AI-Powered Product Management

Think Faster, Ship Smarter, Lead Through the Shift

Tulio Marques Soria

2026

AI-Powered Product Management: Think Faster, Ship Smarter, Lead Through the Shift

First Edition

Contents

Introduction

AI isn't new. What's new is that the interface to intelligence moved from specialists to everyone.

For decades, AI was mostly invisible—models embedded in search ranking, fraud detection, recommendations. You benefited from it without thinking about it. Machine learning powered your email spam filter, your credit score, your Netflix queue. The technology existed. It just didn't feel like a conversation.

Generative AI changed the psychology. Suddenly the output looks like language, decisions, reasoning, and creativity. It feels human enough that people treat it like a teammate. You type a question, and something types back. You ask for a draft, and you get prose. You describe a flow, and you get a diagram.

That shift didn't just add a tool. It changed how products are imagined, built, and shipped.

This Is a Workflow Wave

The mistake is to think this is only a technology wave. It's a workflow wave. It compresses the time between idea and artifact.

A PM can now draft requirements, generate variants, simulate edge cases, summarize research, and produce first-pass UX copy in minutes. That's not magic. It's leverage. And like all leverage, it rewards teams with good judgment and punishes teams with bad fundamentals.

Here's the uncomfortable truth: if your strategy is unclear, AI just helps you ship confusion faster.

I've watched teams adopt AI tools and get worse, not better. They generate more documents nobody reads. They produce more variants nobody tests. They move faster in the wrong direction. The problem wasn't the tool. The problem was that speed amplified existing dysfunction.

I've also watched teams adopt AI and feel like they gained a superpower. They cut weeks from their discovery cycles. They tested hypotheses that used to sit in backlogs. They spent less time formatting and more time thinking. The difference wasn't the model. It was the fundamentals underneath.

What This Book Is About

This book is for Product Managers who want to stay sharp, not be replaced. It's for people who build things and ship outcomes—not people who want to talk about AI at conferences without using it in the work.

I'm going to show you how AI rewires product work, from specs to strategy. You'll get examples, tools, and uncomfortable truths about what's now possible—and expected—from modern PMs.

But let me be clear about what this isn't.

This isn't a book about chasing trends. It's not about hallucinating product ideas or pretending that prompts replace judgment. It's a guide for adults who make trade-offs and ship outcomes. You'll learn how to blend leverage with clarity, how to treat models like interns not oracles, and how to lead in a world where thin-slice velocity beats roadmap theater.

The New Competitive Edge

In the AI era, your competitive edge is not "having AI." Everyone has AI. Your edge is how quickly you learn what matters.

PM work has always been about reducing risk: market risk, usability risk, technical risk, execution risk. AI doesn't remove those risks. It just changes how fast you can run the cycle: hypothesis → prototype → feedback → iteration.

So the bar moves. If your team still takes weeks to produce the first usable artifact, you're going to feel slow in a world where iteration is measured in days. The solution isn't working harder. It's restructuring your workflow: smaller releases, better instrumentation, tighter decision logs, and a bias toward experiments that can fail cheaply.

My Operating Principle

Here's the rule I keep coming back to: AI can propose, but it can't decide.

It can write a user story, but I still define the outcome. It can suggest acceptance criteria, but I still own what "done" means. It can summarize customer interviews, but I still check the raw notes when the decision is expensive.

Leverage is earned through verification.

The moment you treat AI output as truth, you get sloppy. The moment you treat it as a junior assistant that works fast but hallucinates sometimes, you get value.

Using AI as a PM is not about outsourcing thinking. It's about accelerating the parts of the job that are pure throughput—drafting, summarizing, clustering feedback, generating alternatives. The thinking is still yours. The judgment is still yours. The accountability is still yours.

Who This Book Is For

This book is for Product Managers at any level who need to stay competitive in the AI and GenAI era. That includes:

- **New PMs** who are entering a field that looks different than it did two years ago. The expectations have shifted. The tools have shifted. You need to build habits that match the current landscape, not the one described in older PM literature.
- **Experienced PMs** who've been doing this work for years and are wondering how AI changes what they do. The answer is: it changes the speed, not the job. Strategy, alignment, prioritization, stakeholder management—all still your responsibility. But the feedback loops are tighter, and the excuse of "we didn't have time to test that" is getting thinner.

- **PM Leaders** who need to help their teams adopt AI without losing rigor. The risk isn't that your team ignores AI. The risk is that they adopt it sloppily—generating noise, skipping verification, treating outputs as decisions. You need frameworks to prevent that.

What You'll Learn

Each chapter in this book tackles a specific aspect of AI-powered product management:

Chapter 1 explores the new landscape—why this shift feels different, how it changes expectations, and what it means for throughput and team dynamics.

Chapter 2 gives you enough technical context to make smart decisions. You don't need to become a machine learning engineer, but you do need to understand what models can and can't do, and how to frame AI work as trade-offs.

Chapter 3 shows AI in action—how it speeds up specs, clustering, prioritization, and research without making you sloppy.

Chapter 4 focuses on data, feedback, and learning loops. Your job is reducing risk with reality. AI makes it faster to test hypotheses, but only if you measure right and move fast.

Chapter 5 covers building AI-powered products—how to scope, integrate, test, and ship with models in the loop, ethically and responsibly.

Chapter 6 is about leading with leverage—how to align teams, debug incentives, and avoid hero mode in an AI-accelerated org.

Chapter 7 looks ahead at agents, autonomy, and the next interface. The roadmap gets weird. You need to know how to prep without chasing vapor.

The Uncomfortable Truth

If this book is about anything, it's about shifting from "building features" to "building learning machines." The AI era doesn't reward the team with the best slide deck. It rewards the team that can run experiments faster, interpret feedback honestly, and change course without ego.

There's also a personal side to this. If your identity is attached to being right, the AI era will humble you, because the marketplace moves too fast for pride. The healthier mindset is: I'm not here to be right; I'm here to reduce risk. My job is to get closer to reality every week.

The real upgrade isn't AI. It's honesty. Honest assumptions. Honest metrics. Honest post-mortems. AI makes it harder to hide behind process because it exposes how much work was just slow writing and slow coordination.

If you adopt AI and keep the old habits—vague goals, unclear ownership, roadmap-as-wish-list—you'll simply ship confusion at higher speed.

If you adopt AI and tighten fundamentals, you'll feel like you gained a superpower.

How to Read This Book

You can read this book front to back, or you can jump to the chapter that matches your current pain.

If you're skeptical about the hype, start with Chapter 2. It will ground you in what models actually do.

If you're already using AI but feeling sloppy, start with Chapter 3. It will give you structure.

If you're leading a team and worried about adoption, start with Chapter 6. It will help you set guardrails.

If you're wondering whether this whole thing is going to make your job obsolete, read the Conclusion. Spoiler: it won't. But it will change what "good" looks like.

Let's Begin

I wrote this book because I believe the hard part isn't AI. The hard part is admitting we've been shipping opinions dressed up as requirements.

In the AI era, your advantage is not having the fanciest model—it's having the fastest learning loop. If you can ship a thin slice, measure behavior, and iterate weekly, you'll beat teams that spend three months polishing a plan that nobody validates.

That's what we're going to build together. Not a new set of tools. A new operating system for product work.

Let's get started.

Chapter 1

The Long Road to Here: AI's Timeline from 1940 to 2026

This chapter covers ancient history. Not in a boring way. In the way that matters for understanding why 2023 felt different.

If you want to know why your job is changing, you need to know why the previous 80 years of AI didn't change it. The pattern is clear: AI repeatedly hit walls. Walls of computing power. Walls of data. Walls of fundamental understanding. What changed in 2022-2023 wasn't magic. It was the convergence of three things finally becoming true at the same time.

1.1 The Dream Begins (1940s-1950s)

The idea that machines could think isn't new. What's new is that we built machines that could prove it.

In 1950, Alan Turing asked a question that would haunt computer science for decades: *Can machines think?* He didn't pretend to answer it philosophically. Instead, he proposed something practical: if a machine could hold a conversation indistinguishable from a human, then for practical purposes, it thinks. The Turing Test. Simple. Dangerous. Still relevant.

By the mid-1950s, researchers had built the first neural networks. Warren McCulloch and Walter Pitts showed in 1943 that networks of simple artificial neurons could perform logical operations. Marvin Minsky and Dean Edmonds built the SNARC in 1951—the first neural net machine. It worked. It was slow. It was expensive. It would be forgotten for decades.

Then came the Dartmouth Workshop in the summer of 1956. This is where AI got its name and its mission. John McCarthy, Marvin Minsky, Claude Shannon, and others gathered to test a bold assertion: that every aspect of human intelligence could be precisely described and simulated by a machine.

They believed it. They really believed it. Within a decade, one of them predicted machines would match human intelligence. The timeline? Ten years. Twenty years max. Humanity would have artificial general intelligence by the late 1960s.

This is the optimism before the wall.

1.2 Early Wins, Hidden Limits (1956-1974)

The 1960s were impressive. Computers solved algebra word problems. They proved geometry theorems. They beat humans at checkers. ELIZA, a chatbot that barely did anything but restate what you said, fooled people into thinking it was a therapist. The "Look Ma, no hands!" era, as John McCarthy called it. Astonishment at anything clever.

Government funding poured in. Millions of dollars from ARPA (now DARPA) and other agencies. Entire laboratories dedicated to cracking human intelligence. MIT, Stanford, Carnegie Mellon, Edinburgh. The smartest people in the world were working on this problem.

But there was a problem with the solution. Three of them, actually.

First: computers were weak. Not slow. Weak. A model trained on 20 words of vocabulary was impressive because computers couldn't hold more. Memory was measured in kilobytes. Processing speed in kilohertz. When Hans Moravec calculated what it would take to match human vision, he realized you'd need a computer millions of times faster than anything that existed.

Second: the combinatorial explosion. Most intelligent tasks require searching through possibilities—millions, billions, trillions of them. The early approaches tried to search every option. For non-trivial problems, this is impossible. You'd need to search until the heat death of the universe.

Third: common sense. A five-year-old knows that birds fly but penguins don't. That when you knock over a glass of water, it spills. That people have intentions and beliefs. Encoding this into a machine seemed to require billions of atomic facts, manually entered. No one in 1970 could build that database. No one knew how machines might learn it.

The researchers didn't give up. They kept working. But by 1974, the promises had clearly exceeded reality. The U.S. and British governments cut funding. The Lighthill Report in the UK concluded that AI had failed to achieve its "grandiose objectives." The money stopped. Over 300 AI companies shut down or went bankrupt.

This was the First AI Winter. It lasted six years.

1.3 Revival and Optimism Trap (1980-1987)

By 1980, expert systems arrived. Not general intelligence. Something narrower. Systems that encoded expert knowledge in a specific domain—medical diagnosis, equipment configuration, insurance underwriting. They worked. More importantly, they made money.

R1, an expert system for Digital Equipment Corporation, saved the company $40 million a year by 1986. Suddenly corporations around the world were building expert systems. By 1985, companies were spending over a billion dollars on AI, mostly on in-house projects.

The industry boomed. Specialized hardware companies like Symbolics and Lisp Machines existed just for AI. Software companies sprang up. Japan launched the Fifth Generation Computer Project with massive government backing. The U.S. responded with the Strategic Computing Initiative. Funding tripled between 1984 and 1988.

It was another false summit.

Expert systems worked because they worked in small domains with clear rules. They failed when the world was messy. When assumptions changed. When you needed judgment, not just encoding. By the late 1980s, companies realized they'd spent billions on systems that couldn't do what the sales pitch promised.

In 1987, the specialized AI hardware market collapsed. Desktop computers from Apple and IBM had become faster and cheaper. Why buy a million-dollar Lisp Machine when a $10,000 PC could do the job? The hype reversed. By 1991, Japan admitted its Fifth Generation goals hadn't been met. By 1993, over 300 companies had shut down or been acquired.

This was the Second AI Winter. It lasted through the 1990s.

1.4 Hidden Progress (1990s-2000s)

Here's what's strange about the second winter: AI didn't stop. It just stopped being called AI.

Researchers started calling their work "machine learning," "computational intelligence," "informatics," "knowledge-based systems"—anything but AI. The name had been poisoned by broken promises.

But underneath, breakthroughs were happening. Probabilistic reasoning. Statistical methods. Neural networks (nobody called them that anymore). Google's search engine used AI techniques. Banks used AI for fraud detection. Logistics companies used AI for routing. Spam filters used AI for classification.

Nick Bostrom's observation captures it: "Once something becomes useful enough and common enough, it's not labeled AI anymore."

By the late 1990s, the field had fragmented into specialized sub-problems. Computer vision. Speech recognition. Natural language processing. Machine learning. Each became a mathematical discipline with rigorous methods. Success came from collaboration with statisticians, mathematicians, electrical engineers, operations researchers. AI became science again instead of prophecy.

Moore's Law was quietly doing the work. Computing power doubled every two years. By 1997, Deep Blue beat Garry Kasparov at chess—not with human-like thinking, but with brute force and smart pruning. A billion calculations a second, evaluating millions of positions. It wasn't elegant. It was effective.

1.5 The Machine Learning Turn (2005-2017)

By 2005, something shifted. Data became abundant. The internet had created digital libraries. Companies had accumulated massive datasets. Fei-Fei Li released ImageNet in 2009—three million labeled images. Word embeddings like word2vec captured meaning in vectors. You could do vector math and get equivalences: King - Man + Woman = Queen.

But the breakthrough came from an old idea made new. Neural networks returned. Deep neural networks. Multiple layers. A technique called backpropagation (developed in the 1970s but never practical) suddenly worked when you had enough data and enough computing power.

In 2012, AlexNet, a deep convolutional neural network, won the ImageNet competition with significantly fewer errors than traditional methods. It was a pivot point. Within years, every other approach to image recognition was abandoned. Deep learning was the new paradigm.

Deep learning worked because it could learn patterns in data without humans hand-coding features. Give it millions of images. It learns. Give it millions of sentences. It learns patterns of language. The constraint wasn't cleverness anymore. It was data and computation.

Computing power was finally catching up. GPUs (graphics processing units) that were designed for video games turned out to be perfect for neural network math. Cloud computing made that power accessible. Amazon, Google, Microsoft all offered AI services in the cloud.

By 2016, investment in AI was surging. Competitions. Conferences. Companies. DeepMind was solving protein folding. OpenAI had just formed. Sentiment shifted. AI wasn't dead. It was coming back.

1.6 The Transformer Moment (2017-2022)

In 2017, researchers at Google published a paper called "Attention is All You Need." The transformer architecture. At the time, it seemed like an incremental improvement. In hindsight, it was the architecture that would change everything.

Transformers could train on massive amounts of text. They could capture long-range dependencies in language. They could be scaled up without losing efficiency. You could train a transformer on billions of words and it would learn statistical patterns about language, knowledge, reasoning, even style.

By 2020, GPT-3 was released. 175 billion parameters trained on hundreds of billions of words from the internet. When you prompted it, it would generate coherent text. It could write code. Tell stories. Explain concepts. It wasn't perfect, but it was remarkable.

Nobody knew it would change the world in two years.

1.7 The Interface Shift (2022-2023)

ChatGPT launched on November 30, 2022. It hit 100 million users in two months. Faster adoption than any application in history.

Here's why it mattered: AI stopped being invisible. It stopped being a feature inside another product. It became the product. An interface you could talk to. Something that felt like intelligence.

It wasn't technically different from GPT-3 (the underlying model was similar). But the interface was. Simple. Accessible. Free. You could write a prompt and get an answer in seconds. You could refine. Iterate. Have a conversation.

Suddenly, non-technical people experienced AI. Lawyers used it. Marketers used it. Product managers used it. Teachers used it. It worked. Not perfectly. But it worked.

The response from big tech was immediate. Google declared "code red." Released Gemini (formerly Bard). Microsoft integrated the technology into Bing Chat. Amazon, Apple, Meta, all scrambled.

By mid-2023, the venture capital world had moved. Generative AI funding skyrocketed. AI companies were raising billions. OpenAI's valuation approached $86 billion by early 2024.

1.8 Why Now. Why This Is Different.

You could ask: why didn't this happen in 2016? Why not 2018? Why 2022-2023?

Three things converged:

Computing power became cheap and accessible.

The transformer architecture is computationally expensive. Training GPT-3 cost millions of dollars. But once trained, it could run on commodity hardware. Or be rented from the cloud. A startup with $100K could use world-class AI models. No expensive Lisp Machines. No supercomputers.

Moore's Law had finally done the work. Hardware was fast enough.

Data became abundant and labeled.

ImageNet. Word2vec. Wikipedia. GitHub. The internet itself. Billions of texts, images, code samples. Freely available.

In the 1970s, the bottleneck was data. You couldn't get enough labeled examples. By 2020, the bottleneck had flipped. You had too much data. The challenge was efficiently learning from it.

The transformer architecture turned out to be efficient at scale. Bigger models trained on bigger datasets worked better. The scaling laws held. This was the surprise. Some theorists thought you'd hit diminishing returns. You didn't. You kept getting smarter.

The interface made it human.

GPT-3 was impressive to researchers. ChatGPT was impressive to everyone else.

The difference: a conversation. You could interact with it like a person. Ask follow-ups. Refine your prompt. See it struggle with ambiguity and respond honestly about uncertainty.

That human-like interface meant non-technical people could try it. Use it. Form opinions. Ask for more. The network effects kicked in. Every person who tried ChatGPT told five people. Developers built applications on top of it.

The barrier to adoption disappeared.

1.9 The View from 2026

Here's what we know now, that the 1950s researchers didn't:

Intelligence isn't one thing. General intelligence is possible without understanding. These models don't think like humans. They predict the next token with statistical precision. But that prediction, scaled across billions of parameters and trained on trillions of words, approximates something that looks like reasoning, creativity, judgment.

The walls that stopped AI for 50 years—computing power, data scarcity, architectural limits—turned out to be solvable. Not magically. Through engineering, investment, and time.

But new walls emerged. Models hallucinate. They reflect biases in training data. They can't access real-time information. They struggle with causal reasoning. They don't understand when they should say "I don't know." They consume enormous amounts of energy and resources.

And there's a philosophical question that hasn't been answered: if you can predict the next word so well that it approximates human language, have you created understanding? Or just a very sophisticated mirror?

Product managers don't need to solve that. You need to know: it works well enough for most practical tasks. It's improving. It's accessible. And it changes what's possible in your workflow.

The long road from 1956 to 2023 wasn't a failure. It was prerequisite work. Each failed approach taught something. Each winter created space for new ideas. Each boom and bust clarified what actually mattered.

We're not in a new age because we invented intelligence. We're in a new age because we finally made intelligence accessible.

That's a different thing. And it changes everything that happens next.

Chapter 2

The New Product Management Landscape

The bar has moved. If you haven't felt it yet, you will.

Two years ago, a product team could spend three weeks producing its first testable artifact—a clickable prototype, a spec with enough detail to estimate, a research synthesis that clarified the problem. That pace was normal. Nobody questioned it.

Today, that same timeline feels slow. Not because expectations are unreasonable, but because the tools changed. Teams that use AI effectively are shipping first drafts in days, not weeks. They're testing assumptions before the old teams finish their kickoff meetings. The gap is visible, and it's growing.

This chapter is about understanding why this shift feels different—and what it means for how you work.

2.1 The Interface Shift

AI isn't new. What's new is the interface.

For decades, artificial intelligence was the domain of specialists. Data scientists built models. Engineers deployed them. Product managers wrote requirements and hoped for the best. The technology was powerful, but it was hidden behind APIs, pipelines, and technical debt. You didn't interact with AI. You consumed its outputs: a recommendation, a score, a ranking.

Generative AI changed that. Now the interface is language. You type a question. You get an answer. You describe what you want. You get a draft. The barrier between "having an idea" and "seeing a version of that idea" collapsed.

This matters because product work is fundamentally about reducing the distance between hypothesis and evidence. The faster you can turn a guess into something testable, the faster you learn. And learning speed is the only sustainable advantage.

2.2 From Technology Wave to Workflow Wave

Here's the mistake most people make: they think this is a technology wave. It's not. It's a workflow wave.

A technology wave gives you new capabilities. A workflow wave changes how you spend your time. The smartphone was a technology wave. But the real impact wasn't the device—it was the workflow shift: always-on communication, location-aware services, app-based everything. The technology enabled the shift. The shift changed the work.

GenAI is similar. Yes, the models are impressive. But the real impact is what happens to your Tuesday afternoon. Instead of spending two hours drafting a PRD, you spend twenty minutes refining one. Instead of scheduling a meeting to brainstorm edge cases, you generate fifteen in a prompt and then discuss the interesting ones. Instead of waiting for a research synthesis, you produce a first-pass clustering and validate it against the raw data.

The time compression is real. And it changes what's possible.

2.3 What Changes for PMs

Let's get specific about what this means for product managers.

Discovery Gets Faster

Discovery used to be bottlenecked by throughput. You could only conduct so many interviews. You could only synthesize so many notes. You could only generate so many hypotheses before the quarter ended and you had to ship something.

AI changes the throughput equation. You can now:

- Summarize twenty interview transcripts in an hour
- Generate hypothesis variants you wouldn't have thought of
- Draft discussion guides, screener surveys, and synthesis frameworks in minutes
- Cluster feedback from support tickets, NPS comments, and sales calls simultaneously

This doesn't mean discovery is "solved." It means the constraint shifts. The bottleneck is no longer "can we process this information?" It's "do we know what question we're trying to answer?"

Artifacts Get Cheaper

Writing a PRD used to take time. Generating wireframes used to take time. Producing a competitive analysis used to take time. All of these are now dramatically cheaper.

That's good news if you know what you're building. It's dangerous if you don't.

When prototypes become cheap, strategy becomes more important, not less. Because now it's easier than ever to build something that looks impressive and does nothing. If you can generate ten flows in a day, you can also generate ten wrong flows in a day. The constraint shifts from production capacity to judgment.

Iteration Cycles Shrink

The traditional PM cycle looked something like this: identify problem → research → define solution → build → launch → measure → iterate. Each step took time. Each handoff introduced delay. A full cycle might take a quarter.

AI compresses the early stages. You can go from "I think this might be a problem" to "here's a testable prototype" in days instead of weeks. That means more cycles per quarter. More chances to learn. More opportunities to be wrong cheaply instead of wrong expensively.

But here's the catch: if your organization isn't set up for fast iteration, you'll just create a pileup. You'll generate artifacts faster than the team can review them. You'll produce hypotheses faster than you can test them. Speed without structure creates chaos.

2.4 The New Expectations

This shift creates new expectations—some explicit, some implicit.

Speed Is Now Visible

Before AI, it was hard to compare team velocities. One team's "two-week sprint" might produce different outputs than another's. The work was opaque.

Now, speed is visible. If one PM can produce a comprehensive competitive analysis in an afternoon, and another takes two weeks, the difference is obvious. The excuse of "I'm being thorough" only works if the output is proportionally better. Often, it's not.

This creates pressure. Some of that pressure is healthy—it pushes people to work smarter. Some of it is unhealthy—it rewards quantity over quality. Your job is to navigate that distinction.

Stakeholders Expect More

Stakeholders have noticed the tools too. They've seen the demos. They've read the headlines. They've used ChatGPT to draft their own emails.

The result: they expect more from product teams. Faster turnaround. More options. Better documentation. Whether or not those expectations are reasonable, they exist. You need to manage them.

My approach: set expectations early about what AI can and can't do. Explain that AI accelerates drafting, not deciding. Show stakeholders how you're using the tools, so they understand the process. Otherwise, you'll get requests like "can't you just have AI write the roadmap?" and you'll spend more time explaining why that's a bad idea than you would have spent writing the roadmap.

Quality Standards Are Shifting

Here's a nuance most people miss: AI raises the floor and lowers the ceiling.

It raises the floor because anyone can now produce a decent first draft. The gap between a junior PM's output and a senior PM's output has shrunk on the first pass. AI can help you avoid obvious mistakes, structure your thinking, and produce something presentable quickly.

It lowers the ceiling because AI outputs tend toward the average. They're trained on existing content, so they produce variations of what's already been said. Truly original thinking—the kind that creates breakthrough products—still requires human insight. AI can help you get to "good" faster, but it won't take you to "exceptional" on its own.

The implication: if you're competing on speed-to-good, AI is your friend. If you're competing on exceptional insight, AI is a starting point, not a destination.

2.5 The Risks Nobody Talks About

Let's talk about the downsides.

Strategy Drift

When artifacts are cheap, it's tempting to produce a lot of them. More PRDs. More prototypes. More analyses. But quantity is not strategy.

I've seen teams drown in AI-generated documents that nobody reads. They feel productive because they're producing output. But output without direction is just noise. If you don't have a clear strategy, AI helps you ship confusion faster.

The fix: before you generate anything, answer four questions. What user? What moment? What pain? What outcome? If you can't answer those, you don't need AI—you need clarity.

Verification Fatigue

AI outputs need to be checked. They hallucinate. They miss nuance. They produce plausible-sounding nonsense.

When you're producing a lot of AI-assisted content, verification becomes exhausting. It's easier to skim and approve than to actually validate. That's when mistakes slip through.

My rule: the more important the decision, the more carefully I check. For a first-pass brainstorm, I'll skim. For a PRD that's going to engineering, I read every line. For a customer-facing message, I verify every claim. Calibrate your verification to the stakes.

Skill Atrophy

If AI writes all your first drafts, you might forget how to write first drafts.

This sounds minor, but it matters. The ability to structure an argument, identify what's missing, and communicate clearly—these are core PM skills. If you outsource them entirely, you'll lose them. And when AI fails (which it will), you'll have nothing to fall back on.

My approach: I still write some things from scratch. Not because it's efficient, but because it keeps the skill sharp. I treat AI as a sparring partner, not a replacement.

2.6 Restructuring Your Workflow

Given all of this, how should you actually work?

Smaller Releases

If you can produce artifacts faster, you should ship them faster. Not in the sense of rushing—in the sense of getting feedback sooner.

Break work into smaller chunks. Instead of a comprehensive PRD, ship a one-page problem statement and validate it before expanding. Instead of a full prototype, ship a single flow and see how users react. The goal is to learn before you invest.

Better Instrumentation

Speed without measurement is just fast guessing. If you're iterating quickly, you need to know whether your iterations are working.

Invest in instrumentation. Track the metrics that matter. Build dashboards that update in real time. The faster your feedback loops, the more valuable your speed becomes.

Tighter Decision Logs

When you're moving fast, it's easy to forget why you made certain choices. Document your decisions. Write down the assumptions. Note what you expected to happen.

This isn't bureaucracy. It's protection. When things go wrong (and they will), you'll want to know what you were thinking. When things go right, you'll want to understand what worked. Decision logs are the raw material for learning.

Bias Toward Cheap Experiments

Not all experiments are equal. Some require significant investment—engineering time, marketing spend, operational changes. Others can be tested with a landing page, a survey, or a conversation.

Bias toward cheap experiments. Run the quick tests first. Validate assumptions before committing resources. If you can disprove a hypothesis with a twenty-minute test, do that before you spend two weeks building something.

2.7 The Bottom Line

The product management landscape has changed. Not because AI is magic, but because the interface to intelligence shifted—and that shift changed the speed of work.

Teams that adapt will iterate faster, learn more, and build better products. Teams that don't will feel increasingly slow, increasingly behind, and increasingly frustrated.

The adaptation isn't complicated. It's structural:

- Ship smaller
- Measure everything
- Verify AI outputs
- Maintain your core skills
- Anchor everything in strategy

The tools are available to everyone. The edge comes from how you use them.

That's why I keep coming back to one sentence: what user, what moment, what pain, what outcome. If you can't answer that, you don't need a bigger model—you need a clearer problem.

Strategy is the filter that prevents you from shipping noise faster.

Chapter 3

AI Fundamentals for PMs

You don't need to become a machine learning engineer. But you do need to understand what you're working with.

This chapter gives you enough technical context to make smart decisions. Not enough to build a model—enough to ask the right questions, spot the real constraints, and avoid the mistakes that come from treating AI like magic.

3.1 What Models Actually Do

Let's start with the basics. A large language model (LLM) is a statistical system trained on vast amounts of text. It learns patterns: which words tend to follow which other words, how sentences are structured, what kinds of responses match what kinds of prompts.

When you ask a model a question, it's not "thinking" in the way you think. It's predicting what a plausible response looks like, based on patterns in its training data. This is important because it explains both the power and the limitations.

The power: models can generate fluent, coherent, contextually appropriate text across an enormous range of topics. They can summarize, translate, explain, brainstorm, and draft. They can handle ambiguity and produce outputs that feel intelligent.

The limitation: they don't "know" anything in the way you know things. They don't have beliefs. They don't verify facts. They don't have access to current information unless it's in their training data or explicitly provided. They produce plausible outputs, which is not the same as producing true outputs.

This distinction matters every time you use AI in your work.

3.2 The Trade-Off Framework

Every AI decision involves trade-offs. The sooner you internalize this, the better your decisions will be.

Speed vs. Accuracy

AI is fast. Really fast. You can generate a first draft in seconds that would take you an hour to write from scratch. But speed comes with accuracy risk.

Models make mistakes. They hallucinate facts. They misunderstand context. They produce outputs that sound confident but are wrong. The faster you move, the more likely you are to miss these errors.

The trade-off: you can have fast outputs or verified outputs, but verification takes time. Calibrate based on stakes. For brainstorming, speed wins. For customer-facing content, accuracy wins. For internal documents, find the middle ground.

Capability vs. Cost

More powerful models generally cost more to run. They're also slower. The most capable model isn't always the right choice.

For simple tasks—summarizing a short document, generating a list of options, reformatting text—a smaller, cheaper model often works fine. For complex reasoning, nuanced writing, or handling ambiguity, you might need the heavy artillery.

The trade-off: don't use a sledgehammer for a nail. Match the model to the task. If you're building AI into a product, this becomes a direct cost question. If you're using AI as a personal tool, it's still worth considering—do you really need the premium tier for this task?

Context vs. Constraints

Models have context windows—the amount of text they can consider at once. Bigger context windows let you include more information (documents, conversation history, examples) but they also cost more and can slow down responses.

This creates a constraint: you can't always give the model everything it might need. You have to choose what to include. That choice shapes the output.

The trade-off: more context isn't always better. Sometimes a focused prompt with key information outperforms a bloated prompt with everything. Learn to curate what you feed the model.

Customization vs. Generalization

Out-of-the-box models are generalists. They know a lot about a lot, but they don't know your specific context—your product, your users, your terminology, your constraints.

You can customize through fine-tuning, retrieval-augmented generation (RAG), or careful prompting. Each approach has costs: time to set up, ongoing maintenance, potential for errors in the customization layer.

The trade-off: generic models are faster to deploy but less tailored. Custom setups are more relevant but require investment. Choose based on how specific your needs are and how often they change.

3.3 What Models Can't Do

This is the section most AI enthusiasts skip. Don't skip it.

They Can't Verify Facts

Models don't have a truth detector. They generate plausible text based on patterns. If the pattern suggests that a certain fact is true, they'll state it confidently—whether or not it's actually true.

This means: every factual claim from an AI needs verification. Especially for numbers, dates, names, and technical specifications. The confidence of the statement is not evidence of its accuracy.

They Can't Access Real-Time Information

Unless explicitly connected to live data sources, models only know what was in their training data. They don't know what happened yesterday. They don't know your company's current metrics. They don't know what your competitor announced this morning.

This means: for anything time-sensitive, you need to provide the information or verify against current sources. Don't assume the model knows what's happening now.

They Can't Make Decisions for You

This is the big one. Models can generate options, analyze trade-offs, summarize information, and suggest approaches. They cannot tell you what's right for your specific situation.

Why? Because "right" depends on context that the model doesn't fully have: your strategy, your constraints, your stakeholders, your risk tolerance, your values. The model can help you think, but the thinking is still yours.

My rule: AI can propose, but it can't decide. I use it to expand my options, challenge my assumptions, and draft my documents. I don't use it to make choices I'm accountable for.

They Can't Replace Judgment

Related but distinct: models don't have judgment. They don't know when to push back, when to ask clarifying questions, when to say "this is a bad idea."

A good PM challenges requirements, questions assumptions, and identifies edge cases. AI will do what you ask. That's a feature and a bug. It makes AI useful for execution, but it means you need to provide the judgment layer.

3.4 Hallucinations and How to Handle Them

Let's talk about the elephant in the room.

Models hallucinate. They make things up. They invent citations, create fake statistics, and describe products that don't exist. This isn't a bug that will be fixed—it's a fundamental characteristic of how these systems work.

The causes are structural. Models are optimizing for plausible outputs, not true outputs. If a plausible response includes a made-up fact, the model will generate it without hesitation. It doesn't know it's making something up.

Types of Hallucinations

Factual hallucinations: The model states something as fact that isn't true. "The company was founded in 2015" when it was actually founded in 2018.

Citation hallucinations: The model invents references that don't exist. "According to a 2023 Harvard study..." when no such study exists.

Capability hallucinations: The model claims it can do something it can't. "I can search the web for current information" when it has no web access.

Confidence hallucinations: The model presents uncertain information with false confidence. No hedging, no caveats, just wrong statements delivered with authority.

Mitigation Strategies

You can't eliminate hallucinations, but you can manage them:

Verify important claims: If a fact matters for your decision, check it against a primary source. Don't trust AI-generated statistics, quotes, or technical specifications without verification.

Ask for sources: When asking AI to summarize research or cite evidence, ask it to provide sources. Then verify those sources exist and say what the AI claims.

Use retrieval augmentation: If you're building AI into products, consider RAG architectures that ground responses in actual documents. This reduces (but doesn't eliminate) hallucination risk.

Calibrate trust to stakes: For low-stakes brainstorming, hallucinations are a minor nuisance. For high-stakes decisions, they're dangerous. Adjust your verification effort accordingly.

Build verification into your workflow: Don't treat verification as optional. Build it into the process. First pass: generate. Second pass: verify. Make this explicit.

3.5 Prompt Engineering: The Basics

Prompting is how you communicate with models. Better prompts get better outputs. Here's what actually matters.

Be Specific

Vague prompts get vague outputs. Instead of "write a product description," try "write a 100-word product description for a B2B project management tool, targeting engineering managers who are frustrated with tool sprawl, emphasizing simplicity and integration."

The more context and constraints you provide, the more targeted the output.

Provide Examples

Models learn from patterns. If you want a specific format or style, show examples. "Here's an example of the tone I want: [example]. Now write something similar for [new topic]."

This is sometimes called few-shot prompting. It works remarkably well for style, format, and structure.

Break Complex Tasks into Steps

Don't ask the model to do everything at once. For complex tasks, break them into steps. First, outline. Then, draft section one. Then, review and revise. Then, draft section two.

This gives you checkpoints to verify and correct before the model builds on mistakes.

Tell the Model What Not to Do

Negative constraints can be as useful as positive ones. "Don't use jargon. Don't make claims without evidence. Don't exceed 200 words." Sometimes telling the model what to avoid is the clearest way to shape the output.

Iterate

Your first prompt rarely produces the best output. Expect to iterate. Refine based on what you get. Add constraints. Provide more context. Show examples of what's not working.

Prompting is a conversation, not a one-shot request.

3.6 Framing AI Work as Trade-Offs

Let me bring this back to PM fundamentals.

Every AI decision is a trade-off decision. When you're deciding whether to use AI for a task, ask:

- What do I gain? (Speed, coverage, variety)
- What do I risk? (Accuracy, nuance, originality)
- What's the verification cost? (Time to check outputs)
- What happens if it's wrong? (Stakes of error)

For low-stakes, high-volume tasks, AI is almost always worth it. For high-stakes, nuanced decisions, AI is a starting point that requires significant human judgment.

The mistake is treating AI as either magic (it can do anything!) or useless (it's just autocomplete!). The reality is in between: it's a powerful tool with specific strengths and specific weaknesses. Your job is to deploy it where the strengths match your needs and the weaknesses are manageable.

3.7 The Technical Vocabulary You Need

You'll hear these terms in conversations with technical colleagues. Here's what they mean in practical terms:

LLM (Large Language Model): The models that power tools like ChatGPT, Claude, and Gemini. Trained on text, good at generating and understanding language.

Fine-tuning: Taking a general model and training it further on specific data to improve performance on specific tasks.

RAG (Retrieval-Augmented Generation): A technique that connects models to external documents, allowing them to reference specific information rather than relying only on training data.

Context window: The amount of text a model can consider at once. Bigger windows allow more information but cost more.

Token: The unit of text models work with. Roughly a word or part of a word. Pricing and context limits are often measured in tokens.

Temperature: A setting that controls randomness. Higher temperature means more creative/varied outputs. Lower temperature means more predictable/consistent outputs.

Prompt engineering: The practice of crafting inputs to get better outputs from models.

Hallucination: When models generate false or made-up information.

You don't need to master these concepts. You need to recognize them, understand roughly what they mean, and know when to ask for clarification.

3.8 The Bottom Line

AI is a tool. Like all tools, it has strengths and weaknesses. Like all tools, it rewards users who understand what it's actually doing.

The fundamentals for PMs:

- Models generate plausible outputs, not true outputs
- Every AI decision is a trade-off

- Verification is not optional for important work
- Better prompts get better outputs
- AI can propose, but it can't decide

The technical depth you need is shallow but important. You don't need to train models. You need to use them intelligently, know their limits, and avoid the mistakes that come from treating them as something they're not.

Leverage is earned through verification. That's the principle. That's the practice. That's how you use AI without outsourcing your judgment.

Chapter 4

AI in Action: Upgrading Your Workflow

Here's the truth about GenAI in product work: it doesn't replace thinking; it compresses the time between thinking and trying.

You can draft a PRD faster, generate ten variants of microcopy, prototype flows, summarize research, and turn support tickets into themes. But if your strategy is unclear, AI just helps you ship confusion faster.

This chapter is about integrating AI into your actual workflow—the daily work of specs, research, prioritization, and communication. I'll show you where it works, where it fails, and how to use it without getting sloppy.

4.1 The Workflow Integration Principle

Before we dive into specifics, let me share a principle that guides my approach:

AI accelerates throughput. It doesn't fix fundamentals.

If you know what you're building and why, AI makes you faster. If you don't, AI makes you faster at going nowhere. The leverage is real, but it's leverage on your existing direction. Make sure your direction is right before you accelerate.

With that framing, let's look at specific applications.

4.2 Drafting Documents

This is where most PMs start, and for good reason. Drafting is time-consuming, and AI dramatically reduces the time to first draft.

PRDs and Specs

When I'm writing a PRD now, I start with one sentence and refuse to move on until it's sharp: What user, what moment, what pain, what outcome.

Once I have that sentence, I can use AI to expand it into structure. Here's a typical workflow:

Step 1: Write the core sentence myself. No AI here. This is strategy work, and I need to own it.

Step 2: Generate a PRD skeleton. I give AI my core sentence and ask for an outline. What sections should this PRD include? What questions should each section answer?

Step 3: Draft each section with AI assistance. For each section, I provide context and ask AI to generate a first pass. I'm specific about format, length, and what to include.

Step 4: Review and revise heavily. The AI draft is a starting point, not an endpoint. I rewrite sections that don't match my intent. I add context AI couldn't know. I verify any factual claims.

Step 5: Share and iterate. The document goes to stakeholders. I use AI to help process their feedback and generate revisions.

The time savings are real—maybe 50-60% reduction in drafting time. But the quality depends entirely on the clarity of my initial thinking. If my one sentence is vague, the PRD is vague. AI amplifies whatever you feed it.

User Stories and Acceptance Criteria

User stories are a great use case for AI because they follow predictable patterns. Given a feature description, generating user stories is largely mechanical.

My approach:

- Describe the feature and its goals
- Ask AI to generate user stories in a specific format
- Ask AI to generate acceptance criteria for each story
- Review for completeness and accuracy

The review step is critical. AI-generated acceptance criteria often miss edge cases. They assume happy paths. They don't know your system's constraints. Use AI to generate the first pass, then apply your judgment to catch what's missing.

Communication Documents

Status updates, stakeholder emails, meeting summaries—these are high-volume, low-complexity documents. Perfect for AI acceleration.

I'll often dump raw notes into AI and ask for a structured summary. Or I'll describe the situation and ask for a draft email. The output usually needs editing for tone and specificity, but it's much faster than starting from blank.

4.3 Research and Analysis

AI changes how you process information. You can handle more inputs, synthesize faster, and identify patterns that would take hours to find manually.

Interview Synthesis

Traditional approach: conduct interviews, transcribe notes, read through everything, identify themes, write a synthesis. This could take days for a substantial research project.

AI-assisted approach: feed transcripts to AI, ask for themes and patterns, generate a first-pass synthesis, then validate against the raw material.

The time savings are dramatic—hours instead of days. But there's a risk: AI synthesis can miss nuance. It picks up what's explicit and frequent. It can miss what's implied, unusual, or emotionally significant.

My rule: AI can summarize customer interviews, but I still check the raw notes when the decision is expensive. For exploratory research, AI synthesis is a great starting point. For decisions with high stakes, I verify against primary sources.

Competitive Analysis

AI can help you structure competitive analysis and fill in publicly available information. It's particularly good at:

- Generating comparison frameworks (what dimensions should we compare?)
- Summarizing competitor positioning from their public materials
- Identifying gaps or patterns across competitors

The limitation: AI doesn't know what competitors announced yesterday. It doesn't have access to pricing that requires a sales call. It doesn't know insider information. Use it for structure and publicly available synthesis, then fill in the gaps with primary research.

Market Research

Similar dynamics. AI can help you:

- Structure your research questions
- Summarize industry reports and articles
- Generate hypotheses to test
- Identify related trends or analogous markets

Again, verification matters. AI-generated market statistics should be checked against primary sources. The model doesn't know which numbers are accurate and which are hallucinated.

4.4 Ideation and Brainstorming

This is one of AI's strongest applications: generating options you wouldn't have thought of.

Feature Brainstorming

When you're stuck, AI can help you explore the solution space. Give it the problem you're trying to solve and ask for ten different approaches. Or twenty. Or fifty.

Most of the ideas will be mediocre or irrelevant. That's fine. You're not looking for AI to solve the problem—you're looking for AI to expand your thinking. One unexpected idea in twenty is worth the two minutes it takes to generate them.

Edge Case Generation

The most common failure mode in product isn't a bad idea. It's a good idea with undefined edges. AI is surprisingly good at finding edge cases.

Give AI a feature description and ask: "What could go wrong? What edge cases should we consider? What assumptions are we making that might not hold?"

The output won't be comprehensive, but it will often surface cases you hadn't considered. Use it as a checklist to review, not a replacement for your own analysis.

Variant Generation

Need multiple versions of something? AI excels here.

- Ten variations of a tagline
- Five ways to phrase an error message
- Three different onboarding flows
- Multiple framings of a pricing page

Generate variants quickly, then apply judgment to select and refine. This is where the cheap artifact principle pays off: when prototypes are cheap, you can explore more options before committing.

4.5 Clustering and Prioritization

When you have a lot of inputs—feedback, feature requests, support tickets—AI can help you organize them.

Feedback Clustering

Dump a batch of customer feedback into AI and ask it to cluster by theme. You'll get a first-pass organization that you can refine.

This works well for:

- NPS comments
- Support ticket themes
- Feature requests
- App store reviews
- Survey responses

The clustering won't be perfect. You'll want to review, merge some categories, split others. But it's much faster than reading through everything and clustering manually.

Prioritization Support

AI can't prioritize for you—prioritization requires judgment about strategy, constraints, and trade-offs that AI doesn't have. But it can support your prioritization process:

- **Scoring frameworks**: Ask AI to apply a scoring rubric to a list of features, based on criteria you define
- **Trade-off analysis**: Describe two options and ask AI to articulate the trade-offs
- **Impact estimation**: Given information about a feature and your user base, ask AI to estimate potential impact

Use these as inputs to your decision, not as the decision itself. AI can help you think more rigorously, but the final call is yours.

4.6 Communication Efficiency

A lot of PM work is communication. AI can make it more efficient.

Meeting Preparation

Before important meetings, I'll often use AI to:

- Generate an agenda based on topics to cover
- Draft talking points for difficult conversations
- Anticipate questions and prepare responses
- Summarize relevant context that attendees need

This isn't about scripting conversations—it's about walking in prepared rather than improvising.

Asynchronous Communication

Written communication is increasingly important in distributed teams. AI helps you:

- Draft clearer, more structured messages
- Adjust tone for different audiences
- Summarize long threads for people joining late
- Generate FAQ responses for common questions

When I'm writing, I try to avoid the temptation to sound smart. I'm not writing to impress; I'm writing to be understood. AI can help with this—ask it to simplify your draft, remove jargon, or make it more direct.

Documentation Maintenance

Documentation rots. Information gets outdated. Nobody has time to update it. AI can help:

- Identify inconsistencies between documents
- Update boilerplate sections across multiple docs
- Generate FAQs from existing documentation
- Convert documentation between formats

It's not glamorous work, but it's work that often doesn't get done. If AI makes it faster, it might actually happen.

4.7 Where AI Gets You in Trouble

Let me be direct about the failure modes.

Generating Without Strategy

If you can generate ten flows in a day, you can also generate ten wrong flows in a day. The constraint shifts from production capacity to judgment.

Don't use AI to generate more. Use AI to generate faster so you have more time to think. If you find yourself producing artifacts without a clear reason, stop. Go back to basics. What user? What moment? What pain? What outcome?

Skipping Verification

AI outputs need to be checked. Every time. The more you rely on AI without verification, the more errors slip through.

Build verification into your workflow. First pass: generate. Second pass: verify. Make this explicit and non-negotiable.

Over-Polishing First Drafts

AI can generate polished-looking outputs quickly. That's dangerous because polished outputs feel finished. They resist revision.

I'd rather have a rough draft that's clearly a draft than a polished draft that gets accepted without critical review. Sometimes I deliberately ask AI for an outline or bullet points rather than prose, to make clear that work remains.

Outsourcing Thinking

The moment you treat AI output as truth, you get sloppy. The moment you treat it as a junior assistant that works fast but hallucinates sometimes, you get value.

AI is not a replacement for your thinking. It's a tool that accelerates certain parts of your work while requiring judgment on others. If you find yourself accepting AI outputs without engaging critically, you're doing it wrong.

4.8 Building Your Personal System

Here's how I think about AI in my workflow:

High-AI Activities

- First drafts of documents
- Summarization and synthesis
- Variant generation
- Feedback clustering
- Communication polishing

Low-AI Activities

- Strategy definition
- Priority decisions
- Stakeholder negotiations

- Final sign-off on anything customer-facing
- Judgment calls with incomplete information

Always-Verify Activities

- Any factual claims
- Numbers and statistics
- Technical specifications
- Quotes or citations
- Customer-facing content

This isn't a rigid framework. It's a starting point. Build your own system based on what works for your context, your risk tolerance, and your strengths.

4.9 The Bottom Line

AI in action is about integration, not replacement. You're adding a capability to your existing workflow, not substituting AI for your judgment.

The gains are real:

- Faster drafting
- Broader brainstorming
- More efficient research
- Quicker communication

The risks are also real:

- Unverified errors
- Strategy drift
- Over-production of artifacts
- Skill atrophy

Navigate between them by staying anchored in fundamentals. Know your strategy. Verify your outputs. Maintain your skills. Use AI for throughput, not for thinking.

When I'm deciding what to cut, I ask one question: what gives us learning? Not what looks impressive. Not what wins the demo. What gives us signal.

That principle applies to AI too. Use it for what gives you learning. Skip it when it just adds noise.

Chapter 5

Data, Feedback, and Learning Loops

PM work has always been about reducing risk: market risk, usability risk, technical risk, execution risk. AI doesn't remove those risks. It just changes how fast you can run the cycle: hypothesis → prototype → feedback → iteration.

This chapter is about building learning loops that actually work. AI makes it faster to test hypotheses—but only if you measure right and move fast. Speed without measurement is just fast guessing.

5.1 The Learning Loop Mindset

Let me start with a reframe that changed how I think about product work.

You're not building features. You're building a learning machine.

Every release is an experiment. Every metric is evidence. Every decision is a hypothesis about what users want and what will move the business. The goal isn't to be right on the first try—it's to learn faster than your competition.

In the AI era, this mindset isn't optional. It's survival. When iteration cycles compress from months to weeks to days, the teams that learn fastest win. The teams that cling to long planning cycles and big-bang releases will find themselves perpetually behind.

The best PMs I know don't look like geniuses because they have answers. They look like geniuses because they don't hide uncertainty. They make assumptions explicit, they create tests that can fail, and they celebrate learning faster than their ego.

That's the job: reduce risk with reality, not confidence.

5.2 Designing Experiments That Teach

Not all experiments are equal. Some generate learning. Some generate activity. You need to know the difference.

The Learning Hierarchy

Think about experiments in terms of what they can teach you:

Level 1: Does anyone care? This is market risk. Does the problem you're solving matter enough for people to engage? Can you get attention?

Level 2: Can they use it? This is usability risk. Once people engage, can they actually accomplish what they came to do?

Level 3: Does it work technically? This is technical risk. Does the solution perform reliably at scale?

Level 4: Does it move the business? This is execution risk. Does solving this problem translate into metrics that matter?

Work your way up the hierarchy. Don't spend months building something technically excellent if you haven't validated that anyone cares. Don't optimize for business metrics if users can't figure out how to use the feature.

Cheap Experiments First

When I'm deciding what to cut, I ask one question: what gives us learning? Not what looks impressive. Not what wins the demo. What gives us signal.

If two features take the same effort, I pick the one that teaches us more about user behavior, even if it's less sexy.

And whenever possible, I run cheap experiments first:

- A landing page before a product
- A Wizard of Oz test before automation
- A survey before a prototype
- A conversation before a survey

AI makes cheap experiments even cheaper. You can generate landing page copy in minutes. You can prototype flows quickly. You can analyze survey responses at scale. Use that speed to run more tests before committing resources.

Falsifiable Hypotheses

A good experiment can fail. If your hypothesis can't be proven wrong, it's not really a hypothesis—it's a hope.

Write hypotheses in this format: "We believe [doing X] will result in [outcome Y] because [reason Z]."

Then define what would prove you wrong: "We'll consider this validated if [metric] moves by [amount] within [timeframe]. If it doesn't, we'll [alternative action]."

This format does two things. First, it forces clarity about what you expect. Second, it creates an exit condition that prevents you from rationalizing failure as success.

5.3 Measuring What Matters

A lot of teams confuse motion with progress. Shipping five things that don't move a metric is not progress—it's activity. Progress is boring: one clear goal, one measurable outcome, one iteration at a time.

If we can't name the metric, we're basically just hoping.

Leading vs. Lagging Indicators

Lagging indicators tell you what already happened: revenue, churn, NPS. They're important, but they change slowly and they don't tell you why.

Leading indicators predict lagging indicators: activation rate, feature adoption, engagement frequency. They move faster and they're more actionable.

Build dashboards that show both. Use leading indicators for weekly decisions. Use lagging indicators for monthly and quarterly strategy reviews.

The Single Most Important Metric

For any given initiative, there should be one metric that matters most. Not three. Not a balanced scorecard. One.

This doesn't mean other metrics don't matter. It means that when trade-offs arise—and they will—you know what to optimize for. Clarity beats completeness.

When I'm writing a PRD, I include the single metric that tells us if we're delusional. If we can't define success in a week or two, we're probably building a story, not a product.

Instrumentation as a First-Class Concern

You can't learn from what you can't measure. Instrumentation isn't an afterthought—it's a requirement.

Before a feature ships, you should know:

- What events will be tracked
- What the success metric is
- How you'll analyze the data
- What decision you'll make based on results

If you can't answer these questions, you're not ready to ship. You're ready to launch and hope.

AI can help here too. Use it to generate instrumentation specs, review event schemas for completeness, and plan analysis approaches. But the thinking about what to measure and why is still yours.

5.4 The Feedback System

Data tells you what's happening. Feedback tells you why.

Quantitative + Qualitative

Numbers without stories are misleading. Stories without numbers are anecdotes. You need both.

When a metric moves, dig into the qualitative data. What are users saying? What are support tickets showing? What patterns emerge from conversations?

When you hear interesting feedback, check the numbers. How common is this experience? Is it a few loud voices or a widespread pattern?

AI accelerates both sides. Summarize support tickets to find themes. Cluster feedback by topic. But don't let the speed of analysis replace the depth of understanding. Sometimes you need to read the actual words, hear the actual conversations.

Continuous Feedback Channels

Don't wait for quarterly research projects. Build continuous feedback channels:

- In-product feedback mechanisms
- Regular user conversations (even informal ones)
- Support ticket review cadence

- Social media and review monitoring
- Sales call insights

The goal is ambient awareness—a constant, low-level understanding of what users experience. Not exhaustive analysis, but enough signal to catch problems early and validate assumptions regularly.

The Feedback Loop Cadence

Here's a rhythm that works for many teams:

Daily: Check key metrics dashboards. Note anomalies. Skim recent support tickets.

Weekly: Review leading indicators. Discuss in team standup. Adjust priorities if needed.

Bi-weekly: Conduct user conversations. Synthesize feedback themes. Update assumptions.

Monthly: Review lagging indicators. Assess experiment results. Make strategic adjustments.

Quarterly: Evaluate overall direction. Kill projects that aren't learning. Double down on what's working.

This isn't rigid. Adjust to your context. But have a cadence. Without structure, feedback processing becomes reactive—you respond to crises but miss patterns.

5.5 Shipping Small

My personal rule: ship small, learn fast, repeat. It sounds like a slogan until you realize it's an operating system.

A tiny release with telemetry beats a big launch with vibes. A bad prototype beats a perfect debate. If you don't have data, you don't have clarity—just louder opinions.

Why Small Works

Small releases are easier to understand. When you ship one change, you know what caused the effect. When you ship ten changes, you're guessing.

Small releases are easier to revert. If something breaks, you can back out quickly. Big releases have complex failure modes and difficult rollbacks.

Small releases are faster to learn from. You get feedback in days, not months. You can iterate while context is fresh.

Small releases reduce risk. Each one is a small bet. You're not betting the quarter on a single launch.

What "Small" Actually Means

Small doesn't mean incomplete or low quality. It means minimal scope at full quality.

Ask: what's the smallest version of this that would teach us something? Not the smallest version we could ship—the smallest version that would generate learning.

Sometimes that's a feature flag for 10% of users. Sometimes that's a simplified flow without edge cases. Sometimes that's a manual process before automation. The key is reducing scope while preserving learning potential.

The Shipping Cadence

Aim to ship something testable every week or two. Not every ship needs to be a major release—many will be experiments, iterations, or improvements.

But maintain momentum. Long gaps between releases create pressure to make each release "count," which leads to bloat, which leads to longer gaps. It's a vicious cycle. Small, frequent releases break the cycle.

5.6 Learning from Failure

If your experiments never fail, you're not learning. You're either testing things you already know or you're not taking enough risk.

Productive Failure

A productive failure teaches you something you needed to know. It validates that a hypothesis was wrong before you invested heavily.

An unproductive failure teaches nothing. Usually because the experiment was poorly designed, the data was incomplete, or the hypothesis was unfalsifiable.

The goal isn't to fail less. It's to fail faster and learn more from each failure.

Post-Mortems Without Blame

When things go wrong, do a post-mortem. But make it about learning, not blame.

Questions to ask:

- What did we expect to happen?
- What actually happened?
- What assumptions were wrong?
- What would we do differently?
- What did we learn that we can apply elsewhere?

Write it down. Share it. Make failure visible and learning transferable.

We keep saying we want innovation, but we punish uncertainty. Real innovation looks messy: half-formed ideas, imperfect data, and decisions made with incomplete information. The difference between a strong team and a loud team is whether they can make uncertainty explicit, run the smallest test, and accept being wrong without turning it into a blame game.

5.7 AI-Accelerated Learning Loops

Let's bring this back to AI specifically.

Faster Hypothesis Generation

AI can help you generate more hypotheses to test. Given a problem space, ask for ten different explanations or approaches. Most will be wrong, but you'll have more options to explore.

Faster Experiment Design

AI can help you design experiments quickly. Given a hypothesis, ask for ways to test it with minimal investment. Ask for potential confounds or biases to watch for.

Faster Data Analysis

AI can help you analyze results faster. Summarize experiment data. Identify patterns in feedback. Generate insights from metrics.

But be careful: AI analysis needs verification like any other AI output. Don't let speed replace accuracy for important decisions.

Faster Iteration

When an experiment suggests changes, AI can help you implement them faster. New copy, updated flows, revised approaches—all can be drafted quickly and refined through iteration.

The cumulative effect: more cycles in the same calendar time. More hypotheses tested. More learning accumulated. More progress toward product-market fit or product improvement.

5.8 Common Mistakes

Let me save you some pain.

Measuring Everything, Learning Nothing

More data isn't better data. If you track fifty metrics, you'll drown in dashboards without developing intuition. Pick the metrics that matter. Ignore the rest until you need them.

Long Feedback Cycles

If it takes three months to learn whether something worked, you're moving too slowly. Find ways to get signal faster. Intermediate metrics, qualitative feedback, proxy measures. Something is better than waiting.

Ignoring Negative Results

Negative results are results. If an experiment fails to move the metric, that's valuable information. Don't rationalize it away or run another variant hoping for different results without changing your hypothesis.

Over-Optimizing for Metrics

Metrics are proxies for user value. If you optimize the proxy at the expense of value, you'll win the metric and lose the user. Stay connected to qualitative feedback. Remember why the metric matters.

5.9 The Bottom Line

The real upgrade isn't AI. It's honesty. Honest assumptions. Honest metrics. Honest postmortems.

AI makes it harder to hide behind process because it exposes how much work was just slow writing and slow coordination. If you adopt AI and keep the old habits—vague goals, unclear ownership, roadmap-as-wish-list—you'll simply ship confusion at higher speed.

If you adopt AI and tighten fundamentals, you'll feel like you gained a superpower.

The fundamentals of learning loops:

- Ship small and often
- Measure what matters
- Write falsifiable hypotheses
- Learn from failures
- Act on what you learn

In the AI era, your advantage is not having the fanciest model—it's having the fastest learning loop. If you can ship a thin slice, measure behavior, and iterate weekly, you'll beat teams that spend three months polishing a plan that nobody validates.

I'm also trying to apply the same product thinking to my own life: measure, don't guess. When I track spending, calories, or time, I'm not judging myself—I'm collecting data. Shame makes you hide. Data makes you design.

The same principle applies to products. Build the measurement system. Collect the data. Let reality guide your decisions.

Chapter 6

Building AI-Powered Products

When prototypes become cheap, strategy becomes more important, not less. Because now it's easier than ever to build something that looks impressive and does nothing.

This chapter is about PMing AI products—how to scope them, integrate them, test them, and ship them. Not as a scientist or an engineer, but as a product manager who's accountable for outcomes.

6.1 The AI Product Reality Check

Let me start with some uncomfortable honesty.

Most AI features don't work as well as the demo. The demo shows the best case. Production shows the distribution of cases—including the confusing, the wrong, and the embarrassing.

Most AI features don't move metrics as much as hoped. Users are impressed by AI... until they're not. The novelty wears off. What matters is whether the feature actually solves a problem better than the alternative.

Most AI features are harder to maintain than expected. Models need updating. Edge cases need handling. Users find ways to break things that nobody anticipated.

None of this means you shouldn't build AI features. It means you should build them with eyes open. The hype is real. The challenges are also real.

6.2 Scoping AI Features

The biggest mistake in AI product development is starting with "let's add AI" instead of "let's solve this problem."

Problem-First, Not Technology-First

Before you build anything with AI, answer these questions:

- What user problem are we solving?
- How are users solving it today?
- Why is that solution inadequate?
- How would AI make it better (not cooler—better)?
- What's the simplest test of whether AI would help?

If you can't answer these clearly, you're not ready to build. You're ready to generate demos that impress stakeholders but disappoint users.

The AI Advantage Test

AI should do one of these things for your feature:

- **Make something possible that wasn't before**: Truly new capabilities that didn't exist
- **Make something faster that was slow**: Automation of tasks that were manual
- **Make something better that was mediocre**: Quality improvement over existing solutions
- **Make something cheaper that was expensive**: Cost reduction for existing capabilities

If your AI feature doesn't clearly fit one of these, question whether AI is the right approach. Sometimes a well-designed form beats a chatbot. Sometimes a simple rule beats a model. Don't use AI because it's impressive. Use it because it's effective.

Scope Creep Warning

AI features tend to expand in scope. "Let's add a simple summarization feature" becomes "let's add a full-featured AI assistant" faster than you'd expect.

Fight this. Scope AI features tightly. Define what the feature does and—critically—what it doesn't do. Set boundaries early and defend them.

The tightest scope that still tests the hypothesis. That's what you're aiming for.

6.3 Designing for AI's Limitations

AI products fail when they're designed for the best case instead of the real case. Good AI product design accounts for limitations.

Design for Uncertainty

AI outputs are probabilistic. They're not always right, and they're not always confident when they should be.

Design implications:

- **Show confidence levels when appropriate**: Let users know when the AI is uncertain
- **Make editing easy**: Users should be able to quickly correct AI outputs
- **Provide alternatives**: Offer multiple options rather than a single "answer"
- **Enable override**: Users should always be able to reject AI suggestions

Design for Failure

AI will fail. Sometimes it will fail badly. Design for this.

- **Graceful degradation**: What happens when the AI is wrong or unavailable?
- **Clear error states**: Help users understand what went wrong and what to do
- **Feedback mechanisms**: Let users report problems easily
- **Human fallback**: Have a path to human assistance for complex failures

The goal isn't perfect AI. It's AI that fails gracefully and recovers well.

Design for Trust Calibration

Users don't know how much to trust AI. Some trust too much (accepting everything without review). Some trust too little (ignoring useful suggestions). Your design should help calibrate trust appropriately.

- **Explain what the AI does**: Help users understand capabilities and limitations
- **Show the AI's work when possible**: Let users see why the AI made a recommendation
- **Build trust gradually**: Start with low-stakes suggestions before high-stakes ones
- **Make verification easy**: Support users who want to check AI outputs

6.4 The Build Process

Building AI products involves some unique considerations in the development process.

Prototype with Real Models

Paper prototypes and wireframes don't capture AI behavior. You need to prototype with actual models to understand what's possible and what the experience really feels like.

Start with off-the-shelf models and APIs. Test the core interaction before investing in custom solutions. You'll learn a lot about what works and what doesn't before writing much code.

Data Requirements

AI features often need data. Sometimes training data. Sometimes context data. Sometimes evaluation data. Understand these requirements early.

Questions to ask:

- What data does the model need to perform well?
- Do we have that data?
- If not, how would we get it?
- What are the privacy and compliance implications?
- How does data quality affect output quality?

Data problems kill AI projects. Surface them early.

Prompt Engineering vs. Fine-Tuning vs. RAG

You have options for customizing AI behavior:

Prompt engineering: Crafting inputs to get better outputs from general models. Fastest to implement, most flexible, but limited in how much you can customize.

Retrieval-Augmented Generation (RAG): Connecting models to your own documents and data. Good for grounding responses in specific information. More setup, but more control over knowledge.

Fine-tuning: Training models on your specific data to improve performance on specific tasks. Most powerful customization, but most expensive and slowest to implement.

Start with prompting. Move to RAG if you need specific knowledge. Consider fine-tuning only when other approaches are insufficient. Don't over-engineer before you've validated the use case.

Evaluation Before Launch

AI features need evaluation frameworks. Not just "does it work?" but "how well does it work across the distribution of real inputs?"

Build evaluation sets:

- Representative inputs that reflect real usage
- Edge cases that stress the system
- Adversarial inputs that might cause problems
- Clear success criteria for each

Run evaluations regularly. Before launch, after changes, on a cadence. This is how you catch regressions and maintain quality.

6.5 Testing AI Products

Testing AI products is different from testing traditional software.

The Testing Spectrum

Unit testing: Does the code work correctly? This part is traditional software testing.

Integration testing: Does the AI integrate correctly with the rest of the system? Also relatively traditional.

Model testing: Does the AI produce good outputs for the inputs it receives? This is where things get interesting.

User testing: Do users actually find the AI helpful? This is where assumptions get challenged.

All four layers matter. Don't skip any.

Model Testing Challenges

AI outputs are non-deterministic. The same input might produce different outputs. This makes testing harder than traditional software.

Strategies:

- Test against evaluation sets with clear pass/fail criteria

- Use metrics like accuracy, relevance, and safety scores
- Compare against baselines (how good is "good"?)
- Include human evaluation for subjective quality

User Testing Priorities

When testing AI features with users, focus on:

- **Trust**: Do users trust the AI appropriately?
- **Utility**: Does the AI actually help them accomplish their goal?
- **Friction**: Is the AI adding or removing friction?
- **Recovery**: Can users recover when the AI is wrong?

Watch for users who over-trust (accepting everything) or under-trust (ignoring everything). Both indicate design problems.

6.6 Ethical Considerations

AI products carry ethical responsibilities that traditional products don't. Take them seriously.

Bias and Fairness

AI models can encode and amplify biases from their training data. This can lead to unfair outcomes for certain users.

Questions to ask:

- Who might be harmed by biased outputs?
- How can we test for bias in our specific use case?
- What mitigation strategies can we implement?
- How do we monitor for bias in production?

This isn't theoretical. Biased AI products have caused real harm. Think about it before you ship.

Transparency

Users have a right to know when they're interacting with AI. In many cases, they also have a right to understand how decisions are being made.

Be transparent about:

- When AI is being used
- What the AI is doing with user data
- How AI recommendations are generated (at an appropriate level)
- Limitations and potential errors

Privacy

AI features often process sensitive data. Ensure you're handling data appropriately:

- What data is being sent to AI models?
- Who has access to that data?
- How long is data retained?
- What are users told about data usage?
- Does data handling comply with relevant regulations?

Privacy mistakes with AI can be severe. Data sent to models may be used for training. Conversations may be logged. Be intentional about what you're sharing.

Harm Prevention

AI can be misused. It can also cause unintended harm through normal use.

Think through:

- How could this feature be misused?
- What harmful outputs could the AI produce?
- What safeguards can prevent misuse or harm?
- What monitoring will detect problems?

Build safety into the design, not as an afterthought.

6.7 Launching and Iterating

AI features benefit from gradual rollout and careful iteration.

Staged Rollout

Don't launch AI features to everyone at once. Stage the rollout:

- Internal testing first
- Beta users who opt in
- Gradual percentage rollout
- Full launch only after validation

Each stage is an opportunity to catch problems before they affect more users.

Monitoring in Production

AI features need ongoing monitoring that traditional features don't:

- **Output quality**: Are outputs meeting quality standards?
- **User feedback**: Are users reporting problems?
- **Usage patterns**: Are users engaging as expected?
- **Failure rates**: How often is the AI failing?
- **Edge cases**: Are unexpected inputs causing problems?

Set up alerts for concerning patterns. Don't wait for users to complain.

Iteration Based on Reality

Once you have real usage data, iterate based on what you learn:

- What are the most common failure modes?
- Where are users getting stuck?
- What features are users not using?
- What are users asking for that you didn't anticipate?

AI features often need significant iteration post-launch. Plan for it.

6.8 The PM's Role

In all of this, what specifically is the PM's job?

Define Success

You own the success criteria. What does this feature need to accomplish to be worth maintaining? Get specific about metrics and thresholds.

Manage Trade-offs

AI development involves constant trade-offs. Quality vs. speed. Capability vs. cost. Safety vs. utility. You're the one who makes these calls and communicates them to stakeholders.

Maintain User Focus

It's easy for AI projects to become technology showcases. Keep asking: is this solving the user's problem? Would users miss this if it went away?

Coordinate Across Functions

AI products touch engineering, data science, design, legal, and potentially more. You're the integrator who keeps everyone aligned on goals and progress.

Communicate Honestly

Stakeholders need to understand what AI can and can't do. Set expectations accurately. Surface problems early. Don't oversell or hide limitations.

6.9 The Bottom Line

Building AI products is product management with extra constraints. You need to understand the technology well enough to make good decisions, but you don't need to build the models yourself.

The fundamentals:

- Start with problems, not technology
- Scope tightly and resist expansion
- Design for AI's limitations
- Test rigorously across the spectrum
- Take ethics seriously
- Launch gradually and iterate based on reality

If you can generate ten flows in a day, you can also generate ten wrong flows in a day. The constraint shifts from production capacity to judgment.

Strategy is the filter that prevents you from shipping noise faster. In AI product development, strategy matters more than ever—because the ability to build impressive-looking features has never been easier, and the ability to build features that actually solve problems hasn't gotten any easier at all.

Chapter 7

Leading with Leverage

Clarity scales. So do bad decisions.

In an AI-accelerated organization, you can move faster than ever. That's power, but it's also responsibility. When iteration cycles compress, the impact of good leadership compounds—and so does the impact of bad leadership.

This chapter is about leading effectively in this environment. How to align teams, debug incentives, and avoid the traps that accelerated organizations fall into.

7.1 Alignment Is a Feature

Sometimes the simplest truth is the one people avoid: alignment is a feature.

If the team isn't aligned, you pay for it in rework, politics, and "why did we build this?" conversations. I'd rather spend thirty minutes writing a crisp problem statement than spend three weeks arguing about a solution that doesn't match the problem.

The Cost of Misalignment

Misalignment is expensive:

- **Rework**: Building the wrong thing means building twice
- **Politics**: Disagreements that should be technical become personal
- **Frustration**: People feel unheard, undervalued, confused
- **Speed loss**: More meetings to resolve what should have been clear
- **Quality degradation**: Compromises that satisfy no one

In an AI-accelerated environment, these costs compound faster. You can build the wrong thing in a week instead of a month. You can have five misaligned conversations in the time it used to take to have one.

Speed amplifies alignment quality. Good alignment gets better results faster. Bad alignment gets bad results faster.

Alignment as a Deliverable

If we're serious about moving faster, we need to stop treating "alignment" as a meeting and start treating it as a deliverable.

A one-page doc with the goal, constraints, trade-offs, and success metric beats three hours of talking in circles. Meetings feel productive because everyone participated. Docs are productive because they force clarity.

What an alignment document looks like:

- **Problem**: One paragraph describing what we're solving and for whom
- **Success metric**: The single number that tells us if we succeeded
- **Constraints**: What we can't change, what we won't do
- **Trade-offs**: What we're explicitly choosing and what we're giving up
- **Open questions**: What we still need to figure out

Write this before the kickoff. Share it before the meeting. Discuss disagreements explicitly. Update it as you learn.

The Crisp Sentence First

I reduce debates by writing the crisp sentence first.

Instead of having an hour-long discussion that ends with "we need to think about this more," I write a proposal: "Here's what I think we should do and why." That gives people something concrete to react to. Disagreements become specific. Agreements become explicit.

You can use AI to draft these sentences faster—but you have to own the content. The clarity must be yours.

7.2 Managing Stakeholders

Stakeholders love certainty, but certainty is expensive and usually fake. I'd rather say, "We're 70% sure this will work; here's how we'll validate in seven days," than pretend we're 100% sure and spend three months finding out we were wrong.

Confidence without validation is just storytelling.

The Roadmap as Hypothesis

Here's how I frame it with stakeholders: the roadmap is a hypothesis, not a promise. Every item on it carries hidden costs—time, complexity, future bugs, and the opportunity cost of not learning sooner.

So if we still want "one more feature" after we price those costs out loud, fine. But we decide like adults: we trade something for it.

This reframe changes conversations. Instead of "can we add this?" the question becomes "what are we willing to give up for this?" Suddenly everyone becomes a realist.

I've seen too many roadmaps become wish-lists. The roadmap should be a bet slip: "We believe X will move Y because Z." If you can't write that sentence, the item doesn't belong on the roadmap yet. It can be an idea, sure. But don't call it a commitment.

Making Trade-offs Visible

My favorite leadership move is simple: make the trade-off visible.

People can disagree on solutions forever, but they get quiet when you turn it into cost. "If we add this, we delay launch by two weeks or we drop QA coverage. Which one do you want?"

This isn't manipulation—it's clarity. Every decision has trade-offs. Making them visible enables better decisions.

AI can help you analyze trade-offs faster—cost projections, timeline impacts, risk assessments. But the communication of those trade-offs is a leadership skill that remains deeply human.

Managing Expectations About AI

Stakeholders have seen the demos. They've read the articles. They think AI can do more than it can—or that it's more dangerous than it is.

Part of your job is calibrating their expectations:

- Explain what AI actually does in your context
- Show examples of both successes and failures
- Set realistic timelines for AI feature development
- Discuss the ongoing maintenance AI features require

Don't let AI become either a magic solution or a bogeyman. Help stakeholders see it as a tool with specific capabilities and limitations.

7.3 Avoiding Hero Mode

I don't like hero mode. Hero mode feels good because it turns stress into identity: "Look how hard I'm working." But it's a trap.

It creates fragile systems. It teaches the team to depend on emergencies. It's not sustainable. And it makes you feel indispensable while making you exhausted.

I'd rather build a cadence that works on a normal Tuesday.

Why Hero Mode Happens

Hero mode usually emerges from one of these patterns:

- **Poor planning**: Not enough buffer for reality
- **Scope creep**: Taking on more than the team can handle
- **Dependency failures**: Waiting until the last minute for things you don't control
- **Communication gaps**: Problems that fester instead of surfacing
- **Identity attachment**: Feeling valuable when you're "saving the day"

AI can accelerate work, but it doesn't fix these patterns. If anything, it makes hero mode more seductive—you can do more, so you take on more, so you work more.

Building Sustainable Cadence

The alternative to hero mode is sustainable cadence:

- **Realistic planning**: Include buffer for surprises
- **Scope protection**: Say no to additions that break the plan

- **Early dependency management**: Surface blockers before they're urgent
- **Regular communication**: Problems are discussed when small
- **Distributed responsibility**: Multiple people can handle critical tasks

I don't want my calendar to be a graveyard of good intentions. So I work in short blocks. Twenty minutes is long enough to make progress and short enough that my brain can't negotiate. Open the doc, write the paragraph, push the commit, send the email. Stop when the timer ends.

Consistency beats hero mode, every time.

Discipline as Kindness

There's a version of me that thinks discipline is aggression—like I have to punish myself into results. That version is wrong.

Discipline is kindness to future-me. It's choosing the boring thing now so I'm not panicking later. Same reason I like contingency plans: not because I expect disaster, but because calm decisions are cheaper than frantic ones.

This applies to team leadership too. Setting boundaries isn't mean—it's responsible. Protecting focus time isn't selfish—it's productive. Saying no to scope creep isn't negative—it's sustainable.

7.4 Building Team Capability

In an AI-accelerated environment, team capability matters more than individual heroics. Your job as a leader is to build a team that can operate effectively without depending on you.

Teaching AI Skills

Not everyone on your team will adopt AI at the same pace. Some will be enthusiastic early adopters. Others will be skeptical. Both reactions are reasonable.

Your job is to:

- Model effective AI use in your own work
- Share prompts and approaches that work
- Create space for experimentation
- Discuss failures openly (AI doesn't always work)
- Set expectations about verification and quality

Don't mandate AI use for everything. Focus on high-leverage applications where the benefit is clear.

Maintaining Human Skills

If the team outsources all their drafting to AI, they'll forget how to draft. If they outsource all their analysis to AI, they'll lose analytical skills.

Protect core capabilities:

- Have people write some things from scratch

- Require understanding, not just generation
- Rotate who does what to prevent over-specialization
- Value the thinking, not just the output

AI is a tool. Tools augment capability; they don't replace it. Make sure your team maintains the skills that matter.

Fostering Good Judgment

The constraint in an AI-accelerated environment shifts from production capacity to judgment. You can generate more artifacts than ever. The question is whether they're the right artifacts.

Build judgment through:

- **Decision reviews**: Look back at decisions and what you learned
- **Explicit trade-off discussions**: Make reasoning visible
- **Experimentation**: Test hypotheses instead of debating them
- **Failure tolerance**: Let people be wrong and learn

7.5 Debugging Incentives

Systems produce what they're incentivized to produce. If your incentives are wrong, your outcomes will be wrong—and AI will help you get wrong outcomes faster.

Common Incentive Problems

Rewarding output over outcome: Teams produce lots of features that don't move metrics.

Punishing failure: Teams avoid experiments that might not work, so they never learn.

Valuing certainty: Teams make confident predictions that turn out wrong instead of honest assessments.

Celebrating heroes: Teams depend on unsustainable individual effort instead of building systems.

Fixing Incentives

Measure outcomes, not outputs: Track metrics that matter, not just releases shipped.

Celebrate learning from failure: Make it safe to be wrong when you learn something.

Reward honest uncertainty: Praise people who say "I don't know" and propose a way to find out.

Recognize sustainable performance: Value consistent delivery over heroic saves.

Incentives and AI

AI creates new incentive challenges:

- Are people rewarded for quantity of AI-assisted output (bad) or quality of decisions (good)?
- Is AI use celebrated regardless of outcome (bad) or evaluated by results (good)?

- Are people penalized for being slower without AI (bad) or evaluated on effectiveness (good)?

Design incentives that encourage effective AI use, not just AI use.

7.6 Communication at Scale

In fast-moving environments, communication becomes even more critical. You need to share information efficiently without creating meeting overload.

Writing Over Meetings

Docs are productive because they force clarity. Meetings feel productive because everyone participated.

Bias toward writing:

- Share updates in written form first
- Use meetings for discussion, not information transfer
- Make documents the source of truth
- Record decisions and rationale

AI can help you write faster—use it. But the information architecture is your responsibility.

The Right Level of Detail

Different audiences need different levels of detail:

- **Executives**: High-level progress, key risks, decisions needed
- **Stakeholders**: Project status, timeline updates, trade-offs
- **Team**: Detailed plans, technical decisions, implementation notes

Tailor communication to the audience. Don't send the same update to everyone.

Transparency About Uncertainty

The best PMs I know don't look like geniuses because they have answers. They look like geniuses because they don't hide uncertainty.

Be transparent about:

- What you know and what you don't
- What's going well and what's at risk
- What you expected and what actually happened
- What you're confident about and what you're guessing

This builds trust. Stakeholders learn they can rely on your assessments because you don't oversell.

7.7 The Personal Side

Some days I feel like I'm juggling too many roles—leader, builder, father, husband, student. The easiest mistake is to treat that as a failure. It's not failure. It's load.

The answer isn't self-hate; it's systems. Small blocks. Clear priorities. Protecting sleep like it's a business asset.

Managing Your Own Energy

You can't lead effectively if you're depleted. Manage your energy:

- Protect time for deep work
- Take breaks before you're exhausted
- Maintain boundaries between work and rest
- Watch for signs of burnout

AI can help you work more efficiently. Use that efficiency to work sustainably, not to work more.

When Things Get Hard

And when I spiral, I try to remember the simplest move: do the next right thing. Not the perfect thing. Not the heroic thing. The next right thing.

A single paragraph. A single email. A single walk. That's how you rebuild momentum without lying to yourself.

7.8 The Bottom Line

Leading with leverage means using the tools available—including AI—while staying anchored in fundamentals:

- Alignment is a feature
- Make trade-offs visible
- Avoid hero mode
- Build team capability
- Debug incentives
- Communicate clearly

Clarity scales. So do bad decisions. In an AI-accelerated world, the leverage you have is enormous—but it amplifies whatever you put into it.

If you're clear about goals, rigorous about trade-offs, and honest about uncertainty, AI helps you move faster toward good outcomes.

If you're confused about goals, sloppy about trade-offs, and overconfident about uncertainty, AI helps you move faster toward bad outcomes.

The choice is yours. The leverage is real. Use it wisely.

Chapter 8

Inspiring Your Team: Navigating Resistance to AI Adoption

There's a gap between announcing AI and having your team actually use it. Smart people resist. Good engineers push back. Designers worry about being replaced. Researchers question the fundamentals. This is not a problem to solve. It's a reality to navigate.

The mistake leaders make: treating adoption as a technology problem. It's not. It's a human problem. Your team isn't resisting AI. They're resisting uncertainty, extra work, disrupted workflows, and the threat to skills they spent years building.

Your job isn't to convince them AI is good. Your job is to make adoption safe, useful, and worth the friction.

8.1 Why Smart People Resist

Start with this: resistance is information.

The Skills Threat

Your engineer spent five years becoming the SQL expert. Now they hear AI can write SQL. Their internal monologue: my expertise is about to be worthless.

This is real. Not irrational. Not Luddite. The skills economy is real. If AI can do the thing you're known for, what happens to your career?

The honest answer: some technical skills will depreciate. Query optimization, boilerplate code, routine debugging. These will matter less.

The other answer: what matters more is judgment. Architecture. Trade-offs. Knowing when automation is wrong. Understanding what the business is trying to do.

Your job is to help them see the second answer. Not by dismissing the first one.

The Workflow Disruption

Your designer has a process. Sketch → prototype → feedback → iterate. Now you're saying: use AI to generate variants faster.

What they hear: everything you optimized is now slow.

What actually happens: the process changes. Faster iteration means different feedback loops. More variants means better prioritization. But they have to learn new tools. New habits. New collaboration patterns.

That's work. That's friction. That's a reason to say no.

The Quality Concern

Your researcher looks at AI outputs and sees hallucinations. Plausible-sounding nonsense. They think: how can I trust this?

They're right to be skeptical. Models do hallucinate. They do make mistakes. They do confidently say wrong things.

The people who adopt AI successfully don't trust it more. They verify differently. They treat output as draft, not final. They have a different QA process.

But that's not obvious from the outside.

The Effort Paradox

Here's the trap: using AI well takes more work upfront.

Prompt engineering. Evaluating outputs. Setting up guardrails. Building verification workflows. Testing edge cases. Documentation.

It's faster once you've done this work. But initially? It's a project.

People see this and say: I could just do it the old way faster.

They're often right.

8.2 The Real Barriers (Not Fear)

Resistance isn't always psychological. Sometimes it's structural.

Tools Don't Exist

You're telling an engineer to use AI but the AI tools cost money. Don't integrate with their IDE. Require API keys they don't have. Don't work offline.

That's not laziness. That's friction.

The Knowledge Gap

Using AI well requires knowing what models can do. What they're bad at. How to structure prompts. How to evaluate outputs. How to handle edge cases.

Your team doesn't have this knowledge yet. Teaching it takes time.

And until they have it, the AI feels like a toy. Not a tool.

Incentives Point Wrong

Your reviews measure individual contribution. AI leverages group thinking. Your sprint velocity metric assumes stable processes. AI adoption disrupts processes. Your culture rewards heroes who stay late. AI adoption needs team discipline.

If incentives point the wrong direction, adoption won't happen. You can't motivate your way past structure.

No Clear Use Case

You're excited about AI in general. But your engineer is thinking: where would I use this in my actual work?

If the answer isn't specific, adoption stalls.

8.3 The Honest Approach to Adoption

This is where most leaders fail: trying to make adoption painless.

You can't. Adoption is change. Change is friction.

Your job is to make the friction worth it.

Start With a Specific Problem

Not "adopt AI." Not "everyone should learn to prompt."

Specific: "We're writing too many tests manually. Let's use AI to generate test cases for boring paths. Goal: 30% faster test coverage. Success metric: bugs caught."

This is concrete. It has a boundary. It has a metric.

Find the Believers First

Don't start with the skeptics. Find the person who's curious. Who's already tinkering. Who sees the potential.

Get them a win. A small one. Then they become proof.

Proof is more powerful than persuasion.

Make It Safe to Fail

People won't experiment if failure is punished.

Frame it: we're learning. We expect mistakes. We're measuring what works.

This is different from "we're shipping perfect AI features."

The honest frame is: we're running experiments. Some will fail. That's data.

Remove Friction Where You Can

If the tool is annoying, adoption won't happen.

Integrate it into workflows. Get licenses. Build templates. Create shortcuts. Remove 10 small frictions and adoption becomes possible.

Acknowledge the Skill Shift

Don't pretend some skills won't become less valuable.

Say it directly: "Writing boilerplate code matters less. Evaluating AI output matters more. Understanding architecture matters more. Here's how we're investing in those skills."

Then invest. Training. Time. Growth opportunities.

Make the transition real, not theoretical.

8.4 The Resistance You Should Listen To

Some resistance is bullshit. Some is wisdom.

Bullshit

- "We should wait for AI to mature." (It won't get easier if you don't practice.)
- "AI will replace all of us." (Some jobs will change. Panic is not a strategy.)
- "We don't need to learn this." (You'll need to, even if just to understand what's coming.)

- "AI is just hype." (It's generating real value for people using it. Hype doesn't change reality.)

These are fears masquerading as reasons. You can't argue with them. You can create proof against them.

Wisdom

- "This will break our QA process." (Probably true. You need a new QA process.)
- "We don't have the infrastructure to do this." (Maybe true. What would we need?)
- "The outputs aren't good enough for our use case." (Maybe true. Is there a different use case where they're good enough?)
- "This will hurt the user experience." (Possible. How would we measure? What would we change?)

Listen to these. They're telling you about real constraints.

8.5 Building Adoption Into Operations

One-time training doesn't work. Adoption is a system.

Dedicate Time

Don't add AI adoption to existing workloads. Give people time. A few hours a week.
"Figure out how AI could help your work. Try something. Report back."
This signals: this is a priority.

Create Feedback Loops

- Weekly: What did you try? What worked? What didn't?
- Monthly: Share what worked. Create templates.
- Quarterly: Measure impact. Celebrate wins.

Feedback loops create momentum.

Build Peer Teaching

The people who figure it out first become teachers. Peer teaching is faster than top-down training.
Give them time to teach. Create forums. Pair new people with experienced people.

Measure the Right Things

Don't measure AI adoption rate. Measure outcomes.

- Time to first draft (is it faster?)
- Quality of outputs (are they better?)
- Team confidence (do people feel effective?)

- Iteration speed (can we go faster?)

Outcomes matter. Adoption rate doesn't.

8.6 The Uncomfortable Truths

Some People Won't Adopt

And that's okay. You need a core group of practitioners. Not everyone.

Your job: make sure the core is big enough. That the laggards aren't blocking the adopters.

Adoption Will Disrupt

Workflows change. Some processes break. Some roles shift.

This is the actual cost. Not hype. Not fear. Real disruption.

Plan for it. Communicate about it. Support people through it.

You Have to Model It

If you're not using AI, your team knows you don't believe in it.

You don't have to be an expert. But you have to be trying.

Use it in your work. Struggle with it publicly. Share what you learned.

This matters more than any speech.

Timing Matters

Adoption takes time. It's not a sprint. It's a shift.

Plan for 6-12 months. Expect plateaus. Expect people to come back to it later and suddenly "get it."

This is normal. Not failure.

8.7 What Success Looks Like

Phase One: Curiosity (Months 1-2)

A few people are tinkering. Others are watching. No real adoption yet.

Success: people are asking questions. Trying tools. Sharing results.

Phase Two: Patterns (Months 3-4)

Specific use cases are emerging. People are figuring out what works.

Success: repeatable wins. People can articulate when and why to use AI.

Phase Three: Integration (Months 5-6)

AI is integrated into some workflows. New hire onboarding includes AI training. Junior people are learning faster because of AI support.

Success: adoption doesn't require conscious effort anymore. It's part of how work gets done.

Phase Four: Sophistication (Months 7+)

People are building on top of AI. Custom tools. Workflows. Using AI to amplify their expertise, not replace it.

Success: you can't imagine your team working without AI anymore.

8.8 Your Leadership Job

Clarity

Be clear about why this matters. Not "AI is cool." Not "we're falling behind."

Specific: "We can iterate 3x faster with AI help. That means better products. That means job security. That means we can try more ideas and learn what users actually want."

Specificity creates buy-in.

Permission

Give people explicit permission to experiment. To fail. To ask questions.

"I don't know how to use this either. Let's figure it out together."

This is more powerful than expertise.

Protection

Protect the early adopters from people making fun of them. Protect them from pressure to deliver perfect results immediately.

Adoption needs space to be messy.

Celebration

When something works, celebrate it loudly.

"Jordan figured out how to use AI to write database migrations 5x faster. Let's learn from Jordan."

Celebration creates momentum.

Honesty

Be honest about what's changing. What skills matter less. What matters more.

Be honest about risks. About failures. About what you're uncertain about.

Honesty builds trust.

Trust enables adoption.

8.9 The Underlying Issue

Most teams resist not because of AI. They resist because of you.

If you've repeatedly asked them to adopt new things and abandoned them halfway through, they'll resist.

If you've punished failure, they'll resist.

If you've not given them time or resources, they'll resist.

If you've blamed them for not moving fast enough, they'll resist.

AI adoption looks like a tech problem. It's actually a leadership problem.

Fix the leadership. Adoption becomes possible.

Adoption without this work is like trying to ship a product without understanding user needs. You can do it. It just won't stick.

8.10 One More Thing

Your team is smart. They see what's coming. They know AI will change their work.

They're watching to see if you're actually serious about this or if it's another management fad that will pass.

Show them you're serious by:

- Investing in tools and training
- Changing how you measure success
- Protecting time for learning
- Modeling the behavior yourself
- Being honest about what's hard
- Celebrating small wins
- Planning for the long term
- Accepting that some people won't come along

Do this and adoption becomes inevitable.

Not because you convinced them AI is great. But because they see that AI helps them do their job better.

And that's the only reason that matters.

Chapter 9

Looking Ahead: Agents, Autonomy, and the Next Interface

The roadmap gets weird from here.

Agents that take actions. Models that see images and hear audio. Systems that reason across multiple steps. Workflows that run without human intervention. Some of this is real. Some is hype. All of it is changing.

This chapter is about preparing for what's coming—without chasing vapor.

9.1 The Pace of Change

Let me be honest about what we don't know.

A year ago, many of today's capabilities didn't exist. A year from now, we'll have capabilities we can't predict. The pace of change in AI is faster than most people can track, and faster than most organizations can adapt.

This creates a challenge for PMs. You need to plan, but plans built on current capabilities will be outdated quickly. You need to make decisions, but the decision landscape keeps shifting.

The solution isn't to stop planning. It's to plan with uncertainty explicitly acknowledged. Build flexibility into roadmaps. Create decision points where you reassess. Don't lock into solutions that assume today's constraints are permanent.

9.2 Understanding Agents

The next major shift is agents—AI systems that don't just generate outputs, but take actions.

What Agents Are

Current AI is mostly reactive. You give it a prompt, it gives you a response. The interaction ends there. You have to take any action yourself.

Agents go further. They can:

- Break down complex goals into subtasks
- Execute actions (browse web, run code, interact with APIs)
- Monitor results and adjust approach
- Work over extended time periods

- Collaborate with other agents

Think of the difference between asking someone for advice and hiring someone to do the work. Current AI is advice. Agents are delegation.

Agent Capabilities Today

In early 2026, agents are real but limited:

- They work well for structured, well-defined tasks
- They struggle with ambiguity and novel situations
- They make mistakes that require human review
- They're expensive relative to simpler AI
- They need careful setup and guardrails

For PMs, this means: agents are tools for specific use cases, not general-purpose replacements for human work. They can automate certain workflows, but they need oversight.

Agent Implications for Product

As agents mature, they'll change how products are used:

- **Users may delegate to agents**: Instead of using your product directly, users might have agents use it on their behalf
- **API-first becomes critical**: Products that can be accessed programmatically will have an advantage
- **UX shifts**: Interfaces optimized for human attention may need to accommodate agent interaction
- **Trust models change**: When an agent acts on behalf of a user, how do you verify authorization?

Start thinking about these shifts now, even if you're not building agent capabilities yourself.

9.3 Multimodal AI

Models are no longer text-only. They see images. They hear audio. They generate visual and audio content. This expansion of modalities changes what's possible.

Current Multimodal Capabilities

What works today:

- **Image understanding**: Models can describe, analyze, and reason about images
- **Image generation**: Create images from text descriptions
- **Audio transcription**: Convert speech to text with high accuracy
- **Audio generation**: Create voice and music from prompts

- **Video understanding**: Analyze video content (emerging capability)
- **Video generation**: Create short video from prompts (emerging capability)

Quality varies. Text remains the strongest modality. But the trajectory is clear.

Multimodal Implications for Product

Consider how multimodal AI affects your product domain:

- Can users interact through voice instead of text?
- Can visual content be generated automatically?
- Can images or screenshots be analyzed as input?
- Can audio/video content be summarized or searchable?

Not every product needs multimodal features. But if your users deal with non-text content, think about how AI can help.

9.4 Reasoning and Planning

Current models are good at generating responses but weaker at multi-step reasoning. That's changing.

The Reasoning Gap

When you ask a model a simple question, it usually does well. When you ask it to work through a complex problem with many steps, it often makes mistakes—especially if the steps depend on each other.

This is the reasoning gap. Models are fluent but not always logical. They produce plausible-sounding nonsense when problems get complex.

Emerging Improvements

Several approaches are improving reasoning:

- **Chain-of-thought prompting**: Asking models to show their reasoning step by step
- **Reasoning models**: Models specifically trained for logical reasoning
- **Tool use**: Letting models use calculators, code executors, and databases to verify their work
- **Multi-step architectures**: Systems that break problems into steps and verify each one

The implication: tasks that currently require heavy human oversight may become more autonomous as reasoning improves. Keep an eye on which tasks are becoming viable for AI assistance.

9.5 What Not to Chase

In a hype-heavy environment, knowing what to ignore is as important as knowing what to adopt.

Vapor

Don't build products around capabilities that don't exist yet. Demo videos aren't products. Research papers aren't production systems. Wait until capabilities are actually available and reliable before building on them.

Solutions Looking for Problems

"We should add AI to this" is not a strategy. AI should solve specific problems better than alternatives. If you can't articulate the problem clearly, don't add AI.

Technical Sophistication for Its Own Sake

The most sophisticated AI solution isn't always the best one. Sometimes a simple rule works better than a complex model. Sometimes a well-designed form beats a chatbot. Choose the right tool for the job, not the most impressive one.

Hype Cycles

Every few months, there's a new "everything changes" announcement. Most of them don't change everything. Filter hype through practical questions: What can I build with this today? What problems does it solve? What are the limitations?

9.6 Preparing Without Overcommitting

How do you prepare for an uncertain future without betting everything on predictions that might be wrong?

Build on Fundamentals

The fundamentals don't change:

- Understand your users deeply
- Define problems clearly
- Test hypotheses quickly
- Measure outcomes rigorously
- Iterate based on evidence

These practices serve you regardless of how AI evolves. Invest in fundamentals, and you'll be prepared for whatever comes.

Maintain Optionality

Where possible, avoid decisions that lock you into specific technical approaches:

- Abstract AI integrations behind clean interfaces
- Build capabilities to swap models or providers
- Design for human-in-the-loop even if you hope to automate later
- Keep data portable and well-structured

Optionality has value. Don't give it up unnecessarily.

Experiment Continuously

Keep running small experiments with new capabilities:

- Set aside time to explore new tools and models
- Prototype applications even if you're not sure they'll ship
- Share learnings across the team
- Document what works and what doesn't

This isn't distraction—it's investment. You're building intuition about what's possible and what's practical.

Watch the Right Signals

Not all AI news matters. Focus on:

- **Capabilities you can use**: What's available in APIs today?
- **Cost reductions**: What's becoming cheap enough for production?
- **Reliability improvements**: What's becoming stable enough to trust?
- **User adoption**: What are users actually using?

Ignore announcements about research breakthroughs until they turn into usable products.

9.7 The Longer View

Let me zoom out for a moment.

AI as Infrastructure

Over time, AI is becoming infrastructure—something you build on rather than something novel. Like databases, like the internet, like mobile devices. Eventually, AI capabilities will be assumed rather than remarkable.

When that happens, the differentiation shifts. Having AI won't matter because everyone has it. What matters is how you use AI to solve specific problems better than alternatives.

This is already happening. "We use AI" is no longer a differentiator. "We solve X problem better because of how we've applied AI" might be.

The Human Role

As AI becomes more capable, the human role evolves. Less execution, more direction. Less production, more judgment. Less processing, more strategy.

This doesn't mean humans become irrelevant. It means human value concentrates in different places:

- Defining what problems to solve
- Setting direction and priorities
- Making judgment calls with incomplete information
- Building relationships and trust

- Ensuring ethical and responsible use

If your value was in execution, you need to expand. If your value was in judgment, you need to amplify.

The Continuous Learning Imperative

The AI landscape changes faster than formal education can keep up. Staying current requires continuous learning:

- Follow developments actively
- Learn by doing, not just reading
- Build networks who share knowledge
- Accept that knowledge has a shorter shelf life

This isn't optional. It's survival. The PMs who stop learning will fall behind. The PMs who keep learning will keep leading.

9.8 What Stays the Same

In all the change, some things remain constant.

Users Still Have Problems

Technology changes; human needs are more stable. Users still want to accomplish goals, solve problems, feel understood. AI changes how you help them. It doesn't change that helping them is the point.

Judgment Still Matters

AI can propose, but it can't decide. The more capable AI becomes, the more important human judgment becomes for steering it in the right direction.

Strategy Still Matters

If you can generate ten flows in a day, you can also generate ten wrong flows in a day. The constraint shifts from production capacity to judgment. Strategy is the filter that prevents you from shipping noise faster.

Ethics Still Matter

As AI becomes more powerful, ethical considerations become more important, not less. How you use these tools, who benefits, who might be harmed—these questions don't go away.

9.9 The Bottom Line

Looking ahead is not about predicting the future. It's about preparing for multiple futures while staying grounded in the present.

The principles that guide preparation:

- Build on fundamentals that don't change

- Maintain optionality where possible
- Experiment continuously with new capabilities
- Watch practical signals, not hype
- Focus on problems, not technology

The roadmap gets weird. Agents, autonomy, multimodal, reasoning improvements—all of it is coming, in some form. Some predictions will be right. Many will be wrong. The exact shape of the future is unknowable.

But the principles for navigating uncertainty are clear. Stay curious. Stay grounded. Keep learning. Keep building.

If your identity is attached to being right, the AI era will humble you, because the marketplace moves too fast for pride. The healthier mindset is: I'm not here to be right; I'm here to reduce risk. My job is to get closer to reality every week.

That mindset serves you regardless of how AI evolves. That's how you prepare without chasing vapor.

Chapter 10

AI Ethics and Legal: Navigating Copyrights, Privacy, Safety, and Compliance

Here's the uncomfortable truth: the AI boom was built partially on unresolved legal questions. Lots of training data came from the internet without explicit permission. Models were trained on copyrighted books, code, images. Artists and authors are suing. Regulators are watching. And PMs need to care because this becomes a product risk, a legal risk, and a reputation risk.

This isn't about being righteous. It's about not building on sand.

10.1 The Copyright Problem That Won't Go Away

Your model was trained on billions of words from the internet. Probably without permission. Definitely without individual consent. This is how models got smart, but it created a legal gray area that's now becoming a legal battlefield.

The question is simple: is training a model on copyrighted work fair use?

The U.S. Copyright Office released guidance in 2024 saying that AI-generated content generally isn't copyrightable because it lacks human authorship. But that doesn't resolve the upstream question: was it fair use to train on copyrighted work in the first place?

Courts are still deciding. The Authors Guild sued OpenAI. Getty Images sued Stability AI. Music publishers are suing. The arguments:

The Fair Use Defense

Proponents argue: training is transformative. You're not copying the book to redistribute it. You're learning patterns that enable different outputs. Like how Google Books indexed copyrighted works and courts upheld that as fair use.

This is reasonable. Training a model on copyrighted text to build a product that serves a different purpose than publishing the original has some conceptual merit.

But courts might disagree. They might say: the market harm to original creators is real. If writers trained models on your work without permission and now your work is less valuable, the copyright holder should have a say.

The Compliance Risk

From a PM perspective, here's what matters:

If your model was trained on copyrighted data without permission, you're exposed to:

1. **Litigation risk**: You could be sued. Even if you win, you'll spend years and millions defending it.
2. **Injunction risk**: A court could order you to stop using the model or retrain it.
3. **License obligation**: You might be forced to license the training data retroactively or pay damages.
4. **Regulatory risk**: Regulators might ban the practice and force retraining.
5. **Reputation risk**: Users might object to models trained on copyrighted work without consent.

What You Can Do

Copyright-Conscious Decisions Option 1: Use data with explicit licenses. Creative Commons licensed content, open-source datasets, data you own or have permission to use. Slower to scale. Legally clean. Defensible.
Option 2: License the data. Pay rights holders. Sometimes feasible, sometimes prohibitively expensive.
Option 3: Use models trained on licensed data. OpenAI, Google, Anthropic all made choices about training data. If you build on their models, you inherit their legal exposure.
Option 4: Acknowledge the risk and plan for mitigation. Document your training data sources. Have a legal strategy. Be transparent about what you're doing.
You likely can't change history. But you can make honest choices going forward.

The frame: copyrighted training data is leverage earned but not yet verified. The verification is coming in court.

10.2 Privacy: Whose Data Is In Your Model?

Models trained on internet data also contain personal information. That data about real people. Conversations. Medical records accidentally exposed. Social media posts. Personal photos.

When you train a model on this data without consent, you're using people's information to build a product. That's privacy work.

The Privacy Risks

Machine learning models can memorize training data. Researchers have shown that with the right prompts, you can extract verbatim training examples from models. Names, addresses, phone numbers, sensitive information.

It's rare. Models are trained on billions of examples. Extracting specific data requires effort. But it's possible. And it matters.

From a PM perspective:

1. **Data minimization**: Collect and train on only the data you need. Less data means less privacy exposure.

2. **Differential privacy**: Add mathematical noise to training so individual data points can't be extracted. Reduces model quality slightly but protects privacy.
3. **Data deletion**: Let users request deletion of their data from training sets. Harder technically but increasingly expected.
4. **Consent**: Be explicit about what data you're using and why.
5. **Transparency**: If someone asks whether their data was in your training set, have an honest answer.

Regulatory Pressure

GDPR in Europe, state privacy laws in the U.S., regulations in Japan and elsewhere are tightening. The trend is clear: regulations assume you need explicit consent to collect and use personal data at scale. That consent is harder to get for training data than for product usage.

If you're building an AI product that uses personal data, you need a privacy legal review. Not later. Now. Because regulators are watching and enforcement is accelerating.

10.3 Safety, Bias, and Harm

Here's the reality: your model will encode the biases in its training data. It will sometimes produce harmful outputs. It will occasionally fail in ways you didn't predict. And you're legally and ethically responsible for reasonably foreseeable harms.

What Counts as Harm

- **Discrimination**: Your model makes decisions that systematically disadvantage protected groups.
- **Misinformation**: Your model confidently generates false information that people believe and act on.
- **Safety issues**: Your model provides advice that causes injury or damages assets.
- **Manipulation**: Your model is designed or optimized to manipulate people.
- **Privacy violation**: Your model leaks or infers sensitive information.
- **Deception**: Your model presents itself as something it's not (human, infallible, etc.).

All of these are possible. Some are likely.

The Due Diligence Framework

Regulators and courts now expect companies to conduct AI impact assessments. Not perfection. Not zero risk. But demonstrable effort to identify and mitigate foreseeable harms.

AI Impact Assessment Questions

1. What populations could be affected by this model?
2. What are the most harmful failure modes?

3. How will you measure if the model is causing documented harms?
4. What safeguards will you implement?
5. How will you handle reports of harm?
6. Who is accountable if things go wrong?
7. Can users opt out or appeal decisions?
8. Is the model explainable enough for users to understand why it made a decision?

This isn't bureaucratic theater. It's the difference between "we deployed an AI system" and "we deployed an AI system and did the work to make it safer."

Bias Is Not a Binary

You will ship a biased model. You will make decisions that disproportionately affect some groups. The goal isn't zero bias. The goal is:

1. **Measurement**: Actually measure bias across demographic groups. Don't assume fairness.
2. **Transparency**: Tell users where bias might exist and how the model is used.
3. **Intentionality**: Make explicit choices about trade-offs. If accuracy for one group is worse, say so and justify it.
4. **Recourse**: Give users a way to challenge or appeal decisions.
5. **Iteration**: Bias doesn't get solved once. It's an ongoing practice.

The dangerous approach: deploy, assume you're fair, move on. The regulatory approach: document your bias testing, acknowledge limitations, update continuously.

10.4 Compliance: The Regulatory Landscape is Moving

The regulatory environment is fragmenting. EU AI Act. China regulations. U.S. executive orders. State laws. Industry-specific rules (healthcare, finance, criminal justice). It's a mess.

From a PM perspective, the mess has a pattern:

The EU AI Act

Highest-risk systems (criminal justice, hiring, lending decisions) require pre-market approval, transparency, and monitoring. Medium-risk systems need documentation and user notification. Low-risk systems have minimal requirements.

Your system is probably medium or high-risk if it affects people's rights or safety.

U.S. Approach

No single federal law yet. Instead: executive orders, sectoral guidance (healthcare, finance), state laws. The pattern is moving toward:

- Disclosure when AI is used to make decisions about you

- Right to explanation and appeal
- Bias testing and documentation
- Prohibition on certain uses (predictive policing, certain hiring systems)

What This Means for PMs

Compliance Reality Check 1. You need to know where your system is used and whether it's regulated. 2. You need to document your training data, testing, and known limitations. 3. You need to build explainability and user recourse into the product, not as an afterthought. 4. You need to have privacy impact assessments and bias testing. 5. You need legal and compliance input early, not in the last sprint.

The cost isn't theoretical. It's:

- Engineering effort to build explainability and auditability
- Legal review cycles that slow deployment
- Compliance infrastructure to track and report on model performance
- Potential product limitations (some uses might not be allowed)
- Reputational risk if you're on the wrong side of a regulatory decision

This is the trade-off nobody wants to acknowledge: governance costs time and resources. The alternative is deploying without governance and hoping nothing goes wrong.

10.5 What Uncomfortable Honesty Looks Like

The most defensible position is the one you can explain to a reporter, a regulator, or a victim:

We built this model knowing it would have these limitations. We tested it in these ways. We found bias in these areas. We chose to mitigate X and accept risk on Y. Here's who it affects and how they can appeal.

This is uncomfortable. It means admitting your model isn't perfect. It means accepting that someone will be hurt by it even though you tried to minimize harm.

But it's the position that survives litigation and regulatory scrutiny.

The Honest Checklist

Before shipping an AI product:

1. **Training data**: Can you defend where it came from?
2. **Copyright**: Are you using copyrighted material? Do you have a legal position?
3. **Privacy**: Did you get consent? Can you explain to users what data is used?
4. **Bias**: Have you measured it? Can you explain what you found?
5. **Safety**: What's the worst thing this could do? Have you tested for it?
6. **Explainability**: Can users understand why the model made a decision?

7. **Recourse**: If someone is harmed, what's their remedy?
8. **Monitoring**: How will you know if things are going wrong?
9. **Governance**: Who's accountable if there's a problem?
10. **Transparency**: Would you be comfortable publishing your answers to these questions?

You don't need perfect answers. You need honest ones.

10.6 The Real Cost of Cutting Corners

I've seen teams ship without considering these questions. They move fast. They ship. Users complain. Regulators notice. Lawsuits happen. The company spends millions defending something that could have cost a fraction of that to do right.

The calculation isn't hard:

- **Cost of governance**: $500K-$2M to build compliance infrastructure, train teams, do impact assessments.
- **Cost of litigation**: $5M-$50M+ if sued and it goes to trial.
- **Cost of a ban**: Infinite if regulators block your product.
- **Cost of brand damage**: Hard to quantify but real.

Governance isn't nice-to-have. It's risk management.

10.7 Your Job as PM

You're not the lawyer. You're not the ethicist. But you're the person who decides what gets built.

That means:

Ask Hard Questions

- Where did the training data come from?
- Do we have the right to use it?
- Who could be harmed by this?
- How would we know?
- What would we do about it?
- Can we explain our decisions to regulators?

Build Guardrails Into the Product

- Explainability: Users should understand why the model made a decision.
- Uncertainty: The model should indicate when it's unsure.
- Recourse: Users should be able to appeal or get a human review.
- Logging: You should track decisions for audit and accountability.
- Kill switches: You should be able to disable the model if it's behaving badly.

Make the Trade-offs Visible

- Acknowledge what you're accepting and what you're mitigating.
- Document decisions in writing.
- Be transparent with teams about limitations.
- Update your assessment as you learn.

Assume You'll Be Audited

Write your docs as if regulators will read them. They probably will. If you can't defend a decision to a regulator, it's a bad decision.

10.8 Looking Forward

The legal and ethical landscape around AI will crystallize. Courts will rule on copyright. Regulators will enforce compliance. Victims will demand accountability. Standards will emerge.

What won't change: the tension between speed and safety, between innovation and responsibility, between what's technically possible and what's ethically defensible.

Your job is to navigate that tension honestly. Not to solve it perfectly. But to acknowledge it, measure it, and make decisions you can defend.

That's what governance looks like in the AI era. Not permission. Not restriction. Just honesty about what you're building, who it affects, and what could go wrong.

Conclusion

AI won't save bad fundamentals. But paired with sharp thinking, it gives you superpowers.

That's the sentence I want you to remember from this book. Not because it's clever, but because it's true—and because forgetting it will cost you.

The Real Upgrade

The real upgrade isn't AI. It's honesty.

Honest assumptions. Honest metrics. Honest post-mortems.

AI makes it harder to hide behind process because it exposes how much work was just slow writing and slow coordination. If you adopt AI and keep the old habits—vague goals, unclear ownership, roadmap-as-wish-list—you'll simply ship confusion at higher speed.

If you adopt AI and tighten fundamentals, you'll feel like you gained a superpower.

I've watched this play out across teams. The ones that struggle with AI adoption aren't struggling because of the technology. They're struggling because AI exposes weaknesses they'd been living with. Unclear strategy becomes painfully obvious when you can generate artifacts in minutes. Poor alignment becomes expensive when iteration cycles compress. Weak measurement becomes unforgivable when you can run experiments every week.

The teams that thrive are the ones that use AI as an opportunity to get rigorous. They tighten their definitions. They clarify their metrics. They say the uncomfortable truth out loud. And then AI amplifies their clarity.

What This Book Tried to Do

I wrote this book to give you a practical guide for the shift. Not hype. Not theory. Practical guidance for PMs who build things.

Here's what we covered:

The landscape changed. AI shifted the interface to intelligence from specialists to everyone. That compressed iteration cycles, raised expectations, and changed what "good" looks like for product teams.

Fundamentals matter more than ever. AI can propose, but it can't decide. Models generate plausible outputs, not true outputs. Leverage is earned through verification. The technology is powerful, but it requires judgment to use well.

Workflow integration is where the value is. AI accelerates drafting, research, synthesis, and brainstorming. But it doesn't fix unclear strategy or replace human judgment. Use it for throughput, not for thinking.

Learning loops are the core competency. Ship small, measure behavior, iterate weekly. That's the operating system. AI makes loops faster. But only if you have the instrumentation, the hypotheses, and the honesty to learn from results.

Building AI products requires discipline. When prototypes are cheap, strategy becomes more important. Scope tightly. Design for limitations. Test rigorously. Ship responsibly.

Leadership adapts but doesn't change. Alignment is still a feature. Trade-offs still need to be visible. Hero mode is still a trap. The fundamentals of good leadership remain—they just operate at higher velocity.

The future is uncertain but navigable. Agents, multimodal AI, reasoning improvements—they're coming, in some form. Prepare by building on fundamentals, maintaining optionality, and experimenting continuously.

The Uncomfortable Truth

I don't think the hard part is "AI." The hard part is admitting we've been shipping opinions dressed up as requirements.

The hard part is acknowledging that a lot of what we called "work" was actually slow writing and slow coordination that AI exposes as unnecessary.

The hard part is accepting that speed without judgment creates faster failures, and that judgment is a skill that requires cultivation.

The hard part is looking honestly at what we're building and asking whether it actually solves a problem—or whether we're just shipping noise faster.

AI is a mirror. It shows you what you really are. Teams with good fundamentals look great in the mirror. Teams with bad fundamentals look worse than they did before.

Which kind of team do you want to be?

Your Competitive Edge

In the AI era, your advantage is not having the fanciest model—it's having the fastest learning loop.

If you can ship a thin slice, measure behavior, and iterate weekly, you'll beat teams that spend three months polishing a plan that nobody validates.

This isn't about working harder. It's about restructuring your workflow. Smaller releases. Better instrumentation. Tighter decision logs. A bias toward experiments that can fail cheaply.

The teams that win will be the ones that learn fastest. AI makes learning faster—but only if you're set up to capture the learning. Otherwise, you're just moving faster without getting smarter.

A Personal Note

I wrote this book because I believe product management is one of the most important jobs in technology. We're the people who decide what gets built and why. We translate between what users need and what technology can deliver. We make the calls that shape products and, through products, shape how people live and work.

That responsibility doesn't get smaller when AI arrives. It gets bigger.

Because now we can build faster. Now we can experiment more. Now we can reach more people. The leverage is enormous. And leverage amplifies whatever direction you're pointing it.

If you're pointing at real problems, AI helps you solve them faster. If you're pointing at noise, AI helps you generate more noise. The choice is yours.

There's also a personal side to this. If your identity is attached to being right, the AI era will humble you, because the marketplace moves too fast for pride. The healthier mindset is: I'm not here to be right; I'm here to reduce risk. My job is to get closer to reality every week.

That's what I'm trying to do. That's what I hope this book helps you do.

What Comes Next

I don't know exactly what comes next. Nobody does. The models will improve. New capabilities will emerge. Some predictions will prove right; others wrong.

But I know this: the principles in this book will serve you regardless of how AI evolves.

Clear thinking beats fuzzy thinking—always.

Honest measurement beats hopeful guessing—always.

Learning from reality beats defending assumptions—always.

Alignment is a feature—always.

Leverage is earned through verification—always.

These aren't AI principles. They're product principles. They're leadership principles. They're principles for working effectively in a world of uncertainty.

AI just makes them more important.

The Job

Let me end where I started, with a statement of what the job is:

Ship thin slices, learn fast, lead with clarity.

That's product management in the AI era. That's what the best PMs I know do. That's what I'm trying to do, imperfectly, every week.

The tools are more powerful than ever. The pace is faster than ever. The opportunity is greater than ever.

The fundamentals haven't changed. They've just become more consequential.

Learn fast. Decide like an adult. Lead the shift.

That's the job.