an EM, engineering manager, coordinating all of these junior AI engineers. So your point is even if you have a 30-hour block, you can get deep into code, but you can unblock all these AI engineers that are running off doing tasks. Plus, your point is they remind you of just like, "Here's where you left off. Okay. You can just jump into this code, maybe make some tweaks."

中文翻译:

这太有意思了。本质上，每个工程师都在变成一名工程经理（EM），协调所有这些初级 AI 工程师。所以你的观点是，即使你只有 30 分钟的时间块，你也可以深入代码，或者去“解救”那些正在执行任务但卡住了的 AI 工程师。此外，你的观点是它们会提醒你：“这是你刚才停下的地方。好，你可以直接跳进这段代码，做些微调。”

[00:21:17] Nicole Forsgren

English:

Yeah.

中文翻译:

是的。

[00:21:18] Lenny Rachitsky

English:

So interesting. Let me zoom out a little bit and... Before we get into your framework for how to approach developer experience, the latest thinking you've got, beyond just obviously engineers doing more is great, what's your best pitch for why companies should really, really focus on developer experience?

中文翻译:

很有趣。让我稍微拉远一点……在我们深入探讨你关于如何处理开发者体验的框架和最新想法之前，除了“工程师能做更多事显然很棒”之外，你认为公司为什么要真正重视开发者体验？你最好的理由是什么？

[00:21:37] Nicole Forsgren

English:

I hate to say return of investment, but the business value is... the opportunity here is huge. In general, we write software for fun and for hobbies, but we also have software because it meets a business need. It helps us with market share, it helps us attract and retain customers, it helps us do all of these things. And I think DevEx is important because it enables all of that software creation, it enables all of that problem solving. It enables the super rapid experimentation with customers that... Before, you'd need a while for a

prototype and maybe a little bit longer to actually flight it through an A/B test on a production system. You can do it in hours, right now.

中文翻译:

我不想只谈投资回报率（ROI），但商业价值……这里的机会是巨大的。通常我们写软件是为了乐趣和爱好，但我们拥有软件也是因为它满足了商业需求。它帮助我们获得市场份额，帮助我们吸引和留住客户。我认为 DevEx 很重要，因为它赋能了所有的软件创造和问题解决。它实现了与客户的超快速实验。以前，你需要很长时间做一个原型，再花更长时间在生产系统上进行 A/B 测试。而现在，你可以在几小时内完成。

[00:22:21] Lenny Rachitsky

English:

Maybe the opposite end of the spectrum, getting very tactical, before we get into the larger framework, what's just one thing that you think an eng team, a product team can do this week, next week to help their developer experience maybe get more done?

中文翻译:

也许我们可以从另一个极端谈起，谈谈战术层面。在进入大框架之前，你认为工程团队或产品团队在本周或下周可以做哪一件事，来改善开发者体验并可能完成更多工作?

[00:22:35] Nicole Forsgren

English:

Honestly, I think the best thing you can do is go talk to people and listen. I love that the audience of this podcast is primarily PMs because they tend to be really good at this. And I would say start with listening and not with tools and automation. So many times companies are like, "Well, I'm just going to build this tool," or, "I'm going to build this thing." Often you build a thing that you yourself have had a challenge with or that is easy to do, easy to automate. And if you just go talk to people and ask the developers like, "Think of yesterday, what did you do yesterday? Walk me through it. What were the points that were just delightful? What were the points that were really difficult? Where did you get frustrated? Where did you get slowed down? Where was there friction?" If you go talk to a handful of people, a lot of times, you can surface a handful of things that are relatively low lift and still have impact or you can identify a process that's unnecessarily complex and slow.

中文翻译:

老实说，我认为你能做的最好的事情就是去和人们交谈并倾听。我很喜欢这个播客的听众主要是产品经理（PM），因为他们通常很擅长这个。我想说，要从倾听开始，而不是从工具和自动化开始。很多时候，公司会想：“好吧，我要建这个工具”或者“我要做这个东西”。通常你做的是你自己遇到过挑战的东西，或者是容易做、容易自动化的东西。但如果你去和开发者聊聊，问他们：“想想昨天，你昨天都做了什么？带我过一遍。哪些环节让你觉得很顺畅？哪些环节非常困难？你在哪里感到沮丧？哪里让你慢了下来？哪里有摩擦？”如果你去和几个人聊聊，通常能发现一些成本相对较低但影响很大的事情，或者你能识别出一个不必要且复杂的缓慢流程。

[00:23:36] Lenny Rachitsky

English:

So the listening to, I hear, almost is you want to help your teams move faster and be happier eng teams. Your advice is just, "Before you do anything, just go ask them what is bothering you."

中文翻译:

所以我听到的是，如果你想让团队跑得更快、更快乐，你的建议就是：“在做任何事之前，先去问问他们什么在困扰着他们。”

[00:23:46] Nicole Forsgren

English:

Go ask them, yeah. And trust me, most developers are going to be more than happy to tell you what's broken and what's bad. I'll say, there was one company that I had worked with. I remember they had a process that was really difficult and it was on an old mainframe system, and they were going to have to replat the whole thing and so they never went to work on it or talk about it. Everyone hated it because it was this huge delay. I mean, all they had to do was change a process. Sometimes all you have to do is change a process. And they changed it so that instead of... I think someone had to print it out and walk it down three or four flights, and they get approval. And then someone else had to walk it back up, and so it was just that interim. They didn't replat anything. They didn't redesign anything major. They just sent an email.

中文翻译:

去问他们，没错。相信我，大多数开发者会非常乐意告诉你哪里坏了、哪里不好。我曾合作过一家公司，他们有一个非常困难的流程，是在旧的大型机系统上。他们本以为必须重构（replat）整个系统，所以一直没去碰它。每个人都讨厌它，因为它造成了巨大的延迟。其实，他们只需要改变一个流程。有时你只需要改变流程。他们改了流程，以前是有人必须打印出来，走下三四层楼去拿审批，然后再有人走上来。他们没有重构任何东西，没有做任何重大重新设计，只是改成了发邮件。

[00:24:31] Lenny Rachitsky

English:

Let me push on that and... I'm curious just what are the most common things people do. If you're just starting on, "Okay, we need to focus on engineering experience," what do you find are the most... two or three most common improvements companies need to make?

中文翻译:

让我追问一下……我很想知道人们最常做的事情是什么。如果你刚开始想：“好，我们需要关注工程体验”，你发现公司最需要做的两三个最常见的改进是什么？

[00:24:45] Nicole Forsgren

English:

I'll say, I'll kind of echo that process, there's almost always a process that can be improved and that can be improved without a lot of engineering lift or a lot of engineering headcount. Most large companies, in particular, have something that is several, several steps. It's the way it is because it's the way it is, but that's no longer the way it is. And even small companies sometimes is just a little too YOLO, and you don't know what it is and you're kind of chasing everyone around. So if you can create a very lightweight

process, that can also be helpful. That can be one of the best places to start, especially if you have limited exposure to the whole rest of the org. Sometimes just a team process can help.

(00:25:28):

I will say from a business leader's standpoint, a lot of what you can do is provide structure and support for this organizational change. Communicate what you're doing, communicate what the priorities are, communicate why this is important, to celebrate wins. Because if folks try to do this, just like a one-off side fully-isolated project, it's really challenging to get some good momentum, to get people to care, and to get them stay involved. Because it feels like it's just another internal project that isn't going to matter or that isn't going to get celebrated, but it has these huge upside potential returns for the business.

中文翻译:

我会呼应刚才提到的“流程”。几乎总有一个流程是可以改进的，而且改进它不需要投入大量的工程力量或人力。特别是大多数大公司，总有一些环节需要好几个步骤。它们之所以存在是因为“一直以来都是这样”，但现在情况变了。甚至小公司有时也太随性（YOLO）了，你搞不清楚状况，只能到处追着人问。所以，如果你能创建一个非常轻量级的流程，那也会很有帮助。这是一个很好的切入点，特别是当你对整个组织的其他部分了解有限时。有时仅仅是一个团队内部的流程就能起作用。

(00:25:28):

从业务领导者的角度来看，你能做的很多事情是为这种组织变革提供结构和支持。沟通你正在做的事情，沟通优先级，沟通为什么这很重要，并庆祝胜利。因为如果人们只是把它当作一个孤立的一次性副项目，很难获得良好的势头，很难让人们关心并持续参与。因为它感觉就像又一个无关紧要、不会被表彰的内部项目，但实际上它对业务有巨大的潜在回报。

[00:26:10] Lenny Rachitsky

English:

It's interesting, what I'm hearing here is nothing about tools or technologies. It's not like move to this cloud, it's not like install this new deployment system, it's processes and people and org and morale.

中文翻译:

很有意思，我在这里听到的完全没有关于工具或技术的内容。不是说“迁移到这个云端”，也不是“安装这个新的部署系统”，而是流程、人、组织和士气。

[00:26:24] Nicole Forsgren

English:

Yeah. Now, there will be technical pieces that are very important, especially now with AI, where we're rethinking how build and test systems work. We're rethinking feedback to users so that it's very, very customized in terms of what is shared and when it is shared. There are a lot of technical pieces that are involved, but that's not the only thing. It's necessary but not sufficient, and that doesn't have to be the place that you start.

中文翻译:

是的。当然，技术部分也非常重要，特别是现在有了 AI，我们正在重新思考构建和测试系统的工作方式。我们正在重新思考给用户的反馈，使其在分享内容和时机上非常定制化。这涉及很多技术环节，但那不是唯一的事情。技术是必要的，但不是充分的，而且它不一定是你的起点。

[00:26:50] Lenny Rachitsky

English:

I have a hard question I want to ask you that I thought of as you were talking. I feel like this is the question that most founders and heads think about. And the question is just like, how do I know if my eng team is moving fast enough, if they can move faster, if they're just not performing as well as they can? What are just maybe smells, signs that tell you, "Yeah, my team should be moving faster," versus, "This is just the way it works. This is as fast as they can move"?

中文翻译:

在你说话时，我想到了一个很难的问题想问你。我觉得这是大多数创始人和负责人都会思考的问题：我怎么知道我的工程团队动作是否够快？他们还能不能更快？或者他们是否只是没有发挥出应有的水平？有哪些“异味”（smells）或迹象能告诉你“是的，我的团队应该跑得更快”，而不是“这就是它的运作方式，这就是他们能达到的最快速度”？

[00:27:16] Nicole Forsgren

English:

Most teams can move faster, right? Also, given what we know about cognitive load, not all speed gains are necessarily good. Or the upside is going to be kind of limited once you hit kind of a certain point, and most people are not even near that point. I don't know a single team, frankly. But how do you know? You know if you're always hearing about bills breaking, flaky tests, overly long processes, if you have to request a new system or if you need to provision a new environment, or if it's really, really hard to switch tasks or switch projects. So if someone has an opportunity to go work in another part of an org and they don't for reasons that are unclear, and not political, and anyone says anything about the system, that's usually a pretty good smell that there's friction somewhere.

(00:28:20):

Because once you finally figure out your system and you're able to get work done, the switching costs can often be really, really high to go anywhere else. So sometimes people will do that. But I've worked with companies where switching orgs within the company, you had to basically pay the same tax as a new hire because the systems were so different and they were so full of friction, and it was so difficult to do so many things.

中文翻译:

大多数团队都可以跑得更快，对吧？此外，考虑到我们对认知负荷的了解，并非所有的速度提升都是好事。或者一旦达到某个点，收益就会变得有限，但大多数人甚至还没接近那个点。坦白说，我还没见过这样的团队。但你怎么知道呢？如果你总是听到构建失败、不稳定的测试（flaky tests）、过长的流程；如果你必须申请新系统或配置新环境非常麻烦；或者如果切换任务或项目非常困难。如果有人有机会去组织的另一个部门工作，但因为一些不明确且非政治的原因而没去，或者任何人对系统有怨言，这通常是一个很好的迹象，说明某处存在摩擦。

(00:28:20):

因为一旦你终于搞清楚了自己的系统并能开展工作，去其他地方的切换成本往往非常高。所以有时人们会选择留守。但我合作过一些公司，在公司内部调换部门，你基本上要像新员工入职一样重新交一遍“税”，因为系统差异太大，充满了摩擦，做很多事情都极其困难。

[00:28:49] Lenny Rachitsky

English:

I love the first part of your answer especially, which is you can always move faster. I think every founder is going to love hearing that. To your point though, there's diminishing returns over time?

中文翻译:

我特别喜欢你回答的第一部分，即“你总能跑得更快”。我想每个创始人都会喜欢听这句话。不过正如你所说，随着时间的推移，会有边际收益递减？

[00:28:58] Nicole Forsgren

English:

Yeah. And you don't know about the quality, right? So I think that's the other side is that you can always move faster, but faster for what? Are we making the right business decisions? And I think that's especially where PMs come in. We can ship trash faster every single day. We need strategy and really smart decisions to know what to ship, what to experiment with, what features we want to do in what order and what rollout. The strategy is the core piece, and then think about speeding that up. If we don't have the other pieces in place, I mean, garbage in, garbage out.

中文翻译:

是的。而且你不知道质量如何，对吧？所以另一面是，你总能跑得更快，但快是为了什么？我们是否做出了正确的商业决策？我认为这正是 PM 发挥作用的地方。我们可以每天更快地交付垃圾。我们需要战略和非常聪明的决策来确定交付什么、实验什么、以什么顺序开发功能以及如何发布。战略是核心，然后再考虑加速。如果其他环节没到位，那就是“垃圾进，垃圾出”。

[00:29:30] Lenny Rachitsky

English:

I want to follow that thread, but before I do that, just to mirror back what you shared. So signs that your team... There's a lot of low-hanging fruit to improve the productivity of your team as builds are always breaking. There's flaky tests are constantly incorrect, false positives. It's hard to context switch between different projects. You just hear people talking about the system, it's just really hard to work with. Is that roughly right?

中文翻译:

我想继续这个话题，但在那之前，先复述一下你分享的内容。所以，如果构建总是失败、不稳定的测试不断报错、在不同项目间切换上下文很困难、或者你听到人们抱怨系统很难用，那么提高团队生产力就有很多“低垂的果实”。大概是这样吗？

[00:29:52] Nicole Forsgren

English:

Yeah.

中文翻译:

是的。

[00:29:53] Lenny Rachitsky

English:

Cool, okay. So going back to the point you just made, there's a sense that AI is making teams so much faster because it's writing all this code for them. You're going to have all these asynchronous agents, engineers working for you. It feels like a core part of your message is that's just a one part of engineering work and there's so much more, including figuring out what to build... an alignment internally. Maybe just speak to just... There is a lot of opportunity to improve engineering performance productivity, but there's so many other elements that are not improved through AI?

中文翻译:

好。回到你刚才提到的观点：人们感觉 AI 让团队变快了很多，因为它帮他们写了所有代码。你会有所有这些异步智能体、工程师为你工作。感觉你的核心信息是，那只是工程工作的一部分，还有很多其他事情，包括弄清楚要构建什么、内部对齐等。也许你可以谈谈……虽然有很多提高工程绩效和生产力的机会，但还有很多其他元素是 AI 无法改进的?

[00:30:22] Nicole Forsgren

English:

Yes. Or could be in the future, right?

中文翻译:

是的。或者未来可能会改进，对吧?

[00:30:25] Lenny Rachitsky

English:

Mm-hmm.

中文翻译:

嗯。

[00:30:26] Nicole Forsgren

English:

I think there are a lot of ways that we can pull in AI tools to help us refine our strategy, refine our message, think about the experimentation methods or targets of experimentation, or think about our total addressable market, but we need to have that strategy and plan fairly well aligned or at least have two or three alternatives that you want to test. Because now, the engineering can go, or at least the prototyping especially, much, much faster. We can throw out prototypes. We can run any tests and experiments that are customer facing, assuming that we have the infrastructure in place, which allows us to learn and progress much faster before. In some places, it used to take months to get something through production to do A/B testing and get feedback. We can do this in a day or two, definitely under a week. But we want to make sure that we're building and testing the right things, "Are we partnering with the right... Do we have the data that we need?"

(00:31:24):

And I will say AI can actually be a pretty good partner there if you have a good conversation with it, and then also check with you experts, "What type of data should I be looking at? What type of instrumentation do I need? What type of analysis can I do?" Because then, you can also go to your data science team and say, "I'm planning on doing this. I'd like to..." Let's not just YOLO A/B tests because that can be... It's a shame to do a large test and end up disrupting users or disrupting customers, or breaking privacy or security protocols and also end up with data that's unusable because you just can't get the signal that you're looking for. But now, I'm also seeing people kind of accelerate that into a few days versus a few weeks. So they can start those key stakeholder discussions from a much more informed kind of filled out space.

中文翻译:

我认为有很多方法可以引入 AI 工具来帮助我们完善战略、完善信息、思考实验方法或目标，或者思考我们的总可寻址市场（TAM）。但我们需要让战略和计划相当一致，或者至少有两三个你想测试的备选方案。因为现在，工程实现（特别是原型设计）可以快得多。我们可以抛出原型，运行任何面向客户的测试和实验（假设基础设施到位），这让我们学习和进步的速度比以前快得多。在某些地方，以前通过生产环境进行 A/B 测试并获得反馈需要几个月，现在一两天就能搞定，肯定不到一周。但我们要确保我们构建和测试的是正确的东西，“我们是否与正确的伙伴合作？我们有需要的数据吗？”

(00:31:24):

我会说 AI 在这方面其实可以成为很好的伙伴，如果你能和它进行良好的对话，然后再咨询专家：“我应该看什么类型的数据？我需要什么样的埋点（instrumentation）？我可以做哪种分析？”然后你也可以去找你的数据科学团队说：“我计划做这个，我想……”我们不要只是随性地做 A/B 测试，因为那可能会……做一个大型测试结果却干扰了用户、破坏了隐私或安全协议，最后得到的数据还不可用，因为你找不到想要的信号，那太可惜了。但现在，我也看到人们将这个过程从几周缩短到了几天。因此，他们可以在更充分的信息支持下开始与关键利益相关者的讨论。

[00:32:17] Lenny Rachitsky (Ad)

English:

Today's episode is brought to you by Coda. I personally use Coda every single day to manage my podcast and also to manage my community. It's where I put the questions that I plan to ask every guest that's coming on the podcast, it's where I put my community resources, it's how I manage my workflows. Here's how Coda can help you. Imagine starting a project at work and your vision is clear, you know exactly who's what and where to find the data that you need to do your part. In fact, you don't have to waste time searching for anything because everything your team needs from project trackers and OKRs, the documents and spreadsheets lives in one tab all in Coda. With Coda's collaborative all-in-one workspace, you get the flexibility of docs, the structure of spreadsheets, the power of applications, and the intelligence of AI all in one easy-to-organize tab. Like I mentioned earlier, I use Coda every single day. And more than 50,000 teams trust Coda to keep them more aligned and focused. If you're a startup team looking to increase alignment and agility, Coda can help you move from planning to execution in record time. To try it for yourself, go to coda.io/lenny today and get six months free of the team plan for startups.

中文翻译:

今天的节目由 Coda 赞助。我个人每天都使用 Coda 来管理我的播客和社区。我把计划问每位嘉宾的问题放在那里，把社区资源放在那里，用它来管理我的工作流。Coda 是这样帮助你的：想象你在公司开始一个项目，愿景清晰，你确切知道谁负责什么，以及在哪里可以找到你需要的数据。事实上，你不需要浪费时间寻找任何东西，因为团队需要的一切——从项目跟踪器和 OKR 到文档和表格——都集中在 Coda 的一个标签页里。通过 Coda 的协作式全能工作空间，你可以在一个易于组织的标签页中获得文档的灵活性、表格的结构、应用程序的能力以及 AI 的智能。正如我之前提到的，我每天都用 Coda。超过 5 万个团队信任 Coda，用它来保持对齐和专

注。如果你是寻求提高对齐度和敏捷性的初创团队，Coda 可以帮助你在创纪录的时间内从计划转向执行。访问 coda.io/lenny 即可免费获得 6 个月的初创团队计划。

[00:33:33] Lenny Rachitsky

English:

I love that you work with a bunch of different companies and a bunch of different types of businesses. I think very few people get to see inside a lot of different places. What kind of gains are you just seeing in terms of increased productivity with AI? How big of a gain have you seen?

中文翻译:

我很喜欢你与这么多不同公司和不同业务类型合作这一点。我想很少有人能看到这么多不同公司的内部情况。关于 AI 带来的生产力提升，你观察到了什么样的收益？你见过的收益有多大？

[00:33:49] Nicole Forsgren

English:

I'd say it's real, and I would also say we don't have great measures for it yet. We're still trying to figure out what to measure and what that looks like. One of the best is going to be velocity, all the way through the system, how quickly can you get a feature or a product or something through the system so that you can then experiment a test, either from idea to final end or even kind of a feature and a piece through the system so we can test. That's really good. Now, that's also hard to tie back directly to a particular AI tool in the hands of a particular developer. But there are some other things that we can look at and we can see, and that I've seen is, again, this kind of rapid prototyping.

(00:34:36):

I hate lines of code, but I'm going to use the lines of code. We do see... I know I worked with some folks who had kind of a whole set of companies they were looking at, and they found that AI was generating significantly more code for the people who were using it regularly. But then, they also found that for folks who were regular users of AI coding environments, AI ADEs, the tool kind of gave them more code. And then the engineers themselves, the increase was double what the coding agent had given them. So one, I'd say, probably it's kind of a secondary or knock on or just a smell is it can unblock you. It can speed up the work that you would already do. I know sometimes when I work, the first few minutes, it's hard for me to start. But once I get started, I'm there. So they're really good at unblocking and unlocking that.

中文翻译:

我会说这种提升是真实的，但我也要说我们目前还没有很好的衡量标准。我们仍在摸索该衡量什么以及它看起来是什么样的。其中最好的指标之一是全系统的速度（velocity）：你能多快地让一个功能或产品通过系统，以便进行实验测试？无论是从想法到最终结束，还是仅仅是一个功能片段。这非常好。当然，这也很难直接归功于某个开发者手中的某个特定 AI 工具。但我们还可以看其他一些东西，我看到的是，再次强调，这种快速原型设计。

(00:34:36):

我讨厌代码行数，但我还是要用它举例。我们确实看到……我知道我合作过的一些人观察了一组公司，他们发现对于经常使用 AI 的人，AI 生成的代码量显著增加。但随后他们还发现，对于 AI 编码环境（AI IDEs）的常客，工具给了他们更多代码，而工程师自己编写的代码增量竟然是 AI 智能体给他们的两倍。所以，我认为这可能是一种次生效应或迹象：它能帮你“疏通”。它能加速你本来就要做的工作。我知道有时我工作时，头几分钟很难开始。但一旦开始，我就进入状态了。AI 非常擅长打破这种僵局并开启工作。

[00:35:32] Lenny Rachitsky

English:

Something I've seen people on Twitter sharing is how good OpenAI Codex, especially, is at finding really gnarly bugs. And I think it was Karpathy that shared it. He was so stuck on a bug and, no AI tool could figure it out. And then the latest version of Codex spent an hour or something, looking into it, and found it for him.

中文翻译:

我在 Twitter 上看到人们分享，尤其是 OpenAI Codex 在寻找那些非常棘手的 bug 方面表现出色。我想是 Karpathy 分享的，他被一个 bug 困住了，没有任何 AI 工具能解决。然后最新版本的 Codex 花了一个小时左右深入研究，帮他找到了问题。

[00:35:51] Nicole Forsgren

English:

Yeah. I'm hearing incredible things like that, right? Well, and even also writing unit tests and spinning up unit tests, and creating documentation and cleaning up documentation because I know now people are like, "Oh. Well, we have agents. I don't need to read the docs because there's the code there." It turns out, agents rely on good data because it's all about how they've been trained or how they've been grounded. And better data gives you better outcomes, and some of that data includes documentation and comments. The better documentation and the better comments you have, the better performance you're going to get out of your AI tools.

中文翻译:

是的。我也听到了很多类似不可思议的事情。甚至还包括编写单元测试、快速生成单元测试、创建和清理文档。因为我知道现在人们会说：“哦，既然有了智能体，我不需要读文档了，因为代码就在那儿。”事实证明，智能体依赖于高质量的数据，因为这关乎它们是如何被训练或“锚定”（grounded）的。更好的数据会带来更好的结果，而这些数据中就包括文档和注释。你的文档和注释越好，你从 AI 工具中获得的性能就越好。

[00:36:29] Lenny Rachitsky

English:

And AI can help you write that documentation. I've been working with Devin a little bit, and it's really good at that stuff.

中文翻译:

而且 AI 还可以帮你写那些文档。我最近试了一下 Devin，它在这些方面非常出色。

[00:36:34] Nicole Forsgren

English:

Yeah.

中文翻译:

是的。

[00:36:36] Lenny Rachitsky

English:

Okay. Let's talk about this framework, this book. So you're publishing a book called Frictionless, which sounds like a dream, "How do you create a dev team that's frictionless?" It's called Frictionless: 7 Steps to Remove Barriers, Unlock Value, and Outpace Your Competition in the Age of AI. There's a seven-step process to this. Walk us through this and maybe give us just context on this book, who it's meant for, what problem it solves, and then the seven steps.

中文翻译:

好。让我们聊聊这个框架和这本书。你要出版一本叫《Frictionless》（无摩擦）的书，这听起来像个梦想，“如何创建一个无摩擦的开发团队？”全名是《无摩擦：在 AI 时代消除障碍、解锁价值并超越竞争对手的 7 个步骤》。这里有一个七步流程。请带我们了解一下，并介绍一下这本书的背景：它是为谁写的，解决了什么问题，以及那七个步骤是什么。

[00:37:00] Nicole Forsgren

English:

I will say, I also wrote this with Abi Noda who has just... of DX. He has incredible experience in the space. He's worked with hundreds of companies and so it was kind of nice bouncing ideas off of him. Also, thanks to all of the engineering leads and DevEx leads, and CTOs, and engineers that we talked to to make sure that our smells were right. So who is this book for-

中文翻译:

我想说，这本书是我和 Abi Noda 合著的，他是 DX 的创始人。他在这个领域有惊人的经验，曾与数百家公司合作，所以和他交流想法非常棒。同时也要感谢我们访谈过的所有工程主管、DevEx 主管、CTO 和工程师，确保我们发现的“异味”是准确的。那么，这本书是给谁看的——

[00:37:26] Lenny Rachitsky

English:

Let me take a tangent on Abi, and DX, since you mentioned him. This is super interesting, and I think it connects so directly with this conversation. Abi started this company called DX, which is such a great name for a company around developer experience. They just sold the company for a billion dollars to Atlassian. It's a very high multiple on their ARR. It, to me, shows exactly why this conversation is so valuable, just how much value companies are putting into improving developer experience. Atlassian would spend a billion dollars on this. It's an early stage-ish startup. It was doing really well and people loved it, but it was like early stage-ish, a billion dollars. And the idea is they have all these companies working using Jira and all their products. They're all trying to figure out how do we measure productivity. It's worth a lot of money to them. And I know you were an early advisor to them too, so-

中文翻译:

既然你提到了 Abi 和 DX，让我插一句。这非常有趣，而且我认为它与这次对话直接相关。Abi 创办了这家叫 DX 的公司，对于一家围绕开发者体验的公司来说，这名字起得太棒了。他们刚刚以 10 亿美元的价格将公司卖给了 Atlassian。相对于他们的年度经常性收入（ARR），这是一个非常高的倍数。对我来说，这恰恰说明了为什么这次对话如此有价值，说明了公司在改善开发者体验上投入了多少价值。Atlassian 愿意为此花 10 亿美元。这是一家处于早期阶段的初创公司，虽然表现很好且深受喜爱，但 10 亿美元确实惊人。其逻辑是，Atlassian 有这

么多公司在使用 Jira 和他们的产品，这些公司都在试图弄清楚如何衡量生产力。这对他们来说非常值钱。我知道你也是他们的早期顾问，所以——

[00:38:15] Nicole Forsgren

English:

Yeah. Well, I think it also shows us how much value you can get out of this. There's so much low-hanging fruit, there's so much unlocked potential, and it's hard to know where to start a lot of times even in... I've been at large companies that have a lot of expertise and a lot of really, really smart people. But if you haven't kind of been in this space and thinking about it this way, it's hard to know where to start or it's easy to make simple mistakes up front that mean you kind of need to start over later. So I guess it also brings us back to, "Who is this book for?" It's for anyone that cares about DevEx, so definitely technology leaders, anyone who's trying to kick off a DevEx program, or is working on a DevEx DevEx improvement program. I think it's particularly relevant for PMs because if you're PMing something that involves software building and creating software, improving DevEx will only help your team. And also, you have key skills and insights and instincts that are so important to DevEx that many times, I will say, I've seen engineering teams just miss.

中文翻译:

是的。我认为这也向我们展示了你能从中获得多少价值。这里有太多的“低垂果实”，有太多未被释放的潜力。很多时候，即使是在我待过的大公司，那里有很多专家和聪明人，也很难知道从哪里开始。或者如果你没有在这个领域以这种方式思考过，很容易在前期犯一些简单的错误，导致以后不得不推倒重来。所以，回到“这本书是给谁看的”：它是给任何关心 DevEx 的人看的，绝对包括技术领导者、任何试图启动 DevEx 项目或正在进行 DevEx 改进计划的人。我认为它对 PM 尤其相关，因为如果你作为 PM 负责涉及软件构建的产品，改善 DevEx 只会帮助你的团队。而且，你拥有对 DevEx 至关重要的关键技能、洞察力和直觉，而这些往往是工程团队会忽略的。

[00:39:31] Lenny Rachitsky

English:

Okay. What's the framework? What are the steps? Where do people start?

中文翻译:

好。框架是什么？步骤有哪些？人们从哪里开始？

[00:39:35] Nicole Forsgren

English:

The book goes through a seven-step process, and then also kind of provides some key kind of principles at the end. Step one is to start the journey. So assuming you're kicking off, you can start the journey. And this involves what we have already talked about. Go talk to people, have a listening tour, synthesize what you learn, visualize the workflow and tools, get a handle on what the current state is. Step two is to get a quick win. So start small, get a quick win, pick the right projects, share out what you've done. Step three is using data to optimize the work. So establish some of your data foundation, find the data that's there, start collecting new data, use some surveys for some really fast insights and may include example surveys. Step four then is to decide strategy and priority. Once you have some data, then you need to

know of all the things that are potentially broken. And if you've already gotten your quick win of all the things that are left, "What should I do next?" So we walk through some evaluation frameworks there.

(00:40:43):

Step five is to sell your strategy. Once you've decided, now you have to kind of convince everyone else. So now you want to get feedback, you want to share why this is the right strategy right now. Step six is to drive change at your scale. So here, we address folks that have local scope of control. If you're starting on just a dev team, you want to do it yourself, kind of grassroots effort or global scope of control. If you're the VP of developer experience or something, there are some things that you can leverage for a top down, and then how do you drive change when you're kind of somewhere in the middle, because you can leverage both types of strategies. And then step seven is to evaluate your progress and show value, and then kind of loop back around.

(00:41:27):

I will say that we wrote this so that you could kind of jump into any step wherever you are right now. If you're kicking off a team or an initiative, you'll probably want to start at step one. You should definitely start at step one. If you're joining an existing initiative, you could jump into picking the priority or implementing the changes. So those are the seven steps. There's a seven steps, there are a few practices that we also recommend. So thinking about resourcing it, change management, making technology sustainable, and then also bringing a PM lens to this, "How can we think about developer experience as a product, and how do we think about the metrics that we have as a product?"

中文翻译:

书中介绍了一个七步流程，并在最后提供了一些关键原则。第一步：开启旅程。假设你正在启动，这涉及我们之前谈到的：去和人们交谈，进行“倾听之旅”，综合你学到的东西，将工作流和工具可视化，掌握现状。第二步：获得快速胜利（Quick Win）。从小处着手，选择正确的项目，分享你的成果。第三步：利用数据优化工作。建立数据基础，寻找现有数据，开始收集新数据，使用调查问卷获取快速洞察（书中包含示例问卷）。第四步：确定战略和优先级。有了数据后，你需要从所有潜在的问题中选出下一步该做什么。我们在这里介绍了一些评估框架。

(00:40:43):

第五步：推销你的战略。一旦决定了，你就得说服其他人。获取反馈，分享为什么这是目前的正确战略。第六步：在你的规模上推动变革。这里我们针对不同控制范围的人：如果你只是在一个开发团队，你想通过草根努力自己动手；或者你有全局控制权，比如你是 DevEx 副总裁，你可以利用自上而下的手段；以及如果你处于中间位置，如何结合两种策略推动变革。第七步：评估进度并展示价值，然后循环往复。

(00:41:27):

我想说，我们写这本书是为了让你无论处于什么阶段都能切入。如果你正在启动一个团队或倡议，你可能想从第一步开始。如果你是加入一个现有的倡议，你可以直接跳到确定优先级或实施变更。这就是那七个步骤。除了这七步，我们还推荐了一些实践：考虑资源配置、变更管理、使技术可持续，以及引入 PM 视角——“如何将开发者体验视为一个产品，以及如何将我们的指标视为一个产品？”

[00:42:13] Lenny Rachitsky

English:

Awesome, okay. I have questions. Point people to the book real quick. What's the URL? How do they get it? When does it come out?

中文翻译:

太棒了。我有几个问题。先快点告诉大家书的信息。网址是什么？怎么买？什么时候出版？

[00:42:18] Nicole Forsgren

English:

Yeah, developerexperiencebook.com. Right now, you can sign up for the mailing list. We'll let you know when it's out on pre-order, and we'll also be sharing pieces of the workbook. So we've got almost a hundred page workbook that goes along with the book, and then it should be out by end of year.

中文翻译:

网址是 developerexperiencebook.com。现在你可以加入邮件列表，预售开始时我们会通知你。我们还会分享工作手册的部分内容，这本书配有一份近百页的工作手册。预计年底出版。

[00:42:36] Lenny Rachitsky

English:

Okay. So one piece of this is just this term developer experience feels very intentional in that it's not developer productivity, developer work. It's how do we make developer experiences better at our company, which includes they get more done, but also they're happier and things like that. So I think that's an important element of this, right?

中文翻译:

好。其中一点是，“开发者体验”这个词感觉是非常刻意选择的，它不是“开发者生产力”或“开发者工作”。它是关于“我们如何让公司的开发者体验变得更好”，这包括让他们完成更多工作，但也包括让他们更快乐等等。我认为这是其中的一个重要元素，对吧？

[00:42:55] Nicole Forsgren

English:

Yeah, absolutely.

中文翻译:

是的，绝对是。

[00:42:55] Lenny Rachitsky

English:

Okay.

中文翻译:

好。

[00:42:56] Nicole Forsgren

English:

Because, again, it's not just about productivity. We talked about this from the frame and the lens of, "We need to be building the right thing." And you want to be productive, but you also want to be thinking about... and this is what engineers are also just really incredibly good at, give them a problem and don't tell them how to solve it, and then they can solve it better. They have the freedom, they have the innovation, they have the creativity so that they can solve this problem. If it's only about productivity, then it's just lines of code or number PRs or whatever. But we really want to talk about value and how do we unlock value, and how do we get value faster. And that involves, yes, making them more productive and removing friction because then, they have the flow and the cognitive load and the things that we kind of talked about.

中文翻译:

因为，再次强调，这不仅仅关乎生产力。我们是从“我们需要构建正确的东西”这个视角来讨论的。你希望高效，但你也想思考……而这正是工程师们非常擅长的：给他们一个问题，不要告诉他们怎么解决，他们能解决得更好。他们拥有自由、创新和创造力来解决问题。如果只谈生产力，那就只是代码行数或 PR 数量之类的。但我们真正想谈论的是价值，以及我们如何解锁价值、如何更快地获得价值。这确实涉及提高他们的生产力和消除摩擦，因为这样他们才能拥有心流、降低认知负荷，也就是我们谈到的那些东西。

[00:43:41] Lenny Rachitsky

English:

Awesome, okay. And then say someone wants to start this team, what does it usually look like. At Airbnb, I remember this team forming. It was just like an engineer or two, getting it started and taking charge. What do you recommend as the pilot team, and then what does it look like as it grows?

中文翻译:

太棒了。假设有人想组建这样一个团队，通常是什么样的？在 Airbnb，我记得这个团队形成时，只有一两个工程师发起并负责。你推荐的试点团队（pilot team）是什么样的？随着它的成长，又会变成什么样？

[00:43:57] Nicole Forsgren

English:

There are a few ways to do this, right? So if you're doing it yourself, you could do it with a couple of engineers, maybe a PM or a PGM or a TPM to kind of help communicate. Because really, comms plans are just so important here. On a small scale, what we want to do is look for those quick wins, look for things that you can do at small scale. Some folks call them things like paper cuts. There small things that you can do to help people see the value and feel the benefit themselves, "How can a developer's work get better? How can their day-to-day work get better? Kind of build momentum from there?" If you're working from a top-down structure and you have the remit, you still want some quick wins, but those quick wins can look a little more global in scale because you have the infrastructure or the backing to make different types of changes that aren't only local.

(00:44:56):

So an example of a small local change could be just cleaning up your tests, your test suites. Any team could do that, any team could do that. At more global scale, it might be changing organization-wide process that is just overly cumbersome or throwing some resourcing into cleaning up the provisioning environment.

中文翻译:

有几种方法。如果你是自己动手，可以找几个工程师，也许再加一个 PM 或项目经理（PGM/TPM）来协助沟通。因为在这里，沟通计划真的非常重要。在小规模阶段，我们要寻找那些“快速胜利”，寻找小规模就能做成的事。有人称之为“纸张割伤”（paper cuts，指微小但烦人的问题）。你可以做一些小事让人们看到价值并亲身感受到好处，“开发者的工作如何变得更好？他们的日常工作如何改善？并以此建立势头。”如果你是自上而下的结构且拥有职权，你仍然需要快速胜利，但这些胜利在规模上可以更具全局性，因为你有基础设施或支持去做一些不仅限于局部的改变。

(00:44:56):

例如，一个小型的局部改变可以是清理你的测试套件。任何团队都能做到。而在更全局的规模上，可能是改变一个过于繁琐的全组织流程，或者投入资源清理环境配置（provisioning）流程。

[00:45:15] Lenny Rachitsky

English:

What kind of impact have you seen from teams like this forming, on the engineering teams at their companies?

中文翻译:

你见过这类团队成立后，对公司的工程团队产生了什么样的影响？

[00:45:21] Nicole Forsgren

English:

I'll say I've seen a huge impact for smaller companies, hundreds of thousands of dollars for large companies or in the billions. Well, also, we need to learn how to communicate that, "What does the math look like?" Many times, we can look at saving time, we can look at saving costs, we can look at a lot of different things. We can look at speed to value as speed to market. We can look at risk reduction, but the gains really are there. I will mention that it tends to follow something like the J-curve. So you'll have a couple of quick wins and it'll look like a big win, and then you'll hit kind of a little divot where suddenly the really obvious projects, the low-hanging fruit are handled. So now, we need to do a little bit of work. We might need to build out a little bit more infrastructure. We might need to build out a little more telemetry, so that we can capture the things we want to capture. And then once we get that done, then we start to see those benefits really compound.

中文翻译:

我会说我看到了巨大的影响。对于小公司来说是数十万美元，对于大公司来说可能是数十亿。当然，我们也需要学会如何沟通这一点，“背后的账是怎么算的？”很多时候，我们可以看节省的时间、节省的成本，或者很多不同的东西。我们可以看“价值实现速度”即“上市速度”。我们可以看风险降低。收益是实实在在的。我要提一下，这通常遵循类似“J 曲线”的规律。你会先获得几个快速胜利，看起来像个大胜利，然后你会进入一个低谷期，因为那些显而易见的项目、低垂的果实都处理完了。现在我们需要做一些硬活，可能需要构建更多基础设施，构建更多遥测系统（telemetry）以便捕捉我们想要的数据。一旦完成这些，你就会看到收益开始产生真正的复利效应。

[00:46:16] Lenny Rachitsky

English:

So going back to that measurement number, what do you recommend? How do people find these numbers? Because I think that's so much of the power of this is like, "We saved a million dollars doing this." What do you look at to figure that out?

中文翻译:

回到衡量指标的问题，你有什么建议？人们如何找到这些数字？因为我认为这种力量很大一部分来自于：“我们做这个节省了一百万美元。”你通过看什么来计算出这个数字？

[00:46:28] Nicole Forsgren

English:

I think there are a few different things to keep in mind, like who is our key audience, and we usually have a few key audiences. We really want to be able to speak to developers because they're the ones that are going to be using the systems. They'll be partnering with you on either building them or at least providing feedback about what you're doing. So for them, we often want to frame this in terms of things they care about. So time savings. If something gets faster, they can save time. They don't spend time doing setup when they don't need to anymore, related to status reduced toil. So compliance and security are super important. Also, many times it requires several manual steps that... I don't say they're not value add. They're not value add from an individual human perspective. If we can automate as much as possible, that's great, and improved focus time.

(00:47:22):

That's from the developer side of you. Leadership often cares about... They care about those things, but they often care more about other things. So we could talk about usually costs in dollars, "Can we accelerate revenue? What does our time to value look like? What is our velocity? How quickly can we get feedback from customers?" And for folks and organizations that are in really competitive environments, that can be really compelling because it's all about speed. We could talk about saving money. Here, we can look at maybe quantifying savings. One example is test and build. If we can clean up a test and build suite to a developer, they really want to hear about time saved and more reliable systems. There's less toil because they don't have to keep re-running tests or kind of go clean up test suites.

(00:48:13):

From the business perspective, cleaning up a test in a build suite can be cloud cost savings because all of those tests are running somewhere on a cloud. And if they always fail or if it's just kind of a waste of spend, that can be useful, recovering some capacity. We can always talk about time and productivity gains, "How much equivalent developer time are we losing on things that are not necessarily value add?" And then sometimes we can correlate to business outcomes and correlate is usually the best we can do here, but there can be some pretty compelling correlations in terms of speeding up time to value and increase market share, for example.

中文翻译:

我认为有几点需要记住，比如谁是我们的核心受众。我们通常有几个核心受众。我们真的需要能够与开发者对话，因为他们是系统的使用者。他们会与你合作构建系统，或者至少提供反馈。所以对于他们，我们通常用他们关心的东西来表述：节省时间。如果某事变快了，他们就能省下时间。他们不再需要花时间做那些不必要的配置，这与减少琐事（toil）有关。合规和安全非常重要，但很多时候它们需要多个手动步骤，从个人角度看这些步骤并不增加价值。如果我们能尽可能自动化，那就太棒了，还能增加专注时间。

(00:47:22):

这是从开发者的角度。而领导层通常关心……他们也关心这些，但他们往往更关心其他事情。所以我们可以谈论金钱成本，“我们能加速收入增长吗？我们的价值实现时间（time to value）是怎样的？我们的速度如何？我们能多快从客户那里获得反馈？”对于处于激烈竞争环境中的组织，这非常有吸引力，因为一切都关乎速度。我们可以谈论省钱。在这里，我们可以量化节省的开支。一个例子是测试和构建。对于开发者，他们想听的是节省的时间和更可靠的系统，琐事变少了，不用反复运行测试或清理测试套件。

(00:48:13):

从业务角度看，清理测试和构建套件可以节省云成本，因为所有这些测试都在云端运行。如果它们总是失败，那就是在浪费钱。此外还可以恢复一些产能。我们总是可以谈论时间和生产力的提升，“我们在那些不一定增加价值的事情上损失了多少等效的开发者时间？”有时我们可以将其与业务结果联系起来，虽然通常只能做到“相关性”分析，但在加速价值实现和增加市场份额方面，这种相关性可以非常有说服力。

[00:48:54] Lenny Rachitsky

English:

Let me follow that thread and come back to this, what I think is the biggest question people have right now with AI and productivity, and I don't think anyone has the answer yet, but I'm curious to get your take of just what should people do today? What's the best approach to understanding what impact AI tools are having on their productivity? Because they're spending all this money on there. I don't know, what are we getting out of this? So if someone had to just like, "Okay, here's what I should probably try to do," what would be your best advice here for measuring the impact of AI tools on productivity?

中文翻译:

让我顺着这个思路，回到我认为目前人们关于 AI 和生产力的最大疑问。我想目前还没有人有标准答案，但我很好奇你的看法：人们今天应该怎么做？了解 AI 工具对生产力产生什么影响的最佳方法是什么？因为公司投入了这么多钱，却不知道到底得到了什么。如果有人必须说：“好，这就是我该尝试做的”，你对于衡量 AI 工具对生产力的影响有什么最好的建议？

[00:49:28] Nicole Forsgren

English:

I would say it depends. In part, it depends on what your leadership chain really cares about. We are usually pretty good at figuring out what matters to developers and we could communicate that to them. But if we're trying to just identify two or three data points to really kind of focus on, because when we're first starting with data, sometimes it can be challenging, what do they care about? Think about the messaging you've been hearing. Have they been talking about market share? Losing market share or competitiveness in the marketplace, if that's it, focus on speed. Think about ways that you can capture metrics for speed from feature to production or feature to customer or feature to experiment and what that feedback loop looks like if they're talking about profit margin all the time.

(00:50:18):

Now, we always talk about money because this is business. But if that seems to be an overarching narrative, look for ways that you can save money and then translate that into recovered and recouped headcount cost. Or sometimes you'll reinvent, change a process, and then you no longer need as many vendors. So reductions in vendor spent can also help there. I say also it depends because sometimes they'll say something, leadership will say something, and it kind of comes up as a theme. If you could solve a problem that they have or it's something that they're focused on, if you can slightly reframe it even, like if they're calling everything developer productivity, go ahead and call it productivity. If they're

calling it velocity, and velocity is what matters to them, think about how to frame this in terms of velocity. If they're talking about transformation or disruption, how does this help with the disruption? Because then, it will resonate with them. We don't want to make them work to understand what it is that we're doing and the value that we provide.

中文翻译:

我会说这取决于具体情况。部分取决于你的领导层真正关心什么。我们通常很擅长弄清楚开发者关心什么并与他们沟通。但如果我们试图确定两三个真正需要关注的数据点（因为刚开始处理数据时可能会很挑战），那就看领导层关心什么。想想你听到的那些信息。他们一直在谈论市场份额吗？如果是担心市场份额流失或竞争力，那就关注“速度”。思考如何捕捉从功能构思到生产、到客户或到实验的速度指标，以及反馈循环是什么样的。如果他们一直在谈论利润率：

(00:50:18):

当然，作为商业行为，我们总是会谈论钱。但如果利润率是核心叙事，那就寻找省钱的方法，然后将其转化为回收的人力成本。或者有时你重新发明或改变了一个流程，不再需要那么多供应商，那么减少供应商支出也会有帮助。我说“取决于具体情况”是因为有时领导层会提到某个主题。如果你能解决他们面临的问题，或者那是他们关注的焦点，即使只是稍微重新定义一下，比如他们把一切都称为“开发者生产力”，那你就叫它生产力。如果他们叫它“速度”，且速度对他们很重要，那就思考如何用速度来表述。如果他们谈论“转型”或“颠覆”，那就谈谈这如何帮助颠覆。因为这样才能引起他们的共鸣。我们不想让他们费劲去理解我们在做什么以及我们提供的价值。

[00:51:20] Lenny Rachitsky

English:

That is such good advice. Just to reflect back, the advice here is if your company's trying to figure out what sort of impact are AI tools having on our company, first, it's just like, what does the company care about most? What do leaders care about most? Could be market share, could be profit margin, could be velocity. We need higher velocity or we need to transform, transformation. So your advice there is figure that out based on words and phrases you're hearing. Then figure out ways to measure that, ways to measure market share growing, profit margin increasing. I love these examples, like time from feature, idea to production or to experiment, so maybe start tracking that. If it's margin, it's money saved by fewer tests, failing or some vendor you don't have to pay for, things like that. And then velocity, I imagine that's where things like DORA come in of just speed of engineering, shipping, or... What would you think about there for velocity?

中文翻译:

这建议太棒了。总结一下：如果你的公司想弄清楚 AI 工具产生了什么影响，首先看公司最关心什么？领导最关心什么？可能是市场份额、利润率或速度。你的建议是根据你听到的词汇和短语来判断。然后寻找衡量这些东西的方法：衡量市场份额增长、利润率提高。我很喜欢这些例子，比如从功能创意到生产或实验的时间，可以开始跟踪这个。如果是利润率，就是减少测试失败节省的钱，或者省下的供应商费用。至于速度，我猜 DORA 指标就在这里发挥作用，衡量工程交付的速度。关于速度你还有什么想法？

[00:52:16] Nicole Forsgren

English:

I would say it's actually one of those... I would pick as broad a swathe as you can. So if you can go from idea to customer or idea to experiment, how long does that take? How long does it typically take, and how long can it take, and does it take now with improved use of AI tooling and reduction in friction?

That's where I will say, we talk about this a little bit in the book, how do we deal with attribution challenges? What was responsible for this? Was it the DevEx or was it AI? Go ahead and disclose that. Say, "Yes, we rolled out AI tools. We also had this effort in DevEx. They partnered very closely together." Both of them probably contributed to this, right? If we had AI tools without the DevEx improvements, we probably would've had some improvements, but not nearly as much.

中文翻译:

我会说，我会选择尽可能广泛的范围。如果你能衡量从“想法”到“客户”或“想法”到“实验”的时间，那需要多久？通常需要多久？在改进了 AI 工具使用并减少了摩擦后，现在需要多久？这就是我在书中提到的“归因挑战”：到底是谁的功劳？是 DevEx 还是 AI？直接公开说明就好。你可以说：“是的，我们推出了 AI 工具，同时我们也进行了 DevEx 方面的努力。两者紧密配合。”两者可能都做出了贡献，对吧？如果我们只有 AI 工具而没有 DevEx 的改进，我们可能会有一些提升，但绝不会有这么多。

[00:53:00] Lenny Rachitsky

English:

If people were starting to do this today, say they're just like, "I want to start measuring developer experience," are there a two or three metrics everybody basically needs they should just start measuring ASAP?

中文翻译:

如果人们今天开始做这件事，比如他们说“我想开始衡量开发者体验”，有没有两三个每个人都基本需要、应该尽快开始衡量的指标？

[00:53:10] Nicole Forsgren

English:

If you're just starting today and if you have nothing at all, talk to people, obviously. After that, I would do surveys because surveys can give you a nice overall view of the landscape quickly so that you know where the big kind of challenges are. I say that because if you're just starting, you might not have instrumentation through your system, all the metrics. And if you do already, it might not be what you think you want. Metrics that were designed without purpose, questionable. Metrics that were designed for another purpose, they might work for what you want, but they might not, so we can't just assume we have them. That's one reason I like surveys, and we include an example in the book. You can just ask a few questions, "How satisfied are you? What are the biggest barriers to your productivity, or what are the biggest challenges to getting work done?" and let them pick either from a set of tools or maybe a set of processes and then say... Let them pick three, just three.

(00:54:12):

Of those three, how often does this affect you? Is this hourly? Is this daily? Is this weekly? Is this quarterly? Because sometimes it hits you every single day, and you're just mad about it. Sometimes it only hits you once a quarter because it's end of quarter, but it's so onerous, and then kind of open text, like, "Is there anything else we should know?" That can give you incredible signal because by making folks prioritize the top three things... Let them pick everything, it makes the data super, super messy. But three things and how often, you can just come up with a score or a weighted score if you want, and then go kind of dig into, where should that data be? What data do we need? But also, then you've got at least some kind of baseline. It'll be a subjective baseline, but now you'll know what the biggest challenges are.

中文翻译:

如果你今天刚开始且一无所有，显然先去和人聊聊。在那之后，我会做调查问卷，因为问卷可以让你快速获得全局概览，从而知道主要的挑战在哪里。我这么说是因为如果你刚开始，你的系统里可能还没有埋点和所有指标。即使已经有了，它们可能也不是你真正想要的。没有明确目的而设计的指标是值得怀疑的；为其他目的设计的指标可能有用，也可能没用，所以我们不能假设已经拥有了它们。这就是我喜欢调查问卷的原因之一，我们在书中也提供了一个示例。你可以只问几个问题：“你的满意度如何？你生产力的最大障碍是什么，或者完成工作的最大挑战是什么？”让他们从一组工具或流程中选择。让他们只选三个，就三个。

(00:54:12):

在这三个中，这种情况多久影响你一次？是每小时、每天、每周还是每季度？因为有时某些事每天都发生，让你很恼火；有时它每季度才发生一次（比如季度末），但非常繁重。最后加一个开放性文本框：“还有什么我们需要知道的吗？”这能给你提供极佳的信号，因为通过让人们优先排序前三件事……如果让他们选所有事，数据会变得非常混乱。但选三件事并标明频率，你就可以得出一个分数或加权分数，然后去深入研究：数据应该在哪里？我们需要什么数据？同时，你至少有了一个基准。虽然是主观基准，但现在你知道最大的挑战是什么了。

[00:55:04] Lenny Rachitsky

English:

I love how all this just comes back just starting by talking to people and asking them these things, which is very similar to product management and just building great products is, have you talked to your customers? Everyone thinks they're doing this, but most people are not doing this enough.

中文翻译:

我很喜欢这一切最终都回到了“从与人交谈并询问这些事情开始”，这与产品管理和构建伟大产品非常相似：你和客户聊过了吗？每个人都以为自己在做这件事，但大多数人做得还不够。

[00:55:17] Nicole Forsgren

English:

And I will say one thing that's challenging when you think about getting data, so interviews are data and that's important, surveys are a little more quantified because we can turn it into counts, but that's where we also want to be careful. A lot of folks go to write a survey question and they'll say something like, "Were the build and test system slow or complicated in the last week?" You're asking four different questions there. If someone answers yes, was it the build? Was it the test? Was it slow or was it flaky or complicated or something? So it can be really difficult to untangle what the signal is you're actually getting there, and so it is worth the time chatting with someone who's familiar with survey design, having a conversation with Claude or Gemini or ChatGPT around, "Here are the survey questions. Or can you propose some?" And then make sure you take a couple of rounds. Is this a good survey question? What questions can I answer from the data that I get? What problems could I solve? If you can't answer a question with data, don't get it.

中文翻译:

我想说，获取数据时有一点很具挑战性：访谈是数据，这很重要；调查问卷更量化一些，因为我们可以将其转化为计数，但我们也需要小心。很多人写问卷问题时会说：“上周构建和测试系统是否缓慢或复杂？”你这里其实问了四个不同的问题。如果有人回答“是”，那是构建慢？还是测试慢？是缓慢还是不稳定（flaky）或复杂？所以很难理清你到底得到了什么信号。因此，花时间咨询熟悉问卷设计的人，或者和 Claude、Gemini、

ChatGPT 聊聊是值得的：“这是我的问卷问题，你能提些建议吗？”然后多过几轮。这是一个好的问卷问题吗？我能从得到的数据中回答什么问题？我能解决什么问题？如果你不能用数据回答某个问题，那就不要收集它。

[00:56:22] Lenny Rachitsky

English:

And you have example surveys in your book for folks that want to just copy and paste and not have to think about this much.

中文翻译:

你的书里有示例问卷，适合那些只想复制粘贴、不想费脑筋的人。

[00:56:28] Nicole Forsgren

English:

Yeah, example surveys, a lot of example questions. We even recommend what the format, what the flow should look like, how long it should be, how long it should not be.

中文翻译:

是的，有示例问卷和大量示例问题。我们甚至推荐了格式、流程应该是怎样的，应该多长，以及不应该多长。

[00:56:37] Lenny Rachitsky

English:

One thing that I was reading is that you don't love happiness surveys specifically, asking engineers how happy they are, is that true? If so, why is that?

中文翻译:

我读到过你并不特别喜欢“幸福感调查”，即询问工程师他们有多幸福，是真的吗？如果是，为什么？

[00:56:45] Nicole Forsgren

English:

I don't, no. Well, I'll say I don't love a happiness survey because there are too many things that contribute to happiness. Happiness is a lot, right? So happiness is work, happiness is family, happiness is hobbies, happiness is weekends, happiness... There are so many things that contribute to happiness. Now, that doesn't mean I don't care about happiness. I think happiness surveys are not particularly useful here. What can be helpful is satisfaction and people are like, "That's the same thing." It's not because you can ask, "Are you satisfied with this tool?" and then ask some follow-up questions. Now, those two are related because the more satisfied you are with your job and your tools and the work and your team, it contributes to happiness. I used to joke... Remember the golf commercials like, "Happy cows like happy cheese"?

中文翻译:

我不喜欢，是的。我会说我不喜欢幸福感调查，因为影响幸福的因素太多了。幸福是一个很宏大的概念，对吧？幸福关乎工作、家庭、爱好、周末……有太多事情贡献了幸福感。这并不意味着我不关心幸福。我只是认为幸福感调查在这里不是特别有用。更有用的是“满意度”（satisfaction），人们会说：“那不是一回事吗？”其实不是，因为你可以问：“你对这个工具满意吗？”然后问一些后续问题。这两者是相关的，因为你对工作、工具和团队越满意，就越有助于提升幸福感。我以前常开玩笑……记得那个广告吗，“快乐的奶牛产快乐的奶酪”？

[00:57:35] Lenny Rachitsky

English:

No.

中文翻译:

不记得。

[00:57:35] Nicole Forsgren

English:

I had a Calabrian. That was the best. Happy devs make happy code. They write better programs, they do better work, they're better team members and collaborators. But capturing and trying to directly influence happiness, that's not what we are here for. It's too challenging, it's too all-encompassing. Satisfaction can give us some signal.

中文翻译:

那是个很棒的广告。快乐的开发者写出快乐的代码。他们编写更好的程序，做更好的工作，是更好的团队成员和协作伙伴。但捕捉并试图直接影响“幸福感”并不是我们的目的。那太具挑战性了，太包罗万象了。而满意度可以给我们提供一些明确的信号。

[00:57:59] Lenny Rachitsky

English:

In a totally different direction, in terms of just tools you see people using, are there any that just like, "Oh, yeah, this one's really commonly great." For people, this is just a tool people are finding a lot of success with. There's the common ones, Copilot, Cursor. I don't know. Is there anything that stands out that you want to share, just like, "Hey, you should check this tool out. People seem to love it"?

中文翻译:

换个完全不同的方向，就你看到的工具而言，有没有哪些是“哦，是的，这个真的普遍很好用”的？就是人们用得很成功的工具。除了常见的 Copilot、Cursor 之外，还有什么让你觉得眼前一亮、想分享给大家的吗？比如“嘿，你应该试试这个工具，大家似乎都很喜欢”。

[00:58:21] Nicole Forsgren

English:

I think they're huge, right? Copilot, Cursor, Gemini.

中文翻译:

我认为那些大牌的都很棒，对吧？Copilot、Cursor、Gemini。

[00:58:25] Lenny Rachitsky

English:

Claude Code.

中文翻译:

Claude Code。

[00:58:26] Nicole Forsgren

English:

Yep, Claude Code. I love Claude Code.

中文翻译:

是的，Claude Code。我非常喜欢 Claude Code。

[00:58:30] Lenny Rachitsky

English:

I have a whole post coming on ways to use Claude Code for non-engineering use cases.

中文翻译:

我即将发一篇关于如何将 Claude Code 用于非工程场景的文章。

[00:58:35] Nicole Forsgren

English:

Cool. Nice.

中文翻译:

酷，太棒了。

[00:58:36] Lenny Rachitsky

English:

It's so interesting. For example, Claude Code, "Find ways to clean up storage on my laptop," and it just tells you there's a bunch of files. It's just like ChatGPT running on your computer and you could do all kinds of crazy stuff on your computer for you, like a mini God.

中文翻译:

这非常有趣。例如，对 Claude Code 说：“帮我找找清理笔记本电脑存储空间的方法”，它就会告诉你有一堆文件。它就像是在你电脑上运行的 ChatGPT，可以为你做各种疯狂的事情，就像一个迷你上帝。

[00:58:36] Nicole Forsgren

English:

I'm going to do that now. This is great.

中文翻译:

我现在就要去试试。这太棒了。

[00:58:57] Lenny Rachitsky

English:

It's so good. Yeah, that's why I'm writing this. I had Dan Shipper was on the podcast and he said Claude Code is the most underrated AI tool out there because people don't realize what it's capable of. It's not just for coding, and that's what I'm trying to explore more and more. Okay. Is there anything else that you think would be valuable to help people improve their developer experience, help them adapt to this new world of AI and engineering that we haven't covered?

中文翻译:

非常好。是的，这就是我写那篇文章的原因。Dan Shipper 曾上过我的播客，他说 Claude Code 是目前最被低估的 AI 工具，因为人们没意识到它的能力。它不仅用于编码，这也是我一直在探索的。好，关于帮助人们改善开发者体验、帮助他们适应 AI 和工程的新世界，还有什么我们没涵盖但你认为有价值的内容吗？

[00:59:22] Nicole Forsgren

English:

I think something that's important to think about in general is to bring a product mindset to any type of DevEx improvements that are happening, and also the metrics that we collect and capture. By that, I mean we want to identify a problem, make sure we're solving a problem for a set of users. We want to think about creating MVPs and experiments and get fast feedback, do some rapid iteration. We want to have a strategy. We want to know who our addressable market is. We want to know what success is. We want to basically have a go-to-market function. We need to have comms. We need to get continuous feedback from our customers. We want to keep improving. And, at some point, we want to think about sunsetting something. Is it in maintenance mode? Is it sun setting?

(01:00:12):

And I think that's important in general, but I think it's extra important now because when we have AI tools, we're using AI tools, we're embedding AI into our products, things are changing so rapidly that it can be really important to take half a beat and say, "Okay, what's the problem I'm trying to solve right here? Is this metric that we've had for the last 10 years still important or should this be sunset because it's not really important anymore? It's not driving the types of decisions and actions that I need."

中文翻译:

我认为总体而言，很重要的一点是为任何正在进行的 DevEx 改进以及我们收集的指标引入“产品思维”。我的意思是，我们要识别问题，确保我们是在为一组用户解决问题。我们要考虑创建最小可行产品（MVP）和实验，获取快速反馈，进行快速迭代。我们要有战略，知道我们的目标市场是谁，知道什么是成功。我们基本上需要一个“进入市场”（GTM）的功能。我们需要沟通，需要从客户那里获得持续反馈。我们要不断改进。而且在某个点，我们要考虑停用（sunsetting）某些东西。它是处于维护模式，还是该停用了？

(01:00:12):

我认为这在平时很重要，但在现在尤为重要。因为当我们拥有 AI 工具、使用 AI 工具、将 AI 嵌入产品时，情况变化得太快了，停下来思考半秒钟变得非常重要：“好，我现在试图解决的问题是什么？我们用了 10 年的这个指标现在还重要吗？还是说因为它不再重要、不再能驱动我需要的决策和行动而应该被停用？”

[01:00:40] Lenny Rachitsky

English:

Before we get to our exciting lightning round, I want to take us to AI Corner, which is a recurring segment on this podcast. Is there some way that you've found a use for an AI tool in your life, in your work that you think might be fun to share, that you think might be useful to other people?

中文翻译:

在进入精彩的闪电轮环节之前，我想带大家进入“AI 角”，这是本播客的一个固定环节。在你的生活或工作中，有没有发现某种 AI 工具的用法，是你觉得分享出来很有趣、或者对别人有用的？

[01:00:55] Nicole Forsgren

English:

I have been working on some home design and redecorating rooms and stuff. I'm working with a designer because I know what I like, but I don't know how to get there, I'm not good at this. But I've really been loving ChatGPT and Gemini especially to render pictures for me, so I can give it the floor plan, I can give it one shot of the room that's definitely not what it's supposed to look like, and then I can give it pictures of a couple different things, and then I can just tell it change the walls or change the furniture layout or change something. It helps me and it's relatively quick. It helps me kind of visualize the things... Again, I know what I like, but I don't know how to get there, so I know if I like it or not, which is probably a very random use, but it's fun for now.

中文翻译:

我一直在做一些家居设计和房间重新装修之类的事情。我正在和一位设计师合作，因为我知道我喜欢什么，但我不知道该怎么实现，我不擅长这个。但我真的很喜欢用 ChatGPT 尤其是 Gemini 来帮我渲染图片。我可以给它平面图，给它一张目前房间的照片（肯定不是最终想要的样子），再给它几张不同风格的图片，然后告诉它改变墙壁、改变家具布局或改变某些东西。它能帮我快速可视化。再次强调，我知道我喜欢什么，但我不知道怎么做出来，所以它能帮我判断我是否喜欢。这可能是一个很随机的用途，但目前很有趣。

[01:01:41] Lenny Rachitsky

English:

My wife does exactly the same thing. She's sending me constantly, "Here's what this rug will look like in our living room. Here's this water feature." It's so good and it keeps getting better. It's just like, "Wow, that's exactly our house with this new rug," and all you do is just upload these two photos and just like, "Cool. How would this look in our room?"

中文翻译:

我妻子也在做完全一样的事情。她不断发给我：“这是地毯铺在我们客厅的样子”，“这是那个水景”。它太棒了，而且越来越好。就像是：“哇，这就是铺了新地毯的我们的家。”你只需要上传两张照片，然后问：“酷，这在我们的房间里看起来怎么样？”

[01:01:57] Nicole Forsgren

English:

Yeah, I've been impressed a couple times. Definitely the machines are listening to us. It's given me a mock-up of a room or something and then it throws in a dog bed, because I have dogs. I'm like, "I did not tell you to do that, but yeah, that's probably the color and style of dog bed that I should have in this room."

中文翻译:

是的，我有几次都被惊艳到了。机器肯定在偷听我们说话。它给我一个房间的效果图，然后居然放了一个狗窝，因为我有狗。我心想：“我没让你加那个，但没错，那确实是我在这个房间里应该放的狗窝颜色和款式。”

[01:02:13] Lenny Rachitsky

English:

Speaking of that, have you tried this use case, ask ChatGPT, "Generate an image of what you think my house looks like based on everything you know about me."

中文翻译:

说到这个，你试过这个用法吗？问 ChatGPT：“根据你对我所有的了解，生成一张你认为我家样子的图片。”

[01:02:22] Nicole Forsgren

English:

I haven't.

中文翻译:

还没试过。

[01:02:23] Lenny Rachitsky

English:

Because it has memory and it remembers everything you've talked about, and it's hilarious. You got to do it.

中文翻译:

因为它有记忆，记得你谈论过的一切，结果非常搞笑。你一定要试试。

[01:02:29] Nicole Forsgren

English:

Okay, that's on my to-do list.

中文翻译:

好，列入我的待办清单。

[01:02:31] Lenny Rachitsky

English:

There we go. Bonus use case. Nicole, with that, we've reached our very exciting lightning round. I've got five questions for you. Are you ready?

中文翻译:

太好了。额外的用法。Nicole，到此我们进入了非常精彩的闪电轮环节。我有五个问题问你，准备好了吗？

[01:02:38] Nicole Forsgren

English:

Awesome. Let's go.

中文翻译:

太棒了，开始吧。

[01:02:39] Lenny Rachitsky

English:

What are two or three books that you find yourself recommending most to other people?

中文翻译:

你最常向别人推荐的两三本书是什么？

[01:02:43] Nicole Forsgren

English:

Outlive by Peter Attia is fantastic. Another one that's I guess maybe related, I hurt my back so it's not great, Back Mechanic by Stuart McGill is incredible. Shout out to anyone who has hurt lower back. It's for a lay person to read through and figure out how to fix lower back problems. It's kind of a random one. I will say I love How Big Things Get Done. I can't pronounce the names. I think one's... There's Scandinavian, one is. It kind of dissects really large projects through recent-ish history and where they failed and why. And I think it's really interesting for us to think about, especially now in this AI moment where basically all of our at least software systems are going to be changing. So how do we think about approaching what is essentially going to be a very large project? And then, sorry, I'm going to throw in a bonus one, The Undoing Project by Michael Lewis. Matt Velloso recommended it to me, and it's so good.

中文翻译:

Peter Attia 的《Outlive》（超越期待）非常棒。另一本可能相关的，因为我背受伤了，所以感觉不太好，Stuart McGill 的《Back Mechanic》（腰背维修师）简直不可思议。向所有腰背受伤的人推荐，它是写给外行看的，教你如何修复腰背问题。这本有点随机。我还很喜欢《How Big Things Get Done》（大局观，作者名字我读不准，好像是斯堪的纳维亚人）。它剖析了近代史上的一些大型项目，以及它们在哪里失败、为什么失败。我认为这对我们很有启发，特别是在现在的 AI 时刻，基本上我们所有的软件系统都要发生变化。那么我们该如何处理这个本质上非常庞大的项目？最后，抱歉，我还要加一本，Michael Lewis 的《The Undoing Project》（思维的发现）。Matt Velloso 推荐给我的，太精彩了。

[01:03:42] Lenny Rachitsky

English:

Yes, I read that-

中文翻译:

是的，我读过——

[01:03:44] Nicole Forsgren

English:

I audibly gasped at the last sentence.

中文翻译:

读到最后一句话时，我惊讶得叫出了声。

[01:03:46] Lenny Rachitsky

English:

Oh. I was like, "What?"

中文翻译:

哦，我当时也是，“什么？”

[01:03:47] Nicole Forsgren

English:

I was [inaudible 01:03:48]. Yeah, I was not expecting it.

中文翻译:

我（听不清）。是的，我完全没预料到。

[01:03:49] Lenny Rachitsky

English:

I read that and I do not remember that last sentence. Oh, man. Okay, cool. Next question. Do you have a favorite movie or TV show you recently watched and enjoyed?

中文翻译:

我读过那本书，但我不记得最后一句话了。天哪。好，下一个问题。你最近有没有看过并喜欢的电影或电视剧？

[01:03:57] Nicole Forsgren

English:

I'll say I watch Love Is Blind. If I got to shut down at the end of the day, Love Is Blind is fun.

中文翻译:

我会说我看了《爱情盲选》（Love Is Blind）。如果一天结束需要放松大脑，这个节目很有趣。

[01:04:02] Lenny Rachitsky

English:

There's a new season out.

中文翻译:

新一季刚出。

[01:04:03] Nicole Forsgren

English:

Yeah, very excited... and Shrinking. Have you seen Shrinking?

中文翻译:

是的，很兴奋……还有《诚实诊疗室》（Shrinking）。你看过吗？

[01:04:07] Lenny Rachitsky

English:

No. I think I started The Therapist and yeah, I gave it a shot.

中文翻译:

没。我想我开始看了《治疗师》，是的，我试着看了一下。

[01:04:12] Nicole Forsgren

English:

Strongly recommend it. It's cute.

中文翻译:

强烈推荐，很可爱的一部剧。

[01:04:13] Lenny Rachitsky

English:

Sweet. Is there a product you've recently discovered that you really love? Could be an app, could be some kitchen gadgets, some clothing.

中文翻译:

太棒了。最近有没有发现什么你非常喜欢的产品？可以是 App、厨房小工具或衣服。

[01:04:21] Nicole Forsgren

English:

Yeah, the Ninja Creami is-

中文翻译:

有的，Ninja Creami 冰淇淋机——

[01:04:25] Lenny Rachitsky

English:

Did you say this last time?

中文翻译:

你上次是不是说过这个?

[01:04:25] Nicole Forsgren

English:

I don't know. I may have. I don't think so.

中文翻译:

不知道，也许吧。我觉得没有。

[01:04:29] Lenny Rachitsky

English:

Somebody said this and I still remember it. It's like-

中文翻译:

有人说过这个，我还记得。就是——

[01:04:30] Nicole Forsgren

English:

It's so good.

中文翻译:

它太好用了。

[01:04:31] Lenny Rachitsky

English:

... you make ice cream and stuff with it, right?

中文翻译:

……用它做冰淇淋之类的，对吧?

[01:04:33] Nicole Forsgren

English:

Yeah, and you can basically freeze a protein shake and then it turns it into ice cream-

中文翻译:

是的，你基本上可以把蛋白粉奶昔冻起来，然后它能把它变成冰淇淋——

[01:04:37] Lenny Rachitsky

English:

Oh, man.

中文翻译:

天哪。

[01:04:37] Nicole Forsgren

English:

... which is delicious. Another one is a Jura coffee maker. I'd love good coffee and I'm not great at making it, so I can just push the button and it'll give me anything I want, including lattes, cappuccinos or anything. So that's kind of fun.

中文翻译:

……非常美味。另一个是 Jura 咖啡机。我喜欢好咖啡，但我不擅长做，所以我只需要按个钮，它就能给我任何想要的，包括拿铁、卡布奇诺之类的。挺有意思的。

[01:04:51] Lenny Rachitsky

English:

Sweet, okay. Do you have a favorite-

中文翻译:

太棒了，好。你有没有最喜欢的——

[01:04:54] Nicole Forsgren

English:

Just sugar and caffeine. I just need a power through the day.

中文翻译:

就是糖和咖啡因。我只需要它们帮我撑过这一天。

[01:04:57] Lenny Rachitsky

English:

There's the engineering productivity 101.

中文翻译:

这就是“工程生产力 101”啊。

[01:05:01] Nicole Forsgren

English:

Yes.

中文翻译:

没错。

[01:05:01] Lenny Rachitsky

English:

Oh, man. Okay, two more questions. Do you have a favorite life motto that you often find useful in work or life and come back to in various ways?

中文翻译:

天哪。好，最后两个问题。你有没有最喜欢的人生格言，是你经常觉得在工作或生活中很有用，并会以各种方式反复回味的?

[01:05:09] Nicole Forsgren

English:

Yeah, I think one that's come up a couple times, it's not a verbatim thing, I think it's more the vibe, hindsight is 2020, but it's also really dumb. I think if we made the best decision we could at the time with the information that we had available, then it is what it is. If you make a bad decision because you made a bad decision and you knew better, you had the information, not great. I don't think we give ourselves or other people enough grace because we always end up finding more information out later.

中文翻译:

有的。我想到了一个出现过几次的，不是原话，更多是一种感觉：事后诸葛亮（hindsight is 20/20）其实很愚蠢。我认为如果我们根据当时掌握的信息做出了最好的决定，那么结果就是那样了。如果你明知故犯，在有信息的情况下做出了错误的决定，那确实不好。但我认为我们对自己或他人都不够宽容，因为我们总是在事后才发现更多信息。

[01:05:42] Lenny Rachitsky

English:

Hear, hear. Final question. I was going to ask you something else, but as we are preparing for this, you shared that you have a new role at Google. Maybe just talk about that, what you're up to there, why you

joined Google, anything folks should know.

中文翻译:

说得好。最后一个问题。我本来想问别的，但在准备时，你分享了你在 Google 有了新职位。也许可以聊聊那个，你在那里做什么，为什么加入 Google，以及大家应该知道的事情。

[01:05:53] Nicole Forsgren

English:

Sure. I am senior director of developer intelligence and core developer. It's super exciting and super fun because of all of these things we've been talking about. It's focused on Google and all their properties and their underlying infrastructure, how can we improve developer experience, developer productivity, velocity, all of these things we've been talking about and, because kind of the numbers person, how do we want to think about measuring it, how does measurement change, how do feedback loops change, how can we improve the experience throughout and then kind of drive that change through an organization in ways that are meaningful and impactful and faster than they've been before.

中文翻译:

当然。我是开发者智能与核心开发者（Developer Intelligence and Core Developer）的高级总监。这非常令人兴奋且有趣，因为这正涉及我们一直在谈论的所有事情。它专注于 Google 及其所有资产和底层基础设施：我们如何改善开发者体验、生产力、速度，以及所有这些话题。而且，作为一个“数据控”，我还要思考如何衡量它、衡量方式如何变化、反馈循环如何变化、如何全面改善体验，然后以有意义、有影响力且比以前更快的方式在组织中推动这种变革。

[01:06:33] Lenny Rachitsky

English:

Nice job, Google, getting Nicole. What a win. I need to get some more Google stock ASAP. Okay, two follow-up questions. Where can folks find you online and find your book online if they want to dig deeper? And how can listeners be useful to you?

中文翻译:

Google 干得漂亮，请到了 Nicole。真是大赢家。我得尽快再买点 Google 股票。好，最后两个后续问题：如果大家想深入了解，可以在哪里找到你和你的书？以及听众可以为你做些什么？

[01:06:47] Nicole Forsgren

English:

Online, you can find the book at developerexperiencebook.com, I'm at nicolefv.com, and LinkedIn occasionally. Sometimes it's a mess. I try to wade through all of the noise. I get there to be useful, sign up for the book and the workbooks. The workbooks are free. I'd love to get any kind of feedback on what works, what doesn't. I always love hearing those kind of stories.

中文翻译:

在线上，你可以在 developerexperiencebook.com 找到这本书，我的个人网站是 nicolefv.com，偶尔也会上 LinkedIn。LinkedIn 有时很乱，我试着从噪音中理出头绪。如果想帮到我，请注册订阅这本书和工作手册。工作手册是免费的。我很想听听关于哪些有效、哪些无效的反馈。我一直很喜欢听这类故事。

[01:07:15] Lenny Rachitsky

English:

Nicole, thank you so much for being here.

中文翻译:

Nicole，非常感谢你能来。

[01:07:17] Nicole Forsgren

English:

Thanks for having me, Lenny.

中文翻译:

谢谢邀请我，Lenny。

[01:07:19] Lenny Rachitsky

English:

My pleasure. Thanks, again. Bye, everyone.

(01:07:23):

Thank you so much for listening. If you found this valuable, you can subscribe to the show on Apple Podcasts, Spotify, or your favorite podcast app. Also, please consider giving us a rating or leaving a review as that really helps other listeners find the podcast. You can find all past episodes or learn more about the show at lennyspodcast.com. See you in the next episode.

中文翻译:

我的荣幸。再次感谢。大家再见。

(01:07:23):

非常感谢大家的收听。如果你觉得本集节目有价值，可以在 Apple Podcasts、Spotify 或你喜欢的播客 App 上订阅。此外，请考虑给我们评分或留下评论，这能极大地帮助其他听众发现本播客。你可以在 lennyspodcast.com 找到所有往期节目或了解更多信息。下集见。