O'REILLY®

"Matt LeMay's actionable advice should be required reading for both experienced and aspiring PMs."

—Ken Norton,
GV (formerly Google Ventures)

Product Management IN PRACTICE

A Real-World Guide to the Key Connective Role of the 21st Century

MATT LEMAY

O'REILLY®

"If you want to be a great product manager you need to go beyond frameworks and tools. *Product Management in Practice* does just that with an honest, humble, and insightful look at the realities of life as a PM. Rich with real-world stories of success, failure, and common misconceptions that face every product manager, Matt reminds us that, above all, building great products is about building great relationships. A must-read for practicing and aspiring product managers or anyone who is interested in building great products."

—Craig Villamor, VP Product, Salesforce

Product management has become a critical connective role for modern organizations, from small technology startups to global corporate enterprises. And yet the day-to-day work of product management remains largely misunderstood. In theory, product management is about building products that people love. The real-world practice of product management is often about difficult conversations, practical compromises, and hard-won incremental gains.

In this book, author Matt LeMay focuses on the CORE connective skills—communication, organization, research, execution—that can build a successful product management practice across industries, organizations, teams, and toolsets.

For current and aspiring product managers, this book explores:

- **On-the-ground tactics for facilitating collaboration and communication**
- **How to talk to users and work with executives**
- **The importance of setting clear and actionable goals**
- **Using roadmaps to connect and align your team**
- **A values-first approach to implementing Agile practices**

Matt LeMay is a product management consultant and coach who has helped build and scale product management practices at companies ranging from early-stage startups to Fortune 500 enterprises.

US $29.99 CAN $39.99
ISBN: 978-1-491-98227-3

Twitter: @oreillymedia
facebook.com/oreilly

Praise for *Product Management in Practice*

It can seem impossible to learn the nuances of day to day product work without doing it yourself. Matt LeMay weaves together case studies from experienced product managers and frameworks to help teach and reinforce key dimensions of the Product Management job.

—*Ellen Chisa*

This is an outstanding practical primer on the day-to-day job of Product Management. What Matt does here is break down Product Management into specific responsibilities and actions that new PMs can take to outline, plan, build and monitor a smooth product development process. This should be a first-stop resource for any new Product Manager.

—*Blair Reeves, Product Principal, SAS Software*
Author, Building for Business *(O'Reilly)*

What makes this book unique is how it goes beyond the jargon and zeroes in on the practical challenges of product management, with super actionable tips. I was smiling and nodding as I read chapter after chapter—highly recommend for all product managers.

—*Pradeep GanapathyRaj, Director of Product Management, Yammer*

Product Management in Practice is a bold and honest playbook by someone who's truly done the job and understands the real challenges PMs face every day. Matt LeMay's actionable advice should be required reading for both experienced and aspiring PMs.

—*Ken Norton, GV (formerly Google Ventures)*

If you want to be a great product manager you need to go beyond frameworks and tools. *Product Management in Practice* does just that with an honest, humble, and insightful look at the realities of life as a PM. Rich with real-world stories of success, failure, and common misconceptions that face every product manager, Matt reminds us that, above all, building great products is about building great relationships. A must-read for practicing and aspiring product managers or anyone who is interested in building great products.

—Craig Villamor, VP Product, Salesforce

In a digital era for products and teams, Matt LeMay brings the role of product management back to its fundamental human tenets. In order to succeed, both new and experienced product managers should note the core PM values that LeMay emphasizes: empathy, curiosity, and collaboration through communication.

—Cecelia Shao, Product Manager, IBM Watson Data Platform

Product Management in Practice

A Real-World Guide to the Key Connective Role of the 21st Century

Matt LeMay

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Product Management in Practice
by Matt LeMay

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com/safari*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisition Editor: Mary Treseler
Editor: Angela Rufino
Production Editor: Nicholas Adams
Copyeditor: Octal Publishing, Inc.
Proofreader: Rachel Monaghan
Indexer: Angela Howard
Interior Designer: Monica Kamsvaag
Cover Designer: Ellie Volckhausen
Illustrator: Jose Marzan Jr. and Rebecca Demarest

November 2017: First Edition

Revision History for the First Edition
2017-11-07: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781491982273* for release details.

978-1-491-98227-3

[LSI]

Contents

Introduction

Why I Wrote This Book: My First Day as a Product Manager

I never felt more prepared for anything than I did for my first day as a product manager. Ever the eager student, I had made a point of reading up on the basic tenets of user experience, sharpening my programming skills, and learning about product development processes like Scrum and XP. I could recite the Agile Manifesto by heart, and knew about the industry-standard bounce rates for comparable web properties. After settling in at my new workstation, I approached my boss—who was not a product manager himself but had worked with enough of them to know the type—with the overconfident swagger of a young person who had read a lot of books very quickly.

"I'm so excited to really dig into this work," I told him. "Where's the latest version of the product roadmap? What are our quarterly goals and KPIs? And who should I be talking to if I want to better understand the needs of our users?"

He gave me a weary look and took a deep breath. "You're smart," he said, "Figure it out."

Aside from a very generous assessment of my intelligence at the time, my boss gave me the greatest gift a new product manager can receive: a grim but resolute understanding that guidance would be very, very hard to come by. For all the books that I had read and all the methodologies that I had studied, the only thing I was left with when I sat back down at my desk was, "What the hell am I supposed to *do* every day?" If there was no roadmap, how was I supposed to manage the roadmap? If there was no product development process, how was I supposed to oversee the product development process?

Early on in my career, I chalked much of this up to the fast pace and loose job definitions that come with working at a startup. But as I began to do more consulting and training work at organizations of different sizes and types, similar

patterns began to present themselves. Even in highly process-driven enterprise organizations, much of the actual work of product management seemed to be taking place on the margins and in the shadows. Product ideas were being hashed out over coffee breaks, not planning meetings. Highly prescriptive scaled Agile frameworks were being bypassed by savvy politicking. Subterfuge, backchannels, and influence reigned supreme. And those same questions I had asked myself on my first day as a product manager were still being asked by product managers at organizations large and small, at cutting-edge technology startups and slow-moving enterprises, by new and experienced product managers alike.

Product management in practice is very different from product management in theory. In theory, product management is about building products that people love. In practice, product management often means fighting for incremental improvements on products that are facing much more fundamental challenges. In theory, product management is about triangulating business goals with user needs. In practice, product management often means pushing relentlessly to get any kind of clarity about what business "goals" really are. In theory, product management is a masterfully played game of chess. In practice, product management often feels like a hundred simultaneous games of checkers.

To that end, this book is not a step-by-step guide to building great products, nor is it a list of frameworks and technical concepts guaranteed to bring you Product Management Success™. This book is intended to help you through the challenges that no tool, framework, or "best practice" can prepare you for. This book is about the day-to-day *practice* of product management with all its ambiguity, contradictions, and grudging compromises. Simply put, this is the book that I wish I had on my first day as a product manager, and many, many days thereafter.

Who This Book Is For

Product management is the glue of modern organizations—the role that connects user needs with business goals, technical viability with user experience, and vision with execution. The connective nature of product management means that the role will look very different depending on the people, perspectives, and roles that you are connecting.

To that end, even defining what is and is not "product management" can prove infuriatingly nonlinear. For the purposes of this book, "product management" refers to the entire nexus of product and product-adjacent connective roles —which might be "product manager," "product owner," "program manager,"

"project manager," or even "business analyst," depending on where you work. For some organizations, a "product manager" is the person responsible for defining a product's strategic vision, whereas a "product owner" oversees day-to-day tactics. For some organizations, the exact opposite is true. I worked with one organization in which a team of "business analysts" woke up to find themselves magically transformed into "product managers" by executive fiat, with no clear sense of how their day-to-day responsibilities had changed, or why.

Titles, like software tools and product development methodologies, are one way to provide some kind of structure and certainty to a role that offers precious little of either. Successful product management is much less a question of titles, tools, or processes than it is of *practice*. I use this word the same way one might refer to a yoga practice or a meditation practice—it is something that is built up with time and experience, and cannot be learned from frameworks and instructions alone.

This book is for anybody who wants to develop their practice of product management. That could be somebody with a "product manager" or "product owner" or "program manager" title. It could be a startup founder struggling to connect the fast-moving pieces of their fledgling company. It could be a software developer who is feeling isolated from the user-facing impact of their work. Long story short, if you are in a role in which you are making connections between and across people and roles, I think and hope that there will be something for you in this book.

How This Book Is Organized

Each chapter of this book is organized around a particular theme, but these themes necessarily blur and blend into one another. Some of the concepts introduced in the first chapters of this book are referenced in later chapters, and some of the ideas explored at length in the later chapters of this book are set up toward the beginning. To get the most out of this book, I'd recommend treating it more like a novel and less like a reference. In practice, product management often feels more like a series of interrelated novellas than it does a neatly organized textbook.

Note that this book does not get into great detail about any particular roadmapping tools, Agile software development methodologies, or product life cycle frameworks. There is no shortage of very useful information out there about choosing a platform for bug tracking, a development methodology for product teams at medium-sized startups, or a framework for estimating user stories. The

goal of this book is not to address the specific tools you might choose in your product management practice, but rather how to set up your product management practice to get the most out of any tools you choose to use.

For simplicity's sake, I have described the people for whom you are building products as "users," not "customers." Not every product is paid for directly by the people who use it, but every product is used by someone or something. In some cases, such as enterprise B2B software sales, you might have a "customer" who is different from your "user," and you can find yourself having to understand and connect the needs of both. If you're interested in reading more about this distinction and the ramifications it can have on product design, I would recommend checking out Blair Reeves's article "Product Management for the Enterprise".

Also note that this book is not intended to be a high-level introductory glossary of product management terminology. If you encounter an idea, a concept, or an acronym that is new to you, take a moment to look it up.

STORIES FROM WORKING PRODUCT MANAGERS

There's often a knowing, conspiratorial tone to conversations shared between working product managers—like we're all in on the same secret. That secret, for those of you who are wondering, is that our job is widely misunderstood and really, really, really hard. Product managers are much more likely to share "war stories" than "best practices," and are more likely to talk about the mistakes they've made than the meteoric successes they've achieved.

In the hopes of bringing this kind of conversation to people who might benefit from it, I have included stories from working product managers in this book. Most of these stories started with the question, "What is one story from your work that you wish somebody had told you on your first day as a product manager?" As you will see, most of these stories are about people, not frameworks, tools, and methodologies. Several of the product managers I spoke to provided multiple stories which, taken together, paint a more comprehensive picture of the distinct-but-related challenges a product manager is likely to encounter over the course of their career.

Some of these stories are directly attributed to the folks who told them, some are anonymized, and some are composited from multiple sources. But they all represent the messy, complex realities of product management in the field. I have learned—and continue to learn—a lot from these stories, and I hope that you will, too.

TEMPLATES

Throughout this book, I have included templates that you are welcome to adapt and customize for the specific needs of your role, team, and organization. I am a big believer in templates, especially for managing contentious issues like organizational process changes and "emergency" feature requests. Templates codify organizational goals and values into an impartial set of prompts and questions, and have immense power to depersonalize emotionally charged disagreements. When used consistently and transparently, templates become an immensely useful organizational mediator.

To get the most out of these templates, I would suggest that you begin by using them to structure your own ideas and suggestions, and then requesting that others follow suit. Templates can be a critical instrument of rigor and discipline—but only if you are willing to subject yourself to that same rigor and discipline.

"YOUR CHECKLIST"

Every chapter of this book ends with an actionable to-do list called "Your Checklist." Product management can be heady and abstract stuff, and my primary goal with this book is for it to prove useful to working product managers. Each item on "Your Checklist" serves as an action-oriented summary of an idea that was explored at more length within that chapter.

O'Reilly Safari

Safari (formerly Safari Books Online) is a membership-based training and reference platform for enterprise, government, educators, and individuals.

Members have access to thousands of books, training videos, Learning Paths, interactive tutorials, and curated playlists from over 250 publishers, including O'Reilly Media, Harvard Business Review, Prentice Hall Professional, Addison-Wesley Professional, Microsoft Press, Sams, Que, Peachpit Press, Adobe, Focal Press, Cisco Press, John Wiley & Sons, Syngress, Morgan Kaufmann, IBM Redbooks, Packt, Adobe Press, FT Press, Apress, Manning, New Riders, McGraw-Hill, Jones & Bartlett, and Course Technology, among others.

For more information, please visit *http://oreilly.com/safari*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *http://bit.ly/product-management-in-practice*.

To comment or ask technical questions about this book, send email to *book-questions@oreilly.com*.

For more information about our books, courses, conferences, and news, see our website at *http://www.oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*

Follow us on Twitter: *http://twitter.com/oreillymedia*

Watch us on YouTube: *http://www.youtube.com/oreillymedia*

Acknowledgments

Thanks to Mary Treseler, Angela Rufino, Laurel Ruma, Meg Foley, and everybody at O'Reilly Media for turning a pitch about a "very voicey and opinionated book about product management" into a real thing.

Thanks to everybody who provided both formal and informal feedback along the way.

Thanks to Roger Magoulas for bringing me into the fold.

Thanks to every product manager who shared their stories with me. And, special thanks to those who helped me streamline the process of gathering these stories. (Product managers, amirite?)

Thanks to everybody whom I have had the pleasure of working with through the ups, downs, and literal and figurative tears of my own product management career. Your patience and generosity will always mean the world to me.

Thanks to Josh Wexler for the best coffee meetings ever.

Thanks to Andy Weissman for taking a chance on me way back when.

Thanks to Sarah Milstein for getting this all started.

Thanks to Jodi Leo for an exceptionally well-timed gift of encouragement.

Thanks to Tricia Wang and Sunny Bates for showing me the power of true partnership.

Thanks to mom for not obfuscating the message.

Thanks to dad for a grudging but unshakeable love of learning.

Thanks to Joan for everything, every day.

The Practice of Product Management

I recently asked Pradeep GanapathyRaj, director of product management at Yammer, what he wished that every new product management hire understood about their responsibilities. Here are his answers:

- Bringing out the best in the people on your team
- Working with people outside of your immediate team, who are not directly incentivized to work with you
- Dealing with ambiguity

On his third point, he added: "The skill of actually figuring out what you need is probably as important as what you do after you figure it out."

Perhaps the most striking thing about these answers is that none of them are about product, per se. Although many people are drawn to product management by the promise of "building products that people love," the day-to-day practice of product management involves much less actual building than it does supporting, facilitating, and communicating. In this chapter, we discuss the actual, real-world practice of product management, and address a few common traps that product managers fall into when their expectations of the role don't line up with the reality.

What Is Product Management?

So, what exactly *does* a product manager do all day, anyhow?

The answer to that question depends on a lot of things. At a small startup, you might find a product manager cobbling together product mock-ups, schedul-

ing check-in points with contract developers, and conducting informal interviews with potential users. At a medium-sized technology company, you might find a product manager running planning meetings with a team of designers and developers, negotiating product roadmaps with senior executives, and working with their colleagues in sales and customer service to understand and prioritize user needs. At a large enterprise organization, you might find a product manager rewriting feature requests as "user stories," requesting specific data from their colleagues who work in analytics or insights, and attending a whole lot of meetings.

In other words, if you are working as a product manager, you will probably find yourself doing lots of different things at different times—and what exactly those things are can change at a moment's notice. However, there are a few consistent themes that unite the work of product management across job titles, industries, business models, and company sizes:

You have lots of responsibility but little authority

Did your designers and developers miss a launch deadline? That is your responsibility. Did the product you manage fail to meet its quarterly goals? That's also your responsibility. As a product manager, you are the person who is ultimately responsible for the success or failure of your product—regardless of how well supported that product might be by the rest of the organization.

Working in a position of high responsibility is challenging enough, but to further complicate matters, product managers rarely have any direct organizational authority. Is there a designer on your product team who's showing up late to meetings? An engineer whose attitude is proving toxic to the team at large? These are your problems to solve, but you can't solve them with the bludgeon of "because I said so" or "you're fired." You must lead through influence, not authority, which requires developing an entirely different set of skills and approaches.

If it needs to get done, it's part of your job

"But—that's not my job!" is a phrase rarely uttered by successful product managers. Regardless of whether it falls neatly into the boundaries of your written job description, you are responsible for doing *whatever needs to get done* to ensure the success of your team and your product. This might mean coming in early to bring coffee and breakfast to an overworked product team. It might mean insisting that you get face-to-face time with a

senior executive to resolve some ambiguity around your team's goals. And it might mean calling in a favor from elsewhere in the organization if your team simply doesn't have the capacity to do what's needed on its own.

If you work as a "product manager" at a very early-stage startup, you might find yourself spending most of your time doing work that feels like it has very little to do with "product management" at all. Product managers I've known at early-stage startups have found themselves working as ad hoc community managers, HR leads, UX designers, and office managers. If it needs to be done, and it isn't anyone else's job, then—surprise!—it's your job. Even at large enterprise companies, there will almost certainly be times when you need to step up and do something that is not officially part of your job. And, because you are held responsible for the performance of your team and your product, "that wasn't my job" plays as well at a Fortune 50 corporation as it does at a five-person startup.

To make this even more challenging, most of the things you will need to get done as a product manager are not things you can do alone. You do not have the luxury of disappearing for a few weeks, reading a bunch of books, and returning fully skilled up and ready to single-handedly deliver a product. You will need to ask for support, guidance, and hard work from the people around you—often people outside of your immediate team, who might have no obvious reason to help you out.

You are in the middle

As a product manager, much of your job comes down to translating between the needs, perspectives, and skill sets of your stakeholders and users. This would be challenging enough if your only goal were to synthesize these far-flung data points into an actionable product plan. But as a product manager, you also sit in the middle of the *people* who hold these needs, perspectives, and skill sets. You must understand their communication styles, their sensitivities, and the differences between what they say and what they mean.

Even at organizations that are highly structured and systematized, or that claim to be impartial and "data-driven," it is inevitable that at some point you will find yourself navigating a tangled web of unspoken resentment and unresolved conflict. Others might be able to keep their heads down and "just do their work," but making connections between messy, real-world people *is* your work.

What Is Not Product Management?

Product management can be many different things, but it is not everything. Here are a few consistent—and for some people, consistently disappointing—realities of what product management is not:

You are not the boss

I've often seen the role of product manager described as the "mini-CEO" of a product. Unfortunately, most of the product managers I've seen act like a "mini-CEO" are more interested in the status of this honorary distinction than they are in its responsibility. Yes, as a product manager, you are responsible for the success or failure of your product. But, for you to meet this responsibility, you are entirely dependent upon the trust and hard work of your team. That trust can be easily squandered if you carry yourself like a big important boss, ordering people around and failing to show the same in-the-trenches commitment that you're expecting from the other members of your team.

You are not actually building the product yourself

For some people, "product management" evokes visions of brilliant inventors and tinkerers, toiling away to bring their game-changing idea to the masses. If you like to be the person actually building things with your hands, you might find yourself deeply frustrated by the connective and facilitative nature of product management. Furthermore, what feels like a good-natured wish to get into the weeds of technical and design decisions might come off as infuriating micromanagement to the people actually tasked with building the product you manage.

This absolutely does not mean that you should take zero interest in your product team's technical and design decisions. Taking a genuine interest in the work of your colleagues is one of the most important things you can do as a product manager. But product management often poses a particular challenge to those who come from the "never mind, I'll just go do it myself" school of problem-solving. If you, like me, are the kind of person who absolutely hated "group projects" in school and sought to take on as much work as possible for yourself, product management is likely to teach you difficult-yet-important lessons about trust, collaboration, and delegating.

You can't wait around until somebody tells you what to do

As I learned on my first day as a product manager, it's exceptionally rare that you will be given clear guidelines and instructions in this role. Larger companies, especially larger companies like Amazon and Google that have a mature product management practice, are more likely than other companies to have a well-defined set of expectations around the product manager role. But even at those companies, you will have your work cut out for you figuring out what you should do, who you should talk to, and discovering the "unknown unknowns" that are unique to your product and team.

If you're unclear about a directive coming down from senior leadership, you can't sit around waiting for them to clarify it. If you see something in a mock-up that you think will be a problem, you can't wait until somebody else catches it. Just as you need to get out ahead of identifying the work that needs to be done, you also need to get out ahead of any disconnects in communication or understanding that might harm your product or your team, and be relentless about resolving them.

What Is the Profile of a Great Product Manager?

Some organizations are well known for favoring a certain "profile" for product management candidates. Amazon, for example, looks for MBAs. Google, on the other hand, prefers candidates with a computer science degree from Stanford. Generally speaking, the "classic" profile for a product manager is either a technical person with some business savvy, or a business-savvy person who will not annoy the hell out of developers.

Although there are plenty of product managers who fit this profile to some degree, some of the very best product managers I've met—including product managers who cut their teeth at Amazon and Google—don't fit any "classic" profile. The truth is, great product managers can come from anywhere. Some of the best product managers I've met have backgrounds in music, politics, nonprofits, theater, marketing—you name it. Generally speaking, they are people who like to solve interesting problems, learn new things, and work with smart people.

Great product managers are the sum of their experiences, the challenges they've faced, and the people they've worked with. They are constantly evolving and adapting their own practice to meet the specific needs of their current team and organization.

When I consult with organizations that are looking to identify internal candidates for PM roles, I often ask a few people to draw out a diagram of how information travels within the company—not a formal org chart, just an informal sketch of how people communicate with each other. Without fail, a few people continuously show up smack in the middle—these people are the information brokers, the connectors, the expansive thinkers who are actively seeking out new perspectives. These people rarely fit the "traditional" profile of a product manager, and, in many cases, they are entirely nontechnical. But these are the people who have already proven their capability at difficult but critical connective skills, and have demonstrated the initiative to take on this kind of work without much formal recognition.

What Is the Profile of a Bad Product Manager?

Although great product managers rarely fit a single profile, bad product managers are quite consistent. There are some bad product manager archetypes that seem to show up in nearly every kind of organization:

The Jargon Jockey

The Jargon Jockey wants you to know that the approach you're describing might make sense if you were working in a hybrid Scrumban methodology, but is simply unacceptable to a certified PSM III Scrum Master. (If you had to look any of that up, the Jargon Jockey is shocked by your incompetence—how did you even get this job in the first place?) The Jargon Jockey is appalled that you haven't heard of Arie van Bennekum—*one of the signers of the Agile Manifesto!* The Jargon Jockey defines words you haven't heard with other words you haven't heard and seems to use those words more and more when there's a high-stakes disagreement playing out.

The Steve Jobs Acolyte

The Steve Jobs Acolyte Thinks Differently™. The Steve Jobs Acolyte likes to lean back in chairs and ask big, provocative questions. The Steve Jobs Acolyte would like to remind you that people didn't know that they wanted the iPhone, either. The Steve Jobs Acolyte doesn't want to build a faster horse. The Steve Jobs Acolyte wouldn't say that your users are stupid—at least, not exactly—but they are definitely not visionaries like the Steve Jobs Acolyte.

The Hero Product Manager

Have no fear, the Hero Product Manager is here with an amazing idea that will save the whole company. The Hero Product Manager is not particularly interested in hearing why this idea might not work, or that it's already been discussed and explored a million times. Did you hear about what the Hero Product Manager did at their last company? They pretty much built the whole thing single-handedly, or at least the good parts. And yet, the people at *this* company just never seem to give the Hero Product Manager the resources or support they need to deliver on all those amazing promises.

The Product Martyr

Fine! The Product Martyr (Figure 1-1) will do it. If the product didn't launch on time or didn't meet its goals, the Product Martyr takes complete and unequivocal responsibility for screwing everything up (again). The Product Martyr says it's no big deal that they pick up coffee for the whole team every morning, but the way they place the Starbucks tray down on their desk seems just a liiiiiiiiittle more emphatic than it needs to be. The Product Martyr says repeatedly that they've put this job ahead of everything else in their life, and yet they seem outraged and overburdened every time you come to them with a new question or concern.

The Nostalgic Engineer

The Nostalgic Engineer would rather be writing code than sitting in all these meetings. Did you know that the Nostalgic Engineer used to write code? *Really* good code, too, before there was all this "jquery" nonsense. The Nostalgic Engineer is happy for the product team to work on whatever that team thinks is most fun. Did the Nostalgic Engineer mention that they would rather be writing code than be stuck in this stupid meeting? No offense, of course.

Figure 1-1. The Product Martyr, in the wild

These patterns are shockingly easy to fall into—I've definitely fallen into all of them at one point or another in my career. Why? Because by and large they are driven not by malice or incompetence but by insecurity. Product management can be a brutal and relentless trigger for insecurity, and insecurity can bring out the worst in all of us.

Because product management is a connective and facilitative role, the actual value product managers bring to the table can be very difficult to quantify. Your developer wrote 10,000 lines of code. Your designer created a tactile, visual universe that wowed everybody in the room. Your CEO is the visionary who led the team to success. Just what did *you* do, exactly?

This question—and the urge to defensively demonstrate value—can lead to some epic acts of unintentional self-sabotage. Insecure product managers might begin to make big, awkward public displays of their contributions (the Product Martyr). They might begin speaking in gibberish to prove that product management is a real thing that is really complicated and important (the Jargon Jockey). They might even start devaluing their own contributions to the team for the sake

of showing that they *could* be doing higher-status work if they wanted to (the Nostalgic Engineer).

For product managers, the value you create will be largely manifest in the work of your team. The best product managers I've met are those whose teams use phrases like "I would trust that person with my life" and, "That person makes me feel excited to show up for work in the morning." If you're starting to feel insecure about your work, talk to your team and see what you can do to better contribute to their success. But don't let insecurity turn you into a bad product manager—and please, please, don't quote Steve Jobs in a meeting or job interview.

Filling a Leadership Vacuum and Co-Creating Shared Goals (Or, How a Lack of Organizational Objectives Can Create a Team of Product Martyrs)

M.G.
Product leader, nonprofit

I recently joined a nonprofit product shop that primarily builds products for other nonprofits. Like many nonprofits, we face some pretty major resource constraints. When I first sat down to talk with the product owners who report to me directly, each was working on at least three products and seemed stressed out and spread thin. Thankfully, I was able to get some more resources allocated our way to hire a few more product owners. This, I thought, was the answer to our problems: more product owners meant each product owner could devote more time and attention to one thing at a time.

To my surprise, my direct reports did not seem all that excited about this news. In fact, their initial reactions ranged from skeptical to outright defensive. What seemed to me like an opportunity to focus and streamline our efforts as a company was instead being interpreted as a personal attack, "Why are you taking some of my products away from me?" This reaction was particularly confusing to me because it was coming from some of the same people who had been complaining about how overworked they were just a few weeks prior.

It took me a while to realize it, but those initial complaints about being overworked weren't just a reflection of our lack of organizational resources. **They were symptomatic of a much deeper and more com-**

plex problem: our product owners had no way of measuring their own success other than how hard they were working and how many products they owned. Why? Because there was no common sense of the goals we were working toward as an organization. In the absence of these goals, each individual product owner was left to his or her own assumptions and ad hoc success metrics.

The funny thing is, if I had simply gone ahead and hired more product owners without addressing this underlying issue, it just would have made things worse. In the absence of clear, company-wide goals, more product owners would have just meant more competing assumptions about what success looks like. As soon as I started bringing all of our product owners together to talk about shared goals, a lot of the competition and defensiveness I had seen naturally started to melt away. Product owners started looking for ways to share resources and knowledge, knowing that their success would be measured against how well we were able to deliver value to our customers, not how many products they "owned" or how late they were willing to stay in the office on a Friday evening.

Summary: Sailing the Seas of Ambiguity

No matter how many books you've read (including this one), how many articles you've scrolled through, or how many conversations you've had with working product managers, there will always be new and unexpected challenges in this line of work. Do your best to remain open to these challenges and, if possible, enjoy the fact that the ambiguity around your role means that you are likely to learn lots of entirely new things.

Your Checklist:

- Accept that being a product manager means that you are going to have to do lots of different things. Don't become upset if your day-to-day work is not visionary and important-seeming, so long as it is contributing to the goals of your team.
- Find ways to align, motivate, and inspire your team that do not require formal organizational authority.

- Be proactive about seeking out ways that you can help contribute to the success of your product and your team. Nobody is going to tell you exactly what to do all the time.
- Be a connector between teams and roles.
- Get out ahead of potential miscommunications and misalignments, no matter how inconsequential they might seem in the moment.
- Don't get too hung up on the "typical profile" of a successful product manager. Successful product managers can come from anywhere.
- Don't let insecurity turn you into the caricature of a bad product manager! Resist the urge to defensively show off your knowledge or skills.

The CORE Connective Skills of Product Management

Given the ambiguity around the day-to-day responsibilities of product management, it can prove incredibly difficult to pin down the skills that product managers must use in their work. This often results in product management being described as a mish-mash of the skills used in other, easier-to-define roles. A little bit of coding, a pinch of business acumen, some user experience design, and —voilà!—you're a product manager.

As we discuss in this chapter, the connective work of product management requires its own set of connective skills. Defining this set of skills helps carve out a place for product management as a unique and valuable role, and provides much-needed day-to-day guidance for how product managers can excel at their work.

The "Hybrid" Model: UX/Tech/Business

Insofar as there is a commonly accepted skill model for product managers, it is often positioned as a three-way Venn diagram (Figure 2-1) between "business," "tech," and "UX" (user experience):

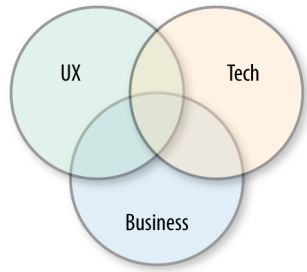

Figure 2-1. The "hybrid" product management Venn diagram, from Martin Eriksson's "What Exactly Is a Product Manager"

I have seen a number of variations on this—sometimes "UX" is replaced with "design" or "people." Sometimes "business" is replaced with "statistics" or "finance." I recently saw a job listing from a major bank that asked for candidates who are proficient in "business, technology, and human"—which in no way sounds like a job listing written by and for sentient robots.

When used well, this model speaks to the connective nature of the product management role. If you happen to be working with a team of designers and developers to build products that solve business needs, you might find yourself standing in the middle of this very diagram. But the skills required to *be* a designer, a developer, or a business leader are very different from the skills required to *create alignment* between designers, developers, and business leaders. Although this model helps describe the roles and subject matter knowledge that you are likely to encounter as a product manager, it does not provide much guidance as to what exactly you are supposed to *do* about it.

The UX/Tech/Business model is further complicated by the vast variability of the product manager role between organizations. At a large enterprise, a product manager might be accountable for a specific product's profits and losses but rarely work directly with developers or designers. At a high-growth startup, a product manager might be working in close collaboration with developers and designers, but remain largely insulated from financial decision making. From company to company, the people you're working with can vary enormously, but it is always your job to connect and align those people.

The CORE Skills of Product Management: Communication, Organization, Research, and Execution

For all the variability in the product manager role across organizations, the skills required to succeed as a product manager are often quite similar (Figure 2-2): a product manager must be able to effectively **communicate** between stakeholders and users, **organize** the product team for successful collaboration, **research** new ideas and perspectives, and **execute** whatever day-to-day tasks are required for their specific role and team. These **CORE** connective skills constitute a new skill model for product managers that better reflects the real-world work of product management across organizations and industries.

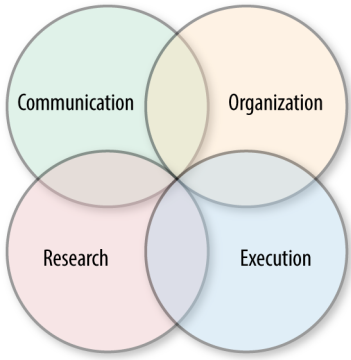

Figure 2-2. The CORE skills of product management

What follows is a breakdown of the CORE skills of product management, with a guiding principle for each skill that speaks to the real-world behaviors and decisions that put these skills into action.

COMMUNICATION

"Clarity over comfort"

Communication is far and away the most important skill for a product manager to develop and nurture. If you cannot communicate effectively between your team, your stakeholders, and your users, you will not succeed as a product manager. Great product managers not only tolerate, but actively enjoy, the challenge of creating alignment and understanding between different roles and perspectives.

The guiding principle for communication is "clarity over comfort." The choice between clarity and comfort is a real one, and one that we are often faced with at the most important moments of our career. For example, you might find yourself attending a meeting in which a senior stakeholder makes a passing reference to a product requirement that has never come up in a formal planning meeting. Not wanting to call out this inconsistency—and not wanting to create an uncomfortable situation—you might decide to let it pass, assuming that when the product is launched you will be able to declare anything not included in the formal spec as definitively "out of scope." But from the senior stakeholder's perspective, your silence might be interpreted as a tacit agreement that the product scope has officially changed. The consequences of this lack of clarity could be trivial—or cataclysmic.

It is no accident that uncomfortable moments like this are often the ones that prove most impactful. Discomfort is often the manifestation of a lack of clarity. It is a valuable signpost that people might not be on the same page, or that expectations have not been clearly set. As a product manager, you cannot fear discomfort—you must actively work through it to get clarity for yourself and your team. We will discuss specific strategies for achieving clarity over comfort in Chapter 5, *The Art of Egregious Overcommunication*.

ORGANIZATION

"Change the rules, don't break the rules"

Beyond using their own personal communication skills, product managers must organize their teams to work well together. This is not (just) a matter of being an expert in product development methodologies—though somebody who has successfully led a team using one of these methodologies likely knows a thing or two about organizing. If communication skills come down to managing one-on-one interactions, organization is about operationalizing and scaling those interactions.

Not all individuals who excel as individual communicators are naturally gifted organizers. Product managers who lack organization skills, no matter how knowledgeable and charismatic they are, often become a bottleneck for their teams. They run around giving directions, unblocking team members, and resolving conflicts, but their teams are incapable of functioning without the product manager's direct and constant intervention. Product managers who lack organization skills are happy to hear questions like "what should we be working on right now?" because these questions put the product manager in the utterly indispensable role of guiding the team's day-to-day priorities and decisions.

By contrast, product managers who excel at organization see the question "what should we be working on right now?" as a sign that something is broken. They strive to ensure that everybody on their team will always know what they should be working on and why, without having to ask that product manager personally. When something goes wrong, organization-minded product managers don't just ask "how can I solve this problem right now?" but also "how can I make sure this doesn't happen again?"

The guiding principle for organization is "change the rules, don't break the rules." If an organizational process or practice is not helping achieve your team's goals, it is your responsibility to work with your team toward *changing* that process or practice. Instead of breaking the rules for "special cases," reflect on why the rules cannot accommodate these cases. How might you collaboratively change those rules to better account for new situations and changing circumstances? Great product managers are not afraid to challenge procedural orthodoxies if it will help their team meet its goals. We'll talk more about this in Chapter 10, *The Wonderful, Horrible Truth About Agile*.

RESEARCH

"Live in your user's reality"

In her excellent book *Just Enough Research* (A Book Apart, 2013), Erika Hall says that "research is just another name for critical thinking." Indeed, critical thinking is a crucial piece of a product manager's job—but just sitting back and being really smart is not enough. For a product manager, research is about seeking out and synthesizing multiple perspectives and sources of information. If curiosity is the key mindset of the product manager, research is how that curiosity is actualized and extended beyond the walls of the organization.

In theory, product managers should know their market and know their user. In practice, this is far from a given and often requires asking difficult questions and challenging deeply held assumptions within an organization. Great product managers never take anything for granted and are constantly seeking out new ideas, confounding variables, and challenging perspectives.

Research is a critical tool for attaining the subject matter knowledge that might be necessary for a specific product management role. At a tech startup, some knowledge of user experience and technology can come in handy for meeting deadlines and making trade-offs. A product manager at a company that primarily sells physical goods might want to study up on manufacturing and supply chains. It is always good to know as much as you can about product's competitive landscape—but that landscape is always changing, and at an ever-faster pace.

What's important is not that you already possess the relevant subject matter knowledge for your specific role, but rather that you are actively seeking out any and all knowledge and information that might help you and your team succeed.

Product managers who lack research skills tend to lead their team steadfastly down a predetermined path, without taking the time to ask *why* they are pursuing that path, or seeking out new information that might compel them to adjust course. These product managers might be able to hit their specific deadlines, but they are constantly playing catch-up with their market and their users.

The guiding principle for research is, "live in your user's reality." Every product has a user—whether it's a consumer, another business, or an engineer utilizing an API. The things that matter to you, such as hitting project deadlines, managing the product backlog, or balancing the profit and loss statement, do not matter at all to them. Your users have their own set of priorities, needs, and concerns, the most critical of which might not seem directly related to their interactions with your product. The most successful product managers I've met give voice not only to their users' immediate, transactional "pain points," but also to their broader and more complex *realities*. When these product managers evaluate a competitor's product, they ask, "What might this product mean to our users," not "How can we achieve feature parity?" By taking an open and holistic view, these product managers are able to identify previously unexplored solutions to fast-changing user needs. We'll talk more about this in Chapter 7, *Talking to Users (Or, "What's a Poker Game?")*.

EXECUTION

"No work beneath, no work above"

Product managers are, of course, still responsible for making sure that stuff gets done. This often means stepping up to do whatever work is needed for your team, even if that work is not technically part of your job description. Execution-minded product managers not only provide critical support for their teams, they inspire everybody on their team to step up and do whatever needs to be done.

Product managers who lack execution skills often become bogged down in the theoretical underpinnings of product management. They are more concerned with finding the "perfect" product management framework than they are with actually getting a product released. They value thinking over doing. And by approaching their work this way, they implicitly devalue the contributions of their teammates who are tasked with actually building their products.

The guiding principle for execution is "no work beneath, no work above." Execution-minded product managers are willing to take on work that might be

seen as low-status in their particular organization, and actively elevate that work by modeling its importance. An execution-minded product manager, for example, will happily go on an early-morning coffee run if that is an important step toward actually getting a product out the door. As Google Ventures' Ken Norton says, "always bring the donuts."

When I started working as a product manager, I was prepared to do my share of donut and coffee runs. What I was not expecting was how many times I would need to step into conversations that felt *above* my proverbial pay grade. During one particularly tumultuous moment early in my career, I was made "VP for a day" so that I could lead a mission-critical negotiation with a major platform partner. Embarrassingly, I was much more focused on the fact that I had not been given a real promotion than I was on successfully leading this important negotiation. An execution-minded product manager is willing to step into critical, high-level conversations for the sake of clarifying and achieving organizational goals, not for personal glory. We'll discuss this more in Chapter 6, *Working with Senior Stakeholders (Or, Throwing the Poker Game)*.

...But What About "Hard Skills"?

The skills I have outlined in this chapter could be described as "soft skills." Generally speaking, soft skills are considered to be the squishy, subjective, "interpersonal" skills that are difficult to quantify or measure. "Hard skills," on the other hand, are considered to be fixed, objective, and measurable. For example, communication and time management skills are often considered soft skills, whereas computer programming and statistical analysis are considered hard skills.

In some contexts, hard skills are considered the absolute essentials to perform a job, whereas soft skills are considered a "nice to have." And for some roles, there is a bar of hard skills that must be met—after all, you wouldn't hire a computer programmer who had never written a line of code, or a dentist who had never attended dental school. But the absolute distinction between "hard" and "soft" skills is often deployed in a way that feels reductive, imbalanced, and unfair to both types of skills. Hard skills like programming still require nuance and craft, and soft skills like communication and time management can still be learned, practiced, and evaluated.

When it comes to product management, though, the distinction between soft and hard skills is particularly insidious. To put it bluntly, far too many people and organizations hire product managers based on hard skills that have precious little to do with the day-to-day work that those product managers will be expected

to perform. I've seen fantastic product managers fail job interviews because they couldn't whiteboard an algorithm or solve a code challenge, even if their day-to-day work will require them to do neither of these things. To this day, one of the most common questions I am asked by aspiring product managers is, "just how technical do I have to be?"

In an article titled "Getting to Technical Enough as a Product Manager," Lulu Cheng provides a swift and definitive take on this issue:

> *The day-to-day responsibilities [of a product manager], and technical bar, varies widely depending on the industry and size of the company, as well as the part of the product you work on. At the same time, the qualities that make someone a universally respected PM rarely have to do with technical expertise.*

Indeed, if you are working on a highly technical product, having a basic knowledge of the systems with which you are working will soften the learning curve and give you a head start. But any assessment of the required hard skills for a specific product management role should begin with the work that a product manager will be expected to do day-to-day—which, in most cases, is much more connective than it is technical.

So why does this focus on hard skills persist? Here are a few myths I've encountered and would like to debunk:

You need hard skills to win the respect of technical folks

The idea that technical folks can respect only somebody who shares their skill set is, frankly, insulting to technical folks. If anything, I've seen product managers who "play" developer initially win over their technical counterparts, only to annoy the living crap out of them later by micromanaging implementation details. As we discuss in Chapter 3, communication skills will help you learn hard skills in a way that respects the expertise of your colleagues and the specific context of your organization.

You need hard skills to challenge technical folks

There is a kernel of truth here—if you have no idea how technical systems work, your developers could tell you that something relatively easy to build will take a million years. But if your team is flat-out lying to you about how long things will take, you have a more fundamental problem on your hands. Product managers with excellent execution skills inspire their teams

to *want* to get stuff out the door quickly, and they don't do it by playing "gotcha" with technical specificities.

You need hard skills to stay interested and engaged with technical work

It is absolutely true that a product manager who is disinterested in the work of their colleagues is likely to fail. But knowledge and interest are two very different things, and I've found that many of the most technically knowledgeable product managers are also those least interested in learning new things and engaging deeply with the work of their colleagues. The best product managers, regardless of their hard skills, are able to take a genuine interest in the technical work of their colleagues and draw compelling connections between technical work, user needs, and business goals.

You need hard skills to do things like query databases, write documentation, and push minor changes

In many cases, this is actually 100% true. In keeping with the idea that "if it needs to get done, it's part of your job," product managers often find themselves faced with technical tasks that do require some hard skills. For example, at a small company, a product manager might be asked to make minor code changes (such as updates to website copy) without enlisting the help of a developer. This will likely require the product manager to develop some basic familiarity with the programming language used by their team, as well as the tools that their team uses to deploy code.

The challenge here is not so much for a product manager to be "technical," but rather to be comfortable exploring and learning about technical as well as nontechnical concepts. I have seen nontechnical product managers excel in highly technical organizations when they approach technical challenges with openness and curiosity, and I have seen nontechnical product managers falter in relatively nontechnical organizations because they see technical work as either uninteresting or unapproachable. The best product managers are just as curious about technical concepts as they are about nontechnical concepts. We discuss this in Chapter 3, *Showing Up Curious*.

Summary: Changing the Conversation About Product Management

Because product management is a relatively new discipline and because the role can vary so much from organization to organization, it is tempting to describe product management as a hybrid of other roles. Unfortunately, this approach

often results in a mismatch between what makes a product manager look good on paper (e.g., "a designer who knows some code" or "a developer with an MBA") and what makes a product manager successful in their day-to-day work. I hope that the CORE skills model will change the conversation about product management in theory to something that better aligns with the day-to-day work of product management in practice.

Your Checklist:

- Embrace the uniqueness of the product manager role. Don't try to be a designer, a developer, or a business analyst—and don't confuse the skills needed to excel at those roles with the skills needed to excel at product management.
- Pursue clarity over comfort to build your communication skills.
- Seek out opportunities to solve organizational problems on the systemic level rather than the individual level. If the rules aren't working, change them, don't break them.
- Don't let the day-to-day organizational conflicts of your work pull you out of your user's reality. Remember that what your company cares about and what your users care about are different things, and be a relentless advocate for the latter.
- Remember that there is no work beneath you, and no work above you. Be willing to do whatever it takes to help your team and your organization succeed.
- Even if you don't self-identify as a "technical" person, avoid saying things like "I'm not a technical person, so I could never understand that!" Trust in your own ability to learn and grow.

Showing Up Curious

When I started working as a product manager, I was incredibly intimidated by data scientists. I was never a "math person," and these folks were writing up complex equations on whiteboards and making nerdy in-jokes that I desperately wanted to understand. I spent the first year or so of my career as a product manager tiptoeing deferentially around the data scientists in my orbit, never really understanding what they did and assuming that they had no interest in explaining it to me. After all, these were, like, actual geniuses. Why would they want to waste their time taking me to data science preschool?

After a year or so, it became clear to me that this approach was making it much more difficult for me to do my job. Even though they weren't on my immediate team, our data scientists had a lot to offer, and I didn't even know what to ask for. So, in a moment of caffeine-fueled anxious desperation, I emailed somebody on the data science team to see if he would be willing to chat. It was a short email, that went something like this:

> *Subj: Coffee?*
>
> *Hey! Hope your week is off to a great start. I'm really curious to learn a little bit more about what you're working on—are you free to grab a quick coffee this week? Maybe Thursday morning?*
>
> *Thanks!*

I hit "send" and logged out of my inbox in an attempt to distract myself from creeping anxiety and embarrassment. Had I just done something totally weird?

Within a few hours, I got a straightforward response that included none of the overcompensatory enthusiasm of my original message. And that Thursday, we had our coffee—I'm hesitant to even call it a "meeting." It was a great conver-

sation that revealed both some mutual interests (turns out we're both musicians!) and some important insights about our work together. It turned out that this data scientist felt just as alienated from the product team as I did from the data science team. It was difficult to admit, but this disconnect was entirely of my own making. In assuming that other people had no interest in what I was doing, I had created the impression that I had no interest in what *they* were doing. Oops.

In this chapter, we discuss the single most important dimension of a successful product manager's attitude and approach: curiosity.

Taking a Genuine Interest

When people ask me how product managers can earn the trust of developers, or data scientists, or compliance officers, or any other person with a specialized and distant-seeming expertise, my answer is: take a genuine interest in the work that they do. "I'm curious to learn more about the work that you do" is the most powerful sentence at your disposal as a product manager, whether it's your first day or you've been working in the field for decades.

A simple gesture of curiosity can have an enormous and immediate positive impact on your work as a product manager. Here are three critical things you can accomplish by reaching out to your colleagues from a place of openness and genuine curiosity:

Learn "hard skills" contextually

No matter how much time you spend trying to learn "hard skills" such as data science or programming, you will never keep pace with the people whose actual job is to use those skills. You will learn more by asking those people about their work than you will by reading a book about data science or Python and then showing up to work trying to "talk the talk." Learning about hard skills from the people tasked with applying those skills ensures that you will learn about the specific hard skills that are most important to your organization right now—and that you're doing so in a way that directly strengthens your bond with technical folks.

Note that this approach applies equally to nontechnical specialized skills. For example, I've seen this strategy work very well for product managers who need to work with compliance experts at financial services companies. As a product manager, it is exceptionally unlikely that you will become an expert in compliance, but getting to know and understand the folks who *are*

experts in compliance will help you make better-informed decisions and work more closely with their teams.

Build bridges before you need something

If you only talk to people when you need something from them, nobody will be particularly happy to hear from you. The relationships you build with people when you don't need something from them will be there when you do need something from them, and the things you've learned about each other will help you collaborate more effectively.

Expand your network of influence

The people you reach out to each have their own networks of influence—people they talk with "off the record" and from whom they are willing to call in favors when needed. By reaching out to folks in your organization beyond the people you are working with every day, you're building a far-reaching network that might lead you to places you never expected.

In my experience, it has been exceptionally rare that "I'm curious to learn more about the work that you do" has been met with anything other than gratitude and some sense of relief. And in the rare instances where these inquiries have been met with mistrust or skepticism, it has provided a jumping-off point for a difficult but critical conversation about mistrust and skepticism in the organization at large.

So, take a moment and reach out to somebody who is not an immediate member of your team. Maybe it's somebody whose team you have worked with in the past, but you are not working with at the moment. Maybe it's somebody whose role you don't fully understand, but you think could have an impact on your work sometime in the future. Maybe it's somebody from a far-off corner of the organization whom you met during an offsite or other company event. There is no wrong person to reach out to—any step toward expanding your network of knowledge and influence is a step in the right direction.

Peeling Back the Layers of an Enterprise Organization

Amelia S.

Product manager, enterprise media company

When I moved from a small technology startup to a large media company, I fully expected that a company of such scale and size would completely have their act together when it came to product manage-

ment. But I found out very quickly that this is not exactly the case. Large companies have a veneer of formality that startups usually don't, but that doesn't mean that things are linear or predictable. Startups are often much more candid about their challenges: "This is messed up; let's fix it." At a large company, it can take months to get people to open up and speak candidly about the challenges they're facing.

Building that kind of trust involves using a lot of classic product manager tricks: having coffee with people, getting drinks with them, getting to know their work and their problems. Starting from a place of openness: "I'm new; I don't know anything. Tell me your problems and we'll figure something out." Nothing transactional, no quid-pro-quo, no "you'll scratch my back…." No expectations. People appreciate that kind of candor.

Part of the challenge at a big company is that the higher-ups usually aren't the people who understand what's really going on. They get reports from people who do the work. But I realized that in order for me to be successful, I needed to have core partnerships with folks within editorial, design, engineering—these very established orgs within the bigger organization. They have the organizational knowledge and history that I just don't. If you have a meeting with marketing, you need someone who can tell you, "Here's what's actually happening." You need to make sure that the human connection is there, and active.

I was genuinely surprised by how much of my on-the-ground work as a product manager at a large organization happened through backchannels. I thought that all the action would be taking place in a big conference room—but really, it's all about getting people's buy-in outside of those formal settings, which I didn't expect.

Cultivating a Growth Mindset

In her pioneering work on learning and success, Stanford psychology professor and author Carol Dweck posits that people operate in either a "growth" or a "fixed" mindset. When operating in a growth mindset, people see failures and setbacks as learning opportunities. When operating in a fixed mindset, people see failures and setbacks as negative reflections of their intrinsic worth. People operating in a growth mindset are able to approach skills and subject matter that

are new to them as an opportunity to, well, grow. People operating in a fixed mindset feel threatened by skills and subject matter that are new to them.

If you have spent most of your life being an overachiever, as many product managers have, it is excruciatingly likely that you are operating in a fixed mindset. Why? Because many overachievers find success not by growing their skills in areas where they struggle, but rather by avoiding these areas altogether. Fixed mindset overachievers dismiss as useless, irrelevant, or counterproductive the things at which they do not immediately excel. By way of self-incriminating example, I avoided taking any music theory courses in college because "formal training takes the soul out of music." But in reality, I avoided these courses because reading music was something that I found to be really challenging. It was easier to create a self-serving lie around a kernel of defensible half-truth than it was to simply admit that there were areas where I could stand to grow my knowledge and skills.

As a product manager, you likely cannot succeed if you are operating in a fixed mindset. There are simply too many new things that you will need to learn, and you will not even know what these things *are* until it is too late for you to give yourself an overachiever-y head start. Like it or not, you will need to acknowledge and address the limits of your own personal knowledge and skills if you want to do right by your team and your organization.

For example, imagine two product managers are facing the same challenge. They have spent the last couple months working on new mobile products for a large financial institution. The week before launch, both product managers receive a letter from their compliance department that their projects are not approved to move forward.

The first product manager is operating in a fixed mindset. He receives the letter and goes red-faced with embarrassment and anger. He whispers furiously under his breath, "my team is going to f****** hate me for this." But he also knows that his team has been through this before, and will probably be more than willing to lay the blame squarely on those jerks in compliance. The next day, he gathers his team together. "Well, guess what? Those jerks in compliance have done it again. Say goodbye to the last six months of work." The team is crushed. The product never launches.

The second product manager is operating in a growth mindset. She receives the same letter and immediately sends an email to the compliance department. In a nicely worded message, this product manager explains that she wants to make sure that she understands *why* exactly the compliance department could

not approve this product. The next day, she has a meeting with the compliance officer who wrote the letter. The product manager—who has no legal background—asks the compliance officer to walk her through the exact process by which compliance evaluated the product. In the course of this walkthrough, the compliance officer reveals that there was one specific user interaction that compelled them to reject the entire product. The product manager proposes an alternate approach, and both parties agree. The team learns an important new consideration for developing future products. And the product launches the following week.

If you are going to succeed as a product manager, you must be willing to engage deeply with people whose knowledge and expertise in a specific area greatly exceed your own. If you want to be the smartest person in the room, you probably will not succeed as a product manager. In fact, if your first impulse upon walking into a room is to defensively compare your own perceived intelligence to that of the people around you, you are probably not going to succeed as a product manager.

Understanding Goals and Motivations Beyond Your Immediate Team

A.G.

Product manager, 500-employee publishing company

When I was younger, I often became frustrated when I felt folks—usually in other parts of the company—weren't doing the right thing. I thought they were either morons or jerks, wielding power for fun and games. If there is one piece of advice I can give to people struggling with politics at work, it's this: assume people are smart and assume the best intentions. This is not some warm and fuzzy kumbaya mantra; rather, it's tactical, practical advice to help you survive and thrive as a product manager.

When I was working at a publishing company, I was tasked with building a product reliant on a large volume of content. Meanwhile, the VP on the content side was going out of his way to restrict the content we could use. I was livid. "This is a disgusting power play and he should be ashamed." My very wise manager advised me to talk to the VP. I did (calmly, surprisingly), and he indulged me with an explanation of how content acquisition actually works. If he gave me what I wanted, there was a high probability it would piss off our content partner. It might

actually put them out of business. He was protecting his relationship and the long-term viability of that source. Well then.

He wasn't a jerk. Not even a moron. I still didn't agree with the decision, but I understood why he made it and I definitely didn't have to get mad about it. His goals and, more important, his customers were different from mine. It was a very humbling experience.

Since then, I've worked in many other industries—retail, social media, food and drink—and this dynamic plays out time and again. Other parts of the business are often optimizing for different things: different goals and different customers. I know that product managers are often told to work closely with their immediate team, but in some ways, it's even more important to get to know people in other parts of the business. With your team, you still have the same set of goals and day-to-day concerns. But you may be totally at odds with another part of the business and not even know it. You may be thinking about the needs of your end user, but unaware of relationships with vendors and partners that are actually critical for keeping the business running.

And here's the thing: as a product manager, it is your job to figure this all out. Your role is cross-functional by definition, but everybody else's role isn't. You were hired to be a communicator, while somebody else may have been hired because they're really good at math or have great relationships with vendors. **Communication is your job, and you can't expect everyone else to be good at it.** My two favorite questions are: "What are your goals?" and "What are you optimizing for?" I use them often and with great sincerity, and my product manager life (and non–product manager life!) are much better off because of it.

The Gift of Being Wrong

Truly cultivating a growth mindset means being open not just to the unknown, but to being flat-out wrong. The most rewarding compliment I ever received as a product manager immediately followed one of the most difficult and contentious meetings I've ever attended. "You know," said a senior leader as we walked out of the conference room, "you walked into that meeting advocating for one path forward, and by the end of it you were advocating for something totally different. I'm really impressed by how you let yourself be convinced by the other people in the room."

Just a few years prior, this comment would have infuriated me. I had gone into a meeting with senior leaders to present my vision for the product, and by the end of the meeting I was effectively advocating for someone else's vision. In a very real sense, I abdicated any claim I might have had to being the company's "product visionary." But I also demonstrated to senior leadership that I was willing to go with the idea that seemed best for the company, even if it wasn't my own. For one of the very first times in my career, I had accepted the gift of being wrong.

This is not to say that product managers should be spineless and just defer to what other people want. For being wrong to be a gift, you need to know exactly *why* you're wrong—and choose to prioritize the overall goals you are working toward above your specific plan for achieving those goals. If somebody else suggests an approach that better reflects what you are collectively working toward, aligning with that plan gives you a chance to reinforce the entire group's commitment to your goals. And, as we discuss throughout this book, clarity around goals is perhaps the single most important organizational prerequisite to a product manager's success.

Spreading Curiosity

Good product managers bring curiosity and openness to their everyday work. Great product managers turn curiosity into a core value for their team and their organization. Genuine curiosity can be contagious, and naturally encourages people to collaborate more closely and better understand each other's perspectives. In a curious organization, negotiations between stakeholders feel expansive rather than combative, and deep conversations about goals and outcomes feel like an important part of your work, rather than an impediment to doing the "real" work. Curiosity makes everything seem more interesting and less transactional.

The first key to spreading curiosity is to model it yourself, relentlessly. "I'm too busy right now" is a very dangerous sentence for product managers. If your colleagues are taking the time to come to you with questions and thoughts, however trivial-seeming those questions and thoughts might seem, encourage that behavior. Similarly, if you're interested in something that a colleague is doing, don't feel bad about asking for some of that person's time—be confident in the knowledge that time spent learning from your colleagues is time well spent. And when you do need some time to go heads-down on a project and work in relative isolation, avoid saying things like "I just need a tiny bit of time to myself so that I

can actually get something done." Remember, the time you spend communicating with your colleagues *is* time spent getting things done.

Another great way to spread curiosity is to cross-pollinate knowledge and skills among your colleagues. If you're working with a team of designers and developers, ask them what other skills they'd like to learn. Maybe you have a designer who wants to learn more about frontend development. Or a web developer who wants to better understand the UX patterns of mobile apps. Make it as easy as possible for people to learn from one another as part of their day-to-day job. I've seen some product managers go so far as to declare one day a week "cross-functional pairing day," in which designers and developers (or developers working in different technical systems) pair with each other for the explicit purpose of expanding their knowledge and skills. Formal practices like this make it clear that you are explicitly valuing curiosity and knowledge sharing among your team.

Finally, organizing "demo days" and other opportunities for product teams to present their work to the organization at large is an incredibly valuable way to spread curiosity far and wide. I've been truly amazed to see how a team's work transforms when they are tasked with presenting to their colleagues once a week—people work harder, collaborate more closely, and begin asking questions about their own work in anticipation of those they will receive from their colleagues. The assumption that, for example, people in marketing couldn't possibly care about a highly technical product is replaced by the question, "How can we present this highly technical product to all of our colleagues in a way that seems interesting?"

Summary: Curiosity Is Key

Every organization is different, every team is different, and every individual is different. As a product manager, it is your responsibility to communicate, align, and translate between people who might have wildly diverging skill sets, goals, and agendas. The only way to do this is by taking an open, genuine, and curious interest in the work that they do. Learning about specialized skills directly from the people who use those skills within your organization is always more valuable than learning about them from a book or a Wikipedia. Indeed, every single channel of open and curious communication that you can establish is an important step toward the success of your team. We'll discuss this more in Chapter 5, "The Art of Egregious Overcommunication."

Your Checklist:

- Reach out to folks in your organization, ask to meet up for coffee (or over Slack, Skype, or whatever other remote collaboration tools you might use if you are not colocated), and say, "I'm curious to learn more about the work that you do."
- Be just as vigilant about getting to know people outside of your immediate team, and take the time to understand their goals and motivations before you need something from them.
- Cultivate a "growth mindset" and open yourself up to learning from people whose skills and knowledge exceed your own.
- Resist the urge to avoid situations that test the limits of your abilities or knowledge.
- Embrace "the gift of being wrong" by choosing the plan that best meets your organization's goals, even if it is not your plan.
- Shake up the work that people are doing and cross-pollinate knowledge and skills to keep your team curious and actively learning.
- Model the value of curiosity for your team and organization.
- Avoid saying "I'm too busy to deal with that right now" and other things that might implicitly discourage your team from asking open and curious questions that don't have an immediate transactional value.
- Encourage your colleagues to learn from one another, and pair up folks who want to learn about one another's skills.
- Organize "demo days" and other opportunities for product teams to share and discuss their work with the organization at large.

The Worst Thing About "Best Practices"

When I train product managers at large and small organizations alike, the first thing they usually ask for is "best practices." "How does Facebook do product management?" "How does Google define the difference between a product manager and a program manager?" "What are the things we can do to make sure we're running product like a best-in-class organization?"

These are great questions to ask, and the answers to these questions are great to know. But implicit in these questions is often an unspoken and counterproductive addendum: "How does Facebook do product management...because we want to do exactly the same thing."

The appeal of this thinking is not difficult to understand. Given the ambiguity around the work of product management, it makes perfect sense to look for guidance from the companies that in many ways defined product management in its current form.

But the dangers of this thinking are a bit more insidious. There are three particular ways in which I've found that a focus on best practices can actually make it more difficult for working product managers to succeed:

Focusing on best practices leads to an incurious mindset

> Reducing product management to a set of repeatable best practices means wishing away all of the messy, elusive, and absolutely critical human complexity that must be navigated in the role. Product managers who rely too much on best practices become deeply incurious about the people they work with—and sometimes even the product they're working on. Anyone

and anything that does not conform to the best practice becomes a threat to the one-size-fits-all approach that they are hoping will lead them to success.

Best practices often focus on operational stories, not stories of user value
: Most stories about product management best practices end with happy and efficient product managers, designers, and developers, but not (necessarily) with happy users. This focus on the operational success of product management best practices can inadvertently shift goalposts within organizations from "Are we successfully delivering value to our users?" to "Are we successfully doing product management the same way that another company does?"

Magical thinking around best practices inevitably leads to sadness and disappointment
: Initial conversations about best practices are often full of optimism and hope. But as these best practices inevitably run up against an organization's existing habits and rhythms, this quickly gives way to fatalism and frustration. Why aren't these best practices working for us? Whose fault is this? Who doesn't "get it"? These questions usually end with a grim and decidedly unhelpful conclusion such as "Our organization is just too hierarchical to be good at product management," or "The rest of the organization just didn't give us the support we needed to make these changes." The very things that make an organization unique wind up being an unstoppable impediment to change, rather than guiding the ways in which change is implemented.

None of this is to say that conversations about best practices should be avoided altogether—after all, this book is full of them! Let's look at a few important things to keep in mind when learning and communicating about best practices to make sure that they become valuable resources and not broken promises.

Don't Believe the Hype

Sometimes, when I'm fielding a volley of questions about how Company X manages to be so good at product management, I'll ask people to do a simple two-step exercise:

1. Write down all the things you've heard about how Company X is great at product management.

2. Spend five minutes using Company X's product, and write down all the product issues and problems that seem obvious to you—the ones you would want to fix on day one if you wound up working there.

The goal here is not to leave people feeling disillusioned and defeated, or even to suggest that "best in class" companies are missing obvious flaws in their products. Instead, the goal is simply to point out that the world looks very different from *inside* these companies than it does from the outside. Every company has political struggles, resource constraints, deeply held assumptions, and organizational inertia. No matter how many stories you've heard about how product managers at Google are locked in a constant free-snack-driven high-five with their developer counterparts, or how product managers at Facebook are given free rein to *literally push any code change to a billion users whenever they feel like it because startups, maaaaaaan*, the day-to-day challenges that product managers at these organizations face probably look a lot like the day-to-day challenges that product managers in your organization face.

To be blunt, most published case studies and articles about best practices are largely recruiting propaganda. Companies that are competing for product and engineering talent have very little reason to paint an accurate picture of their workplace situations, let alone an even remotely negative-leaning one. Rather than looking for a formal case study from a best-in-class company, talk to the working product managers in your network. Their stories and insights will likely align much more closely with the challenges you will face in your organization, and they will be much more realistic about the limitations and potential drawbacks of the approaches they've chosen.

"But This Worked at the Last Place!"

As product managers move between different organizations, they tend to amass their own set of "best practices" from past companies. These best practices are often the stories that product managers tell in job interviews: "We implemented this new Agile process, and were able to hit all of our release targets for the next year," or "We started setting rigorously quarterly goals, and were able to increase revenue faster than projections." And when a product manager starts at a new organization, there is often an expectation that whatever worked at the last place will yield similar results at the new place.

What often gets lost in these conversations is that the "best practices" that product managers pick up at a particular organization usually succeed for rea-

sons that are specific to that particular organization. Odds are, a lot of trial and error went into getting those best practices to a place where they could even be described or imagined as "best practices" at all. And, like it or not, that same process of trial and error, testing and learning, failing and adjusting needs to take place at *every* organization.

One of the biggest mistakes I've seen product managers make is showing up and immediately trying to make their new organization work exactly like their last organization. They implement so many changes at the same time that it becomes effectively impossible to observe and measure the effect of each individual change. And when the changes that worked for another organization inevitably fail to solve the specific problems of this new organization, a product manager can very quickly lose the trust of their team.

The best product managers always take time to learn about what makes an organization unique *before* they start implementing—or even suggesting—specific best practices. And when they do start implementing those best practices, they start small and build incrementally. By contrast, the worst product managers usually wind up blaming their colleagues when a fast and furious deluge of "best practices" fails to deliver the promised results. Here's a fun fact: the product managers who wind up getting frustrated because "The idiots at this new company just couldn't understand these better ways of working" are often the same product managers who talked about the idiots at their *last* company when they were being interviewed. Product managers who put abstract best practices above the people with whom they work tend to repeat this pattern over and over again.

A Slow and Steady Approach to Building Team Process

Ashley S.
Director of product management, ad tech company

When I started my job at a growing ad tech company, I was eager to apply the best practices I had used at my previous job. I showed up full of enthusiasm, ready to hit the ground running and to transform a disorganized group of high-performing individual contributors into a true software product team. And yet the team didn't seem to share my excitement. They acknowledged that they had a lot of room to improve, but they seemed deeply skeptical of the changes I was suggesting—changes that I had just seen work at another company. What was going on?

I was very lucky to have somebody on my team sit me down and say, "Think about it this way: maybe we want to start small, see what works, and then go from there." It stopped me in my tracks and I realized, "Wow, I know better than this." I was so full of genuine enthusiasm and wanting to make things better that I was ready to roll out everything I had seen work at my last company. But everything was different now; the team was different, the needs were different, company communication was different.

So instead of trying to rebuild the approach I used at my last job, I stepped back and tried to learn where the communication problems were at my new company. I worked with my team to figure out what needed fixing, and what steps we could take to make things better. We introduced changes slowly and steadily, constantly refining our approach based on what worked and what didn't work. One sprint, we would introduce daily standups. The next sprint, we would tweak the way we do our release notes. Slowly but surely, we built up a process that helped the product team collaborate and deliver better products.

So much of being a product manager is about going through the pieces that you think will work, and then feeling the pain of them when they don't work as intended. Once you know where the pain points are, you can start making adjustments. **It's not about changing everything all at once, and it's not about forcing a team to work in a certain framework or adopt a certain set of rituals. It's about really just iterating on your process constantly. When it doesn't work, you figure out why it doesn't work, and you try something else.** It's always a process to get to your process.

Goals First, Then Practices

One way to make sure you are implementing best practices in a manner that respects and aligns with the needs of your organization is to start with the specific needs and goals of your organization, and *then* think about practices that might help you achieve these goals. In the absence of this approach, you run the significant risk of implementing a change that is poorly understood, met with skepticism and resistance, and ultimately destined to fail.

Suppose, for example, that you just joined a team that is struggling to create space for communication between two geographically far-flung offices (a chal-

lenge you will read more about in Chapter 5). At your last company, you had a yearly offsite "product summit" that helped colleagues get to know one another in a more informal setting and align around a plan for the coming year. In your first meeting with the CEO at your new gig, you share this experience and say, "I know how hard it can be to get a distributed team aligned around product vision. I think that this 'product summit' could have truly transformative results." Your CEO is pretty far removed from the day-to-day challenges around collaboration—and who doesn't want transformative results?—so she agrees.

Two weeks later, you send out an email announcing COMPANYCO'S PRODUCT SUMMIT OFFSITE, to be held for a week at a swanky hotel in a location halfway between the two offices. You are expecting a hero's welcome from your colleagues, who you've heard grumbling about communication challenges since your first round of interviews. To your great surprise, the responses you receive range from muted to passive-aggressive. "My team has a deadline the following week; we'll sit this one out, thanks." "Is this the best use of our budget? I was asked to let one of my best developers go last year." Some people seem vaguely excited, some people seem frustrated, and most people seem confused and wary. You begin to hear some rumblings that this entire thing is just a pretext to announce company-wide layoffs. An executive from another part of the organization writes a furious email to the CEO asking why "product" is trying to further consolidate its power and influence at the expense of marketing and sales, both of which are already frozen out of critical roadmap decisions. The situation is, simply put, a big mess.

So, what went wrong? You recognized a problem, and you suggested a solution that you have seen work at another organization. But between two organizations, the same symptom can be caused by a very different disease—and the cure for one disease might make the other much worse. Maybe the reason that remote teams are struggling to work together at this organization has nothing to do with the lack of face-to-face time, but rather stems from a fundamental misalignment of goals and incentives. Maybe there are a few people in each of the offices who just don't like one another—and the idea of meeting up for an offsite is just about the very worst thing that any of them could imagine. Maybe the word *product* is interpreted by some people in this particular organization as, pointedly, "not marketing or sales."

If you don't take the time to truly understand the problem you are trying to solve, and how the organization's current beliefs and practices are contributing to that problem, then any best practice you implement is effectively a shot in the

dark. Here is a basic template you can use to begin understanding a specific challenge your organization is facing, before thinking about potential solutions:

My organization is facing the following challenge: This challenge is affecting our ability to deliver value to our users in these ways: I believe that this challenge is caused by the following current beliefs and practices:

This kind of structured thinking will help keep you focused on what challenge you're actually trying to solve, and can reframe operational problems as a question of lost user value. In some cases, taking this approach might help you realize you gravitated toward a particular problem not because it was directly affecting your users, but rather because you recognized it as a problem you've seen and solved before.

The Best Thing About Best Practices

With all that said, there is one particularly great thing about best practices that can serve as a crucial first step toward making positive change in an organization: because best practices often come with the halo of authority from a well-respected organization, it is much easier to get people to give them a try. Saying, "Let's try this weird thing where we write out three to five goals for the quarter and then also measure some things and call the things we measure 'results' even though they're actually more like 'indicators,'" will probably not get you far. But saying, "Let us adopt the Objectives and Key Results framework, just as they have done with great success at Google" sounds like a pretty reasonable thing to suggest.

Summary: A Place to Start, Not a Guarantee

Just remember: best practices are a place to start, not a guarantee of success. Keep a close eye on what is working for your organization and what can be improved and refined. And, above all else, keep the goals of any best practice you use in mind, so that you have a clear sense of what "working" means in the first place.

Your Checklist:

- Approach best practices as a place to start, not a prescriptive one-size-fits-all solution.
- Ask yourself how a particular best practice might help your team deliver user value, instead of just how it will change the way you work.
- If you're curious about how a particular company approaches product management, try to find some people who have actually worked there and ask them.
- When you are bringing a best practice from one organization into another organization, acknowledge and appreciate that every organization is different.
- Take the time to truly understand the goals and needs of your organization before rushing to implement any specific practices.
- Use a "slow and steady" approach to implementing best practices, so that you can test and measure the impact of every incremental change.
- Avoid the temptation to solve the problems that seem the most familiar to you, as opposed to the problems that are having the most impact on your users.
- Utilize the "organizational halo" effect of best practices to get buy-in toward trying new things, but be prepared to continuously adjust course based on what is working and what is not working.

The Art of Egregious Overcommunication

The title of this chapter is in many ways a joke. But for working product managers, it is also deadly serious. The biggest mistakes I've made as a product manager, and the biggest mistakes I've heard about from many of the product managers I've spoken with, involve failing to communicate things that seemed either too politically dangerous or too inconsequential to openly address.

Sometimes, things can seem both dangerous and inconsequential. For example, suppose that you're in a meeting with your team, and a developer rattles off a detail about a product that feels ever so slightly different from something you discussed in a separate conversation with a senior stakeholder. You begin to feel a little embarrassed—maybe you forgot to clarify an implementation detail with that senior stakeholder, or maybe you didn't do enough to communicate senior leadership's expectations to that developer. In any case, the last thing you want to do in this moment is pause the conversation to draw attention to your own possible mistakes. It's just a tiny thing. Nobody will notice. It would make *no sense* if this turned out to be a big deal! It's fine.

Cut to two weeks later. Your team is presenting a demo of this product, and the senior stakeholder in question begins to make a sour face. Her nose scrunches up and her eyes narrow. She mouths the words, "What is that?" juuuuuust distinctly enough for your heart to completely stop. Head shaking back and forth, she interrupts your developer mid-sentence: "I'm sorry, but this looks *very* different from what I signed off on. I'm very confused right now." Your team stops in its tracks. Everyone's eyes turn to you. After the string of expletives in your head concludes, you say to yourself, "You were afraid this was a big deal, it is a big deal, and now it's too late."

For most working product managers, this scenario is not a hypothetical. It happens all the time, and it keeps happening, even when you swear a million times over that you will never let it happen again. The potential downside of undercommunicating is cavernous and terrifying. The potential downside of overcommunicating is, realistically, a few eye-rolls and maybe some snarky comments. In theory, at least, this is an easy one.

In practice, however, the day-to-day work of comprehensive communication can prove very difficult. Choosing to communicate in the moment is much more challenging than choosing to communicate in the abstract. This chapter provides some tactical guidance for making egregious overcommunication a part of your product management practice.

Asking the Obvious

If there's such a thing as a ten commandments of product management, it's Ben Horowitz's "Good Product Manager, Bad Product Manager," a document Horowitz composed as a kind of ad hoc training for Netscape product managers way back in the days of the first internet boom. "Good Product Manager, Bad Product Manager" is a short and simple document, but it accomplishes something very important: it lays out in exceptionally clear terms the day-to-day expectations for product managers at that particular organization at that particular time. "Do this, not that." Every company should have a document like "Good Product Manager, Bad Product Manager" that spells out the day-to-day responsibilities of the job in clear, instructive, behavioral terms—and also explicitly names the behaviors that are to be avoided.

My very favorite part of "Good Product Manager, Bad Product Manager" says quite simply: "Good product managers err on the side of clarity versus explaining the obvious. Bad product managers never explain the obvious."

When I began working as a product manager, I wondered what exactly this meant—why would explaining "the obvious" be important? The answer, it turns out, is that the things that seem obvious are those that carry the most potential for disastrous miscommunication. This is in no small part because the things that seem obvious are often the ones nobody wants to address explicitly. Being the first person in the room to call out what everybody *thinks* they already know can be a risk both socially and professionally. Will people think that you are being condescending? Will you be wasting everybody's time? Or, even worse, will your own interpretation of the obvious prove to be deeply, embarrassingly wrong?

The answer to any and all of the above questions is often "yes." Asking the obvious can be, and often is, tremendously uncomfortable from a personal perspective. But from a team perspective, asking the obvious has precious little potential downside. The worst outcome for the team is that it turns out everybody was already aligned in the first place—which is not a very bad outcome at all. The best outcome for the team is that you reveal some underlying misalignment that would have proven disastrous down the road—in which case, your willingness to take on personal discomfort might have saved your team from a very bad situation indeed.

Meetings Are Good, If You Want It

Especially at startups and more engineering-driven organizations, product managers can come to be seen as the people who drag everybody away from their interesting engineering work and force them to attend meeting after meeting after meeting. Early on in my career as a product manager, I did my best to defuse this with a healthy(-seeming) dose of self-deprecation. "Here comes the product manager to drag you into another meeting...*ha ha ha ha ha*." I told the engineers I worked with that I would do my best to make sure they had to attend as few meetings as humanly possible. If an (ugh) meeting had to be scheduled, I would take it on behalf of the team—that was my job, not theirs.

For a while, at least, this did seem to improve my personal standing. But it also made it incredibly difficult for me to run effective meetings when there was a good reason for my team to work through something together. In effect, I had created a self-fulfilling prophecy that every meeting my team attended would be approached as if it were a waste of time—and, in turn, would *become* a waste of time.

In his book *Death by Meeting* (Jossey-Bass, 2004), Patrick Lencioni makes a critical point about meetings: if people approach them with a bad attitude, no amount of procedural tweaking will actually solve your problems. I've seen this happen at organizations that adopt rules like "no meetings unless a decision is being made" or "no meeting unless there's an agenda," only for these rules to actually make meetings more unwieldy and frustrating. If you're treating meetings as a problem to be fixed, they tend to become an unfixable problem.

Avoid the temptation of being president of the meeting-haters club. Don't apologize when you're asking for somebody's time—instead, make sure that everyone's time is spent well. Ask your team about the most valuable and well-

run meetings they've attended, and work with them to create a clear and achievable vision for what a "good" meeting looks like.

One Weird Trick for Better Meetings: Disagree and Commit

Maybe this has happened to you: you're in a meeting in which somebody is presenting a new idea for the company. Maybe it's a product idea, maybe it's an internal process improvement. Whatever it is, it is a very bad idea. You know it won't work. You've seen it not work. It is maybe the worst idea you've ever heard. But the person presenting it seems convinced that it's a good idea, and it's not really *your* job to convince him otherwise. He finishes the presentation and asks, "Any questions?" You look around, nod your head gently, and faintly mutter, "Nope." As the meeting adjourns, you smirk and think to yourself, "Wow, this is going to be a complete trainwreck."

Or, maybe this has happened to you: you're in a meeting, presenting a new idea for the company. Maybe it's a product idea, maybe it's an internal process improvement. Whatever it is, you're pretty sure it's a good idea. You're doing your very best to explain the idea as thoroughly as you can, and nobody seems to be objecting to anything. There's a bit of a weird energy in the room, but nobody offers up a dissenting opinion. You ask for questions. Nobody has any. As the meeting adjourns, you breathe a sigh of relief and think to yourself, "Wow, we're really on track with this one."

The preceding scenarios are prime examples of why "clarity over comfort" is so important for product managers. Far too many meetings adjourn without critical conversations taking place because nobody wants to publicly offer a dissenting opinion. Thankfully, the good folks at Intel pioneered a technique called "disagree and commit" that is designed to solve for this very set of problems. The idea behind disagree and commit is simple: the goal of a meeting should not be to get consensus, but rather to get commitment. How are these things different?

In theory, consensus means:

- Everybody agrees!

In practice, consensus means:

- Lots of people disagreed but didn't say anything.

- The decision being made was so important that nobody wanted to speak up and be held accountable for it.
- Everybody grew tired and just wanted the meeting to be over, so they agreed to...whatever it was the meeting was about. (What was that meeting about, again?)

By contrast, commitment means that everybody in the room is explicitly taking responsibility for any decision that is made in the meeting. Prompting people to affirmatively commit to a path forward rather than letting them slink out of the meeting with no sense of accountability naturally encourages people to share dissenting and complicating information. After all, nobody wants to be held accountable for a plan that they know is going to fail.

At its heart, disagree and commit is about achieving the following goals:

- Encourage people to share dissenting and complicating information that might prove critical in deciding upon a path forward.
- Avoid consensus-driven compromise solutions that placate meeting participants but fail to meet underlying goals.
- Force a clear decision, and create shared accountability around that decision.
- Allow participants to pick their battles by committing quickly to low-stakes decisions that are often prone to disagreement (i.e., "What's for lunch?")

As with any best practice, the way you achieve these goals will vary based on the particulars of your team and your organization. Here are some tips for trying out disagree and commit:

Introduce disagree and commit before you use it

Because disagree and commit is a formalized best practice—and one with the likes of Intel and Amazon behind it—you can introduce it as an agreed-upon procedural experiment. This is important because it will help avoid any situations in which people might feel like you are implementing disagree and commit as a kind of passive-aggressive personal criticism directed toward any particularly noncommittal members of your team.

Interpret silence as disagreement

This is a tricky one, and usually the moment in a meeting when it is clear that something different is going on. In most meetings, silence is interpreted as implicit agreement. Somebody suggests a path forward, concludes their pitch with "any questions?" and if nobody responds, it's more or less a done deal. With disagree and commit, nothing short of affirmative commitment is accepted, which means that silence amounts to *disagreement*. Be very clear with participants: "If you are silent, I am going to assume that you are disagreeing with me. Let's go through and have each person share your insights and concerns." The first time you try this, it might be one of the most uncomfortable moments of your product management career, but you'll be amazed at the insights that can emerge from the quietest people in the room.

Ask for affirmative commitment

Ask each person in the room for an affirmative commitment. Look them square in the eye and say, "Are you committed to this approach?" Make it clear that affirmatively committing to a plan means being held accountable for faithfully executing that plan.

Set clear goals, test, and learn

So, what if people simply won't commit to a path forward? This is, believe it or not, a great sign. It means that the people in the room are engaged enough that they will not commit to something that they think is wrong. One way to move this conversation forward is to establish success criteria and plan to revisit the decision at a later time, so that you can validate whether the approach you choose is working and make adjustments accordingly.

For example, suppose that you are in a meeting with your engineering team, and there is a disagreement about whether your product development cycle should be two weeks or six weeks. Rather than trying to get everybody to reach consensus—which might never happen—you could say, "What if we commit to trying two-week development cycles, and touching base in a month to see whether this decision is helping us meet our team goals or whether we want to try something else?" This ensures that a decision happens, and creates a shared sense of accountability for measuring its success and adjusting course if needed.

Don't completely misinterpret the entire point of this and say, "Well, it doesn't matter if you agree because we're doing disagree and commit!"

I almost can't believe I have to write this, but in a few cases, people have taken the idea of disagree and commit to the ridiculous extreme of flat-out barking at their colleagues "IT DOESN'T MATTER IF YOU AGREE WITH ME; WE ARE DOING DISAGREE AND COMMIT." This is an example of how "the worst thing about best practices" can come into play. If you don't understand the *goal* of doing disagree and commit and you are just approaching it like a magical ritual that will guarantee success, there is plenty of room for dastardly characters to take it in the wrong direction. Remember, one goal of disagree and commit is to encourage people to share dissenting opinions. If your implementation of disagree and commit leaves people terrified to share their point of view and berates would-be dissenters into submission, you are doing it wrong.

Using Disagree and Commit to Uncover Better Solutions

J.A.
Product management consultant

I was working with a small consultancy in California that builds products for media conglomerates. We were having a meeting to discuss internal processes, and the question came up of how to handle client emails that come in after working hours. This had clearly been a source of tension among the team, and most people fell silent when the question was posed directly.

Finally, one of the more senior people in the company chimed in with, "Well, it's important that we respond to our clients in a timely manner. So if you happen to see the email, then I guess you should just respond to it." Somebody else asked, "But what if two people are on the thread?" A third person volunteered, "Maybe what we can do is, if you see an email come in after hours and you plan to respond to it, first send a Slack message to the other people cc'ed on the message, tell them you've got it, and THEN send a response to the client." There were a few nods from around the table. A collaborative solution had been reached, and we had a path forward.

But the faces around the table were still tense, and a few people were still suspiciously quiet. I had just learned about disagree and commit, and this seemed like a good moment to try it. I told everybody in the meeting that they would need to affirmatively commit to this approach—and that if they were silent, I would assume that they disagreed with the approach. As we went around the table, most people offered a commitment to the tune of "Yes, I'm willing to try this for a while and see what happens." But one person—the person who had been quietest for most of the meeting—volunteered, "Yeah, I mean, this plan seems fine...but I don't see why we need to email them back that night. When a client emails me at night, I write back to them the next morning. And over time, those clients are kind of trained out of the behavior. They email me at the start of the work day rather than late at night, and everything usually works out better since I'm not rushing to get back to them."

The energy in the room shifted so dramatically in that moment—it was like a window blew open or something. People who had been willing to make a lukewarm commitment to the previous approach started sharing stories of late-night emails gone awry, bad decisions made in haste, and dinner plans ruined by unclear client management expectations. **The entire team committed—enthusiastically—to a new path forward, and that path would never have emerged at all if we hadn't taken a "disagree and commit" approach.**

Creating and Protecting Space for Informal Communication

Meetings are one important aspect of communication, but they are not the only one, by any means. In many product organizations, the most important and impactful conversations happen *around* the meetings, during informal conversations that play out over coffee breaks, during walks, or while spontaneously huddled around somebody's computer watching cat videos.

In many organizations—particularly very formal enterprises and high-pressure startups—there can be an implicit belief that anything other than "heads-down work" will be looked upon badly. This can create a situation in which there is simply no space for communication beyond formal, structured meetings. As a result, the natural flow of information between coworkers is cut off, and the informal "real talk" that often serves as a much-needed escape valve

in high-pressure organization is suppressed. A key part of a product manager's job is to hold and protect space for this informal communication to take place.

For example, I genuinely believe that the most impactful thing I did at my first high-pressure startup job was making room for a 3 PM coffee break. When a team is heads-down trying to meet deadlines, taking a 10-minute walk to the "good coffee place" can seem like a frivolity. However, those coffee breaks did things for the company that no formal processes could. They gave engineers and sales people a chance to talk to one another outside of contentious, transactional meetings. They gave members of my team a chance to vent about the new product manager who scoffed at their single-serving coffee pods and wouldn't shut up about the "good coffee place." (Sorry, former colleagues.) And, most important, they gave everyone who wanted it a chance to snap out of the moment-to-moment rhythms of their tactical responsibilities and take a broader view of their work, their team, and the organization at large.

Keep in mind that your organization's "3 PM coffee break" might not be a coffee break at all. Sometimes, it's a prework breakfast. Sometimes it's a monthly lunch. Here are a few tips for creating a regular time and place for informal communication in your organization:

Look at times and places when people are naturally motivated by something other than the day-to-day demands of their work

> Meal and beverage breaks constitute a great opportunity for informal communication, largely because they are times during which people are motivated by something other than the day-to-day demands of their work. There's a reason that the "water cooler" is the canonical location for informal conversation: it is a place where interactions are guided not by rank and silo, but rather by who happens to be thirsty (for water or for gossip) at that particular moment.

Avoid the most conventionally productive times of the day

> Does your organization experience a dip in energy in the mid-afternoon? Are people showing up early because they can't get anything done before they have that second cup of coffee? These might be great opportunities to regularly convene your team for informal, opt-in gatherings. People are much more inclined to see the value of informal communication when it is not competing for their most conventionally productive hours.

Don't force it!

Part of what makes spaces for informal communication so valuable is that they are informal—which is to say, not mandatory. Model the value of these informal spaces by showing up consistently and making clear that everybody is welcome. Casually mention to your colleagues that you've been showing up early to enjoy a delicious breakfast with people who you don't get to see as much as you would like during your day-to-day work, or taking an afternoon walk to give your eyes a break and think through your strategic challenges for the day. If you can clearly demonstrate how these informal spaces are creating value for *you*, your colleagues are likely to see the value as well.

Working with Distributed Teams

Creating and protecting space for communication is difficult enough when your team is all based in one place, but these challenges are greatly compounded when your team is distributed across offices, cities, and/or time zones.

Frankly, I don't think anybody has yet to fully solve the puzzle of distributed work. The tools and technologies available to distributed teams can help solve the logistical problems that these teams face. But addressing the fundamental communication challenges posed by distributed teams means taking into account the specific people with whom you work, the specific needs and practices of your team, and the specific goals of your organization. As always, there is no one-size-fits-all solution.

There is, however, one fundamental truth that must be acknowledged when you're working with a distributed team: no matter how many tools and technologies you throw at it, distributed work is not the same as colocated work. Joining a meeting via video chat is simply not the same thing as attending a meeting in person, just as attending a meeting at 8 PM is not the same thing as attending a meeting at 10 AM. Rather than treating distributed work as a "stand-in" for colocated work and trying to re-create it by proxy, approach it as its own way of working with distinct rhythms, advantages, and setbacks. Here are a few specific strategies I've used in working with distributed teams:

Keep it brief and keep it focused

One of the biggest mistakes I've made in managing distributed teams is holding looooooong meetings with some people participating in person and some people joining remotely. If you've never attended a four-hour

planning meeting as a remote participant, I sincerely hope you never have to. Losing interest in a rambling and unfocused meeting is easy enough in person, but it's even easier when that meeting is in a tiny window that you can mute while you check your email. Accept the challenge of remote participation as a "canary in the coalmine" of your colleagues' attention spans, and make your formal, scheduled meetings as focused and efficient as possible.

Document everything

Another challenge posed by distributed teams is that decisions made during informal in-person or one-on-one conversations might not find their way to your remote colleagues quickly or easily. Lucky for you, this is *also* a problem faced by fully colocated teams, where an idea hashed out over a coffee break can have serious ramifications for people outside of the immediate conversation. Working with a distributed team is a great opportunity for you to become more disciplined and rigorous about where and how you document your team's work.

Pick up the phone

Even though many people prefer remote participation via video, there's still something to be said for an old-fashioned phone call. Video chat can easily become one of eight hundred tabs open on a cluttered screen, but the telephone has an immediacy and intimacy to it that can be incredibly valuable. If you just got out of a meeting and you're not sure if a remote participant was able to follow along, pick up the phone and call that person.

Make space for informal communication

Perhaps the most challenging thing about working with a distributed team is finding some way to make room for the informal communication that occurs naturally when people are working in the same place. Again, there is no one-size-fits-all solution here. On some teams, an intentionally off-topic intraoffice chat room quickly becomes a thriving hub of company in-jokes, music recommendations, and opinions about reality TV. On some teams, it is perceived as a pointless distraction and quickly falls into a state of neglect and disrepair. Creating space for informal communication on a distributed team is not easy, and will likely require a good deal of trial and error.

Creating Space for Informal Conversations with a Distributed Team

Tony Haile
CEO, Scroll

When I was the CEO of Chartbeat, our whole team worked in the same room. This had its downsides in terms of hiring and ambient noise, but there was one major upside: incidental conversations. When you hear about teams performing well, trust is a huge part of that. And trust often develops through conversations outside of formal, scheduled meetings. I like face-to-face interactions, but with a distributed team this isn't always possible.

At Scroll, we have one office in Portland and one office in New York. This left us with an interesting challenge: how can we create room for incidental conversations when the team is distributed between two physical locations?

To help solve this problem and foster a sense of shared space, we created an always-on video link between our two offices. Now, when people show up to work, they can see all of their colleagues in the other office on a big screen right in the middle of the room. And if they have a quick question to ask or thought to share, they can press a button to activate an audio link. Rather than having to reach out and schedule a formal meeting, you can just slap the button and say, "Hey!" much like you would if you were actually working in the same room. Our inspiration came from Gawker Media, who had used a similar approach to connect teams in New York and Hungary.

I don't know how this approach would scale if we had a ton of people in different locations all working remotely. But for us, it has opened up space for conversations that simply would not be happening otherwise. It has made a real difference in terms of building camaraderie and an ease of communication among the team. And when you're building products, that is something you absolutely need.

Don't Deflect, Be Direct

Several years ago, I received an unexpected text message from my manager at about 9 PM on a Thursday night. It read, "Hey, would be really great if we could get that new release of the iPhone app submitted to the App Store tonight!"

I was confused. Was this a demand? A friendly but low-priority request? Was I being asked to do something *right now*? Or was I simply being shown a 113-character window into this person's vision of a better world? In most situations, I probably would have just dragged my Product Martyr-y self to my computer, grumpily submitted the app, and sent back a tellingly over-enthusiastic message like, "OF COURSE — NO PROBLEM!!"

But this time, I was actually out at a concert, a solid hour away from my computer. (And yes, shame on me for checking my phone during a concert.) Unable to fall back into my usual routine of passive-aggressive overwork, I stepped out and called my manager.

"Hey, I'm really sorry, but I'm actually at a concert right now. If you need me to go home and submit the app, though, I can totally do it."

The voice on the other end of the line was hesitant.

"Oh, umm, yeah, I mean, it would be really great to get it into the store tonight!" A pause. "But if you're out at a concert…I mean…you know what, don't worry about it, we can talk about it in the morning."

I let out a cheery "sounds good, thanks" and was immediately overcome by a deep sense of dread. Had I just overstepped some invisible line of work–life balance? Had I done something bad for the company for my own selfish purposes? Was I, as I had long suspected, a terrible, selfish person?

The next morning, I prepared to receive my punishment. My manager, though, seemed remarkably nonchalant. "Oh, yeah, I mean, I realized last night that it would have been nice to submit the app right away, but it's totally fine to submit it today, it's not like it's going to make all that much of a difference with the final timing."

In an uncharacteristic moment of emotional clarity, I said, "Okay. Can I ask you a favor? In the future, can you be very, very clear when you are actually asking me to do something *right now*? When I got your message, it was hard for me to tell how urgent the situation really was. If you ever urgently need me to do something, then I will do everything in my power to make sure it gets done. But if it's more like a 'nice to have,' could you just be as clear as possible about that?"

For about 10 seconds, I felt deeply proud of myself for being so direct. I then realized that most of the requests I was making to my colleagues still started with some version of "It would be great if…" or "Hey! Do you think you could, maybe, perhaps…." or "HEY NICE DAY HOW'S THE WEATHER I LIKE SANDWICHES DO YOU LIKE SANDWICHES anyways I was just wondering if you maybe had time to…?"

Because product managers rarely have direct organizational authority, it can be tempting to couch any requests for specific actions—especially actions like staying late to release a product or redoing work that was already completed—in the "nicest" terms possible. But being ambiguous about what you're asking for—and whether you're asking at all—is not nice. It is a deflection of responsibility, a passive-aggressive attempt to get the result you want without being the "bad guy."

The pull toward any and all kinds of deflection, overwrought apologies, and general self-deprecation is strong for product managers. But it is also damaging and dangerous—both to you and to your team. For many years, I used self-deprecation to slither out of situations in which I felt like people might be mad at me. When I was coming to my team with a tough deadline or a request for new work, I would often say something like, "Guess what, here comes the PRODUCT MANAGER with another FUN DEADLINE FOR EVERYBODY!" It felt like a good way to alleviate the tension, to show that I was "one of the team" and that I understood that the thing I was asking for was totally annoying. And most of the time, it got at least a little chuckle.

But the long-term effect it had on the team was neither good nor particularly funny. By using self-deprecation to spare my own feelings, I was doing absolutely nothing to communicate to my team *why* I was asking them to meet a tough deadline or revisit something that they thought was already finished. My goal was not to get the team aligned around our purpose, but rather to end the conversation as quickly as I possibly could. I was communicating, intentionally or not, that the work I was asking for was meaningless—because if I took responsibility for conveying its meaning, I would be the person asking for the work. And nobody likes the person asking for the work.

When you are a product manager, there will be times when you need to ask people to do things that they don't want to do. But as tempting as it is to deflect the heavy sighs and icy stares with self-deprecation or by blaming somebody else, it is up to you to turn those heavy sighs and icy stares into affirmative commitments and steely resolve. If you need something, ask for it, and be absolutely clear about *why* you're asking for it.

Accounting for Different Communication Styles

Creating and protecting space for communication, in meetings and in more informal settings, is one huge part of a product manager's responsibilities. But within that space, you are likely to encounter a whole lot of different people and a

whole lot of different communication styles. For many product managers, open and easy communication comes naturally—that's part of why they wound up in this role in the first place. And through that lens, people with different communication styles might seem like "bad" communicators, or even people who are actively resisting open communication because they have some kind of nefarious agenda.

As a product manager, it is critical for you to remember that not everybody is going to share your style of communication, or your inclination toward communication. Here are a few general patterns I've encountered that might help you take a more empathetic approach to teammates who initially come off as bad communicators:

Visual communicators

Some people simply cannot grasp a concept until they have seen it visualized. As a person who primarily uses words to communicate, it took me a long time to understand this. I would often become frustrated and just find myself using *more* words when my meticulously composed messages were met with blank stares. If you are not a visual communicator, visual communicators on your team can offer you a critical opportunity to refine and focus your own thinking by quickly sketching out or visually prototyping your ideas.

Asynchronous communicators

On numerous occasions, I have had somebody confront me after a meeting because they feel like I put them on the spot when I was simply trying to involve them in the conversation. Initially, I wrote this off as a kind of juvenile defensiveness. But I've come to accept that some people need to think things through before they talk them out. Whenever possible, give asynchronous communicators on your team a heads-up, and let them think through a particular question or challenge before sharing their thoughts. Also make sure that they know in advance that they will be asked to speak or present in a meeting.

Confrontation-averse communicators

In the day-to-day work of product management, receiving an uncomplicated "yes" or "looks good to me" can feel like a rare and precious moment of pure positivity and encouragement. But these encouraging "yes" answers are not always motivated by a thorough and nuanced evaluation of the question at hand. As a product manager, putting clarity over comfort is part

> of your job, but it is not everybody else's job, nor is it their inclination. If you need feedback from somebody whose first reaction always seems to be "yes," ask for that feedback in a way that does not allow for a yes/no answer. Accept that person's implicit challenge to be both more precise and more open in the way that you ask for feedback, and it will likely help you gather better feedback from everybody in your organization.

Of course, everybody's communication style is different. The more you can take the time to learn about the specific people on your team and appreciate their individual communication styles, the better you can facilitate communication for your team and your organization.

Egregious Overcommunication in Practice: Three Common Communication Scenarios for Product Managers

Though product managers might find themselves communicating with a lot of different people in a lot of different contexts, there are a few scenarios that tend to present themselves time and time again, organization to organization. In this section, we look at three common communication scenarios for product managers, and how you might approach each of them. After reading the setup for each scenario, you might want to take a moment to think about how you would be inclined to handle it. This will help you square these suggestions with the rhythms, personalities, and issues at play in your specific organizational context.

SCENARIO ONE

Account Manager: *We have to build this feature in two weeks, or we will lose our biggest client.*

Developer: *That feature will take at least six months to build if we want to do something that's even remotely stable and performant.*

Figure 5-1. An "emergency" request meets technical pushback

What's really going on

This is a classic and common case of misaligned incentives. The account manager's job is to retain customers. The developer's job is to create software that is not embarrassing, buggy, and held together with tape and string. The account manager is not directly incentivized to care about things like whether the software is performant. The developer, on the other hand, is not (usually) directly incentivized to care whether the customer is retained—if anything, one less unreasonable customer is one less set of last-minute demands. Both the account manager and the developer are advocating for their own respective short-term goals.

What you might do about it

There are multiple assumptions at play in both the account manager and the developer's positions here. Does this customer really need this *exact* feature? Will we *really* lose the customer if we don't build it? Does the developer fully understand the customer need, or is she using the six-month timeline as a more defensible way to say "no"? Rather than debating the specific feature that your account manager is asking for, dig deeper into the fundamental problem the customer is

having. Enlist the account manager as a partner in better understanding the customer's needs, and the developer as a partner in exploring possible solutions. You might discover that no new feature is required at all, just a quick conversation with the customer to help them better understand an existing feature.

Patterns and traps to avoid

> *Okay, well, let's decide whether we're looking at two weeks or six months...*

Both two weeks and six months might be totally arbitrary timeframes. The account manager might have said "two weeks" as shorthand for "really soon," and the developer might have countered with "six months" as a way of saying "hell no, I do not want to work on that." Avoid the false choice, and get to the heart of the issues.

> *Yes, I agree that we need to solve this in two weeks. And I agree that the software needs to be performant and stable.*

Don't try to play both sides! This will simply not work. There are likely opportunities to level up to a more goals-oriented conversation, and it is your job as a product manager to facilitate that conversation. Best-case scenario, you will discover a solution that takes less than two weeks and incurs minimal concerns about performance and stability. Keep the conversation open and exploratory, but don't try to score quick points by telling people what they want to hear.

> *Our planning process happens every two weeks, and we're all full up. Come back to me later.*

This is a tough one, and there are likely folks out there who have very good reasons for disagreeing with me. If you are working in a world of truly fixed iterations, last-minute additions like this are to be avoided at all costs. There are times —especially when there are no real financial deadlines for an organization— when using these guardrails is important. But when there is real urgency, it is usually better to have a process in place for addressing these requests, rather than shooing them away entirely. We discuss this more in Chapter 9, *Realistic Roadmaps and Painless Prioritization*.

SCENARIO TWO

Designer: *I made three different versions of this design—which one do you like most?*

Figure 5-2. A designer presenting multiple options

What's really going on

The designer might have created three versions that he feels are equally well suited to the goals of the project, but with subjective differences (such as color choice) where he doesn't have a strong point of view. Or, the designer might not be clear about the goals of the project, and is trying to defer responsibility by forcing you to make a choice. Or, the designer might have one approach he's really hoping you'll pick, and has made a few "dummy" options to create the illusion of choice.

What you might do

This is an opportunity for you to demonstrate your trust for the designer by asking him which option he feels best aligns with the goals of the project. If he feels that one choice is clearly superior, this prompts him to think about that choice in

the context of goals rather than preferences. And, if he doesn't have a strong preference, it might compel you both to have a conversation about whether the goals of the project are sufficiently clear, and sufficiently connected with the designer's day-to-day work. If multiple options seem equally viable, you might discuss with your designer how you could test these options to see which one best meets the goals of the project.

Patterns and traps to avoid

I like choice two—let's go with that!

This is an easy temptation—after all, the designer asked you what *you* think. In some cases, the question really is that simple—the designer doesn't care and just wants you to pick between a few subjective variations. But you're better off digging a little deeper than you are rushing to a decision without a clear set of reasons beyond your personal preference.

Let's bring all three of them to the whole group and see what they think!

For a long time, this was my strategy, until a thoughtful UX designer informed me that I was driving our visual designer to the point of quitting with "design by committee." Nothing is worse than having a whole bunch of people spout their opinions at you about the job you were hired to do.

I don't care. Whichever one you want is fine.

It's very rare that somebody puts in the work of doing something three different times unless there's a reason. Don't dismiss that effort—and the deeper issues that might be underlying it—by refusing to engage.

...And a bonus question

What if the designer only gave me one option?

Avoid the temptation to launch immediately into a critique, even if it feels like a generous critique. Instead, ask the designer to walk you through how he arrived at the design. This will give you an opportunity to learn more about how the designer understands the project's overall goals, and might reveal a subtle miscommunication or two that you can work to resolve.

SCENARIO THREE

Developer: *Sorry, I just don't understand why you're trying to force us to follow all this unnecessary process. Can you just let me do my job?*

Figure 5-3. A developer protesting "unnecessary process"

What's really going on

Although phrases like "This process is just way too heavy for us," or "I don't want to follow all of these unnecessary steps," or "This is big-company corporate bulls***" might seem like generic grumblings of the process-averse, they are all important and valuable signals that you have some work to do. If your team does not feel invested in your development process, and/or if they see that process as an impediment to getting their work done, you might have fundamentally failed in your role as a communicator and facilitator, even if you have succeeded in getting your team to formally adopt a certain development framework or methodology.

What you might do

First and foremost, take your developer's feedback seriously. Thank him for his candor, and make it clear that your team can succeed *only* if people are upfront about sharing their concerns. Rather than trying to resolve his concerns in an offline one-on-one conversation, ask if he can repeat this feedback during the next team meeting. This will help establish that you are not looking to be the brutal enforcer of your team's processes, but rather a facilitator who helps the team identify and adopt the processes that will best meet their goals.

Patterns and traps to avoid:

> *Just try it for a while—I promise it will make your life easier!*

There is a subtle but critical distinction between "I promise this will work" and "Let's collaborate to make this work." Asking anybody on your team to uncritically accept process changes is a fundamentally dismissive gesture, not a connective and supportive gesture. If your team does not feel invested in its process, that process is doomed to fail.

> *You're right. Forget this process stuff—what do* you *want to work on?*

Although giving engineers free rein to work on whatever they want might feel like an empowering or at least suitably deferential gesture, it ultimately leaves them deeply disconnected from the user- and business-facing impact of their work. Eventually, somebody is going to hold your team accountable for the actual results of what they build. And the longer your team goes without any kind of process to connect the work they're doing with the goals of your organization, the worse that day of reckoning is going to be.

> *I know, I know, I'm the worst, but my boss said we should put some more process in place. I promise I'll try to make this as painless as possible!*

Self-deprecation is a common coping mechanism for product managers. But if you cast yourself as an unwitting pawn in somebody else's quest to instill meaningless process, you are guaranteeing that the process will be meaningless. If you don't believe that the process you're using is the right one, but, hey, your boss asked for some process, it is time for you to have an uncomfortable conversation with your boss. We discuss this more in Chapter 6, *Working with Senior Stakeholders (Or, Throwing the Poker Game)*.

Summary: When in Doubt...

The day-to-day work of communication requires attentiveness, adaptability, and nuance. But the most important decisions you make as a product manager will often come down to this simple question: are you willing to bring up something that might seem obvious, uncomfortable, or both? The more fearless you are about starting these conversations—and the more space you create within your team and organization for these conversations to play out—the more successful you and your team will be.

Your Checklist:

- Err on the side of overcommunication. When you aren't sure whether something is worth mentioning, mention it.
- Don't be afraid to ask "the obvious." In fact, the more obvious something seems, the more insistent you should be about making sure everybody is in fact on the same page.
- Create a document like "Good Product Manager, Bad Product Manager" that clearly lays out the behavioral expectations for product managers in your organization.
- Avoid the temptation of being a "meeting-hater." Don't apologize when you're asking for somebody, but make sure that their time is well spent.
- Ask your teammates about the most valuable and well-run meetings they've ever attended, and work with them to set a clear vision for what a "good" meeting should look like in your organization.
- Make sure that people are given a chance to voice their opinions in meetings by using "disagree and commit" or any other approach that achieves similar goals within your organization.
- Create and protect space for informal communication in your organization, like team lunches and coffee breaks.
- Acknowledge that distributed and remote work is simply not the same thing as colocated work, and cannot be transformed into an exact proxy for colocated work through tools and technologies.
- Remember that people have different communication styles. Don't write somebody off as a "bad communicator" or assume that they have bad

intentions if they are not as open and extensive a communicator as you are.

- Avoid starting sentences with phrases like "It would be great if..." or "Do you think it might be possible to..." that deflect responsibility. If you are asking for something, ask for it—and be clear about why you are asking for it.
- Level up tactical conversations about things like design choices or development timelines to strategic conversations about goals and user needs.

Working with Senior Stakeholders (Or, Throwing the Poker Game)

The first time my dad met his future father-in-law, he was invited to join in for a friendly game of after-dinner poker. My dad, much like myself, is not somebody who has generally excelled at competitive rituals associated with male bonding. He also, much like myself, is not very good at cards. However, in this particular situation, he wasn't too concerned about his skill level. My dad's goal was not to win the poker game, but rather to make sure that his potential future father-in-law won the poker game. From what both of my parents have told me, this worked quite well.

I've thought about this story many times during my career as a product manager, especially when I've found myself sitting in meetings with people who have much more organizational authority than I do. In most high-stakes meetings—as in some high-stakes poker games—"winning" doesn't necessarily mean the same thing to everybody at the table. And when you are working with senior stakeholders, the best way to "win" is often to help somebody else win.

In theory, working with senior stakeholders should be no different from working with anybody else in an organization. In practice, this is rarely the case. For better or worse, senior stakeholders often wield the power of "because I said so." They can override your priorities. They can shift goalposts when you're midway through a project. Or, they can shift goalposts after you've finished a project, and then fire you for failing to meet your new surprise objectives. Senior stake-

holders will always win the poker game. Your mission, should you choose to accept it, is to ensure that your business and your users win along with them.

In this chapter, we look at some real-world strategies for working with senior stakeholders, a challenge often referred to in business parlance as "managing up."

Managing Up for Clarity at All Costs

Occasionally, somebody will ask me if I think there is a situation in which a product manager simply cannot succeed. I always have the same answer: a product manager cannot succeed if there is not clarity among senior leaders about a company's strategy and vision.

Though this can sound a bit fatalistic, it is downright empowering if taken the right way. Basically, this means that your job is to push upward for clarity at all costs. If you don't have that kind of clarity, there is no way for you to succeed in your role. Which also means that, if you don't have that kind of clarity, you have literally nothing to lose.

A lack of clarity about company strategy and vision might arise when multiple senior stakeholders are pushing and pulling for their own strategy and vision. Perhaps one senior leader sees the company pivoting, whereas another feels that it is important to stay the course. Perhaps there are multiple senior stakeholders implicitly competing for their boss's job, each working to sabotage and undermine the others. As a product manager, it is all too easy to become collateral damage in somebody else's quest to vanquish their organizational foes.

In many cases, though, the challenge is not that there are competing visions for the company, but rather that there is *no* vision at all. Showing up to work every day with no real goals or strategy to work against can feel liberating—you're free to do whatever you want! If the developers on your team want to spend three months refactoring code, *sure, we can do that*! But any true vacuum around vision is only temporary. At some point, somebody is going to step into that vacuum and ask you why you've been doing what you've been doing. And if you don't have a good answer, that person will likely not have a lot of faith in your ability to succeed as a product manager.

In keeping with our guiding principle of "no work beneath, no work above," this means that you must be willing to step up and work towards defining the company vision if nobody else will. If there are senior stakeholders pushing competing visions for the product, go as high up in the organization as you can to get a clear sense of what goals you should be executing against. If you have a CEO or

a departmental leader who won't commit to goals or clearly articulate a vision for the product, do everything within your power to get some face-to-face time. And then let that person take full credit for the vision that you helped create. Remember, CEOs or departmental leaders are going to win one way or another, and you are much better off if they win by the rules that you created together.

Having the Courage to Challenge Executive Decisions

Ashley S.
Product manager, enterprise electronics company

I was working on a product for a large electronics company, and my team was responsible for building a workflow and asset management tool. At the project's outset, we got word from a senior executive that we were to build our product around a specific piece of workflow management software that was being used in one of our European offices—the idea being that as long as one team was already using this particular software, we could just build around it and expand upon the core functionality.

As soon as we started talking to the company that had built this software, it became clear that the path forward would not be so easy. We kept hearing, "Our software doesn't do that." They couldn't even meet some of our basic functional requirements. At every roadblock, we kept moving forward with a series of ever-more-complicated workarounds. We often found ourselves asking, "Why was this technology even selected?" The answer usually amounted to, "We've already invested in this, so keep going." The longer the project went on, the harder it was to revisit that decision, in part because we had spent so much time developing workarounds specifically for the software we were instructed to use. We didn't show the product to users until it was "finished," and by that point, we had gotten so wrapped up in technical workarounds that we had failed to really understand our users' needs. Our users told us, "This sucks," and we said, "Yup." At the end of the day, we had to decide not to launch the product at all. So, all of that concern about sunk cost actually led us down a path where *everything* we invested in the product was just a straight-up loss.

If I could do it over again, I would have pushed back on the top-down decision around technology. **One of the things that I feel has contrib-**

uted the most to my career as a product manager is having the courage to push back, and to have challenging conversations. We are trained to follow the chain of command, so this can be very difficult when you're in a room full of executives. You have to first understand that their questions and criticisms are not personal. I've seen that in some junior product managers—they interpret questions and criticisms from senior leaders as personal attacks. You need to remove yourself emotionally. You need to have the courage to say, "Can I push back on this? Can we talk about why you're making this assumption?"

As a product manager, you're always being asked, "Why does this take so long?" You have to be able to unpack that question without getting defensive about it. Help senior leaders understand that the decisions they make aren't made in isolation—help them see the "invisible" work that they don't think about when they decide that they want a new feature. Give them options and make them aware of the trade-offs of each approach. Always make sure that the decision is in their hands, that they have ownership. That way it's not an us-versus-them situation—it's just us.

"Our Boss Is an Idiot," or, Congratulations—You've Ruined Your Team

When product managers fail to push upward for clarity, they tend to retreat into building cohesion within their own team at the expense of their team's connection with the organization as a whole. Early on in my career as a product manager, when a request came in from senior leadership that felt unreasonable, my first thought was always, "Oh no, my team is going to blame me for this." In an attempt to emerge blameless, I would end the conversation with senior leaders as quickly as possible, go back to my team, and say something to the effect of, "Can you believe how those idiots are jerking us around? WELP, I guess this is what we've got to work on now. *Cough* NOT MY FAULT."

In the moment, this can feel like the only way to keep the trust and respect of your team while also placating senior stakeholders. But in the long term, it never works out. The second you go to your team and say something like "our boss is an idiot," you have effectively ruined your team. They will begin to see any and all requests that come down from senior stakeholders as arbitrary and unreasonable. The time and energy they spend working on projects that align with organizational goals will feel like grudging concessions to the powers that be. And the

time and energy they spend working on projects that do *not* align with organizational goals will feel like "sticking it to the man." They will see your role as protecting them from senior stakeholders, rather than connecting them with senior stakeholders. And you will have backed yourself into a corner where the trust and support of your team hinges on you dutifully performing this role. In the interest of protecting and defending your team, you will have set them up to fail on the organization's terms.

So, what do you do when a directive comes down from senior leaders that doesn't make sense? You manage up for clarity. You explain to those senior leaders why it doesn't make sense, what trade-offs are at play, and what the downstream implications of their demands might be. Treat them like partners, not like idiots. If you feel that their suggestions don't align to a clear vision or set of goals, help them articulate that vision and define those goals. When you go back to your team, take ownership of the decisions that were reached. Don't blame the suits. Explain why these choices were made, share your point of view, and tie it back to goals and vision. Even if your team is frustrated in the moment, you are setting them up for success on the organization's terms, rather than pretending that you will somehow be able to win on your own terms.

The Dangers of "Protecting" Your Team from Business Goals

Shaun R.

Product manager, growth-stage ecommerce startup

I was working as a product manager at an ecommerce startup in London, and my team was responsible for creating a Black Friday sales page. Black Friday is a big deal for ecommerce companies, and the business had a very clear idea of what success would look like. We had an idea for a product that felt very closely aligned with our user needs, but relatively high-risk from a business perspective. I wanted my team to feel emboldened toward this more user-centric approach, so I didn't worry them with the specific goals the business had in mind.

Everything went along smoothly—until we actually launched the product. Even though the product we built did meet the underlying user needs as we understood them, it did not level up to the success metrics that the business had in mind. Had I been more upfront with my team about how the business was looking to define success, we could have built a solution that better aligned the needs of our users with the con-

cerns and constraints of our business. Instead, we wound up having to defensively rethink the product after it had already launched, on a tighter timeline and with a significant drop in morale.

In hindsight, this is a moment when I insulated the team rather than trying to surface the underlying conflict. I created a situation in which I saw myself as the filter for any problems between my team and the business at large, which can help you feel like you're protecting the tech folks from the nontech folks, and can seem manageable in the short term. **But when there's a fundamental mismatch between what the business wants and what your team wants, you can't resolve it by ignoring the business in the name of "protecting" your team.**

No Alarms and No Surprises

Several years ago, I was tasked with putting together a new roadmap for a company where I worked as a product manager. I spent countless hours slowly getting buy-in from all parts of the organization, hearing people's concerns, making adjustments, and putting something together that seemed both impactful and achievable.

After the meeting where our leadership team collectively agreed to this roadmap, one senior stakeholder pulled me aside. "You're such a creative person," he said, "I'd love for you to present a more creative option the next time we all meet." Oh *heck* yes! I put my "creative person" cap on and spent the better part of the next week putting together something *truly awesome*—the plan that I had really wanted all along.

The day before the next week's roadmap meeting, I sent that senior stakeholder an email, which, to the best of my recollection, was approximately 10,000 pages long. It detailed my plans for a brave new direction and thanked him for unleashing my creativity. I slept very well that night, confident that I had the blessing of one of the most senior and important people in the organization.

The next day's meeting was, in short, a bloodbath. No sooner did I begin to present my great new idea than another senior stakeholder jumped in with, "Wait, I thought we had already agreed to a roadmap last week? What the hell is this?" To my great shock and indignation, the very senior stakeholder who had asked me for a more "creative solution" then began berating me for taking the project so thoroughly off-course. I threw my arms up in exasperation and did my best to fight back my tears. *How could he do this to me?*

I was very angry about this for a long time. But in retrospect, I made no fewer than two huge mistakes in the way I approached that fateful meeting. First: I essentially betrayed the trust of every other person who had bought into the original roadmap I worked so hard to synthesize and socialize. Second: though I had laid out my "creative" product vision to this senior stakeholder in an inexcusably long email, I had no idea if he was actually supportive of it or not. These two huge mistakes added up to one colossal mistake: I surprised senior stakeholders with something completely new to them in an important, high-stakes meeting. And, to make matters worse, I did this at an important, high-stakes meeting full of senior stakeholders who I knew were wrestling with different visions for the future of the company. However much blame there was to go around, a lot of it fell on me.

The solution here is pretty simple: nothing that you are telling a senior stakeholder in a "big" meeting should ever be a surprise, ever. There are many reasons why it is always a good idea to individually walk senior stakeholders through a new idea *before* you present it in a group setting. But, to return to the heavy-handed metaphor at the heart of this chapter, a senior stakeholder is always going to win the poker game. And if you've taken the time to ensure that every senior stakeholder in the room is invested in your idea, then the odds are very good that, whichever senior stakeholder wins this particular hand, your idea will win along with them.

Getting Incremental Buy-in and Avoiding the "Big Reveal"

Ellen C.

Product management intern, enterprise software company

When I was working as an intern at a large software company, my first project was building a closed captioning system for a popular office product suite. I was provided a good amount of guidance going into the project: there was a clear business case, a set of well-understood regulations around implementation, and a pretty well-defined sense of what success would look like. I was able to manually test out a bunch of different approaches to execution, and run them by stakeholders. It went really well.

The second project I was given to work on was a system for commenting in that same office suite. I was really excited about this one. Although closed captioning had not been something that I personally

needed, I had a lot of ideas about what I could use from a commenting system. I had grand plans for making something really, really cool. I worked really hard on a specification that covered everything from the "why" to specific design and execution details. This was going to be my Big Win.

When it came time to review the spec I had put together, it did not go so well. It was terrible, actually. I was expecting everybody to be, like, "This is the greatest thing ever, why haven't we done this yet?" Instead, everybody told me all the reasons why it wouldn't work. A lot of things that seemed really obvious to me had contexts behind them that I just didn't understand. And because I was so emotionally invested, I didn't want to accept the feedback.

In retrospect, I made a mistake that I've seen a lot of new product managers make: trying to sell in everything at once with a "big reveal." I hadn't done the work of getting people to agree on the core user need, or presenting different possible paths to addressing that need. Instead, I just laid out, "This is exactly what we should do and why." People don't know where to give you feedback when you present everything all at once in a big meeting that way. If you go to people individually, they might say, "This is terrible—and here's how to fix it." But when you try to do "the big reveal," there's really no way forward.

Staying User-Centric in a World of Company Politics

Navigating company politics can seem like a lot of work—and in most cases, it is a lot of work. However, it is critical for you to remember that your success ultimately hinges not on your ability to make stakeholders happy, but rather on your ability to make users happy. If you build something that your boss and your boss's boss like, but which falls short of delivering any real value to your users, then you are failing to follow one of our guiding principles of product management: "live in your user's reality."

Here are a few tips for staying user-centric even as you navigate company politics:

Let your users make the case for you

When you find yourself trying to argue for specific ideas and approaches, remember that you are ultimately building a product not for your stake-

holders, but rather for your users. If you are doing the work of regularly talking with and getting feedback from your users—which you absolutely should be—you should have plenty of information at your disposal to bring your users' needs to life as you present to senior stakeholders. If you find that you don't have a clear sense of why your users might need the thing you are proposing to build, you probably shouldn't be advocating for it in the first place.

Connect user needs and business goals

It is not uncommon for product managers to find themselves feeling like they are advocating for "what's good for the user" against executive mandates to build "what's good for the business." But the biggest problem in this scenario is not an imbalance between user needs and business goals, but that these two things are perceived as being at odds with each other in the first place. If you feel like you are caught in a tug-of-war between business goals and user needs, the solution is not to pull harder, but to make sure that a clear and positively correlated relationship has been established between the needs of the user and the goals of the business.

When proposing a specific feature or product, be very exact and precise in explaining how you see the relationship between user needs and business goals. For example: "If we can make our onboarding experience faster and less burdensome, we believe that we can increase new user registrations by about 20%. Given that each new user is worth about $1.00 in advertising revenue, we see this as a critical step toward meeting our revenue goals for the quarter."

Flip the script and ask senior leaders about users

If you're looking to encourage user centricity throughout the organization, ask your senior leaders what *they* know about the needs of your users. Make it clear that your goal is to help them deliver value to users and meet the goals of your business. Invite them into a conversation in which you are collaboratively exploring multiple solutions to a well-understood user need, rather than debating a single, predetermined solution.

Avoid "compromises" that negatively affect user experience

In far too many cases, I've seen product managers make concessions to placate senior leaders, even when these compromises stand to negatively affect user experience. If you find yourself compelled to do something that is good for your stakeholders but bad for your users, it might very well

mean that the internal politics and incentives within your organization are fundamentally misaligned with the needs of your users. This is a perfect moment for you to push upward for clarity, rather than taking the expedient path of placating stakeholders at the expense of users.

The Not-So-Mysterious Case of the Disappearing Search Bar

M.P.

Product manager, nonprofit

When I was working as the product manager at a medium-sized nonprofit, I was tasked with overseeing a major site redesign. I knew that this would be a difficult task; the organization had a lot of senior stakeholders with very strong opinions about how their particular corner of the organization should be represented.

I put together a steering committee with the people whose direct sign-off I needed for the final design. I showed incremental work every week, and was able to keep up momentum on the project. Sure enough, there were a few times when people fought hard for their particular department to be featured more prominently, but we were always able to find a compromise that was amenable to the group. Miraculously, we were able to launch on time and on budget.

The project seemed like a huge success, until a few weeks later, when I actually had to use the website to find some information about an event we were hosting. From a product manager's perspective, the site felt like a success, but from a user's perspective, it was a confusing mess. The top-level navigation mapped out perfectly to the departments whose leaders I had to wrangle, but these categories made little sense from a user's perspective. And, worst of all, the search bar—which had no senior stakeholder advocating for it—was completely buried.

I realized in retrospect that I had become so wrapped up in keeping senior stakeholders happy that I had completely forgotten to advocate for the user's needs. Now, whenever I'm working with senior stakeholders, I make a point of starting with the user's needs before trying to reach any kind of conclusion, so that we're making the decision that's best for the user, not best for the egos of the people in the room.

Tactical Trade-Offs, Not Emotional Manipulation

Being user-centric in your conversations with senior stakeholders also means resisting the temptation to show off just how hard you're working, and just how much you're willing to personally sacrifice for the company. In many of my early meetings with senior stakeholders, I would find myself responding to questions like, "Do you think we could have this product launched by the end of next week?" with answers like, "Oh, jeez, well, I don't know, I mean, I'd have to work *all through the weekend* to get that done." These responses would usually stop the conversation dead in its tracks. And, really, there is no good resolution when you've opened up a can of "look at how hard you're making me work." Either the senior stakeholder is asking you something that seems unreasonable, or you are choosing to put your personal life above the needs of the business. Nobody feels particularly good in either scenario. And, more important, neither outcome is based on any kind of sense of what your users need and why.

Rather than playing up the effort involved in a task—which is a one-way road to becoming a Product Martyr—help enlist the senior stakeholder in a decision about tactical trade-offs. "My team and I currently have these deadlines ahead of us. To accomplish this task, we can move one of these deadlines back by one week. What do you think is most important for us to work on right now?"

But let's back up for a second. Remember that the original ask from the senior stakeholder in this scenario was in the form of a question: "Do you think we could have this launched by the end of next week?" Nobody was actually demanding that you do anything! In your rush to say "yes," you might actually find yourself asking unreasonable things of your own team rather than taking a question from a senior leader at face value. I've seen product teams embark upon grueling weekend work sessions because a senior leader asked an offhand question, and a product manager was too afraid to explain in clear and nonemotional terms what trade-offs would be involved.

Throwing the Poker Game in Practice: Two Common Scenarios for Senior Stakeholder Management

Let's look at two common scenarios you are likely to encounter as you work with senior stakeholders. These are both variations of the dreaded "swoop-and-poop," in which a senior stakeholder swoops in on work in progress and delivers some kind of criticism or out-of-nowhere demand for something different. Every product manager I know has experienced at least one swoop-and-poop. As with the

example scenarios from Chapter 5, take a moment to reflect on how *you* might handle this situation before reading on.

SCENARIO ONE

> Executive: *I just saw the work your designer is doing. I don't like the colors, and it doesn't look anything like the product I signed off on!*

Figure 6-1. A classic swoop-and-poop

What's really going on

The executive in this case feels like he's out of the loop. Something is moving forward that he doesn't recognize, and it implicitly threatens his sense of authority and control. Returning to our CORE skills, an organization-minded product manager might recognize this as a sign that something is fundamentally broken in the way that her team is communicating with senior stakeholders, and look for scalable ways to fix that disconnect.

What you might do

First, you might want to straight-up apologize. If the executive is seeing something that feels new to him, then something about the way you are getting product ideas and designs approved is not working as intended. Explain to the executive that you never want anything to feel like a surprise or catch him off guard. Ask what you can do to make sure that he can see works in progress on a timeline that makes sense for him—does he want to carve out a standing weekly meeting? Where in the process does he feel that he lost touch with the direction of the product? Look for ways in which you can address the fundamental problem, rather than trying to wriggle out of this particular uncomfortable moment.

Patterns and traps to avoid

> *This is the exact thing you signed off on. The only differences are purely cosmetic!*

Because you are speaking with somebody whose power and authority greatly exceed your own, you likely do not want to go into litigation mode. What is the real issue here? Is it the changes themselves, or the fact that this executive is seeing something that seems new to him?

> *I sent you the updated mock-ups last week and asked if you had any feedback, and you never responded!*

Anything short of affirmative and specific buy-in is not really buy-in. If your updated mock-ups were one of 10,000 messages to hit somebody's inbox and you received no response—or even a generic "looks fine" response—you might as well have not sent those mock-ups at all. Trying to escape on a technicality won't help you.

> *Okay, we'll change the colors to whatever you want.*

If you read the executive's comment carefully, you might notice that he is not actually asking you to change anything about the product. He is essentially describing a communication problem, not a product problem, and looking to change the latter won't fix the former.

> *Yeah, well, that's just, like, your opinion, man.*

Even if "the colors are all wrong" is just an opinion, you're better off not going down this road. Getting into a battle of opinions with anybody—let alone a senior stakeholder—is a bad move in the best of circumstances, and doing it in this context will rarely result in a positive outcome.

SCENARIO TWO

> Executive: *I know that your team is already working on something this week, but I'm really excited about that other feature we discussed a few days ago. Do you think you could find a little time to work on this as well?*

Figure 6-2. An executive swooping in with a new feature request

What's really going on

On the surface, this might seem like an executive trying to sneak her own pet projects into your team's tightly scoped work plan. But if you take this executive at her word, her motivation is *excitement*, not sabotage. If an executive is taking the time to come to you and express her excitement about a specific feature—even if it's one that you are not currently planning to build—this is a great oppor-

tunity for you to better understand her priorities, and how they align with your team's priorities (see Figure 6-2).

What you might do

Have an open and transparent conversation about what makes this specific feature so exciting to the executive *and* why it was not prioritized by your team as part of the current week's work. Maybe this executive was part of a high-level conversation that potentially shifted your organizational goals, and you simply were not aware of that conversation. Or, maybe this executive is really just super excited about this idea, and is not aware of the specific goals against which your team is executing. Be open to the possibility that the feature that this executive is suggesting might actually be more important to the organization than what your team is building. But, rather than breaking the rules and manually overriding the work your team has currently prioritized, talk with this executive about how you might change the rules to make sure that the most important work is prioritized moving forward.

Patterns and traps to avoid

Yes!

Immediately agreeing to a request like this not only undermines the existing processes your team uses to prioritize work, it also sets up the executive making this request for disappointment. Unless you have taken the time to fully understand *why* this feature is being requested, you are in no position to promise anything.

No!

If an executive has taken the time to come to you and share their excitement about a specific feature, there is almost certainly an important reason. Even if you ultimately plan to stand firm on your team's original priorities, take the opportunity to understand why this executive is so excited about this feature.

Maybe. We'll see how much time we have.

The fundamental question here is not whether your team will have time to work on the new feature, but rather why this executive is so excited about the new feature in the first place. If you make this merely a matter of capacity, you are missing out on a critical opportunity to better understand the goals and motivations of a senior stakeholder in your organization.

Summary: This Is Part of Your Job, Not an Impediment to Your Job

Working with senior stakeholders is a particularly challenging and high-stakes part of a product manager's job. Even when it feels politically dangerous, and even when it feels above your proverbial pay grade, you must do everything in your power to make sure that senior stakeholders are being clear and consistent in articulating company goals and vision. Don't be afraid to push for this clarity, and don't forget to keep your user at the center of the conversation.

Your Checklist:

- When working with senior stakeholders, don't set out to "win." Help empower them to make great decisions, and demonstrate that you can be a valuable and supportive thought partner.
- Push upward for clarity around company strategy and vision, no matter how challenging it is. In the absence of this clarity, you cannot succeed.
- Don't try to "protect" your team from senior stakeholders by talking about how ignorant, arrogant, or out of touch these senior stakeholders are. Instead, bring your concerns directly to these senior stakeholders and help walk them through making the trade-offs that will best serve your company's overall goals.
- Never surprise a senior stakeholder with a big idea in an important meeting. Socialize ideas slowly and deliberately in one-on-one meetings.
- Don't let company politics drown out the needs of your user. Let user needs guide your decision-making, and bring the user's perspective to life in meetings with senior leaders.
- Make sure that business goals and user needs are not seen as at odds with each other, but are instead aligned with each other, both for specific product initiatives and within the organization's overall vision and strategy.
- When senior stakeholders ask you questions like, "Can this be done by Tuesday?" take their questions at face value. Let them participate in making tactical trade-offs, rather than rushing to make yourself the Product Martyr.

- When confronted with a swoop-and-poop, don't try to litigate the details of past conversations. Look for opportunities to diagnose and address the underlying issues so that that the swooper/pooper does not feel out of the loop moving forward.
- If a senior stakeholder suddenly wants your team to work on something different, find out why. There might have been an important high-level conversation of which you were not aware.

Talking to Users (Or, "What's a Poker Game?")

Now imagine that you are at a very different poker game. You are working for an online gaming startup, and you've asked a group of poker players if you can sit in on their game to better understand their fundamental needs and behaviors around card games. After a round of introductions, your host asks, quite generously, "I'm not sure how familiar you are with poker. Do you want me to walk you through the rules?"

You might not realize it, but in this moment, you could be opening yourself up to a breakthrough insight about your users—or you could be completely shutting it out.

You pause for a second. You want these people to trust you, and if they think you're a total rube, why will they volunteer anything particularly interesting or insightful? With an air of knowing confidence, you offer, "Oh yeah, I love to play poker! Are we doing Texas Hold 'em or Omaha?" You get a cordial response of "Hold 'em!" and the game begins. *Whew*. They bought it. That time you spent on Wikipedia last night really paid off. As the first hand begins, you get right down to business: "Thanks so much for inviting me to your game tonight. As you know, I'm here doing some research about online card games. What would y'all want from your ideal online poker app?"

The people around the table are slow to answer. Nobody seems all that excited about the question. After a bit of a lull, the person next to you chimes in with, "I installed a poker app once, I think, on my phone, a few years ago. Didn't wind up using it much, though."

Yes. You're getting somewhere. "What didn't you like about it?"

"Well, I don't really remember, to be honest. I guess it just didn't hold my interest."

So close. "Why do you think it didn't hold your interest?"

"Uhh, I don't know. I guess they just didn't make the game very exciting."

You nod vigorously. *Nailed it.*

On the way home, you pull out your notebook and write down the following sentence:

People need an online poker app to be exciting.

You can see it all now: high-res graphics. Loud music. Explosions. The most action-packed online poker game ever made. That tag line is pretty good, actually. And, hey, when you ran that "what do you want from a poker app" survey a few months ago, "graphics" and "sound" both showed up as relatively high priorities. You are going to build the most successful online poker app ever.

Now, let's imagine you chose to take the other path.

After a round of introductions, your host asks, quite generously, "I'm not sure how familiar you are with poker. Do you want me to walk you through the rules?"

You pause for a second. You don't want these people to think that you're a total rube, but you also don't want your own assumptions about the game to get in the way of learning what the game means to *them*. Timidly, you respond, "You know, I'm actually pretty rusty. Would you mind walking me through it?"

A few people roll their eyes. You begin to feel flush (no pun intended) with embarrassment. But your host is up for it, and begins explaining the rules to you as if you know absolutely nothing about poker. As your host continues walking you through the game, some of the people at the table begin exchanging knowing glances with one another and chuckling. "I'm sorry," you say, "am I missing something?" "No, no," says the person next to you, "It's just that we've been playing this game together for so long, I guess we've made up some of our own little rules along the way. If you tried to play this game the way we do with anybody else, they'd probably throw you out!" Everybody laughs.

On the way home, you pull out your notebook and write down the following sentence:

Players changed the rules of the game to meet the particular needs and expectations of their social group.

You furrow your brow for a moment. This is really different from what you heard when you ran that "what do you want from a poker app" survey. You've uncovered something genuinely surprising—and something that might have major implications for the product you're building. You are left with a lot of questions you hadn't thought to ask previously. What role do informal rules play in card games? How is playing cards with strangers different from playing cards with friends? You aren't quite sure where these questions will lead you, but you sure are glad that you know to ask them before you launch a product.

Stakeholders and Users Are Different

In many ways, talking to users seems like it should be the easiest part of a product manager's job. I mean, how difficult can it be? Find some users, talk to them. BOOM. But talking to users can actually be the most difficult thing for a product manager to learn. Why? Because many of the behaviors and approaches that can help a product manager successfully work with stakeholders are exactly the wrong behaviors and approaches for talking to users.

When talking to your stakeholders, you want them to walk away feeling deeply invested in whatever it is you're building. You want clear and affirmative commitment. You want to connect high-level goals with specific executional details. You want to build alignment and camaraderie.

When talking to users, your goals are very, very different. You do not want them to walk away feeling deeply invested in a specific solution. You do not want a clear and affirmative commitment that they will use your product. And, if you approach your conversations with users hoping that they will simply validate your current product direction, you are setting yourself up to learn very, very little.

When talking with users, your job is not to convince, or to impress, or to align. Your job is simply to learn as much as you can about their needs, their world, and their perspective. If you are to follow the guiding principle "live in your user's reality," you must understand your user's reality in bright and vivid detail. In many cases, this means that your best approach is not to sound smart, but rather to "play dumb" and create as much space as possible for your users to communicate with you in their own words and on their own terms.

Yes, You Need to Learn How to Talk to Users

In some organizations, a product manager might be the sole "voice of the user." In other organizations, a product manager might be working with an extensive team of user experience designers and researchers who are tasked with conduct-

ing exploratory interviews, developing user personas, and overseeing usability tests. To my ongoing consternation and disappointment, I often hear about product managers who dismiss or devalue the work of their counterparts in user research. This, to me, is not only a dereliction of a product manager's connective duties, but also flat-out counterproductive. If you are lucky enough to have people in your organization who are trained experts in learning from users, why wouldn't you want to work closely with those people?

The answer, I think, comes back to the same insecurity that drives our five archetypes of bad product managers. "Owning" the relationship with your user can seem like an important source of authority for a product manager. And the specific tools and techniques deployed by user researchers can feel like implicit accusations that you don't really know as much about your users as you should. But the truth is, you could always stand to learn more about your users. And any new tools or techniques that can help you learn about your users should be taken as a gift.

I did not always subscribe to this belief. Early on in my career as a product manager, I straight-up threw a fit when a UX designer I worked with suggested that we create some user personas—composite profiles of user needs and behaviors designed to help us get out of our own heads and focus on the people for whom we are truly building. *Yeah, um, I don't have to make up a bunch of* fake *users because I understand our* real *users, thank you very much.*

For all of my attempts to sabotage and undermine the efforts of my persona-peddling colleague, I actually wound up finding this tool tremendously useful. Yes, we were building for "fake" people, composited from interviews with real users. But having anybody other than ourselves in mind as we designed our product assisted us in making better decisions. User personas—like any tool or technique—is not without its limitations and pitfalls. But reflexively dismissing specific tools for user understanding does nothing to help you understand their respective limitations and pitfalls—and guarantees that you will enjoy none of their benefits.

Long story short, you can always learn more about how to talk to your users. You should read books and articles about user research. (I've recommended one of my favorites in the appendix of this book.) You should seek out the user researchers in your organization and ask if they can mentor and guide you. Practice every new technique you learn whenever you can. Remain open and curious not just about learning from your users, but also about learning how to learn from your users.

"Quick Wins" for Better User Research

The extent to which a product manager is formally tasked with doing "in-the-weeds" user research can vary enormously from organization to organization and team to team. But informal user research happens everywhere—from coffee lines to family barbecues. Even if user research is officially somebody else's responsibility, there is always something to be learned from speaking directly with your current and prospective users, and it is always in your best interest to have a few quick and dirty tools at your disposal to ensure that you're getting the most out of these interactions.

Over the past handful of years, I have been extraordinarily lucky to work with some fantastic user researchers and ethnographers, including my business partner and research mentor Tricia Wang. The single biggest thing I've learned is that user research requires skill, discipline, and lots and lots of practice. That said, there are a few relatively straightforward tips and tricks that I have found particularly helpful when talking to users:

Ask about specific instances, not generalizations

This one comes up a lot in books and courses about user research, and it remains the single most helpful on-the-ground tactic I've absorbed. In practice, this means that rather than asking, "What do you usually eat for lunch?" or "What is your favorite food," you might use a prompt like, "Walk me through the last meal you ate." The idea here is that people will be able to more candidly and directly walk you through their own experiences when relaying a specific instance, as opposed to trying to synthesize a general statement about their tastes and preferences. I've found this tactic particularly useful when speaking to users about things like music and food, for which there might be significant value judgments associated with certain tastes and preferences. For example, people are generally pretty quick to talk about specific instances of listening to music ("I put on the new Kendrick album on my last run") but freeze up instantly when you ask them about their "taste in music" or "favorite artists."

Don't get too excited if you hear what you thought you wanted to hear

Sometimes, a user will outright volunteer the exact thing you were hoping they would say fairly early on in a conversation. When I began talking to users, I would often jump in with, "Wow, you know, that's exactly what we've been talking about doing! AWESOME! Thank you SO much!" It took me time—and some gentle redirection by my mentor—to see how this was

actually stopping me from getting a deeper understanding of the user's real needs. It is entirely possible that a user will describe the same solution you were imagining for entirely different reasons. And having a clear understanding of those reasons as you go into actually building the product gives you much more room to think through implementation details without compromising the product's core value.

Don't ask users to do your job for you

There is a great story that is often trotted out in design and design thinking workshops about the OXO measuring cup. When the company's researchers asked customers "what do you want from a measuring cup?" they rattled off a list of reasonable-sounding features. "I want it to be sturdy! I want it to have a comfortable handle! I want it to have a smooth pour!" When those researchers asked users to actually interact with a measuring cup, they saw a consistent pattern: after filling up the measuring cup, the user would squat down next to it so that their eye was level with the reading on the side. And so, the "read from above" measuring cup was born (Figure 7-1).

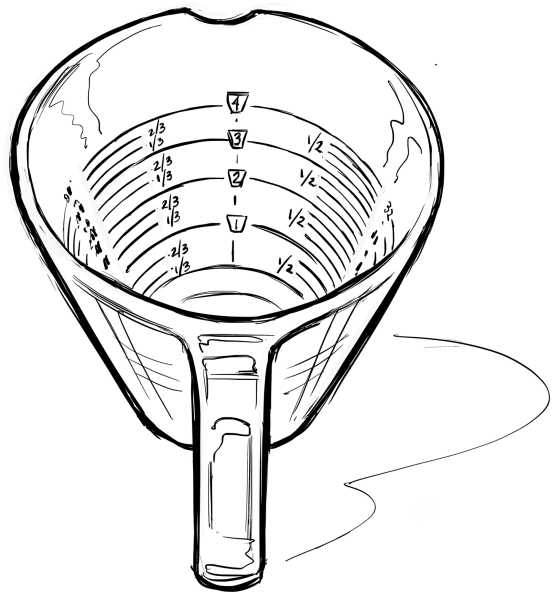

Figure 7-1. The "read from above" measuring cup

This story illustrates the pitfalls of asking your users to do your job for you. If you ask users to rattle off a list of features, you might feel emboldened going back to your team and saying, "I know this is a good idea—our users specifically asked for it!" But your users are not aware of all the organizational intricacies, all the internal trade-offs, and all the specific downstream concerns that go into building a product. In many cases, they are not even aware of their own needs because they are so accustomed to addressing those needs with the tools that they already have. Understanding and clearly articulating the needs of your users is your job, not theirs.

The Siren Song of "Power Users"

Jonathan Bertfield

Product manager, early-stage publishing startup

When I was working on an audience development tool for authors, we were very excited to get early prototypes and mock-ups in front of some real users. It seemed like we were in a great position to do so: we had great contacts in the publishing world, a well-respected leadership team, and a product that was solving a real problem for a small and well-defined user base.

We started reaching out to folks through our professional networks and were very encouraged by the response we got from some pretty high-profile authors. This was the early days of social media, and these were the folks at the forefront of using platforms like Twitter and Facebook to reach fans directly. They usually had their own employees or teams of employees managing their online presence, and they were eager for new tools. We had a clear signal that this product was destined for success.

At the same time, however, we started to hear some pretty clear dissenting voices coming from lower-profile authors and from professionals in the publishing industry. A lot of authors told us flat out, "I won't do that," and many professionals in the publishing industry were telling us, "Authors won't do the things that you are expecting them to do." But we didn't want to hear that; after all, we had more successful authors who were excited to use our product. We believed that the authors who did not think they wanted the product would change their behavior when they saw what we were offering.

This story does not have a happy ending: the startup failed. We just didn't get enough customers. **We had listened only to the people who were already successful, and those were not the actual target customers we were going after**. *Those* users were telling us very clearly that they did not have the time, the resources, or the understanding to do what we wanted them to do. But we chose only to listen to the people who were telling us what we wanted to hear—and when the product hit the market, we paid the price.

Leveling Up Versus Zooming In, or Another Way to "Why"

In the interest of uncovering high-level user needs and motivations, the word *why* can come in tremendously handy. But it can also be dangerous. In many cases, why can feel like an accusation. "Why are you doing things that way?" "Why do you like that band?" "Why did you go shopping on a Saturday morning?" The word *why* has a particular way of putting people on the defensive. And when people are on the defensive, they are generally not inclined to share very much with you about their needs, behaviors, and rituals.

So how do you ask why without asking "why"? In my own work, I've found it helpful think through the following prompt before asking a specific question (see Figure 7-2): is my intent with this question to *zoom in* on specific details, or to *level up* to overall goals and experiences?

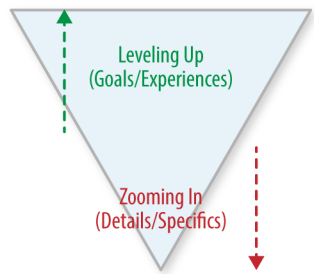

Figure 7-2. Leveling up versus zooming in

If a question you are planning to ask is designed to "level up" the conversation, you are effectively asking a "why" question—even if that question begins with "tell me about the last time..." or "what was it like to..." or "how was the experience when you...." Having a clear framework for thinking about whether

you are directionally moving the conversation upstream and out or downstream and in gives you a way to ask why without reflexively turning to that *exact* word in a way that might make your users feel defensive.

But the value of this framework goes beyond providing us with another way to ask "why" questions. Applying this lens to the questions I ask has helped train me out of one of the worst habits I displayed as a novice user researcher: asking lots and lots of "zooming in" questions for the sake of sounding knowledgeable and keeping the conversation moving.

Let's look at an example to see why this habit was so bad in the first place. Imagine that you are doing research on behalf of an ecommerce company to better understand people's needs and behaviors with respect to shopping. You ask a user about the last time she went shopping. She thinks about it for a moment and responds, "I went to the supermarket earlier today and bought apples."

In the interest of keeping the conversation moving and asking some questions, you might begin asking some very specific questions. "How many apples did you buy?" "What kind of apples were they?" "What did you do with the apples?" Before you know it, you might find yourself embroiled in a deep conversation about the finer points of Granny Smith versus Red Delicious. If you are simply trying to ask "why" questions, you might find yourself asking a question like "why are Granny Smith apples better than Red Delicious apples?" Although this question technically begins with the word *why*, it can easily come off as both accusatory and trivial.

The bigger problem, though, is that you are not there to learn about apples. If the purpose of your interview was to better understand user needs about shopping, and you spent most of the interview asking questions about apples, you are unlikely to discover anything new, interesting, and actionable. Instead, think about some questions you could ask to "level up" to the core issues you are trying to learn about. Use questions and prompts like, "What was your shopping experience like?" or, "Walk me through the whole shopping trip." In both of these cases, you are giving the user the opportunity to lead you to what *she* thinks is important, rather than zooming in on a specific detail that might or might not be significant to the user's overall experience.

Summary: No, Seriously, You Need to Learn How to Talk to Users

As all of these examples illustrate, talking to users is not something that comes easily or naturally to every product manager. Developing the skills needed to "live in your user's reality" often means unlearning specific behaviors that have

helped you to successfully manage internal stakeholders. And it *always* means taking an open and curious approach to anyone and anything that might help you to better see the world from your users' perspective.

Your Checklist:

- Talk to your users!
- Accept and acknowledge that talking to users is a real skill that takes time to develop.
- Remember that talking to users and working with stakeholders are different things, and require different approaches.
- Don't try to impress users with your knowledge or expertise. Create as much space as you can for them to explain their reality to you, even if it feels like "playing dumb."
- If there are user researchers in your organization, reach out to them and ask for their help walking you through the tools and approaches that they use.
- When talking to users about their experiences, ask about specific instances rather than broad generalizations.
- Don't ask users to do your job for you! Do everything you can to understand their needs, and *then* think about the specific products and features that might best address those needs.
- Use "leveling up" questions and prompts to get to core goals and motivations without an accusatory "why."
- Let your users lead you to what they think is important, rather than making that assumption for them with lots of detailed "zooming in" questions.

"Data, Take the Wheel!"

These days, it seems like everybody wants to be—or hire—a "data-driven product manager." And why wouldn't they? For a product manager, "data-driven" can be a handy shorthand for "in this ambiguously defined role full of squishy human complexity, I know how to do *serious data business things*." And for a hiring manager, "data-driven" can be a handy shorthand for "don't make any mistakes, ever." What could possibly go wrong?

In all seriousness, there is a lot to be gained from looking to user, product, and market data to help guide our decision-making, as opposed to, say, "whatever the last thing is that my boss told me." But if the goal of a data-driven approach is to guide us toward better decisions, how much of the driving can data actually do? Even if we are using data to inform our decision-making, we still need to know what we are trying to decide, why that decision is important, and what the possible outcomes of that decision might be. Not all decisions are created equal, and not all data will lead us to the best possible decisions for our business, our product, and our users.

At its best, a data-driven approach encourages us to use the information at our disposal to better understand our product and our users. At its worst, a data-driven approach encourages endless busywork that actually makes it more difficult for us to succeed as product managers. Here, again, we are compelled to follow the guiding principle: "live in your user's reality." If we spend all of our time as product managers with our heads buried in spreadsheets, charts, and dashboards, we are literally living in a different reality from that of our users. In the world of charts and dashboards, information is tidy, neatly categorized, and easily manipulable. In the real world…yeah, not so much.

In your work as a product manager, you will likely encounter a wide variety of off-the-shelf and custom data tools, dashboards, and widgets. In this chapter, we focus on the high-level, toolset-agnostic approaches that will help you to use data to your advantage without handing over the wheel.

The Trouble with the "D" Word

Let's begin with the word *data* itself. This word can be used to describe a *lot* of things. In theory, data describes objective information, whether it is qualitative or quantitative. In practice, I have often seen the word *data* thrown around to describe conclusions drawn from information, filtered and structured representations or visualizations of data—or "anything that kinda looks like a number or a chart." In its common and colloquial use, the word *data* does little to clarify what it's actually describing, while handily imparting an air of certainty and rigor. The word *data* is dangerous for the very reason it is useful: it wields authority without specificity.

To that end, I've often advised product managers I work with to implement a counterintuitive-seeming rule if they want to take a truly data-driven approach: don't use the word *data*. If you're discussing a particular set of information, describe that specific set of information. If you're discussing conclusions that you've made based on that information, describe those specific conclusions and how you reached them.

Take, for example, this somewhat-hypothetical sentence: "Our data shows that millennials are highly receptive to our value proposition." Now imagine rephrasing this as, "The email survey we conducted shows that millennials are highly receptive to our value proposition." There are still many points here that need clarification. (What is the value proposition? How does the email survey show this?) But at the very least, this rephrasing opens up a more meaningful conversation about what information was gathered, how it was gathered, and how it is being interpreted.

To use a more general example, imagine replacing the overused and often misapplied phrase "social data" with something more specific and descriptive like "sentiment analysis conducted on our customers' Tweets." The latter phrasing seems to invite more questions, but these are the very questions that make information both accessible and actionable. The absence of the d-word makes it easier to distinguish information from assumption, to have an informed conversation about methodology, and to set clear and reasonable expectations.

Don't Hide Your Assumptions—Document Them!

As we turn toward data to help guide our decisions, it is absolutely inevitable that we are going to have to make some assumptions. We might assume that the data we are working from is representative of the people we are trying to understand. We might assume that the outliers in a particular dataset are not important for us to consider. In many cases, we assume that the data that does not support our initial hypothesis is somehow less accurate, important, or worthy of consideration than the data that does support our initial hypothesis.

Perhaps the single most meaningful step a product manager can take toward being truly data-driven is to be completely upfront and transparent about these assumptions. In many cases, this means presenting data that contradicts our decisions as well as data that supports our decisions. This is by no means an easy thing to do, given that product managers are ostensibly supposed to know the user and know the product—not show up with a bunch of wishy-washy ideas and blunt admissions that their best guesses about the user and the product are still riddled with untested assumptions.

Here, our first product management principle "clarity over comfort" can offer crucial guidance. Clarity does not mean presenting a single absolute, uncomplicated point of view. It means being clear and transparent about the limitations and assumptions at play in any conclusions you draw, and any datasets from which you draw those conclusions. Doing this is, at times, very uncomfortable, especially when you are presenting your conclusions to people who are hoping that a data-driven approach will remove all uncertainty and risk. But choosing not to document your assumptions does not mean that you are not *making* any assumptions. Instead, it means that you are failing to draw attention to the specific assumptions that informed your decision-making, and in turn depriving your organization of an opportunity to address those assumptions.

Imagine, for example, that you are working at a ride-sharing startup that is planning to expand from America to Europe. You have been tasked with proposing the three European markets in which you think your product should launch. You want to be data-driven in your approach, so you seek out publicly available information that might help you make your decision. And *bingo*, Google's public data platform has some great figures about motorization rates in European countries. (Fun fact: it really does!) You know that, as a ride-sharing company, you need a pretty big fleet of potential drivers. So, you look for the countries with the highest motorization rates. Italy, Cyprus, Malta. Data-driven decision *made*.

The next week, you present your findings to senior leadership. "I consulted a canonical open data source," you say, "and based on that data, I concluded that the areas where we will have the most seamless market entry from an operational perspective will be Italy, Cyprus, and Malta." That sounds pretty legit!

Senior leaders, meanwhile, have their own concerns. "What about regulation?" one of them asks. "Who are the competitors in those markets?" asks another. You promise them that you will research their questions thoroughly, and you prepare to defend your brilliant data-driven decision-making.

There are, necessarily, a lot of assumptions that go into making any decision of this magnitude. But there is one specific assumption you made in your data-driven work that nobody thought to ask about: you assumed that a higher motorization rate was an indicator that a country would make a better market for your initial launch. In making this assumption, you also implicitly made the assumption that supply was a more important factor than demand in identifying your three initial markets. And when you presented to senior leaders, you gave them absolutely no opportunity to understand, confirm, or correct this assumption.

Of course, one of the most challenging steps toward documenting your assumptions is identifying them in the first place. Depending on how anxious and/or sneaky you are, you might be acutely aware of the assumptions you are choosing to gloss over in the interest of presenting an uncomplicated and definitive path forward. When I feel like I'm struggling to identify the assumptions at play in my work, I like to begin with the question, "what other things would need to be true in order for my interpretation to be correct?" But in some cases, the assumptions you're making might be so deeply held that you yourself are not aware of them, even if you set about to think them through carefully. This is one of the many reasons why it is so important for you to open up a conversation about assumptions with your colleagues. It is inevitable that there are some important assumptions that you will not identify—but your colleagues might.

One of the best ways to open up a conversation about assumptions is to build a formal template for every data-driven decision that provides an opportunity to document goals, assumptions, and questions. Here is a rough template you can start with and customize for the particular needs of your organization:

The decision I'm trying to make or problem I'm trying to solve:

The data I'm using to make this decision:

Why I believe that this data will help me make this decision:

What I believe the data is telling me:

What assumptions are present in my interpretation of this data:

How we might test those assumptions:

The next steps I intend to take:

Returning to our previous example, this template would force a product manager to explain why they believe that motorization rate is a helpful statistic for deciding which European markets would be suitable for launch. Even if the product manager does not identify this as an "assumption," this template asks them to reflect on the thought process behind their decision, and allows that product manager's colleagues to participate in the conversation in a more constructive way.

Again, you can customize this template however you'd like; what's important is that you frame documenting your assumptions as a critical part of working with data, not a misstep to be avoided. All work with data includes assumptions. It's up to you to make sure that your organization can navigate, discuss, and test those assumptions as needed.

Focusing on Metrics That Matter

With all the data available to you, it is critical that you have a strong point of view about which metrics matter and why. Benjamin Yoskovitz and Alistair Croll's book *Lean Analytics* (O'Reilly, 2013) provides an excellent framework for addressing this challenge: "One Metric That Matters." Although the idea of actually committing to a single success metric might seem implausible, it is incredibly valuable as a thought exercise. Do you know what the single most important metric for your product is right now? Can you clearly state why it is the most important metric? Are your company's overall strategy and vision making it easier or more difficult for you to know what metric you should be looking at and why?

Committing at the outset to what you are going to measure and why helps you to avoid what *Lean Startup* author Eric Ries describes as "vanity metrics." In short, these are whatever metrics make it look like you're doing a good job, even if they are not tied to your underlying goals. If you do not do the work of connecting specific metrics to your underlying goals, all of your metrics are essentially "vanity metrics," because you have no idea what they are really telling you or what you should do about it.

Suppose, for example, that you're a product manager working on a search product. You see a sudden decrease in daily page views. What does this mean? And what do you do about it?

This is a classic Google product manager interview question, and for good reason. If you are working on a product for which the goal is to get people the right information as quickly as possible, a decrease in page views might be a good thing. If you are working on a product whose revenue is directly proportional to its page views, a decrease in page views might be a very, very bad thing. The same metric can mean very different things depending on how it aligns with the overall goals and strategy of your product and organization.

Having a strong point of view about metrics might also help you discover things that you are not currently measuring but should be. If none of your current metrics align as closely as you would like to the overall goals you are working toward, this could be a critical signal that you need to change what you are measuring and how you are measuring it. More than once, I have worked on a product that was developed with little to no thought about how exactly its success would be measured, only to find myself working frantically to add basic instrumentation and metrics *after* the product has already launched. The sooner you develop a strong point of view about what metrics matter, the less likely you are

to find yourself having to retool an existing product because you cannot effectively measure its success.

Trusting Your Instincts to Find "Invisible" Evidence

Shaun R.
Product manager, B2B advertising software startup

When I started at my first product manager job, there were a lot of different things I could work on but not a lot of guidance as to where I should start. But the business itself was asking me to demonstrate that we should go down this route or that route. I got the sense that they were looking for hard data to support any particular product direction or prioritization decision.

At the time, we had a user interface that was clunky, but serviceable. If you spent time learning how to use it, it made sense. But, out of the box, it was very confusing. I had a strong suspicion that making a very simple, modern, usable interface that would effectively cut down training time and create more affinity for the product. But there was no specific data that proved that this was the right approach. I was hesitant in suggesting that we work on this because I felt like I was missing some kind of incontrovertible signal that this was the right thing to do.

After about three months, the business reluctantly accepted my suggestion to work on the dashboard, "Okay, there aren't any other easy opportunities, so go ahead and do it." But the further along we got, the more positive the response seemed to be. By the time we released the new dashboard and got feedback from our users, the response from senior leaders was, "We're surprised that this has had such a good effect!" I wanted to say, "I've been saying this all along!" But I realized—I actually hadn't been saying it all along. I was embarrassed about what felt like missing evidence, and did not yet know how to make a case for using data to measure changes moving forward, rather than looking for data that already exists.

From that experience, I learned how to start with a conjecture and say, "On this basis, we expect these metrics to be affected." How do you measure that something is working? Are you expecting an increase in sales? An increase in conversions? If you look at science—which has existed for much longer than product management—it relies upon the

initial conjecture to set up the experiment, that first hypothesis or leap of judgment. **Truly data-driven experiments often involve following your intuition, then establishing a feedback loop of some sort to test whether your intuition is correct.**

"Up and to the Right" Is a Signal, Not a Strategy

A few years ago, I was doing some training work with an ad agency that represents a large automotive brand. About halfway into the workshop, an executive stormed into the room and excitedly announced that the sales numbers for that automotive brand had exceeded their projections for the last quarter. A cheer went up throughout the room. "That's great," I said, "Why do you think that is?" The mood changed very quickly. The same executive who had been applauding just seconds earlier looked around and said, somberly, "I guess we've got some work to do."

We all want things to go up and to the right...right? But if we don't understand why we're doing well, we are powerless to build on that momentum, and completely helpless if the trend reverses. As product managers, we need to be as relentless about understanding the "why" behind metrics going in the right direction as we are about understanding the "why" behind metrics going in the wrong direction.

In practice, this often means that our quantitative data simply must be supplemented with some kind of qualitative data. If we see an increase in monthly active users, but we aren't actually speaking with any of our new users to understand what drew them to our product, we basically have no idea what is actually happening or what to do about it. Maybe a competitor just went out of business. Maybe an influential person shared a recommendation. Or, maybe our overall category is seeing significant growth, and our product is actually falling behind. The specific actions we might take are wildly different in each of these scenarios. And unless we take the time to dig deeper, any metric, even a metric that we decided upfront has clear and immediate importance to our business, is a vanity metric.

From "Accountability" to Action

In the past, I've often recommended that product managers ask to be held accountable for specific success metrics. In theory, this ensures that product

managers have a clear point of view about what metrics matter, and stay focused on what will move the product and the company in the right direction.

In practice, though, I've often seen this backfire pretty badly. When product managers are held directly accountable for hitting a specific quantitative number, they often disengage when they feel that number is out of reach. If you're being held accountable for a certain percentage increase in user growth, for example, and a competitor launches a product that you know is going to chip away at your market share, you might be tempted to just throw your hands up and get ready for an unpleasant quarterly review. And, as we discussed earlier in this chapter, you might be just as disengaged if you realize early on that the metric against which you're being evaluated is on a one-way ride to successville.

Therein lies one of the most uncomfortable and difficult challenges around data-driven "accountability" for product managers. If you are being held accountable for a specific number, but the fate of that number is ultimately outside of your control, how do you stay focused on the *actions* you should be taking to move that number in the right direction?

This is a tough question, and there is no obvious or all-encompassing answer to it. However, I have found one shift in framing that helps product managers feel more connected to what they can do, rather than what is being done to them: Rather than being accountable for *hitting* a specific metric, product managers can seek to be held accountable for the following things:

- Knowing which metrics matter and why
- Having clear targets for these metrics
- Knowing what is going on with these metrics right now
- Identifying the underlying issues that are causing these metrics to do what they are doing
- Determining which underlying issues can be effectively addressed by you and your team
- Having an action plan for addressing these issues

In other words, rather than attaching your value as a product manager to a number that might be outside of your control, ask for accountability in a way that makes you wholly responsible for seeking out and acting on the things you *can* control. If the numbers for which you are accountable are moving in the right

direction, but you don't know why or what to do about it, you are not doing your job as a product manager. And, if the numbers for which you are accountable are moving in the wrong direction, but you have taken the time to understand why and develop a plan of action, you *are* doing your job as a product manager.

Acknowledging the Limitations of Obfuscating Quantitative Proxies

What if there were a single number that could tell you everything you need to know about your product or your users? That would be pretty great, wouldn't it? It would certainly make your job as a product manager a whole lot easier. And, hey, it would also make that whole "One Metric That Matters" thing a lot easier, too, because there would literally be only one metric that matters!

Numbers and "scores" that purport to tell you everything you need to know about complex user behaviors and product health issues seem to be popping up in more and more articles, consulting proposals, and company reports. I have taken to calling these numbers and scores Obfuscating Quantitative Proxies (or OQPs)—which is to say, single-dimensional numerical answers for broader qualitative questions that obscure the complexity of the underlying issue. Here are some signs that you might have an OQP on your hands:

- OQPs purport to capture complex and widely variable trends or behaviors in a single number or score.
- OQPs generally disguise or omit the raw data from which they are generated, and the manner in which that data was collected.
- OQPs are presented as being equally applicable to all organizations regardless of those organizations' specific goals and needs.

That last point clearly differentiates OQPs from the "One Metric That Matters." While the idea of "One Metric That Matters" is designed to focus your efforts on the specific goals and priorities of your team and organization at this very moment, OQPs offer the compelling fantasy that you don't have to focus at all; the OQP will figure out what's important for you.

Again, this is a pretty appealing pitch. But behind every OQP is a set of decisions, assumptions, omissions, and limitations. Of course, you can't reduce complex human interactions and behaviors down to a single, easy-to-understand score. But you can summarize those behaviors in a way that might or might not be useful for solving specific problems and answering specific questions. If you

find yourself confronted with an OQP, I would suggest using the following template to assess how you might put it to use:

What is this OQP attempting to represent?
What raw information went into generating this OQP?
How was that raw information turned into this OQP?
What questions can this OQP alone not answer?
How might we answer those questions?
Based on the above, how might we use this OQP to meet our specific goals and needs?

This template provides you with an opportunity to assess both the limitations and the usefulness of OQPs, moving the conversation beyond "it's perfect" versus "it's garbage."

Let's take a look at an OQP that has been widely debunked but remains maddeningly popular: Net Promoter Score. Net Promoter Score is a number (from –100 to 100) that purports to represent how likely your customers are to recommend your product. The *Harvard Business Review* introduced this score in 2003 as "The One Number You Need to Grow." It is a widely used best practice among marketing and research teams all over the world.

Net Promoter is a classic OQP—a single score that purports to tell a complex and comprehensive story. But using the template above, we can begin to understand the real-world limitations of this score, and in doing so, gain a clearer

sense of how we might actually use it to our benefit. To answer the first question in our template, Net Promoter Score purports to capture the likelihood that your customers will recommend your product or service moving forward. Sounds good so far. But what raw information goes into Net Promoter Score? And how is that raw information actually turned into a single score?

The entire process for calculating Net Promoter Score is actually quite simple and straightforward for an OQP. You use a one-question survey to ask your customers how likely they are to recommend your product to a friend or colleague on a scale of 0 to 10. Then, you add up all the 9s and 10s, and subtract all the 0s through 6s. You take the resulting number as a percentage of the total number of customers who responded to the survey, and that's your Net Promoter Score.

So, what's the problem? Well, that all depends on how you're trying to use Net Promoter Score. If you simply want to measure high-level trends in likelihood to recommend—that is to say, whether people are more or less likely to recommend your product today than they were six months ago—Net Promoter might work out just fine. Whatever detail and complexity is lost in generating that score is presumably lost the same way whether you're running the survey now, six months ago, or six months from now.

But suppose that you're using Net Promoter Score to measure the likelihood to recommend of two different products—or the same product in different markets. Can you say definitively that a "7" for a toaster oven is the same as a "7" for a peer-to-peer marketplace that facilitates the sale of toaster ovens? Does a "7" mean the same to somebody in Dubai that it does to somebody in Dubuque? Is recommending something to a colleague the same—and equally valuable for a given product—as recommending something to a friend?

The goal of critically evaluating OQPs is not to unmask them as wholly useless, but rather to reveal the specific ways in which they can maximize our knowledge while minimizing our untested assumptions. OQPs often have real value as ways to summarize complex things enough to recognize broader trends that might become lost in the details. But they do not make the details any less relevant or impactful.

Keeping It Accessible

Early in my career, I was very easily intimidated by all things data-related. Any references to specific data tools, technologies, or concepts—even something as simple as the word *algorithm*—could leave me feeling hopelessly shut out of a

conversation. I spent many long nights scrolling through Wikipedia pages in the hopes of being able to contribute something to these conversations, even if it was just enough to prove that I had even the faintest idea of what these data people were talking about.

Eventually, though, it became clear to me that just being able to define these terms wasn't enough. As a product manager, it is your job to connect and align between people with specialized skill sets, including people who are experts in data. In many cases, these people were hired because they excel at a specific technical skill, not because they are expected to connect that skill with the needs of your users and the goals of your business. That is *your* job.

In practice, this means breaking down technical challenges around data into easy-to-understand and goal-oriented concepts that can be understood by anybody. If, for example, you are working with your engineers to choose between a "relational" database and a "NoSQL" database, take the time to walk through with them what the relative tradeoffs of each approach might be—in clear and descriptive terms that are tied back to the product's goals.

This becomes particularly important when you're working with data scientists—those who hold the "sexiest job of the 21st century" according to the *Harvard Business Review*, and who are often described as "unicorns" for their exceptional value and rarity. The technical jargon around data science can be particularly intimidating, precisely *because* it is placed at such a high value. And for this very reason, it is even more important for you to create a strong connection between the technical concepts deployed by data scientists and the plainspoken goals of your product and organization. In a truly excellent article titled "Recommendation Engines Aren't for Maximising Metrics, They Are for Designing Experiences," data scientist Michael Dewar explains what happens when a product manager does not recast tactical data science decisions in more accessible terms:

> *You will have left your design to the vagaries of the data scientists who will take their favorite metric (mine's a Hamming distance) and figure out how to apply that to your problem of return purchases, or time on page, or completion rate, or whatever.*

In other words, if you do not have a plainspoken conversation about the goals of a complex data-related initiative, you cannot assume that these goals are driving specific implementation decisions. As a product manager, it is your responsibility to make sure that technical conversations are opened up and made

accessible, so that people with different types of expertise can collaborate on deciding how to address a given user need. This is true not only of working with data systems, but of working with *any* kind of technical system you encounter as a product manager.

Turning a Complex Algorithm into a Fun and Accessible Game

Janet Bruncknorst
Principal product manager, Carbon Five

We were working with a client to build an app that could guide patients through a physical therapy regimen. The client knew a lot about physical therapy, but didn't know how to answer the overly technical questions we initially brought to them. We had been thinking, "let's just get these questions out of the way, and then our engineers will build it." But we really had a much more interesting challenge ahead of us: how could we take all of their subject matter knowledge, and transfer it into a technical system that we could actually build?

I sat down with our lead developer on the project and asked, "How could we implement this?" What we needed was some way for the computer to understand, "I did this, so next I should do that." And computers are very good at that. That's basically what an algorithm is. So, we interviewed a trainer on their end and walked him through a number of scenarios, asking what he would do in each situation. We kept giving him weirder and weirder scenarios, until it was like: "Your patient is overseas, and they are going hiking, and they can't call you, and they've got no internet, so what would you tell them to do?" And he was able to answer these questions because these questions were based on his work, and asked in his language. We weren't asking, "How would we design a computer algorithm to solve this problem?" We were asking how he would solve a bunch of problems, and then building an algorithm that would make decisions the same way.

We knew that we needed to validate the algorithm—the core set of rules we were working on—with the client before we actually built it. So, we put a bunch of rules down on index cards, and told them, "We're going to play a game with you, and see if the rules of the game lead to the outcomes that you think are right." For example, one rule might be: "Once you've successfully completed three sessions of one-legged squats, then

you progress to one-legged squats with weights." And as we walk through a particular scenario, the client might say, "Oh no, no, no, there's no way this person would be ready for that yet." So, we ask, "Okay, what should happen instead?" And then we update the rules and try it again.

This client was particularly open to a light-hearted approach, and I don't think this would work for everybody. **But we were able to build a really great product for this client by walking them through real scenarios that they could imagine, and not asking them to think about things in technical terms that they didn't understand.**

Summary: No Shortcuts!

The notion of data-driven product management can promise an almost magical-seeming, worry- and risk-free future. But in reality, data often becomes a black hole into which unasked questions and untested assumptions disappear. If used thoughtfully and thoroughly, data can be a critical tool for understanding your users and your product—but it won't do your job for you.

Your Checklist:

- Recognize that a data-driven approach still means that you will have to set priorities and make decisions.
- Avoid using the word *data* to generalize specific information. Say what that information is and how it was gathered.
- Rather than hiding or erasing the assumptions that go into working with data, document those assumptions so that you and your team can address them together.
- Have a clear and strong point of view about what metrics matter and why.
- As a thought exercise, ask yourself to decide on the "One Metric That Matters." If you're having trouble focusing in, go back to your high-level goals and see if you can make them more specific and actionable.
- Think through how you will measure a product's success *before* you launch it, to avoid having to go back and add instrumentation after a product is already released.

- Be just as curious and active about understanding metrics moving "the right way" as you are about metrics moving "the wrong way."
- Rather than being accountable for a number hitting a target, seek to be accountable for knowing why that number is moving toward or away from that target and having a plan for addressing whatever underlying issues are within your control.
- Resist the siren call of scores and numbers that purport to tell you "everything that you need to know" about anything. Take the time to understand how these quantitative proxies are developed, and do the work of figuring out what specific questions they can and cannot answer based on your goals and priorities.
- No matter how complex the data systems you're working with, resist the pull of jargon. Keep conversations about technical decisions rooted in high-level goals that can be understood by everyone in the organization to make as much room as possible for collaboration.

Realistic Roadmaps and Painless Prioritization

In theory, a roadmap represents what you plan to build in the future and prioritization is the process by which you decide which of those things is worth building right now. In practice, though, it is never that simple or straightforward. A roadmap can represent anything from an ironclad and airtight executional plan to a vague set of reassurances aimed at placating specific stakeholders. And when it comes time to actually allocate resources, even the most robust roadmap might end up thrown out the window if the organization's goals and needs have shifted —or if an important person complains enough.

In this chapter, we look at how product managers can use both a roadmapping document and a prioritization process as tools to connect and align, not to compete and fragment.

It's Not the Roadmap, It's How You Use the Roadmap

One of the best pieces of advice I ever received as a working product manager was to think of roadmaps as a strategic communication document, not as an actual plan for what will be executed and when. Unfortunately, I immediately misinterpreted this advice to mean that *everybody already understood* that the roadmap was not an actual plan for what will be executed and when. This got me into hot water more than once, as I had to explain to various stakeholders (ranging from engineers to [clears throat] board members) that the roadmap I had provided did not actually reflect what my product team was planning to build. After all, somebody really smart told me that a roadmap isn't a promise, but rather a strategic communication document. Didn't they get the memo?

If my misstep illustrates one key lesson, it is this: your organization needs to have an explicit and shared understanding of what a roadmap means and how it is to be used. Is it a hard-and-fast promise? A set of high-level "maybe" ideas? Are the next four years of your product roadmap as set in stone as the next six months? Unless you have taken the time and effort to discuss and clearly answer these questions, you likely do not have this understanding. If you are not sure whether you have this understanding, you do not have this understanding. And in the absence of this understanding, roadmaps are not only useless, but actively harmful.

Here are a few guiding questions to help you get started with creating a clear sense of how your organization intends to use its roadmap:

- How far into the future should our roadmap go?
- Does our roadmap make a distinction between "short-term" and "long-term" plans?
- Who has access to the roadmap? Is it user-facing? Public-facing?
- How often is the roadmap reviewed and by whom?
- How are changes to the roadmap communicated and how often?
- What criteria does something need to meet to be added to the roadmap?
- What could somebody within the organization reasonably expect if they see a feature on the roadmap three months from now?
- What could somebody within the organization reasonably expect if they see a feature on the roadmap two years from now?

The answers to these questions will vary based on your product, your organization, and your stakeholders. What is most important is not how you answer these questions, but rather that you ask and answer these questions at all.

Going from 0 to 1 with Organizational Roadmaps

Josh W.

Product executive, ad tech startup

When I started working as a product leader at an ad tech company, we didn't have a roadmap at all. I knew that we needed one, but I also

knew that a roadmap can be a very dangerous document. I had worked in sales previously, and I know that when junior sales people have access to a roadmap, they will use it to sell. That's not a bad thing by any means—they *should* be using everything they can to sell. But at an organization that had never used a roadmap before, there was significant danger associated with sales people seeing a roadmap as a set of promises when we were still figuring out what that document should even look like.

So, the first thing I did was ask if I could do a presentation about "thinking like a product person" at the sales team's offsite. I know that a sales offsite can sound like a nightmare to a lot of product folks, but it's situations like those that really give you an opportunity to bond with people outside of your role or function. Rather than trying to tell them that the way they approach things is wrong, I wanted to help them understand why product people might seem frustrated when requests come in from sales. I wanted to make sure that they knew where the product team was, and why the roadmap we were working on was a work in progress and not a series of promises.

When we finally did create a roadmap—and our first roadmap was, of course, a mess—I was very sure to label it "VERSION 0." I always use highly visible versioning to communicate when something is a work in progress, and nearly everything is a work in progress. When I shared this document with the head of sales, I communicated with him very clearly that this document was not to be used by sales folks as a set of promises, and I made him accountable for managing how the document was communicated with his team, and for any problems that would arise if it was used incorrectly. This helped ensure that sales people knew that they were accountable to their direct manager if they used the roadmap the wrong way. As a person on the product side, I had no direct authority, but the head of sales certainly did.

Every quarter, I would sit down with the leadership team and run a retrospective on both the roadmap itself and how we used it. By the third quarter, everybody could clearly see the value of having a roadmap, and we had gotten much better at understanding what information we needed on the roadmap and what information was extraneous or misleading. **I don't think we could have gotten there if we hadn't taken the time to truly retrospect on the roadmap itself, but also on how and why we were using the roadmap**.

"The Product Manager Owns the Roadmap!"

I recently had coffee with a friend of mine who had just started working as a product manager at a large education company. A few weeks prior, he had attended a training for new product managers that was intended to provide him with some more clarity on what would be expected of him in his day-to-day work. In walking through the high-level responsibilities that participants could expect to fall on their shoulders, the trainer stated, "The product manager owns the roadmap." My friend—whose willingness to ask an uncomfortable question speaks to his skill as a product manager—interjected, "What about product manager roles where you don't own the roadmap?" The trainer, genuinely confused by this question, responded, "No. The product manager *owns* the roadmap." My friend did not press further.

This story speaks to the enormous disconnect between a product manager's theoretical ownership of a product roadmap and their day-to-day ability to direct and influence that roadmap. Even when a product manager "owns" the roadmap on paper, this ownership is never absolute, easy, or uncontested. Long story short, everybody has ideas about what the company should build, and everybody feels that their success or failure at the organization is at least partially a function of their ability to get these things built. Although owning the roadmap might feel like a glittering prize for a product manager, it is more often a focal point for disagreement, perceived slights, and the writing of proverbial checks that cannot be proverbially cashed.

Here's an example of how this might play out. You recently started as a product manager, and you're eager to show your value by participating in high-level, strategic work. So, you start putting together a new version of the roadmap, culling together bits and pieces from other documents around the organization. You are going to create the one roadmap to rule them all, and in doing so, you are going to position yourself as the true keeper of the organization's product vision.

Unsurprisingly, everybody has a lot of ideas for the roadmap and is eager to meet with you. Sales and account management have major priorities that they need to get in there. Marketing has some ideas, too. You do your best to put something together that meets everybody's needs—or at least the needs that align with your vision for the product. You begin to feel kind of like roadmap Santa Claus, handing out spots in the roadmap to all the good kids and lumps of product coal to anybody whose vision doesn't align with yours (and maybe a few people who have personally annoyed you in the past). You are very strategic about who gets to see the roadmap and who does not. After all, the last thing you need

is somebody trying to add their own ideas to the roadmap without going through you first.

As you begin to execute against your roadmap, you run up against some product interdependencies that you hadn't fully thought through. For your product vision to succeed, you're going to need a few other product managers in the organization to make nontrivial changes to the way their products are designed. It dawns on you that these product managers are the very people you excluded from your initial roadmapping conversation out of concern that they would derail your vision or compromise your status as the true owner of that vision. You begin to get a little bit nervous.

You schedule a meeting with one of these product managers, prepared to beg and barter your way to victory. But it turns out that your efforts to exclude this product manager were even more successful than you realized. Unbeknownst to you, this product manager already had a roadmap for their own product—and the changes you need are not on it. You are suddenly hit with the realization that your "one roadmap to rule them all" is just one of many roadmaps—and all you're left "owning" is a great big mess.

As a product manager, your job is not to covet and defend the roadmap; rather, it is to open the roadmap to a shared, company-wide discussion about what you are building, who you are building it for, and why. As a rule, the product roadmap should be something that encourages collaboration and focuses that collaboration around high-level goals.

Answering the questions in the last section of this chapter is one step toward not feeling like a roadmap is something that needs to be protected and "owned." If everybody in the organization knows how a roadmap is being used, there is much less chance of it being used in a counterproductive way. As a rule, it is helpful to ask, "What steps do I need to take that will make me more comfortable sharing this roadmap with the entire organization?" rather than making excuses for why the roadmap would be disastrous if it ever got into the hands of a particular person or team.

Structure and Facilitate Ideas for the Roadmap, Don't Own Them

Many people are drawn to product management because they want to be the person who "has great ideas" for how to build or improve a product. The truth is, everybody at every organization has lots of ideas about their product. I have never once walked into an organization where there was an actual deficit of product ideas going around. Come to think of it, I don't think I've ever worked on execut-

ing an idea that was truly "mine" in the first place. The challenge for a product manager is not so much to have ideas, but rather to establish criteria that can be used to consistently evaluate ideas against user needs and business goals.

In practice, this means providing your colleagues with a productive way to share their ideas with the organization at large. Early on in my career, I tried to do this with a "product idea dump" document. This failed for two reasons. First, the very idea of a "product idea dump" spoke to the, uh, esteem in which I held my colleagues' ideas. Second, simply giving people a place to "dump" their ideas did nothing to structure their ideas, or to communicate how these ideas would be evaluated and why.

Rather than giving people a place to dump their ideas—the very idea of which suggests that those ideas are not to be taken very seriously—consider providing templates that structure these ideas. The specifics of this template will vary depending on your product and your organization, but here is a generalized place to start:

Product idea:
Suggested by:
Which of our users (current or prospective) this is for:
How this idea will improve their experience:
How this idea will help our business:
How we will measure success:

Even a simple template like this will help structure and filter the ideas coming in from your colleagues around user needs and business goals. I've found that templates like this turn "playing product manager" from an annoying thing your colleagues do into an effective filter against half-baked ideas, or ideas that unintentionally move organizational goalposts.

If you're worried about giving up your status as the company's idea person, worry not: the product ideas that you are most likely to execute successfully are *not* your own. If you are truly willing to throw your full weight behind somebody else's idea, you already have a crucial evangelist in the person who initially suggested the idea. You have left no room for people to assume that you are only pushing through an idea because it is your idea. And you have demonstrated that the ideas that best align with the organization's overall vision are the ones that will be built. If you can get past the need to be perceived as an "idea person," it is truly a win–win.

Your Product Spec Is Not Your Product

As you are working on your product roadmap, you might find yourself spending a lot of time describing at great length how exactly each item on the roadmap will look, feel, and work. Product specifications (usually referred to as "product specs"), like roadmaps, are strategic documents that help guide and facilitate the creation of your product. But they are not your product. Working on an extensive, beautiful, comprehensive product spec can feel like a productive and valuable way to contribute to a product's development—something you can actually do on your own time with your own hands, rather than just "facilitating" or "supporting." But your users see no benefit from that spec—they are using the product you have actually released. Putting *some* time into a product spec can be critical for making sure that you understand what you're building and why you're building it. But putting too much time into a product spec can create a false sense of certainty that actually makes it harder for your team to successfully collaborate.

One approach I've found very useful is to treat product specs like conversation starters, rather than hard-and-fast plans. If a question comes up about how we will build something, I include that question in the spec itself, rather than providing the answer that I think is best. Doing so communicates clearly to my team that they are co-creators of the product and trusted partners in deciding how to build it, not just "code monkeys."

Keeping this rule in mind can be useful regardless of how exactly you write your product and feature specifications. Many teams operating under the broad

banner of "Agile," for example, write out feature and product ideas as "user stories," using the format "as a [role], I want [feature] so that [reason]." These user stories are often accompanied by "acceptance criteria," which are the specific conditions that must be met for this feature to be deemed acceptable in its implementation. So, if you were planning to build a social sharing feature for a music app, you might write, "As a user, I want to share a playlist with my friend, so that they can hear the songs I think they will like." This user story might have acceptance criteria such as "User must enter a valid email address for the person with whom they are sharing," and "Playlist being shared must have at least one song."

At their best, user stories help keep product development solidly grounded in user needs. But they can also perpetuate the assumption that critical prioritization and research work has already taken place. Do we know that our users actually want to share a playlist? And how will launching this feature help us achieve our specific team and company goals? As with any highly prescriptive practice, writing "user stories" can inadvertently shift the goalpost from writing a useful and actionable product or feature spec to writing a formally correct product or feature spec.

Rather than trying to address every possible downstream detail in your product spec—or insisting that your product spec adheres perfectly to a format like user stories—make sure that you spend as much time as possible tying every product, feature, and idea back to a clear goal. This will help structure your thinking as it becomes time to plan and prioritize the actual work of building something, and it will give you a chance to reevaluate the roadmap if your goals change. Again, leveling up the conversation from technical details to higher-level goals will give everybody, including your technical partners, more room to find the best possible solution when the product or feature is actually being built.

The Unanticipated Ramifications of Complex Product Specs

Jonathan Bertfield

Executive producer, large publishing company

About 15 years ago, I stepped into a big job at a publishing company in New York. My title was executive producer, which was kind of like head of product management in those days. I was working on a really high-profile project within the company—it had launched an early beta that was in pilots, but it wasn't scalable. The job was, "Come in and turn this into a real business." I took that directive as, "Rewrite the specifications

for this project in excruciating detail." The documents we had were kind of all over the map and haphazard, and I assumed that *this* was why the product was not scalable. How could the product team possibly execute against this vision, if there was no blow-by-blow for them to follow?

So, I said, "Let's sit in my office with one subject matter expert and spend four months hacking together a detailed product spec." As a result of this approach, two disastrous things happened. First, we didn't speak to a customer for that entire time. What we ended up producing was way more complex than it needed to be, because we effectively stopped learning what it needed to be from our customers. Second, because we wound up writing such a complex product spec, we decided that we must need a really sophisticated development team. So, we got a fancy product agency to sign on, and threw the spec over the wall to them. And from there, I largely excused myself from dealing with the day-to-day work of building the product—after all, I had written everything out in the spec.

We finished the product, and it was a trainwreck. It took 18 months longer than it should have, and it was a total disaster. But the lessons I learned informed everything I've done since then. Writing stuff down is very much a double-edged sword. The more you write down, the more time it takes, and the further removed you can become from the actual work that needs to happen. **Writing a long and detailed spec might help you feel like you're doing a lot of work toward building a product, but it isn't always the right work**. As a product manager, you can never think that just because you did a brain dump, you are excused from the work of translating that brain dump into an actual product.

Wait, You Mean We Actually Have to Build This Now?

In practice, short-term prioritization is often a much bigger pain point than medium- and long-term roadmaps. Why? Because it's much easier to say, "We'll build that in six months" than it is to commit the organizational resources needed to actually build something in the next week, month, or time-boxed "sprint" of work. Roadmaps often become bloated, unwieldy, and unreliable because it is so very easy to say, "Sure, we can add that to the roadmap." But when it comes time for prioritization, you are working with a fixed amount of

capacity, and there are always more super-important things to build than there is time and resources to build it.

For this reason, it is often short-term prioritization, not medium- and long-term roadmapping, where your organization's goals are really put to the test. The bottom line with prioritization is this: prioritization will be as easy or as difficult as your goals are clear, well understood, and actionable. For that reason, prioritization tends to be a microcosm of organizational dynamics at large. Is your organization slapdash, disorganized, and chaotic? Your prioritization process will likely be the same. Is your organization noncommittal, bureaucratic, and accountability-averse? Welcome to your prioritization process. Prioritization is where organizational goals and vision turn into actual decisions about timing and resource allocation. As such, it provides us with a critical opportunity to see how well our organizational goals and vision are actually serving our day-to-day work.

This means that the most important step you can take toward effectively prioritizing might not be sorting through a million product ideas, specifications, and user stories. Instead, you must do whatever you can to make sure that your organization's goals are as clear and actionable as possible. The most important work of prioritization often happens before you actually sit down for a formal prioritization meeting, when you define and clarify the goals that will guide your prioritization decisions.

By way of example, suppose that your team has been tasked with prioritizing the next two-week "sprint" of work from a "backlog" of potential product and feature ideas. As you look through the product backlog in preparation for your planning meeting, you realize that it is totally unclear which of the things you could build is most important. What do you do?

You could send an uncomfortable email to your manager asking for some time to help clarify. Or, even better, you could approach your manager in person and explain that you need some help understanding your company's goals, and that you need to get this sorted before your prioritization meeting. This will likely not be an easy conversation. You might need to ask the same question several times. You might need to walk through different scenarios with somebody who just wants to get on with their day. But by the time your conversation is over, you will (hopefully) have a better sense of what your team needs to do to meet the company's goals and serve your manager's strategic vision.

Now, let's imagine that you're having the kind of day where you reeeeeeally don't want to mess with any of this "goals" stuff. "It's fine," you say to yourself,

"We'll figure out what makes the most sense during the prioritization meeting and work on that. If it doesn't align with our goals, it's my manager's fault for not being clear in the first place!"

Imagine the planning meeting that follows in both of these scenarios. If you were working as an engineer or a designer, which meeting would you rather attend? And, if you were a senior leader in the organization, which meeting would you rather let determine how your well-compensated employees are spending their valuable time?

Using a Simple Impact-Versus-Effort Matrix to Encourage Collaboration on a Remote Team

Janet Bruncknorst

Principal product manager, Carbon Five

We were working with a client who had a remote development team in another time zone. Though they were supposedly working in an Agile way, their process was still very disconnected: the product manager would write out a bunch of user stories in Jira, prioritize those user stories, and throw them over the wall to a remote team of designers and developers. This meant that the product managers were making a lot of assumptions about how easy or difficult it would be to deliver a particular feature, and it left the development team feeling pretty far removed from critical product decisions.

As the team embarked upon a particularly big and important project, it was clear that something had to change. So, we got all the product managers, designers, and developers together, and had a very different kind of conversation about what we would build and why. First, before addressing any technical or tactical concerns, we had an open conversation about the core user needs we were trying to address. Then, we asked the open-ended question: "How might we address these needs for our users?" We collected ideas, and plotted them on a very simple 2 x 2 impact-versus-effort matrix (Figure 9-1): how hard is this going to be versus how much of an impact will it have for our users? This gave the developers the opportunity to talk through the effort each idea would involve, and the product managers the opportunity to describe the user-facing impact.

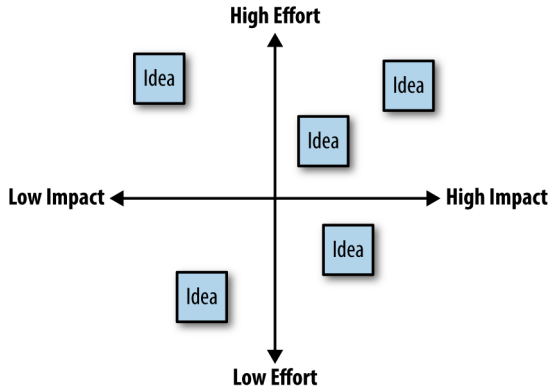

Figure 9-1. An impact-versus-effort matrix

By the end of this meeting, we had a major breakthrough. The product manager realized that the solution that they wanted the most—but assumed would be way too difficult—wouldn't actually be any harder to execute than the other approaches they had considered. We were all able to commit to a path forward that delivered the most value to the product's users, and made the best possible use of the developers' time. And the developers thought it was awesome, too! **They were able to participate in a conversation that defined the product, rather than just being asked to execute against a predetermined set of tasks.**

SMART Goals, CLEAR Goals, OKRs, and so on

There is no shortage of information out there about different ways to bring rigor and structure to your organizational goals. You've got your SMART goals (Specific, Measurable, Achievable, Relevant, Time-bound); you've got your CLEAR goals (Collaborative, Limited, Emotional, Appreciable, Refinable); and you've got your Objectives and Key Results or OKR framework. Which of these works best for you will depend enormously on your specific product, team, and organization—as well as your own preference for more numbers-based approaches versus story-based approaches. Even a cursory scan of SMART versus CLEAR can help you think through whether your particular team would be more receptive to "measurable" goals or "emotional" goals, for example.

In my own work, I have come to favor the Objectives and Key Results framework, primarily because it allows for both a qualitative rallying cry (objective) and

quantitative measures that indicate that you are moving in the right direction (key results). There are a number of great resources out there for learning more about the OKR framework, but I am particularly fond of Christina Wodtke's book, *Radical Focus* (Boxes & Arrows, 2016). Wodtke writes in vivid narrative detail about the common pitfalls that teams encounter when implementing OKRs, and makes an incredibly strong case for the subtractive and focusing power of having clear and well-understood goals.

One quick caveat for product managers looking to implement the OKR framework: every single time I've seen a product manager try to bring OKRs to their entire organization—with the exception of very, very small and early-stage startups—I've seen it struggle to take hold, and usually outright fail. The "by-the-book" way to do OKRs is to implement cascading goals at the individual, team, and organizational level. But building a solid practice around OKRs for your organization—changing the rules, not breaking the rules—takes time and experimentation. My suggestion is to begin small and simple, by reframing your team's goals, seeing what works and what doesn't work, and using those learnings to scale the practice beyond your immediate team to the organization at large.

Later in this chapter, we'll take a look at an example of how rewriting goals in a lightweight OKR-like format—or, really, any format that encourages discipline and rigor—can help make prioritization much easier.

Taking Your Goals for a Test Drive

Regardless of what framework you and your team use to articulate your goals, there is one sure-fire way to determine whether they are easy to prioritize against: take them for a test drive.

When I was consulting for a growth-stage startup, I made a point of sitting down with the CEO every quarter to review the organization's goals before they were shared with the product team. (The mere fact that the CEO was writing out a document with definitive and transparent quarterly goals put this company ahead of many other similarly sized companies I have encountered.) The way we reviewed these goals was quite simple: We went through a backlog of potential product and feature ideas to see whether the goals that the CEO had written out would clearly tell us what we should work on next. If we had one project that would increase revenue and one that would potentially bring us new users, which one was more important? If there was a long-standing user complaint about an

existing feature, how could we quantify its value compared to a shiny new feature?

Working directly with a senior leader to test-prioritize feature and product ideas against goals has a number of important, positive effects beyond simply improving the quality and clarity of the goals. First, it helps senior leaders feel connected to the product decisions being made by your team, making it less likely that they will question or override your decisions down the road. Second, it gives your team more leeway to change executional details if needed, because you have asked someone in a leadership position to commit to goals, not to solutions. And finally, it opens up a channel of helpful and mutually constructive communication between you and senior leadership, which can prove very useful for many different reasons.

Making the Connection Between Product Roadmap and Company Strategy

Blair Reeves

Senior product manager, enterprise technology company

When I was working in the B2B digital analytics group of a large technology company, we acquired a mid-sized company with a complementary hybrid software-hardware solution. This acquisition made a lot of sense, and we initially had some really exciting ideas for how we could integrate the two products to better serve our customers' needs.

As soon as the acquisition went through, though, we realized that the path forward was not going to be quite so easy. We had imagined simply merging our roadmaps—but as soon as we met with the acquired company's product team, it became clear there were a lot of higher-level questions about integrating our business models and strategies that needed to be addressed first. As a product team, we felt like these decisions—which had material impacts on what and how we were selling—ultimately needed to be made by people higher up in the organization. So, we made our recommendations and then waited. And we waited. And asked about progress. And then we waited some more.

When we finally did get a go-ahead from senior leadership, it wasn't the product-level integration approval we had been hoping for. Instead, our executives were asking us to build our roadmap around *their* specific, and pretty shallow, idea for a product integration—a product integration

that we felt was only scratching the surface of what was possible, and not delivering the value that we knew could be found. Our further recommendations were pretty much ignored, and the integration remained pretty shallow. A big missed opportunity.

I think about this story often when I hear product managers and product teams complaining about how senior leadership is demanding that they put certain things on the product roadmap. Without a doubt, this is something that happens in large and small organizations alike. But in our case, there was an element of self-fulfilling prophecy that's hard to ignore: **we were expected to stand back from a conversation that was above our proverbial pay grade, and were upset when that conversation didn't reflect what we as a product team knew about our users and our product.** Knowing that both strategic and product-level decisions are going to be made at a higher level in the organization, it is worth doing absolutely everything in your power to make sure that senior leaders have as much on-the-ground product knowledge as possible.

Making Room for Old Stuff and (Truly) New Stuff

One of the biggest challenges around short-term prioritization is ensuring that you aren't making decisions based on what people on your team feel like working on or think is cool. In theory, prioritizing against a clear set of goals helps solve for this. But having good goals in place will take you only so far if the only ideas you are actively looking to prioritize are ideas that you've already decided are cool, new, and interesting. So what do you do if the options you're discussing all feel more shiny and new than they do useful and goal oriented?

For starters, don't outright dismiss shiny and new things. Work with your team to understand why these things are so exciting. What about this new technology is so appealing? What about this new feature idea has your team advocating so enthusiastically for it? Rather than fighting the momentum behind shiny and new ideas, see if you can direct that momentum toward the ideas that will deliver the most value for your users and your business. If a shiny new idea could solve a specific problem for some of your users, but some quick revisions to an existing product could solve that same problem for *more* of your users, you are in a great position to bring new excitement to old products.

For example, imagine that a developer on your team is really excited about allowing your users to log in with their credentials from a cutting-edge social net-

work. This developer arrives at your prioritization meeting with a few emails from users who have requested this very feature. You pause for a moment and try to suppress your frustration. How in the world could this person think that something so marginal and untested is worth prioritizing? In an effort to swat away this suggestion by way of a goal-based question, you say, "Interesting... how many of our actual users do you think are currently *unable* to log in because this feature is not available?" The developer nods with resignation. You kept the team on track.

Unfortunately, you also might have missed an important opportunity. Something about this idea had your developer excited enough to dig through user feedback before your prioritization meeting. Perhaps this developer is really passionate about improving the overall login experience for your users, which might open up an important discussion about *less* shiny and new but potentially more impactful ideas, such as improving your password recovery workflow. Perhaps this developer read about how the cutting-edge social network in question took a truly novel and interesting approach to its authentication workflow, which might open up an important discussion about what makes an authentication workflow successful in the first place. Something about this idea got your developer truly motivated, but you won't be able to harness that motivation if you look for ways to dismiss their suggestion outright.

Taking this approach, you might find yourself talking through a brand new idea that you haven't had time to formally define and spec out. And if you approach prioritization as a way to answer the question "what well-defined product and feature ideas will we use our capacity for the next iteration to build?" then this idea may simply fizzle out. But your team's time is not always best spent building. It is critical that you also make time for learning, researching, and experimenting—and that you protect that time by explicitly making it part of the work that you are prioritizing.

In Agile parlance, a finite amount of time devoted to learning, researching, and/or experimenting (as opposed to building or coding) is often called a "spike." In keeping with "the best thing about best practices" from Chapter 4, using this specific language can be helpful in communicating that you are not haphazardly taking time away from executional work, but rather committing a specific and finite amount of time to exploring how best to approach that executional work.

Some of the most important product decisions I've seen made by members of my team have happened when the first item on their to-do list for the next week wasn't "write a bunch of code to complete this feature," but rather

"research five possible implementation approaches that would help us better understand this feature." Just because we are going into execution mode does not mean that we should stop learning, or that the only work we should value and prioritize is work that results in an immediate product output.

Using Prototypes to Validate or Invalidate Feature Ideas

J.D.

Product manager, 50-person entertainment startup

When I was working at a 50-person entertainment startup, we had an idea for a really, really cool geolocation-based feature. The idea had been floating around for a while when I started, but I took the lead on writing up a lightweight spec, getting buy-in from across the organization, and making sure it had a place on our roadmap.

After a few months of planning, we were ready to start building. During a prioritization meeting with the product team, we discussed a few potential approaches we could take to getting the product started. At first, the conversation gravitated toward a fairly technical discussion about how we would implement a geolocation-based feature. One of our developers wanted to use an open source option that would require more work, but carry no additional cost. Another developer had a vendor they preferred, whose service was costly but would require less work on our end.

Always up for a good technical challenge, the developer who preferred the open source option offered a solution: give her two weeks to see if she could make a geolocation-aware prototype to test the open source solution, and then make a decision. I was a little nervous about kicking off this project with a prototype that wouldn't necessarily lead us to our actual feature, but everybody seemed excited about this, so we agreed to move forward.

Two weeks later, we set up a time for our developer to demo the prototype she had built. It was, from her perspective, a complete technical success. She was able to create a basic proof-of-concept app that simply fired off an alert when any of the geolocation criteria we had written into the product spec were met. And she was able to do it using the free, open source solution she favored.

But as she walked us through the solution she had built, a few of us started to have the same creeping thought: *How useful would this feature actually be to our users*? Hearing our developer walk through the way she had used this prototype left us with some big questions about how valuable our users might find the full-fledged feature we were imagining. So, rather than moving forward with building the feature, we decided to give the prototype app to a few of our colleagues and see whether it pointed to a feature that would actually be useful.

After just a week, it became very clear to us that this feature was not actually as valuable as we had imagined. The specific geolocation criteria we thought we could use to customize an entertainment experience were not met as often as we had hoped—and furthermore, didn't seem to align with the actual needs and preferences of my colleagues in the moments when the alerts were triggered.

Yes, ideally you should test a prototype with users from outside your organization. But the biggest risk of testing something internally is that you will validate an idea that isn't valuable for your users, and I'm proud of the fact that we were actually able to *invalidate* an idea that we had all thought would be great. **What we thought was just a technical proof of concept actually turned out to be a critical way for testing whether the feature we planned to build would be valuable for our users**. A two-week prototype probably saved us six months of development time that would not have helped us meet our organizational goals.

But This Is an Emergency!

In theory, one of the primary functions of both a roadmap and a prioritization process is to determine both what will and what will not be built in a given timeframe. But in practice, every single organization must deal with "emergency" feature requests. (We walked through one such request at the end of Chapter 6.) In keeping with the spirit of changing, not breaking, the rules, I often suggest that product managers set up an official process and/or template for handling last-minute requests. The following basic template can provide a good place to start:

What is the issue?

Who reported this issue?

How many users is it affecting?

Is there revenue directly tied to this issue?

If so, how much?

What would happen if this issue were not addressed in the next two weeks?

What would happen if this issue were not addressed in the next six months?

Who is the contact person for further discussing/resolving this issue?

This particular template assumes that the requests coming in are likely to take the form of "this thing is broken," but depending on the specifics of your organization, you can customize it to accommodate for feature-happy marketing teams, account management teams that request last-minute custom work, and even developers who seem to habitually prioritize a new bug that they discovered over the work they had set out to do in your prioritization meeting. You can also fine-tune the specific questions around the number of users affected and the potential revenue ramifications based on how broadly accessible that information

is within your organization. (Or, even better, use this template as a starting point for making that information more accessible!)

In many cases, I've found that the mere presence of a template like this makes the volume of emergency requests drop significantly. After all, it's much easier to storm into a room and say, "THIS NEEDS TO BE FIXED RIGHT NOW," than it is to sit down and work through figuring out the actual impact of the thing you're about to ask somebody else to work on for you.

Prioritization in Practice: Same Features, Different Goals

Imagine that you are working as a product manager at an ad-supported video startup. Your company pulls in video from around the web to create "personalized video playlists" that people can enjoy at parties, on their commute, or just for killing time. You know that video is *huge* right now, and you know that there's a lot of potential in personalized aggregation as the video market becomes more fragmented.

As you sit down to approach your next prioritization meeting, you see five items on the roadmap slated for the next quarter:

- Connect to new display advertising network
- Add social sharing feature
- Build functionality for sponsored video playlists
- Improve personalization algorithm
- Launch iPad app (currently iPhone and Android-only)

Some of these ideas have been kicked around within the company since it started. Some of them were supposed to be built last quarter, but were pushed back. And some of them wound up on the roadmap because senior stakeholders kept asking for them, and you found it much easier to say, "*You've got it*, it's going on the roadmap!" than to debate the finer points of something you wouldn't really need to think about for a while.

As is often the case, these things are all very different from one another. It's not entirely clear why you would build one over the other, or even what kind of resourcing they would require. So, you turn to your organization's goals. Wait, goals? This is a *startup*. You rummage through some old emails from your founder, and find the closest thing to a mission statement that you can. It reads:

Our goal is to completely transform the way that people consume video. By aggregating video from all over the web and using machine learning to create a "personalized playlist," we can disrupt the media industry and create a better experience for our users.

You read that over a few times, and blink. How would you prioritize these five potential ideas against those goals? It's not easy, is it?

Now, suppose that, rather than just winging it, you decide to sit down with the founder and take these potential ideas for a test drive. You begin with the simple question, "How can we make this quarter's goals clear enough that we wind up building the things that *you* would want us to build, even if you aren't present at our prioritization meetings?"

After going back and forth a few times, you wind up with the following Objectives and Key Results–style goal for the next quarter:

Our high-level goal for the next quarter is to get our product in front of anybody who is currently engaged in the behavior of watching videos across multiple platforms. We will know we are on the right track toward achieving this goal when:

- *Weekly app downloads have increased by 50%*
- *Weekly new user signups have increased by 50%*
- *The average number of connected video platforms per user has increased from 1.3 to 2*

You know that this goal and its corresponding success metrics are not perfect or 100% comprehensive, but when you went through them with the founder, you were able to strike a few items off your list (especially those that might increase revenue without contributing to user growth). There are a few ideas you need to research a bit more (how many users are you currently losing to not having an iPad app?) but at the very least you have a clear sense of how to proceed. Some of the sales folks might be annoyed that you won't be working on their priorities first, but you're not too worried about it because your choices are clearly being driven by company-level goals.

Or, imagine that your conversation with the founder goes very differently. After the meeting, you wind up with the following Objectives and Key Results–style goal for the next quarter:

> *Our high-level goal for the next quarter is to increase the company's revenue while minimizing the need for customized development work. We will know that we are achieving these goals when:*
>
> - *Overall revenue has increased by 30%*
> - *The percentage of revenue coming from automated ad systems has increased from 30% to 60%*
> - *We are able to continue at or exceed our current rate of user growth*

Again, these are not perfect or 100% comprehensive. But they provide some clear guidance about what you might want to build right now, as well as some guidance about how you might want to implement some items on the roadmap (could you build a "sponsored video playlist" system in a way that would not contribute to increasing customized development work?)

When it comes time to actually prioritize these ideas, there will still be discussion to be had and work to be done. You might find yourself using prioritization techniques such as stack-ranking or an impact-versus-effort matrix. But whatever technique you use, the resulting decisions will be as good as the goals that guided them.

Summary: Let Them In!

Creating and populating a roadmap and managing short-term prioritization can often feel like the most high-status tasks for product managers. Not only does this work give you the chance to make important decisions about what is actually being built, but it also affords you a rare opportunity to point to an actual *thing*—like a roadmap document or an in-depth product spec—and say, "I made that all by myself!" But as with all aspects of product management, roadmaps and prioritization are best approached not as sources of authority, but rather as opportunities to connect and align.

Your Checklist:

- Give up on being the person who "owns" the roadmap. Instead, look to facilitate the way that your entire organization uses roadmaps.
- Don't make assumptions about how your organization uses roadmaps. Ask lots of questions, and create a clear and well-documented understanding of how roadmaps are to be used within your organization.
- Open up the roadmap. It should be a conversation starter and a tool for alignment, not something to be closely guarded and manipulated under cover of darkness.
- Give your colleagues the opportunity to suggest ideas for the product roadmap, but don't let it turn into a free-for-all. Use simple templates to structure your colleagues' thinking around the goals of their ideas, not (just) the ideas themselves.
- Advocate just as fiercely for ideas that are not your own, if not more so. Don't get hung up on wanting to be the "idea person."
- Don't spend so long on product specifications that you close off avenues for true collaboration.
- If you are using a formal practice for writing product and feature specifications such as "user stories," remember that a formally correct spec is not necessarily a good spec.
- Make sure that everything on your roadmap is tied back to a "why" so that if that "why" changes, you can adjust the roadmap accordingly.
- Be prepared for short-term prioritization to be much more challenging than creating a long-term roadmap.
- Do everything in your power to make sure that the goals against which you are prioritizing are clear, well understood and actionable.
- If you can, take your goals for a "test drive" with the senior leaders who are setting the company vision and strategy. See if the goals can serve as a stand-in for their vision, and change your goals if they aren't giving you the guidance you need.
- If your team is excited to build something new and shiny, don't reflexively shoot it down for "not meeting our goals." Find out why your team is so

excited, and see what you can do to direct that excitement toward whatever solution will deliver the most value to your users and your business.

- Remember that learning, testing, and experimenting is still valuable work, and should be treated as such. Prioritize tasks like creating prototypes and researching implementation approaches alongside the work of actually building your product.

The Wonderful, Horrible Truth About Agile

First: a hearty and heartfelt hello to those of you who picked up this book and skipped immediately to this chapter. For many product managers—especially those whose job description skews more toward Scrum Master or Agile Product Owner, navigating the finer points of Agile processes can seem like the sum total of one's job. There are countless books, manuals, and step-by-step guides out there to implementing the Agile framework of your choice, whether it is Scrum or XP or a scaled framework like SAFe or LeSS.

For better or worse, this book is not one of them. Regardless of how orthodox, prescriptive, and by-the-books you are in your implementation of Agile, you can't process-ize out the human complexity of product management. Regardless of which Agile (or non-Agile) processes and practices you choose to implement, you still need to connect, communicate, and collaborate. The wonderful truth about Agile is that it revolves around a set of values that reinforce and strengthen the connective work of product management. The horrible truth about Agile is that the work of implementing these values is never truly over, and requires constant reflection and refinement.

This chapter focuses on strategies and approaches that will help guide you through the successful implementation of *any* practices, processes, and frameworks that could fall under the broad banner of "Agile." Throughout this chapter, keep our second guiding principle of product management in mind: "Change the rules, don't break the rules."

Debunking Three Common Myths About Agile

Over the past decade and a half, the word *Agile* has gone from a tactical distinction among software developers to an inescapable nugget of business jargon. Before we talk about the specific history of Agile and how we can use its core values and principles in our work, let's look at a few common myths and misconceptions about Agile that I've encountered many, many times:

Agile is a rigid and prescriptive methodology
: Fun fact: Agile is not really a methodology at all. As we will discuss, Agile is a *movement* that began when people who had worked on multiple software development frameworks and methodologies came together to discuss the common values expressed in their respective approaches. Many practices that are implemented under the guise of "doing Agile" actually run fundamentally counter to these values.

Agile is a way to do more work, faster
: On numerous occasions, I have stood in meetings while senior leaders describe Agile as a way to "increase our output" or "get things done quicker." If I could capture the looks on the faces of senior developers during these meetings, I would be happy to simply present them as the entirety of this chapter and call it a day. Agile is not a matter of working more or of working faster, but rather of working differently. In fact, following the core values of Agile often means slowing down, at least momentarily, to reflect on how we currently work and how we could work better.

Agile is only for software development teams
: Because the Agile movement emerged from software development, people often assume that it is relevant only to software development. This is not at all true. In fact, when software development teams implement Agile practices in a way that disconnects them from the broader organization, they too are often "doing Agile" in a way that actually runs counter to the core values of Agile.

Turning to the Agile Manifesto

The Agile movement kicked off in earnest in 2001, when a group of 17 software developers gathered together at a ski resort in Utah to discuss alternatives to the "documentation driven, heavyweight software development processes" of the day. The resulting Agile Manifesto reads, in its entirety, as follows:

> *We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:*
>
> ***Individuals and interactions*** *over processes and tools*
>
> ***Working software*** *over comprehensive documentation*
>
> ***Customer collaboration*** *over contract negotiation*
>
> ***Responding to change*** *over following a plan*
>
> *That is, while there is value in the items on the right, we value the items on the left more.*

It is worth taking the time to read this over thoroughly, many times. More than once, I've had it taped up over my desk as a team with which I'm working begins to explore Agile practices and methodologies. Fundamentally, Agile is not about following a single prescriptive set of rules; rather, it is about designing and implementing practices that align with a set of values. Core to these values—and reinforcing the approach to product management described in this book—is the embrace of human complexity. Truly valuing individuals means looking beyond titles and org charts to understand the actual people with whom you're working. Processes and tools can help facilitate our connection with those people, but they cannot *replace* that connection.

Here's my very favorite thing about the Agile Manifesto: it actually came *after* the development of some of the most popular Agile development methodologies such as Scrum and XP. When the people who developed these methodologies met up, they didn't argue incessantly over why "Scrum is for kindergartners" or "XP is just way too difficult for real-world organizations." Instead, they were able to reach a clear and definitive agreement about the core values that these methodologies share. Feel free to bring this up the next time you see product managers engaged in a heated debate over which Agile methodology is "the best."

From Manifesto to Monster

Those who have spent a good deal of time navigating the world of Agile software development and "Agile business transformation" might note the irony in the Agile Manifesto citing "individuals and interactions over processes and tools." In the years since the Agile Manifesto's signing, the Agile ecosystem has become a dizzying Lovecraftian swirl of frameworks, practices, tools, and certifications. This irony is not lost on many of the people who actually wrote the Agile Manifesto. And even if you suspect me of being a process-hating anarchist, surely you

must be interested in what *they* have to say. To that end, Agile Manifesto signer Andy Hunt wrote a blog post called "The Failure of Agile" that lays out his perspective on how a set of inspiring ideas got turned into a prescriptive ideology that fundamentally violates its own core values:

> *In the 14 years since [the Agile Manifesto], we've lost our way. The word "agile" has become sloganized; meaningless at best, jingoist at worst. We have large swaths of people doing "flaccid agile," a half-hearted attempt at following a few select software development practices, poorly. We have scads of vocal agile zealots—as per the definition that a zealot is one who redoubles their effort after they've forgotten their aim. And worst of all, agile methods themselves have not been agile. Now there's an irony for you.*

Hunt goes on to describe why he feels that Agile has been so badly misinterpreted:

> *Agile methods ask practitioners to think, and frankly, that's a hard sell. It is far more comfortable to simply follow what rules are given and claim you're "doing it by the book." It's easy, it's safe from ridicule or recrimination; you won't get fired for it. While we might publicly decry the narrow confines of a set of rules, there is safety and comfort there. But of course, to be agile—or effective—isn't about comfort.*

Again, we return to our first guiding principle: clarity over comfort. As we discussed in Chapter 8, clarity does not mean absolute unflinching certainty. Achieving and maintaining clarity is ongoing, difficult, and at times deeply uncomfortable work. At its best, Agile provides us with a way to value and protect that work. But it cannot do so if we turn to it only for certainty, for absolutism, for "the right way to do things" regardless of the specific individuals involved.

Using Alistair Cockburn's "Heart of Agile" to Bridge Values and Practices

The tragedy of Agile's sloganization is that many of the practices used in Agile software development really can help us enact its stated values. Alistair Cockburn, another signer of the Agile Manifesto, responded to what he called the "overly decorated" state of modern Agile by distilling the entirety of Agile practices and processes into four actions at the "Heart of Agile":

- Collaborate
- Deliver
- Reflect
- Improve

Cockburn explains how the simplicity of these four actions provides a needed counterpoint to the jargon-heavy discourse around modern Agile practices:

> *The nice thing about these four words is that they don't need much explanation. They don't need much teaching. With the exception of "Reflect," which is done all too little in our times, the other three are known by most people. You know if you're doing them or not. So simply saying, "Collaborate. Deliver. Reflect. Improve." already says most of what you need to say and do.*

These four actions provide a bridge between the values of the Agile Manifesto, and the practices that come with specific Agile frameworks and methodologies. They get to the very, well, heart of what separates Agile from the siloed, heavyweight processes that came before it and still persist in its wake. And, most important, they provide a simple and plainspoken yardstick against which you can measure the success of any specific Agile initiative.

One of my very favorite things about Agile—and about Cockburn's "Heart of Agile" in particular—is that it contains within it the blueprint for its own success. If you are taking time to truly reflect and improve, then wherever you start, you will wind up somewhere better. The single biggest mistake I've seen organizations make when implementing any kind of Agile process is to take an all-or-nothing approach in which a framework or set of practices is implemented and then declared an outright failure when it does not work perfectly right away. If, per Cockburn's actions, you don't take the time to reflect on the way you're working and improve the things that don't work, then *any* Agile practices will stagnate, fall into disrepair, and ultimately fail to have any real impact.

Four Steps to Future-Proof Agile

Depending on the specifics of their role, a product manager might have more or less direct authority over the Agile rituals, practices, and methodologies used by their team and organization. But it is critical to remember that reflecting upon and improving Agile practices is, itself, an Agile practice. Even if you see yourself as simply being a practitioner of Agile, thinking about how you can improve the way your team works is still part of your job.

Here are the four steps that I've seen lead to the most successful Agile adoptions for large and small organizations alike.

1. Begin with a "North Star" such as the Agile Manifesto or the "Heart of Agile"

   Before rushing to implement specific rules or methodologies, share a high-level vision for *why* you are making these changes in the first place. This can be the Agile Manifesto, or Alistair Cockburn's "Heart of Agile," or anything else that provides guidance on the order of values, principles, and outcomes, as opposed to practices, ceremonies, and outputs. In the absence of this North Star, it is impossible to know whether the specific practices you are implementing are actually helping you achieve your goals—because nobody knows what your goals are in the first place.

   I recommend putting some kind of visual reminder of your North Star—such as a poster of the Agile Manifesto or a visualization of the "Heart of Agile"—somewhere central and visible to your entire team. This helps create a shared sense of purpose and accountability, and prompts your team to constantly reflect on why it is adopting any specific practices or ceremonies.

2. Pick an off-the-shelf Agile methodology

   Pick a methodology—any methodology! Because you're committed to adapting as necessary based on what is or is not helping you achieve your goals, you don't need to worry all that much about where you choose to begin. As a rule, though, I would recommend starting with an existing, well-documented methodology like Scrum or XP. Starting with something that already exists and is well documented gives you a "referee" to settle any initial confusion or disagreement—which in turn makes it easier to be very clear about what is working and what is not working.

3. Regularly evaluate the specifics of that methodology against your North Star

   In most Agile methodologies, there is a "retrospective" in which the team commits to changing how it will work moving forward. I've seen many product managers either half-ass the retrospective or omit it altogether, because it does not involve actually producing software. Often, this is because the mandate they have been given is to work faster, so anything that does not involve cranking out product is seen as a waste of time. This might feel like a defensible short-term optimization, but it has serious ramifications in the longer term. Without making time to reflect and refine your process, that process will eventually fall apart. And without having that initial North Star against which to evaluate the specific practices you've chosen, you can easily fall into the trap of constantly changing course without knowing which way you're actually headed. Having a North Star in place and making time to reflect empowers you and your team to ask questions like, "Are the meetings we are regularly holding actually helping us *collaborate*?" and "Is the way we work enabling us to *deliver* value to our users as often as possible?"

4. Work with your team to change the specifics of your methodology accordingly

   Here's where things get a bit trickier. If you've started with an off-the-shelf methodology, and found that some of the specific practices within that methodology are not working for you, you are likely going to need to go "off-manual" and implement new practices that are designed to meet the specific needs of your organization. This is where you and your team must commit to full accountability for the way that you work together, to changing the rules, not breaking the rules, to giving up the safety and comfort of pointing at a book or manual and saying, "I'm just doing what it tells me to do."

   When changing specific practices within an organization, I've found it very helpful to document the change being made, the goal of that change, and how we will know when it is succeeding. In Chapter 4, I provided a simple and straightforward template that can be used to bring a goals-first approach to organizational practices. You can easily customize that template for working within Agile practices and methodologies:

We used this Agile practice in our last iteration of work:

We implemented it because we thought it would have this effect toward achieving our North Star goals:

The actual effect of this Agile practice was:

So, for the next iteration of work, we are changing it in this way:

We hope that making this change will have the following effect toward helping us achieve our North Star goals:

We will know that this change is succeeding when:

This template provides an opportunity for you to tie back any specific practices or ceremonies you adopt to your North Star goals, and to explicitly track the difference between what you think a new practice or ceremony will accomplish and what happens when you actually implement that practice or ceremony. This can be tremendously helpful in better understanding the fundamental patterns and rhythms of your organization.

As you might have noticed, these four steps are not a "one and done" approach to implementing Agile practices. Following these steps means taking time to constantly adjust your approach based on what is working and what is not working. Again, the Agile Manifesto itself tells us that responding to change is more important than following a plan—even if that plan is for implementing Agile practices.

Bringing this approach to your work, you might find yourself changing some things that feel like immovable orthodoxies of Agile software development. That is totally fine. Nearly every product manager I've worked with has at some point made a major change to Agile sacraments like the daily standup meeting or the writing of user stories. Going "by the book" is a great place to start. But the book doesn't know the specific individuals and interactions that drive your team and your organization. At the end of the day, it's up to you.

Killing the Daily Standup

A.J.

Product manager, enterprise analytics startup

When I started working as a product manager, I didn't know all that much about Agile processes. But as the company grew, it became clear that the ad hoc system we were using to build products wasn't working very well. I picked up a few books about Agile and Scrum, and turned to a few developers in the organization who had experience with Agile development.

One thing that every book and person seemed to agree on was that we needed to have something called a "daily standup meeting." For anyone who has not worked in Agile software development, this is a meeting, usually at the beginning of the day, where everybody on the product development team stands up and says what they've completed since the last standup, what they're working on, and what is currently blocking them. So, as a very first step toward building out an Agile process, I started holding daily standup meetings with my team.

These meetings were...not great. They felt kind of like elementary school book reports, with everybody grudgingly standing up and reading off a list of what they had done. One of the developers I worked with started referring to the daily standup as the "what have you done for the company lately?" meeting. I knew that the meetings weren't working well, but I had no idea what to do about it. Everybody had agreed that you needed a daily standup to "do Agile." And as a relatively junior product manager, I was certainly in no position to assume that I knew better.

There was one developer on my team who always seemed particularly averse to the daily standup meeting. He would show up late, roll his eyes, and just generally be a pain about it. Ironically, it was this very

developer who wound up giving me the courage to reevaluate whether the standup was working for our team. During one particularly drab Monday-morning standup, he remarked that he had been blocked on something "since Friday afternoon." The developer who had been blocking him offered, sincerely, "Why didn't you tell me?" He responded, "because that's what this meeting is for."

That exchange helped me understand that the daily standup meeting was actually doing the exact opposite of what it was supposed to do. And furthermore, it helped me understand that I had never taken the time to talk with my team about what it was supposed to do in the first place. **The meeting that was supposed to help people get unblocked had actually been giving people an excuse to stay blocked**. After talking it over with the team, we decided to kill the daily standup, and that if anybody was blocked it was their job to jump on our team chat and say what was blocking them immediately. It wasn't "by the book" Agile, but it wound up doing what the "by the book" practice hadn't done for our specific team.

A Few General Caveats About Agile

I am a firm believer in the core values of Agile. But Agile was not intended to be a cure-all for every organizational challenge. Here are a few potential limitations of an Agile approach that are worth keeping in mind:

Agile does not always explicitly address the "why"

Agile provides wonderful guidance as to *how* modern teams can work together. But most Agile practices do not intrinsically solve for "why." As a product manager, you can ostensibly succeed at doing Agile while executing against a roadmap that still delivers no clear value to your business or your users. Resist the temptation to hide behind process and say, "Well, as long as I'm doing all this Agile stuff, I'm doing my job." Even if you have a clear North Star guiding your Agile *process* in the right direction, you still need to make sure that you're using that process to build the right things.

Agile rituals can become a false stand-in for user centricity

One common Agile practice, which we discussed at more length in Chapter 9, is writing out feature specifications as user stories. At its best, this practice can help product teams stay focused on user value rather than get-

ting bogged down in implementation details. But at its worst, this practice can give product teams a false sense of user-centricity, even as they fail to interact directly with their users in any meaningful way. If you are writing user stories, I would strongly recommend using the template from earlier in this chapter to clearly spell out why you are using this practice—and how you will know whether you are succeeding in meeting those goals. For example, you might explicitly set as a success metric that writing "user stories" will result in your product team spending *more* time interacting face-to-face with users. Remember, no matter how many user-centric rituals you incorporate into the daily workings of your Agile team, you still need to talk to your users.

Agile needs to extend beyond your product team

This is not to say that your entire organization suddenly has to "do Agile." But your entire organization *does* need to understand why your team is adopting Agile practices, and what the implications are for the organization's overarching rhythms and deadlines. Again, the core values of Agile tell us that collaboration is key, but we must be vigilant about extending that collaboration beyond our immediate team, to ensure that we are not ultimately insulating ourselves from organizational dynamics that will prove critical to our success or failure.

Once again, a successful implementation of Agile comes back to the CORE skills of product management. No matter which Agile methodology you choose to implement—or whether you choose to implement no Agile methodology at all—there is still a lot of work for you to do as a product manager to make sure that your team remains communicative, well organized, truly user-centric, and focused on executing the right work to achieve your team and organization's goals.

You Are Here

Depending on how mature the product management practice is at your organization, you might be working within a very well-understood and well-documented product development process, or you might be starting from scratch. Or, rather, you might *think* that you are starting from scratch. But the truth is, the lack of formal process is very much a process in and of itself. Teams that are used to operating with zero formal structure and guidance often resist change for the same reason that teams with a very mature practice might resist change: because

they've gotten used to doing things a certain way, and they don't want something to upset the status quo.

Regardless of whether you are working in an organization that has a formal product development process in place or one that is using an ad hoc system, I always find it helpful to take the time to sit down and map out how products are developed at your organization right now. How do you decide what to work on next? How do you estimate how long something will take? How do you break down something on a roadmap into actual tasks that can be completed? And how do you know when something is *done*?

Even if your organization has no formal process in place, answering these questions will help you communicate to your team that there *is* a way that things are currently done, which will make it easier for you to do the always-challenging work of changing organizational processes to better meet your goals. You should never make any changes to your process without having a clear sense of where you are, and where you're going.

Setting Expectations When Transitioning from Waterfall to Agile

Noah Harlan
Founder and partner, Two Bulls

Before we adopted Agile, we were working in a very "waterfall" way, starting each project with these huge spreadsheets that listed out every feature we intended to build. When we were working this way, our clients tended to feel great on day one of a project. Everything felt very certain and very finite, "In four months, we'll have our product, and we know exactly what it will be!" For some bigger products, this could even be a year or two years. But a lot can change in a year or two years, or even a month. Competitors change, technology changes, regulatory environments change. Apple might release a new version of iOS that breaks a finished product right after you launch it. That feeling of comfort and certainty naturally starts to decline when you go about building products in the real world.

Adopting Agile practices meant that those initial conversations with clients had to change pretty dramatically. Rather than haggling over how many features we could build within a client's budget, we explained to them that we would start down a certain path, track our velocity, show them the work every two weeks, and allow them to work *with* us on

changing, adding, or subtracting features as the product took shape. We got a lot of responses like, "Yeah, but how much does it cost and when will I get it?" At first, we struggled to answer those questions. But now that we've been working in Agile for a number of years, we have a much clearer sense of what can be accomplished in a given time box and we can provide some fence posts along the way. With Agile, you constantly refine and explore the delta between your estimated velocity and the reality of your work, which is actually much more powerful than trying to predict it all at the beginning (see Figure 10-1).

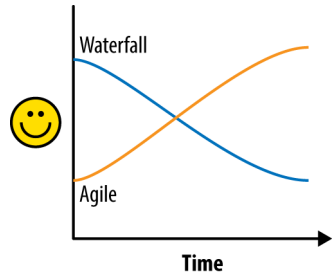

Figure 10-1. Happiness over time in Agile and Waterfall projects

Working in Agile our team feels more like *your* team. As the project goes on, it becomes clearer and clearer that our interests are truly aligned. With Agile, our profit is maximized by you being successful and continuing to develop the product, whereas the end of a Waterfall project is us trying to maintain our profitability by limiting the number of things you can squeeze in, and limiting the warranty. **Though waterfall projects may provide your clients with a seductive sense of certainty at the beginning of a project, they set you down an inherently adversarial path.** Working in Agile, we've been able to collaborate much more closely with our clients and deliver better products.

Summary: Ambiguity Lives Here, Too

With all of its frameworks, methodologies, and "best practices," Agile might seem like a way to bring standardization to a role riddled with ambiguity. But at its heart, Agile is *all about* learning to respect and embrace uniqueness—of indi-

viduals, of interactions, and of the inevitable curveballs that will lead you away from your best-laid plans and into the great unknown.

Your Checklist:

- Avoid ambiguous and misleading jargon around Agile—say exactly what you intend to do and why you intend to do it.
- Understand the core values and principles of Agile before evaluating any specific practices or frameworks. If you just start implementing process without purpose, then you have no way to evaluate whether the process is working or not.
- Socialize a North Star vision around Agile values and principles, so that everybody on your team knows why they are "doing Agile" in the first place.
- Start with an off-the-shelf Agile methodology so that you have an impartial "referee" to resolve any specific questions about whether you're "doing it right"—and then be fearless about changing that methodology if it is not helping you reach your North Star vision.
- Create and protect time for your team to evaluate Agile practices against the goals you've set. Document process changes along with their intended goals, so that there is clarity around what people are doing and why.
- Don't let the operational details of doing Agile distract you from higher-level organizational goals. Remember that if you are executing against a product roadmap that delivers no value to your users or your business, it doesn't matter how Agile your execution is—your product is still going to fail.
- Don't let user-centric Agile rituals serve as a stand-in for actually talking to your users.
- Communicate the goals and rhythms of your Agile practices to people outside of your immediate team, so that they know what to expect and how to work with you.
- Understand that "no process" is still a process, and take the time to really understand how your organization is currently building products.

- If you feel that your organization is becoming too zealous about Agile, feel free to print out a whole bunch of blog posts by people who actually wrote the Agile Manifesto, describing how Agile zealotry and ornamentation have derailed the movement they started.

In Good Times and Bad

As of 2017, there were about 2.2 million apps in the Apple App Store, and 2.8 million in the Google Play Store.

According to a 2015 study by Forrester, most smartphone users spend the vast majority of their time interacting with only *five* apps that did not come preinstalled on their device.

That leaves a lot of disappointed product managers.

I cite these statistics not to be fatalistic, but rather to set the expectation that building a product that sees the kind of meteoric success associated with companies like Apple, Uber, Facebook, or Google is exceptionally rare. Great product managers work on products that fail all the time. There are no "best practices," no perfect prioritization frameworks, no *Agile-cadabra* magic words that will guarantee the success of your product.

Product managers working on established and successful products face their own set of challenges that can be just as galling. Established companies tend to become risk-averse, bureaucratic, and political, at times making it excruciatingly difficult to implement minor changes that have clear user value. Even when the numbers are going in the right direction—*especially* when the numbers are going in the right direction—getting out ahead of fast-changing user needs can prove incredibly difficult.

Product management is not an easy job—but the practice of product management can make everybody's job easier. It can help programmers become better communicators, marketers become more excited about technical work, and executives understand the tactical ramifications of high-level strategic decisions. Great product management turns insidious tension and misalignment into opportunities for learning, sharing, and collaborating—in good times and in bad.

The Soothing Lull of an Organization on Autopilot

In nearly every product organization, particularly more mature product organizations, there are usually stretches of time where teams go into "autopilot" mode. Sometimes, this is because external conditions are so favorable that the numbers are all going in the right direction, and nobody is feeling all that much pressure. Sometimes, this is because people have stopped paying attention to the numbers, and the product team is operating with minimal accountability and oversight. And yes, sometimes, this is because all the right pieces are in place and you are operating like a well-oiled product machine.

But this autopilot mode comes with substantial risk. When a team has gone too long without a new challenge or a fresh perspective, it can begin to feel like "the way things are" is the only path to success. New ideas that don't support the status quo are watered down or waved away. Teams become more insular and less curious, important questions go unasked, and critical opportunities are missed.

When your team feels like it is on autopilot, it becomes more important than ever for you to seek out challenging ideas and alternate explanations. Talk to users who have abandoned your product, even if there aren't that many of them, and try to figure out what went wrong. Explore competing products, and catalog what they seem to do better than you. Bring challenging questions back to your team—what if the direction you're taking is totally wrong? What if the linear growth you're experiencing is a fraction of what's possible? Model openness and curiosity by asking questions that directly challenge the work for which *you* are most directly responsible.

Finally, channel these challenging questions into hands-on collaborative work via time-boxed prototypes. What if we reinvented the product from scratch in one week? What if the core user need has changed, and we only had three days to transform our product accordingly? I personally favor prototypes that directly challenge the fundamental assumptions of a product over "hack day" and "20% time" projects that are framed as a break from working on the "real" product.

The Good Times Aren't (Always) the Easy Times

So, if a lack of immediate challenges and the resultant autopilot mode is not the sign of truly good times in a product organization, what are the signs that a product management practice is improving organizational health? Here are a few general indicators that you are on the right track:

Conflicts are discussed in the open with minimal personal attacks

A healthy product organization is not marked by a lack of conflict; rather, it is marked by the ability to address and resolve conflict in the open with minimal defensiveness, ego-driven attacks, and passive-aggressive flailing. As we discussed in Chapter 5, disagreement can actually be a critical tool for making good decisions as a group.

Everybody feels invested in the work that they are doing

In a healthy product organization, everybody is invested in the work that their team is doing, and in the way that their team is working together. If you suggest a new product idea or a new process improvement and it's met with a chorus of shrugs, this does not mean that you have the full and unwavering support of your product team. So long as disagreement is being handled in a healthy way, disinterest is actually much more dangerous for a product team.

People see new information (and new people!) as an opportunity, not a threat

In a healthy product organization, people never shy away from signals that they are on the wrong path. They do not wait until their quarterly review to share that they are unlikely to hit a quantitative goal. Their mission, per the CORE skills of product management, is to bridge their user's reality with their organization's goals—and any information, people, or ideas that help them get there are seen as a gift.

When I've interviewed for product roles in the past, one of the informal metrics I've relied upon most is, simply, how I'm treated when I walk into an office for the first time. Am I eyed with skepticism and hushed whispers? Or are people open, curious, and engaged? In a healthy product organization, new information and new people are seen as an opportunity to learn, not a threat to "the way things are."

To summarize, the truly good times as a product manager are not necessarily the easy times. Nor are they necessarily times when the company itself is doing well, though that certainly helps. The times when product management is at its most successful and powerful are those when new challenges are actively sought out and approached with openness, curiosity, and candor.

It is not an accident that these times tend to coincide with major product launches, last-mile pushes to get a new feature out the door, and other high-stakes, high-pressure situations. The moments that require the most collabora-

tion, the most camaraderie, and the most connection tend to be the moments when product management has an opportunity to truly shine. The real challenge is to bring that same level of energy and excitement to your work every single day.

Carrying the Weight of the World

Early on in my career, a mentor of mine told me that a product manager's job is "to think about every single little thing that could possibly go wrong, before it goes wrong." I replied, "Well, that's pretty much what I do all the time anyhow, so this job should be perfect for me!"

For people who are inclined to carry the weight of the world on their shoulders, product management can be a little bit *too* perfect. Being a product manager can leave you feeling like every problem you encounter is yours to solve, from new products launched by competitors to personal disagreements among your colleagues. And during moments of particular organizational difficulty or dysfunction, product management can feel both relentlessly demanding and absolutely futile—like pushing a boulder up a hill while 10 larger, more important boulders roll down past you.

Many of my worst moments as a product manager have occurred during times like this, when the sheer weight of the job felt unmanageable. I have thrown tantrums in front of well-intentioned colleagues, stormed out of meetings with senior leaders, and withheld critical information from my own team out of fear that they would be mad at me. And the vast majority of this bad behavior has been motivated by the same dangerous fallacy: "I am the only thing keeping this team (or this company) from completely falling apart."

This is where the connective nature of product management can serve as an amplifier of organizational discord. As a product manager, you are responsible for connecting people throughout the organization, and the more broken and misaligned these connections are without your direct and constant intervention, the more you can start to feel like you are the only thing standing between your team (or your company) and oblivion. During these times, you might begin to feel like you need to be everywhere at once in order to put out fires and resolve disputes. You might catch yourself grumbling to your friends, and sometimes even to your colleagues, about what a mess this whole thing is. But it's *your* mess, and you can't even imagine how it would continue to operate without you.

To revisit our bad product manager archetypes, this is where the thin line between the Hero Product Manager and the Product Martyr starts to blur. If you

begin to feel like you are the only person who can save your team and your company, you are going down a dangerous path. Here are a few steps you can take to avoid falling on the double-edged sword of product heroism and martyrdom:

Make a list of the things outside of your control

Did a tech giant just launch a product that directly competes with yours? Are two senior leaders in your organization embroiled in a battle for the CEO position? While both of these developments might have a serious impact on your job, you can't control another company's roadmap or another person's ambitions. Make a list of the things that are outside of your control, to serve as a reminder that it is not your job to solve every problem for everybody.

Look for opportunities to delegate important things

One way you can break the cycle of heroism and martyrdom is to delegate truly important things to your colleagues. Rather than trying to insulate your team from organizational dysfunction, ask your team to step up and take on responsibility for something that is actually mission-critical for the organization. Delegating important things to your colleagues means that they will likely run up against some of the same friction and frustration that you have been experiencing. While this is not an easy thing, it is often a good thing. It will give you a chance to address these challenges as a group, rather than you feeling like you are the only person capable of and responsible for solving them.

Protect the routines and rituals that bring your team together

During challenging times, it's easy to let things fall through the cracks—especially things that don't reflect the perceived urgency of the moment. Team lunches, coffee meetings, high-level brainstorming conversations—these are usually the first things to disappear from your calendar when the going gets tough. You might assume, as I often have, that your team will be just as happy to enjoy a delicious meal short one very stressed-out product manager. But your absence sends a powerful and dangerous message that the time you spend with your team is just not that important. Your colleagues might, in turn, wonder why *they* don't have anything more important to do.

One of the very best things you can do as a product manager is to protect the time that your team spends together doing normal, fun, and routine

stuff. Show up, be present, and model for your team that, even in the midst of major organizational challenges, it is of the utmost importance that you find time to step back, communicate, and connect.

Summary: It's Hard Work, but It's Worth It

From the thrill of a product launch to the frustrations of organizational dysfunction and inertia, product management tends to have some pretty extreme highs and lows. Product management necessitates being smack in the middle of whatever is going on with your team and your organization, which means that if there's a lot of difficult stuff going on, there's going to be a lot of difficult work to do.

It is for this very reason that product managers can have such a profound positive effect on the lives and experiences of their colleagues. Because you are in the middle, the actions you take are likely to be outsized in their impact. As the informal ambassador between your team and the rest of the organization, you can set the tone for how people communicate with one another, listen to one another, and demonstrate respect for one another's time and perspective. And during times of organizational crisis, you can choose to be the fearless protector of the very best things about you team and your company.

Your Checklist:

- Be wary of your organization and team falling into autopilot. Actively bring new ideas and challenging perspectives to your team at all times.
- Use time-boxed prototypes to explore alternate product directions, even when there is no immediate or obvious pressure to change course.
- Remember that a good product organization is not one free of conflict, but rather one in which conflict is handled openly and without personal attacks.
- Try to bring the energy and enthusiasm from your best and most exciting moments as a product manager to every day of your work.
- If you begin to feel like you are the only person keeping your team or your organization from falling apart, take a step back. Make a list of the things you can't control, delegate impactful work to your colleagues, and make sure you are protecting your team's most valued routines and rituals.

- Understand that the connectedness of your role carries great responsibility, but also great opportunity. Do everything in your power to protect and embody the very best things about your team and your organization.

Conclusion: Whatever It Takes

Nearly 10 years ago, I hoped that simply having the title "product manager" would grant me power and influence. The word *manager* suggested that I would be in charge of something. The word *product* suggested that the thing I was in charge of would be an entire product and, in turn, all the people whose work goes into building that product. *Who wouldn't want that job?*

But this could not be further from the truth. As a product manager, your title gives you nothing—no formal authority, no intrinsic control over product direction or vision, and no ability to get a single meaningful thing done without the help and support of others. To whatever extent you are able to lead through influence and trust, you must earn that influence and trust every minute, every day. And you must chart your own path to earning that influence and trust in a role full of irresolvable ambiguity and irreducible complexity.

Without exception, this means that you will make mistakes—glaring, egregious, embarrassing mistakes—as you build your product management practice. You will be evasive when you need to be direct. You will be impulsive when you need to be patient. You will follow "best practices" to the letter, and they will *still* backfire in ways you never could have imagined. The mistakes you make will have real repercussions for yourself, your team, and your organization. You will be humbled by the generosity and forgiveness shown by your colleagues. And over time, you might even become more forgiving toward yourself.

And therein lies the true beauty of product management. No matter how smart you are, product management demands that you learn how to be wrong. No matter how charismatic you are, product management demands that you learn how to back up your words with actions. And no matter how ambitious you are, product management demands that you learn how to respect and honor your

peers. Product management does not give you an airtight job description or a veneer of formal authority to hide behind. If you want to succeed, you will need to become a better communicator, a better colleague, and a better person.

A few months ago, I delivered a training session at a large, process-driven enterprise financial services company. When the topic turned to the day-to-day responsibilities of a product manager, a recent hire expressed his frustration at the unexpected ambiguity of his new role: "I feel like every day I show up for work, this is a totally different job." The other product managers in the room just smiled. Eventually, he started smiling, too. Like many product managers before him, he had asked the question, "Just what am I supposed to *do* all day, anyhow?" And without even realizing it, he had already found the answer: *whatever it takes.*

A Reading List for Expanding Your Product Management Practice

Given the connective nature of product management, drawing on multiple ideas and disciplines is key for truly excelling in this deeply cross-functional role. Product management always brings new challenges, and a wide base of knowledge will help you be better prepared for whatever comes your way.

To that end, I've found that the books and articles that have helped me the most as a product manager **are rarely about "product management."** What follows is a list of the books that have proven most impactful for me in building my product management practice, as well as a few notes about what you might be looking for if this book is right for you, and how it helped me build my product management practice.

Influence Without Authority (Wiley, 2005) by David L. Bradford and Allan R. Cohen

If You're Looking For: Strategies for leading through influence in low-authority roles and scenarios.

How It Helped Me: The content in this book is useful, but the in-depth case studies are *really* useful, and were the inspiration for the product manager stories included in this book.

The Trusted Advisor (Touchstone, 2011) by David H. Maister, Charles H. Green, and Robert M. Galford

If You're Looking For: Actionable strategies for building trust with customers and senior stakeholders.

How It Helped Me: The Trusted Advisor is that most helpful kind of book, where the specific behaviors it recommends *against* are behaviors that you might find yourself exhibiting all too often. Reading this book for the first time was the very last time I said something like "I'll put my best people on it" when scoping out a consulting gig.

Mindset: The New Psychology of Success (Random House, 2006) by Carol Dweck

If You're Looking For: A way to work past your overachiever tendencies and open yourself up to being wrong and learning new things.

How It Helped Me: In Chapter 3 of this book, we discussed how cultivating a growth mindset is key to succeeding as a product management. This book helped me understand how and why I was often operating within a fixed mindset, and even opened up some space for me to understand how moments that made me feel smart or accomplished might be doing material harm to my team and organization.

The Advantage: Why Organizational Health Trumps Everything Else in Business (Jossey-Bass, 2012) by Patrick Lencioni

If You're Looking For: A way to better understand organizational health (and dysfunction).

How It Helped Me: The Advantage is the first business book I recommend to most people, because it describes common patterns of organizational dysfunction with unparalleled clarity and generosity. Reading *The Advantage* helped me understand that many of the patterns of organizational dysfunction I had encountered in my career as a product manager were real and widespread. It also provided me with critical tools for addressing these patterns.

Good to Great (HarperBusiness, 2011) by Jim Collins

If You're Looking For: A meticulous, scientific breakdown of what makes an organization achieve great results.

How It Helped Me: Good to Great is an exhaustively researched, illuminating, and entertaining guide to *why* some companies succeed where others fail. There are great lessons here about organizational leadership, which I've found crucial for understanding when, why, and how to provide candid feedback to senior leaders. The follow-up *How the Mighty Fall* is also a great read.

Crucial Conversations (McGraw-Hill Education, 2011) by Al Switzler, Ron McMillan, Joseph Grenny, and Kerry Patterson

If You're Looking For: Strategies for having difficult conversations without becoming defensive, shutting down, or freaking out.

How It Helped Me: An absurdly high percentage of the work of product management comes down to suppressing and working through defensive and counterproductive reactions to other people's comments and questions. This book is an incredible resource for avoiding common communication traps for product managers, and is equally helpful for navigating difficult conversations in a personal context. The idea of "victim stories" as a way of dealing with conflict helped me understand and work through my own tendencies toward being a Product Martyr.

The Scrum Field Guide: Agile Advice for Your First Year and Beyond (Addison-Wesley Professional, 2012) by Mitch Lacey

If You're Looking For: Practical guidance on implementing Agile frameworks.

How It Helped Me: Early in my career as a product manager, I read a *lot* of books about Agile software development—and this was my favorite. Specifically, this book really helped me understand and navigate the reactions I could expect from my team as we began implementing Agile practices, and appreciate the value of starting with an off-the-shelf framework.

Data Science for Business (O'Reilly, 2013) by Tom Fawcett and Foster Provost

If You're Looking For: A genuinely useful and jargon-busting explanation of how data science works and how it can help meet business goals.

How It Helped Me: This book explains the fundamentals of data science in clear, concise, and actionable terms. It completely changed the way I was able to collaborate with data scientists by giving me the language to tie their work directly to business goals.

Just Enough Research (A Book Apart, 2013) by Erika Hall

If You're Looking For: Straightforward and useful guidance on conducting research to learn about stakeholders, competitors, and users.

How It Helped Me: This book provides an incredibly valuable balance of specific research approaches and high-level guidance about *why* and *how* to conduct research. It is both a handy reference and a fun read, concise and

useful and engaging through and through. I usually keep it on hand whenever I'm embarking on a project that involves any kind of research, both to read up on specific techniques and to realign my overall approach as needed.

Lean Analytics (O'Reilly, 2013) by Benjamin Yoskovitz and Alistair Croll

If You're Looking For: A no-nonsense way to use analytics to understand what's actually going on with your product and business.

How It Helped Me: There are many great books in the *Lean Startup* series, but this one is my very favorite. Even as a person who has sometimes balked at the idea of quantitative goals, I found this book very helpful in thinking through how and why analytics can be used to improve the way that organizations work.

Radical Focus (Boxes & Arrows, 2016) by Christina Wodtke

If You're Looking For: More information about the Objectives and Key Results framework, or just a fresh take on setting organizational goals overall.

How It Helped Me: Having implemented the Objectives and Key Results framework with varying degrees of success at different organizations, I was thrilled to find a book that describes OKRs in compelling, narrative terms. Nearly every mistake a team can make when implementing OKRs is brought to light here, and Wodtke's emphasis on focus as a goal serves as an important reminder that the goals we set also provide critical guidance about what *not* to do or build.

Articles and Blog Posts Cited in This Book

"Product Management for the Enterprise" by Blair Reeves.
http://bit.ly/ReevesMedium

"Leading Cross Functional Teams" by Ken Norton.
http://bit.ly/NortonCross

"What Exactly Is a Product Manager" by Martin Eriksson.
http://bit.ly/ErikssonProd

"Getting to Technical Enough as a Product Manager" by Lulu Cheng.
http://bit.ly/ChengTechnical

"Good Product Manager, Bad Product Manager" by Ben Horowitz.
http://bit.ly/HorowitzProd

"The One Number You Need to Grow" by Frederick F. Reichheld. In *The Harvard Business Review*.
http://bit.ly/HBROneNumber

"Recommendation Engines Aren't for Maximising Metrics, They Are for Designing Experiences" by Michael Dewar.
http://bit.ly/DewarMedium

"The Failure of Agile" by Andy Hunt.
http://bit.ly/HuntFailure

"Rediscovering the Heart of Agile" by Alistair Cockburn.
http://bit.ly/CockburnHeart

Index

D

N

O

P

R

S

T

U

V

W

About the Author

Matt LeMay is a product management coach and consultant. He has helped build and scale product management practices at companies ranging from early-stage startups to Fortune 500 enterprises. Matt was selected as a Top 50 Product Management influencer by the PM Year in Review for both 2015 and 2016.

Matt is cofounder and partner at Sudden Compass, a consultancy that helps organizations take a cross-functional and customer-centric approach to working with data. In his work as a technology communicator, Matt has developed and led digital transformation and data strategy workshops for companies like GE, American Express, Pfizer, McCann, and Johnson & Johnson.

Previously, Matt worked as Senior Product Manager at music startup Songza (acquired by Google), and Head of Consumer Product at Bitly. Matt is also a musician, recording engineer, and the author of a book about singer-songwriter Elliott Smith. He lives in Brooklyn, NY, with his wife Joan and their turtle Sheldon. You can find more of his work online at *mattlemay.com*.

Colophon

The cover illustration is by Jose Marzan Jr. The cover font is Guardian Sans. The text font is Scala Pro; the heading and sidebar font is Benton Sans.