

THE ULTIMATE GUIDE TO GETTING A PM JOB

A NO-BS GUIDE TO GETTING YOUR
FIRST, OR YOUR NEXT, PRODUCT
MANAGEMENT GIG

AAKASH GUPTA

The Ultimate Guide to Getting a PM Job

Copyright © Aakash Gupta 2023

This book is only available for purchasers.

Electronic distribution is strictly prohibited.

Table of Contents

[Introduction](#)

[SECTION 1: The Tactics](#)

[Chapter 1: How to Get Interviews](#)

[Chapter 2: How to Succeed in Interviews](#)

[Chapter 3: How to Negotiate the Offer](#)

[Chapter 4: Special Case – No PM Experience](#)

[SECTION 2: The Learning Material](#)

[Chapter 5: Key PM Skills At Each Level](#)

[Chapter 6: The History of PM](#)

[Chapter 7: The Languages of PM](#)

[SECTION 3: The Advanced Skills](#)

[Chapter 8: Advanced Roadmapping](#)

[Chapter 9: Why OKRs Work](#)

[Chapter 10: How to Impact Size Features](#)

[Chapter 11: When to A/B Test](#)

[Chapter 12: How to Deal With Tech Debt](#)

[Epilogue](#)

[References](#)

[Acknowledgments](#)

Introduction

Getting a Product Management (PM) job is tough. Whether you have experience or not.

And in today's market, it's tougher than ever. With thousands of laid off big tech PMs on the market, the supply and demand dynamics are brutal.

Experienced PMs are finding their recruiting funnels turned upside down. One Director of Product I worked with for this book had 3 offers the last time he was recruiting, in 2018. He hadn't even applied to more than a handful of jobs.

When he reached out to me, he had applied to over 110 jobs, had over 15 interviews, and still hadn't received an offer. This gentleman's situation is not uncommon in the new job market of Product Management.

The world of 3 simultaneous offers was a zero percent interest rate phenomena (ZIRP). Drunk off topline growth, tech companies grew without abandon. That market is totally gone now.

But that doesn't mean you can't go get an upgrade in your pay or scope by finding another job. I'm happy to report we secured that Director of PM a job which pays 15% better than his last one, at a company whose mission he more closely aligns with.

I myself got a VP of Product job in this market, an upgrade in title and scope. So, I speak from experience: You can do it.

More importantly, you need to do it. Here's why.

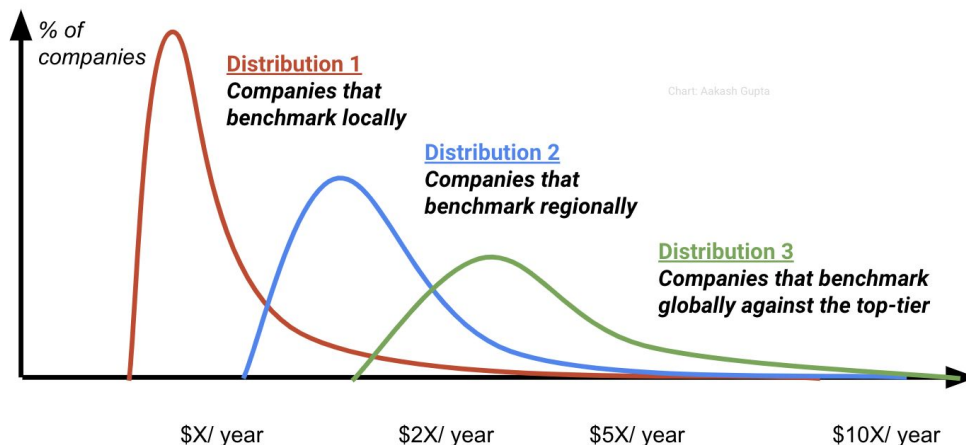
The Trimodal Distribution of PM Compensation

Many people don't realize quite how much more they could be making.

The overall distribution earnings in Product Management is Trimodal. That's because it's actually made up of three separate distributions:

The Trimodal Distribution of Product Management Total Earnings

Sources: Gergely Orosz, Nikhyl Singhal, Levels.fyi, H1B Database



Here's how it works.

1. **Most companies benchmark locally.** These are typically your small to medium businesses. They usually look at the employers in their city or state. For instance, where I live in Research Triangle Park, NC, most startups will pay \$70K for a Product Manager. That Product Manager actually feels well paid, too. That's the most common pay in the whole distribution.
2. **Then there are some companies that benchmark regionally.** These are typically your large businesses that still tend to hire from the region. They have a wide band of pay, because they do hire globally sometimes. But, normally

they'll pay a Product Manager something like \$120K in my area. That Product Manager usually feels rich! They make almost twice their friends.

3. But there are also companies that benchmark globally against the top-tier. These are your >\$100B market cap companies, as well as other large companies that want to hire out of those behemoths. Both Google and Apple pay Product Managers in my area \$550K! That's fully 8x more than the startups - where, mind you, no one was unhappy.

This phenomenon is global and exists everywhere. In India, there are Product Managers working for venture-backed startups in Bangalore for ₹ 7.5 Lakhs per annum. Across town, similarly tenured PMs working for Google earn more than ₹ 78 Lakhs per annum. That's more than a 10x earnings gap.

PM is one of the few careers in the world where you can earn substantially more just by upgrading where you work. Why? Because, you're not going to double, triple, or 10x your salary within your current company overnight.

To do that, you need to have the skill of getting PM jobs. Even in this market. And that's where this book comes in.

You Don't Need PM Experience

Moreover, you can break into this lucrative career without any experience. In this market, where every job seems to ask for three years minimum of PM experience, it's hard to believe.

But people are making it happen. Some are doing internal transfers. Others are breaking into entry level roles.

Many of them are changing their lives. For instance, a customer support person in Texas I worked with for this book went from making \$35K a year to \$100K. That's life-changing money for her and her two kids.

It all comes down to using the tools in this book.

How to Use this Book

You don't need to read this book front to back. It's more like reference material than a single narrative. Here's a couple study paths:

1. No PM experience
2. For PMs with some experience
3. For PMs or product leaders with significant experience

1. No PM Experience

If you have no Product Management experience, start with Chapter 4. Chapter 4 because it introduces how to become a PM without experience. Then read the book straight through.

2: For PMs with some experience

Read the book straight through. But you can skip Chapter 4. You'll love the history of Product Management, Chapter 6, but may find Chapters 5 and 7 a bit junior. Make the call yourself. Section 3 will be perfect for you.

3: For PMs or product leaders with significant experience

You are going to get the most value out of Chapters 1-3. Section 2's History of Product Management, Chapter 6, will be fun for you. In Section 3, pick and choose the areas that you may want to refresh on if you haven't used them in a while.

This vs Other Books

What's the difference between this book and the several other good books that exist out there to get a PM job (like *Decode and Conquer* and *Cracking the PM Interview*)?

1. **Those books are primarily targeted at early-career PMs:** This book is packed with information for people who want to succeed in interviews as *tenured* product professionals.
2. **Those books are outdated:** The best-selling books were written many years ago. The PM market was very different back then. Plus, now that so many PMs have read those books, the alpha from their tips has evaporated.
3. **Those books are focused on the interview step:** The interview step isn't even the hardest these days. It's actually *getting* an interview. This book actually helps you with that step, as well as negotiating the offer after.

Without further ado, let's get into it!

SECTION 1: The Tactics

Chapter 1: How to Get Interviews

The market is absolutely brutal for job seekers. The number of applicants per job is enormous.

I was chatting with one highly tenured product manager who had this to say about his experience job searching:

200 apps, 4 interviews, and only one full loop completed so far.

This is a professional who, when he was job searching 2 years ago, submitted 40 apps, had 20 interviews, 10 final rounds, and 5 offers. His entire funnel has been turned upside down.

- Apply to interview rate has gone from 50% → 2%
- Initial interview to final round rate has gone from 50% → 25%
- Final round to offer rate has gone from 50% → 0%

My friend is not the only person who is feeling this way. A Facebook staff engineer went viral because he couldn't make the mortgage payments on his \$1.5M Seattle home.

Laid Off And Can't Pay Mortgage

Hi all,

Last year I bought a 1.5m home in the Queen Anne area in Seattle. My monthly payment is around 8500. The unexpected happened a few months ago and I was laid off from my job at Meta where I was a staff level engineer. I really need help because I depleted most of my cash in the down payment for my house and I can't afford the monthly anymore without a job. I've applied to more than 50 companies and I've put out more than 500 applications but I can't get an offer for the life of me.

Next month will be the first month I won't be able to make my mortgage payment and I'm terrified. I've thought about selling my house and moving in with friends but I've lost so much money in house value I can't justify it.

Please help me.

Now Deleted Reddit Post from Meta Staff Engineer.

He's had the same rough go of it as my friend. It's not clear exactly what the numbers are, but it sounds like he's gotten a 10% interview rate (50 companies/ 500 applications).

A few years ago, ex-Meta folks were some of the most sought-after people in tech. Simply getting a job at Meta meant you had a fantastic resume.

Now, they can't even get interviews. So, it's clear that it's not just my friend, but it's the market. There's been an enormous shift in the past few years. What's driving it?

Employer Side: Supply is Down

On the employer side, the rise in rates has shifted investor's barometer from revenue to profit - and this has compressed jobs. Most companies and teams that had seen new headcount year after year just haven't gotten it this year, as they focus on profitability. This includes the vast majority of FAANG, which was hoovering up tech talent over the past few years.

When I Tweeted in September 2022, there were 426,000 jobs worldwide open on LinkedIn for the search product manager. As of May 2023, there are only 290,000. **The market for open jobs has decreased by >31%**. All FAANG is barely hiring, and all the startups who can't live up to their 2021 valuations aren't hiring.

So, all that's left are companies that are doing well.

Employee Side: Demand is Up

On the employee side, the massive layoffs in tech have turned scores of tech workers into job seekers. For years, the market had not seen an influx like this. Suddenly, hundreds of thousands of workers hit the market.

The market turned in September 2022. That's when Twilio and DocuSign set off a wave like the tech industry hasn't seen since 2001. Microsoft, Meta, and Salesforce all followed with RIFs (reductions in force). Like me, you may have forgotten that the wave even reached Google!

These are some of the most desirable employers in the world. They used to have their pick of the best talent. So, the talent that is in the market right now isn't just a lot of people, **it's a lot of really high-quality people.**

The result is basic supply and demand. With supply down and demand up, the qualified applicants per job has gone up dramatically.

You Need a Strategy to Compete in this Market

Back to the product manager who wasn't getting interviewed. After he messaged me, I knew I had to spill to him the formula for how to get interviews.

The results are almost too good to be true. He applied to 8 jobs, and he got interviews at all 8 of them. Now, he's spending all day preparing for and taking interviews, instead of applying.

So, I'm here to share the exact steps that I shared with my friend **to get you interviews**.

Here's the secret to my strategy: **you're not going to apply for a single job**. We're going to follow a six-step process to get you into the interview stage without a single online application. Those are a waste of time in this market.

- Step 1: Adjust LinkedIn for Candidate-Market Fit
- Step 2: Find a Target Job
- Step 3: Make First Reach-Out
- Step 4: Crush the Call
- Step 5: Customize the Resume
- Step 6: Follow Up Frequently

The mental model here is **while everyone else is dropping online resumes, you're going to do enterprise sales**.

Step 1: Adjust LinkedIn for Candidate-Market Fit

Find Candidate-Market Fit

Phyl Terry of *Never Search Alone* has this wonderful concept of “candidate-market fit.” It’s so good that it’s going to enter the general lexicon and lose the quotes soon. The basic premise is this: you need to figure out the profile of job that fits your experience and what the market wants.

If you were job searching from 2011-2021, you could reach for jobs slightly higher in seniority. If you were a senior product manager, you might try for a lead product manager role. That strategy worked fine then.

But, in today’s market, that approach doesn’t work. You need to be really targeted and specific about where your candidate-market fit is. The key is to be really realistic: what size company, in what industry, and at what seniority do I have great fit?

My friend had previously defined it as a public company in any industry at the principal product manager level. We had to dramatically finesse that. Here’s what we changed it to:

- Principal or Staff Product Manager at a late-stage company; Director of Product at an early-stage venture-backed company
- Either for a growth team or a Social Media company

This at once expanded and narrowed his aperture. It **expanded** his aperture beyond public companies. They pay the best, but they’re not hiring nearly as much. Just by expanding to late stage venture, he saw a nice uptick. We also added a more senior role at an earlier-stage venture-backed company. Those companies need to grow to

survive and are still hiring. These two criteria added a bunch of companies to the mix who actually needed his services.

But then it also **narrowed** his aperture. He only had experience with growth teams and the Social Media industry. So, we focused only on either growth roles or roles in Social Media. See, in up markets, employers are willing to hire someone without some expertise. But, in down markets, companies double down on some sort of expertise within PM more broadly.

Adjust With An Advisor

You should leverage your trusted advisors to tweak your own profile here. You want to narrow it into places that matter and expand it in places you hadn't thought of. The goal is to be able to identify jobs that you **are** qualified for.

If you're a student, that means trimming out jobs requiring experience. It's tough advice, I know. I had given the exact opposite direction in another market a year ago. But we live in tough times.

The one thing I want to draw your awareness to is: **do you tend to underestimate yourself?** Studies show people from disadvantaged backgrounds and minority groups tend to. So, be aware of your tendency. If you overestimate yourself, downwards adjust. If you underestimate yourself, adjust upwards.

The biggest thing you can do to help yourself here is sanity check everything with advisors: girlfriend, dad, best friend, job search council, whomever, or all. We have a tendency to be able to analyze other's situations better than our own.

Tailor Your LinkedIn

I'm assuming that you already have the basics that come in the free guides you see all over: tons of keyword packed detail for each job with numbers (that you can share), a clear narrative in your about me tying everything together, etc.

Now, you need to start becoming your own biggest skeptic. What are the areas, in your candidate-market fit, where an employer would be skeptical? For my friend, he had always worked at public companies and now we were throwing startups into his mix. So, the area employers were going to be most skeptical is startup interest.

Dig into your background to figure out real ways to address that. For my friend, it meant highlighting how he was actually part of a “startup within the larger company.” He also advised several startups. He added those positions listed separately and up until the present so they were first. This made him look like a startup guy.

You need to analyze yourself in the same way. What are your most significant faults? Your LinkedIn should really be completely revamped to make the answers to those questions flip from “maybe not a fit” to “emphatically a fit.”

Finally, you should fork out the money for LinkedIn Premium and set your profile to open. This makes it free for recruiters to send you a message if they like what they see. Respond to every InMail immediately even if you’re not interested, because recruiter dashboards display your responsiveness.

Step 2: Find a Target Job

Create a Company List

You want to work somewhere great. Be rigorous about defining your top 10 companies you want to work at. Create a list of those companies. Follow them on LinkedIn. This will help the algorithm surface their jobs first.

Smartly Define LinkedIn Alerts

In LinkedIn, you should click the “open to” button and set really smart categories. You should set “worldwide remote” for your target seniority level (for my friend we did lead product manager). Enable separately for your work location as well. Then you should also set your other options (for my friend, director of product). You have 5 slots. Use all of them.

This will create a nice set of emails that LinkedIn will send you every single day (five of them) when there are jobs that meet your criteria. This will get you a whole bunch of opportunities to consider.

Find a Niche Board As Well

LinkedIn is actually an expensive place for employers to be. So you don't just want to track LinkedIn.

Find another job board for your niche that has some good jobs and set up alerting there as well. It might be ZipRecruiter, Indeed, Lenny's - whatever. Do some investigation and pick one. Don't get too worried about your choice. You should be getting plenty of opportunities if you follow all of the above steps.

Prioritize the #1 Job

As you get all the opportunities every day, you should stack rank them. First, use candidate-market fit: you want to prioritize a job that

you really want and are very well qualified for. Because we're about to go do a whole bunch of work to get an interview for that job.

Also, prioritize recency. If a job has been up for a while, the pipeline will be much harder to break into versus if it was just opened up. Once you have your single top job of the day, take the next step.

Re-Customize Your LinkedIn

As necessary you should prepare your profile for the single job you have in mind. It should really sing that you are the perfect candidate for the job. The reason is, we're about to use LinkedIn to reach out.

Step 3: Make First Reach-Out

Rank Potential Reach Out Candidates

Now that your LinkedIn is singing for this job, you are ready to reach out to people. Do you have a connection at the company? If so, they are the perfect person to refer you. They are going to always rank #1, because they're likely to help you.

If not, search for keywords about your job on LinkedIn. Then use the **company filter** and set it to the company. And use the **result type filter** and set it to Posts. This will get you any posts from hiring managers or recruiters about the job. If there's a post from the hiring manager or recruiter for the role, that's the second most ideal person you can reach out to.

But, most of the time, we're not so lucky. In that case, you have two options:

1. Send a free InMail to someone with LinkedIn Premium and an open profile
2. Send a personalized connection invite

You need to weigh the options against the type of people you find about the company. Depending on your role, be strategic. For my friend, we decided that would be Directors of Engineering. He had a lot of experience and they were likely to be intrigued by his profile, since they hire lots of Engineering Managers.

So, he looked for a Director of Engineering that had LinkedIn Premium and an open profile. LinkedIn shows you if they have LinkedIn premium with a Gold In icon if you use search. To find out if they have an open profile, you have to hit Message and it will display in the bottom right-hand corner.

This usually gets you someone. If you can't even find that target role, then go to the next most interesting role until you do.

Crush the Cold Note

Now, you're ready to write a note to the person. You need to personalize each and every cold note. Here are the four things you'll want to do:

1. Try to pull together some detail that ties you both to demonstrate you researched them.
2. Qualify yourself for the position and name it.
3. Ask them for a referral and say you would be up for a call.
4. Include a short link to the job you are interested in.

If you are sending a personalized invite, you really only have 3-4 sentences. I've actually found that's still the right length for InMails as well. Be really concise.

Here's an example *[italicized brackets are notes]*:

Hi Tin-Yun,

It's so cool to see Business Insider recognize Kumo.Ai. *[We found this from his posts section.]*

I saw that you are hiring a Director of Engineering for FinTech Platforms. I think I would be the perfect fit because I have spent 6+ years doing that work at places like X and Y. *[We qualify for ourselves.]*

Can I earn your referral? I'd love to hop on a chat. *[We make the ask.]*

Thanks,

Name

Link *[We Link the Job.]*

A tip for the link: everything after a ? mark can be deleted. LinkedIn always adds a bunch of tracking you don't need. Just test the shortened link before sending it.

Go Through the Sequence

If you don't get a response in 24 hours, message your second-ranked person. And so on and so forth for up to 14 days.

You are bound to get someone to respond if you did a good job with your cold note and profile.

The only complication is, as these things drag on, customizing your LinkedIn profile gets harder and harder. So don't go on too long. My friend usually found results within his first two days.

Step 4: Rock the Call

Massively Prepare

About 50-70% of the time, they will want to have a call before referring you. They want to judge your vibe, make sure you are who you say you are, etc.

You should totally flip that on its head and be massively prepared for the call. Study the role, study the product, study the company's situation. Use the call to convert the person into your biggest fan. Here are the four things you should nail:

1. Say you want to work at the company stage and domain they are in
2. Show you have direct experience in the challenging parts of the role
3. Share ideas you have that you would implement were you in the role
4. Demonstrate an understanding of the company's potential challenges and business problems

Execute in Interview

Write these down on the piece of paper you have for the interview. Literally check off as you make the points succinctly to make sure you get them all in.

You want to make them think, "Wow, they are so excited and a perfect fit." This will seal the deal to help get you a referral.

Most people also appreciate some time back, so don't carry on too long. Impress them, and when the deal looks sealed, read them to see if they want to talk more or not. Many times, it's best to end early.

Step 5: Customize the Resume

Fine Tooth Comb Over Every Bullet Point

You are still not in the clear yet! The final step once they agree to refer you is sending them your resume. The recruiter has to decide whether they want to interview you.

So, you want to customize the living heck out of that thing. There's basically zero limit at this point to how customized you want it to be. You should make virtually every bullet point relevant to the job you're applying for.

Way too many people make the mistake of just sending over their generic best accomplishments. That's way too lazy. At this point, you know the recruiter is going to give you a serious look, since you're a referral. Write your resume as if it's a brand new 1-pager selling you to the recruiter. Do your best to nail it.

The question I like to ask is: "What is the recruiter scanning for?" There's always 1 key skill that's hard to find for a job. That's probably what they really need to distinguish you. Find that **and then bold all your details relevant to it**. Let's say it's a job in a highly niche ML area, **then bold parts of your ML experience**.

This is actually an awesome preparation step for your interviews anyways, because it helps you collect the stories you need, and identify what's most attractive about your candidacy. You always want to select stories as relevant to the job at hand as possible.

Put a Pin On It At the Top

Put a few lines at the very top of your resume that answer the key questions that the recruiter could be most skeptical about. If you don't have a specific type of experience very clearly, outline where and how you have that experience.

This should be easy, because you should have prioritized a job with candidate-market fit. But it's still worth going through this exercise.

As an example, my friend was sending over a resume for a Director at a series C startup job. He had never even been a Director. So the skeptical point would be: Could he lead large teams and be a leader? So, at the top, he said:

Product Leader. Part of core product leadership teams at X & Y. Managed orgs of 6-12 in last 2 roles. 11 YOE in Product, 5 as a Manager.

That summary nicely addressed the main skepticisms of his candidacy. It showed he viewed himself as a leader. It demonstrated he has years of experience comparable to a Director. And it showed he has managed large organizations.

Do the same for yourself.

Step 6: Follow-Up Frequently

With The Person

Follow-up is the name of the game in this process since we were so targeted.

The person recommending you is busy. Follow up with them till you get an email confirming their referral from the company's ATS system.

If you haven't heard a few days after the confirmation email, you can assume the recruiter passed on you. That's okay. It's still not a hard no.

So, ask your contact if they know who the hiring manager is. See if they'd be willing to put you in front of them or share their name so you can message them. Many times, they will know the hiring manager and message directly. If not, at least get the name so you can message the Hiring Manager. This works really well and getting an interview.

If it doesn't work and you still haven't gotten an official no, you can even ask your contact to message the recruiter and ask about a status update. Sometimes, that final push gets you the interview!

But if you get an official no, go ahead and move on the next thing.

Also, quick note: be sure to be extra responsive in your response times with all these people. That helps convey your excitement.

Recursively Iterate Through More Referrals

You can also have multiple people recommend you for the same job. So, go ahead and follow up on the job too.

If you didn't get a no and you had no success with your first contact, move on to a second contact.

If you're seeing traction getting referral calls, you're not wasting people's times. You're doing the right thing. The recruiter may just have missed out on your potential. But if three people refer you to a job, a recruiter is much more likely to give you an interview.

So, the name of the game is not just follow-up, but follow-up at every step of the journey. And cut bait only after an official no. 99% of the time, this process will help you get an interview or an official no.

Final Words

That's it! That's how you will for sure get interviews. You may deviate from my plan in certain ways. That's fine. But the key takeaway is: **stop dropping resumes and start doing sales**. You got this.

Chapter 2: How to Succeed in Interviews

Interviews are the most important skill they don't teach you in school. Too many of us are forced to do the process on our own - prepare, practice, and give the interview - all without guidance or support.

Remote interviews make this process **especially hard**. You have to deal with technical issues, awkward silences, and a lack of body language.

Once it's all over, you just get an email rejection. There's no humanity to it.

The crazy thing is - once you reach the interview stage, you **don't need to be the smartest or most qualified person to get hired**. You need to be the person with the best interview performance. So, today I'm going to help you all conquer this incredibly tough skill.

The Process is Tougher than Ever

I was chatting with one recently laid-off big tech lead product manager who had this to say about his grueling interview processes:

I've had 7 processes now where I had **10+ interviews** and still haven't secured an offer.

Many of you are going through the same. With the crazy lay-offs at companies like Lyft, Dropbox, and Unity, high-quality talent is competing with you for the same limited number of jobs. All of the factors in the market are conspiring against you.

Supply is up a ton, while demand is down a ton. It's a nightmare. But, you can succeed!

Here's what I've learned working with several candidates.

You Can't Use An Outdated Strategy

The fact is, most of you haven't been on the market in a few years. You had strategies to research and answer questions that work for you in the last cycle.

And they worked. You got a great job.

But the market is **much more competitive now**. The techniques that worked for you in interviews a few years ago just won't cut it now.

To crack the remote interview world we live in now, you can't use the "Cracking the PM Interview" - or book of choice - tips from a few years ago. You need something **for this moment**.

Working In This Market

The good news is, I have exactly that type. I'm going to share the exact strategy that I've used to get 6 laid-off big tech senior PMs jobs. Each of these individuals was struggling with their post-layoff experience and came to me. But our efforts together paid off.

Here's what one said:

AAKASH! I couldn't have done it without you. Thanks so much man for helping me think differently about that final round. I GOT THE OFFER!

So, I've put together advanced tips & tricks to implement the exact system I have been coaching them on:

1. **Key strategies** before the interview
2. **Templates** for what to research and how to answer questions
3. How to **maximize** the post-interview
4. The exact places and ways to **use ChatGPT** to help

So, let's get onto it!

The Sales Strategy to Winning At Interviews

The secret to my recruiting strategy is you are doing **sales, not being interviewed**. This is a crucial mindset shift for most of you.

What is the Hollywood depiction before interviews?

Worried, nervous, stressed. People go into interviews expecting to get some tough questions or case studies thrown their way.

The key to performing in interviews is to not just focus on their questions, but actually advance your own interview. You have to **sell yourself**. Let's go over how.

1. Key Strategies Before the Interview

So, how do you do sales well? 80% of the results come from what happens **before** the call. Here's the mental model I want you to remember:

- Product: YOU
- Resume & LinkedIn: Marketing documents they read before the call
- Interview: Your sales calls

Now, let's get into the strategies.

Strategy 1 - Schedule Very Few on the Same Day

The temptation when you are interviewing is to run many simultaneous processes. That's fine, but don't schedule too many interviews on the same day. In fact, you should only have 1-2 companies (even if they have multiple interviews) on a given day.

Why? The trouble with remote Zoom interviews is they are totally **energy-sapping**. Your brain goes to total mush after a few.

Having a small number of interviews per day has a couple of benefits:

1. **Allows you to do deep research before each interview.** It's better to have a funnel where you convert most of your interviews than a funnel where you just get a ton of interviews.
2. **Lets you leverage the magic of sleep.** I want you to review your research note and prepare the evening before your interview. Your brain is going to do a magic processing of your performance the day before to do even better the next day.
3. **Keeps your energy levels high.** The last thing you want is to not be excited about your interview. If you go in not

wanting it without all your energy, you're bound to fail.
Because someone else is going to go in with that energy.

So, how do you decide what interviews you have on each day?

You should keep a **stack-ranked list** of all the processes you are currently in. Things closer to the top of that list should always get top priority on days coming up sooner. You wouldn't want them to go ahead and like another candidate who is moving faster than you. The things lower on your list should get pushed out a bit so you can maintain your spacing.

If you find yourself in too many processes to maintain this list, ease up on the top of your funnel. It sounds counter-intuitive in this market, but it still makes most sense to focus on the conversion of the processes you have got in. This brings us to the next strategy.

Strategy 2 - Keep Your Overall Process Targeted

The easiest way to fail in interviews is to get yourself into processes you aren't a good fit for. This might be because:

- You don't have experience in that domain
- You don't have experience in that specific functional part of the job (eg, you have growth PM experience not platform PM experience)
- You haven't built that type of product before

You're essentially trying to force your way into the position. Instead of scattershot applying and getting into many interviews, if you do just one application a day, you can stay targeted on roles you are a good fit for.

If you stay targeted, you can avoid the dreaded Zoom Fatigue, which is your enemy in this remote interview process.

No sales person is going after 100 accounts at a time. Don't do that with your jobs either. Have a targeted list that are good fit where you can focus lots of time.

Strategy 3 - Prepare a Research Note

From the day you get a recruiter reaching out to ask you for your availability, you should start up a note with the following sections (and gradually fill them in before the first interview).

1. **Company:** this should include information like company stage, number of employees, last fundraise, valuation/market cap, and latest financial information if they are public.
2. **Role:** this should have a link to the job description, key bullet points listed, cover if you have direct reports, and other details you collect as you progress through the interviews.
3. **Product:** this should include the specific part of the product org you will be in, the specific artifacts you will work on, your hypothesis of their key metrics, and your own evaluation of the strengths/weaknesses of the product as it is today.
4. **Key Questions & Answers:** this should include the top 4-5 things that the hiring committee might be skeptical about your candidacy, and how you are going to proactively work to address them.
5. **Interviewer:** this should capture all the content from their LinkedIn posts, any interviews they have given, and other information that reveals their opinions about your role and function.

The sections most people miss are the key questions and Interviewer. The key questions helps you proactively frame yourself in response to skepticisms.

The interviewer research helps you charm them. If you can espouse their same philosophies, they will love it. And they won't remember they said it in an Interview or LinkedIn post.

This doesn't need to take forever. I highly recommend GPT-4 Bing to help you with this. It can scour the web and assemble all this for you.

Here are the exact prompts I would use:

- Company: Can you help me understand the stage of company lifecycle [Company] is in? How many employees do they have? Are they public; if so, can you summarize their recent financials? Are the private; if so, can you summarize their recent fundraises?
- Role: What are some of the key KPIs an [Role] at [Company] would focus on?
- Product: What exactly is the product and artifact of [Company]? What software artifacts would a product manager of [role] own? Can you attempt to describe the buyer and user personas?
- Key Question & Answers: Please read my LinkedIn [Link]. What would be the top skeptical areas [Company] for [Role] would have about my candidacy?
- Interviewer: Can you help me find [ROLE TYPE] content than [INTERVIEWER NAME] has written or been a part of online, and summarize their key points of view?

Strategy 4 - Master “Tell Me About You”

The one question that you know they are going to ask is “Tell me about you.” So, you need to bring a spicy, unique way to answer this question. The thing **not** to do is walk through your work history sequentially.

What I suggest instead is the “Let me show you why I’m the perfect fit for this” strategy. It’s three simple parts:

1. 2-3 sentence summary of history
2. 3 accomplishments relevant to the job
3. Why you’re interested in the particular role

All done in less than 2 minutes. It’s magic.

The way to really make this strategy sing is to proactively **address why they might be skeptical about your candidacy**. Maybe it’s

your experience in B2B, if you have worked mainly in B2C. Maybe it's the stage of company. Pick 3 accomplishments that demonstrate that they **shouldn't** be skeptical about you.

Strategy 5 - Practice Your Top 10 Questions

The hardest part about practicing your top 10 potential questions, is getting started. It's hard to come up with them. I suggest asking GPT-4 to brainstorm the questions. Here's a prompt I would suggest:

What would be the top 10 questions a VP or Product at Spotify, who is the Hiring Manager, would ask in a Group Product Manager for Self-Serve Ads interview?

It does a helluva job:

What would be the top 10 questions a VP or Product at Spotify, who is the Hiring Manager, would ask in a Group Product Manager for Self-Serve Ads interview?

As a VP of Product at Spotify looking to hire a Group Product Manager for Self-Serve Ads, the key areas of focus would include product strategy, user experience, technical knowledge, data-driven decision-making, and leadership skills. Here are the top 10 questions that could be asked in such an interview:

1. How do you approach defining a product strategy for self-serve ads? Can you provide an example of a successful strategy you've implemented in the past?
2. Walk us through your process for prioritizing product features or improvements. How do you balance user needs, business goals, and technical constraints?
3. What's your experience with ad platforms or self-serve ad products? Can you discuss the specific challenges you've faced in this space and how you overcame them?
4. How do you ensure a seamless user experience in the context of self-serve ads, while also optimizing for advertiser ROI?
5. Tell us about a time when you had to make a difficult trade-off between user experience and ad revenue. How did you approach the decision and what was the outcome?
6. How do you stay up-to-date with the latest trends in digital advertising, and how do you apply these insights to drive product innovation?
7. Can you provide an example of a data-driven decision you made in a previous role that had a significant impact on the product's success?
8. How do you collaborate with cross-functional teams, such as engineering, design, and data science, to ensure the successful development and launch of new ad products?
9. Describe your leadership style and how you've managed teams in the past. How do you empower your team members to take ownership and make decisions?
10. How do you measure the success of a self-serve ad product? What KPIs do you track, and how do you use this data to drive continuous improvement?

We might look at this list and tweak some things, but overall this is a great set to practice.

So how should you practice? The key thing to do here is to **take a video of yourself**. Then slowly go through and analyze that video. Look for:

- Not having smart or creative points of view
- “Apple pie” or “platitudes” statements
- Not answering the question
- Crutch words or filler
- Being too verbose

Then **practice again** after analyzing yourself.

I recommend doing this practice the day before, and also some practice **right before** the interview. This will help you do simple things like make sure you are looking at the camera, know what you look like on the other side, check your background, warm up your interview voice, and be ready to crush.

Let’s do an example together.

Advanced tip: on questions like “why do you want to work here?” swap your target company for one of their competitors. If your answer still makes sense, it’s not specific enough. Show you understand the company’s specific challenges and goals.

Strategy 6 - Build a Work Product

One of the people I was advising had one role they clearly wanted more than everything else on their plate. It was at a higher level than the other roles she was interviewing for, the company was a perfect fit, and the hiring manager was magical.

So, what we did is, we spent a few days building a pitch deck specifically for the job. This pulled out an old trick I had used to get the Director of Product job at thredUP back in 2014.

In this case, she had already been shared lots of data on the product, and the hiring manager was chatting with her regularly over email. So, she went ahead and put together a 7-pager draft product

strategy based on what she knew, and sent it over to the hiring manager asking for their feedback.

The hiring manager loved it. They went head over heels telling every subsequent interviewer, before the interview, how excited they were about the candidate I was coaching. This helped her walk into the interviews with a bunch of social proof. And the interviewers themselves were excited to be talking to her. The mutual good feelings lead to great interview performances.

As you might have guessed, she got the job.

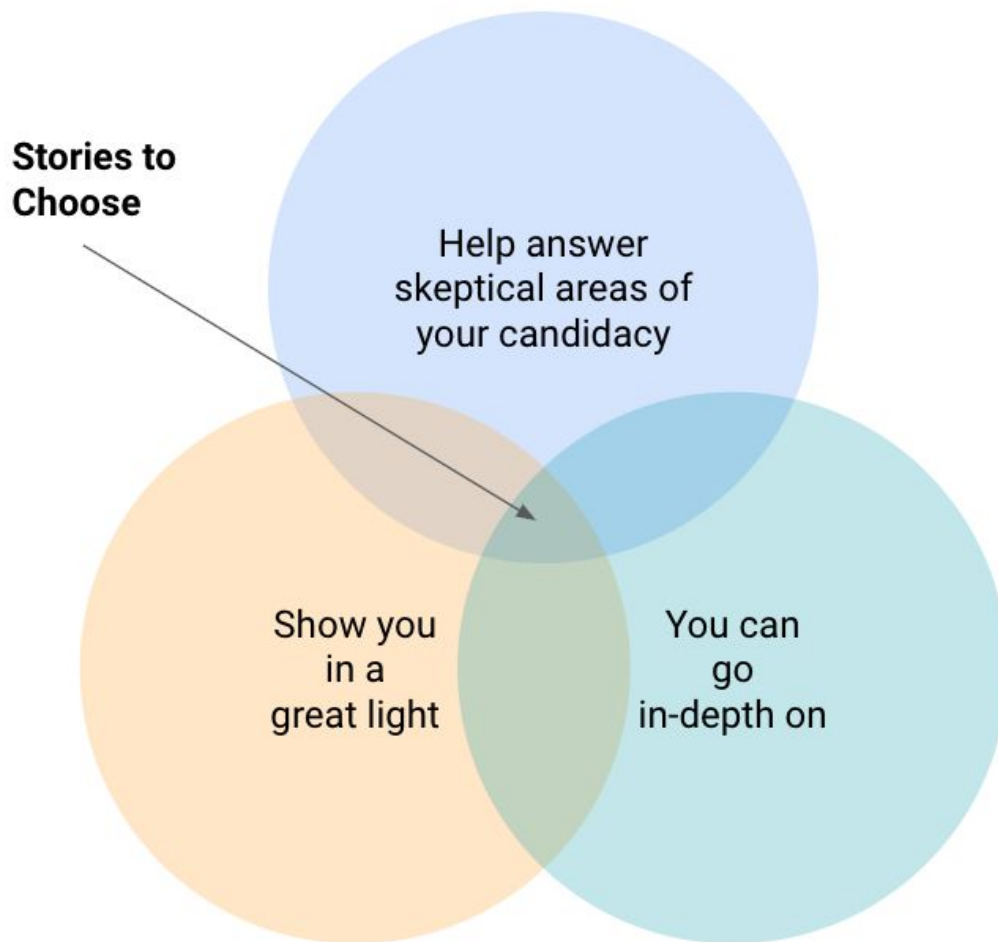
So, if you have a role that is clearly better than the rest, go the extra mile. Send the extra emails. Build a document or presentation. That will make your interviews, once you have them, much easier.

After all, what does every sales person do? They build a doc.

Strategy 7 - Master Your Stories

You should put together 3-5 stories that you really want to use in the interview.

What I like to do is find stories that sit at the intersection of three values: what helps answer skeptical areas of your candidacy, that you can go in-depth on, and that show you in a great light.



Then practice these stories. Write them down at the top of the piece of paper you're going to use in the interview, so you don't forget them. (Physically check them off as the interview progresses.)

When you practice them, avoid what my friend Peter Yang calls the "Disney version" of your story. Remember, great storytelling has challenges, problems, or obstacles that you overcome. Include those details to make your stories compelling.

On the Day Of, Optimize Your Setting

Since you only have a few interviews per day, schedule a nice long window before your interview. Use it to:

1. **Get Outside to Avoid Brain Fog:** The worst thing is to have a bad energy going into your interview. Getting outside, even if it's raining or cold, and going for a walk does wonders. It helps you think ahead to the interview in a calm setting. If breathing or meditation are your thing, go for those too.
2. **Rehearse Your Stories:** Find some time before the interview to write down the few stories you want to tell and rehearse them. If you can, film yourself practicing like I demonstrated above for 1-2 questions.
3. **Login to the Video Provider and Tweak Your Settings:** The last thing you want to do is feel awkward and self-conscious at the beginning of the interview because you need to tweak your microphone, speaker, or video camera. Jump onto the Zoom or Google Meet early and get all that fixed. Then leave.

Optional Advanced Step (\$\$)

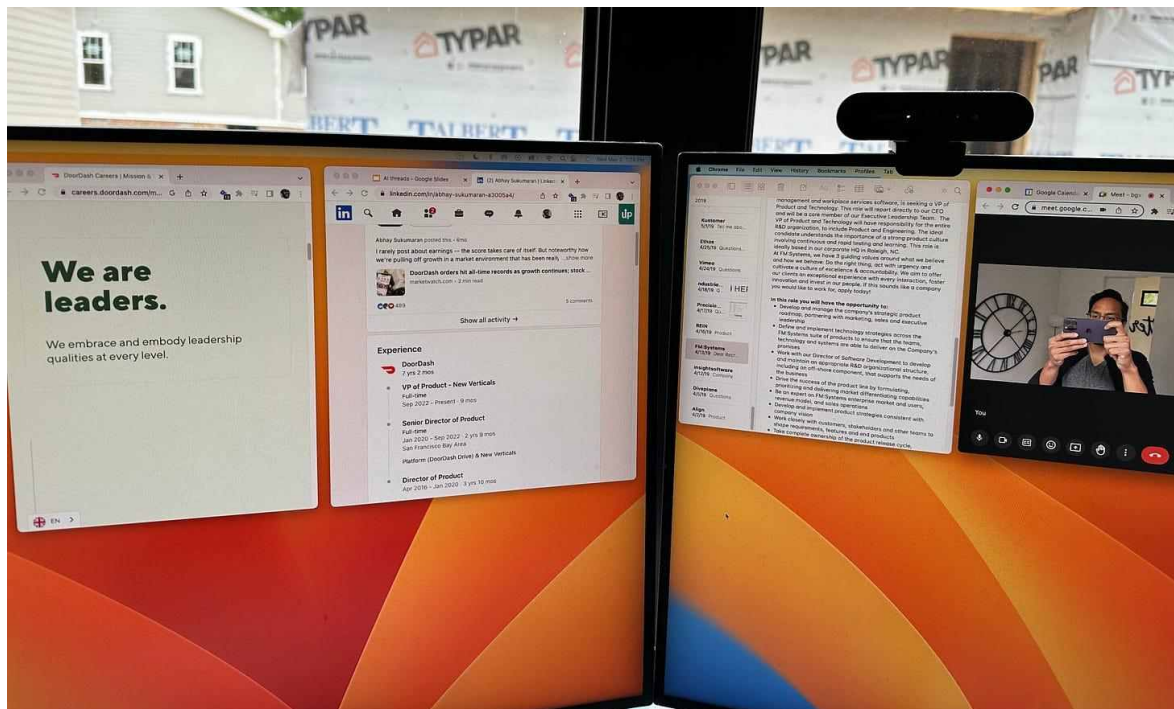
In remote interviews, as much as you hate to hear it, how you look and sound matters. I recommend investing in a great 4K webcam (or setting up your mirrorless camera if you have one), good lighting, a professional microphone with an amp and proper XLR cables. This will give you video and audio quality to stand out.

Even if it has a 1% impact, in this market it can be worth it. And great web conference quality is not going to hurt you once you're on the job either, so it's an investment that pays off in other ways.

2. Tips During the Interview

Tip 1 - Position Screens Smartly

There's no rule in interviews against having notes. It's not cheating. I recommend having two things right under your video camera: the video and your notes.



In this example, my webcam is above my right monitor. So directly under it I have my key notes and the video app. To the left, I have the interviewers' LinkedIn, and to the left of that, the company values. You can see how I've made all the windows short so I stay looking at the webcam.

This is advanced screen positioning for success.

Tip 2 - Ask for Time & Write Things Down

In remote interviews especially, it feels awkward to ask for time. And when you do, the seconds count down like minutes. You feel the

urge to immediately come back to the conversation instead of refining your thoughts.

Combat this with all your energy.

After most questions, ask for time to structure your thoughts and write down what you want to say. And before you jump into your response, look at what you've written to make sure it's going to stand out from the crowd.

Many interviewers are asking the same question to all interviewers and then comparing their responses later. It's not about having a great conversational flow. It's about having stellar answers. Asking for time and writing down your points before making them helps you have stellar answers.

This should be part of your practice. Specifically, see how long it takes you. 1 minute is the sweet spot. If you're thinking and writing for longer than that, shorten it up. But if you're shorter, consider writing up an even better answer.

Tip 3 - Gut Check And Course Correct

Observe your interviewer at all times. Are they a fan of you? Is this interview going well? Is this interview not going well?

If you get the sense that the interview is not going well, use the next time you ask for a minute to structure your thoughts to figure out how to turn around the interview.

- Are they skeptical about your experience?
- Are they finding you long-winded?
- Do they not like something about your style?

Immediately adjust.

In this market, even a single no from one interviewer can result in not getting an offer. So it's best to try to ask yourself halfway through

every interview if it's going well. If not, figure out how to turn it around.

Here are two tactics you can use to course correct:

1. **Break frame:** You can say something like, “just before I launch into this answer, I was getting the feeling maybe you weren’t sure about my background in B2B. Would it be worth us chatting about that for a second?” This can be a powerful way if things are going really south. Why continue the rest of the interview? Go back and fix what you need to.
2. **Use the question time:** When they turn the tables over to you, you can ask something like, “I sensed you were skeptical about my [X], what do you think if I addressed that and we talk through it?” This shows a lot of maturity and helps you address the problem head on.

Of course, these tactics are extreme and shouldn’t be used often. But if the situation calls for it, go ahead make the Hail Mary pass to try and save the interview.

Tip 4 - Record Meetings & Use Poised

This might be the most powerful tactic I’m suggesting that no one uses. Record your meetings! In two ways:

1. **Have a screen recording of your screen and self using a tool like OBS.** It will automatically save it for you and allow you to go back later.
2. **Use a tool like Poised to give you Transcripts.** Poised will give you a transcript of the meeting and analyze your transcript to give you recommendations. It easily highlights filler words

The other benefit of Poised is it gives you real time feedback in the interview on your clarity, confidence, and speed.

This is another vector where you can take advantage of the remote meeting format and stand out from other candidates.

Tip 5 - Actively Inject Your Stories

As you go through the interview, make sure to actively inject the 2-3 stories you identified in your prep work as addressing your weak areas. You want to work these into an answer if you can.

If not, consider working them into a question! Explain the success and then ask a question about it.

One of the accomplishments I had at [X was]. Have you guys tried something similar?

By the end of the interview, you should be able to physically check off on your sheet in front of you that you actively injected your key stories.

Tip 6 - Go Virtual With Your Whiteboard

If they toss you an interactive case question, or any sort of question where in real life you would write on a white board, consider breaking out the virtual whiteboard.

You can screen share anything: Notes, Notion (what I recommend since it's pretty), Google Docs, Miro, whatever you are comfortable with.

Being conversational always works well in interviews. Just have practiced this before the interview and be comfortable doing it.

Tip 7 - Keep Asking Questions

Act like the most curious, most enthusiastic about the position candidate they have ever interviewed.

Reserve some time after your interview so you can go over. Then when the interview time ends, ask something like, "Do you have a

few minutes to go over? I wanted to ask you..." You want to end the meeting on the high note of being someone who is really interested in the job. Just keep asking questions.

This leaves just the right taste in their mouth.

3. Answering Specific Questions

In this section, we'll cover how to answer the most common behavioral questions, product cases, and "what questions do you have for me?"

Type 1 - The 10 most common behavioral questions

Here are the 10 most commonly asked interview questions - and how to respond to each so you stand out & get the offer:

1. "Tell me about yourself."

- 2-3 sentences on your summary
- 2-3 recent accomplishments relevant to the role
- End on why the role is a great fit for your goals

Keep it to two minutes. Go on offense by sticking to the "role relevant" stuff only. Brief over the rest.

2. "Why do you want to work here?"

- I love the mission & vision
- The role is a great fit for my career goals & experience
- I've enjoyed meeting the people & the company's values

Add in the actual information about the company, role, and people you've met. Look excited.

3. "What is your greatest weakness?"

- I'm always improving
- My most recent performance cycle had this feedback
- Here's how I'm working on it

Show you don't have a fixed mindset. But be respectful of the question and show it's a weakness. Then end with how you're growing.

4. "What are your compensation expectations?"

- The high end of the band for this level
- I am concerned about the whole package
- I am keen to see how the interviews proceed in terms of fit

Don't throw out a number. And don't act like you wouldn't like their number either.

5. "What work accomplishment are you most proud of?"

- There was this tough problem
- I took these specific actions
- It resulted in this business impact

Captivate them with a story. Be specific about your contributions. Use numbers in the impact to make it stick.

6. "What is your biggest failure?"

- I can choose a work failure
- Here's what I did wrong
- Here's how I grew

They may drill down on why it's your biggest failure or what you did wrong. So choose a real example. But end on a high note - show you have a growth mindset.

7. "How do you handle competing priorities?"

- I prioritize based on ROI

- I let affected parties know early and often
- If needed, I change my tune. Here's an example

Show your framework and outline with a specific example. Act like it's normal and you handle it well.

8. "Tell me about a conflict at work."

- Here's why there was a conflict
- What I did to resolve it
- How I'm applying the learning going forward

Use it as an opportunity to demonstrate your teamwork & EQ. Don't come across as blaming anyone else.

9. "What are your career goals?"

- I want to progress in the job function of this role
- I especially like the domain this company is in
- Companies of these size are perfect

Show the role, company, and size are a fit for you. Tie it back to authentic things you care about.

10. "What is your approach to giving and receiving feedback?"

- Early and often
- Use 1:1s to develop personal relationships
- View feedback as a gift

Show that you like the process of growth. Demonstrate you can handle tough conversations with EQ.

NOTE: These are just templates to inspire you. Personalize to your situation, and always be truthful.

Type 2 - How to rock product cases

One of the most nerve-wracking interviews in the quest to get a PM job is the case interview. These tend to come in many flavors: Product Design, Product Execution, Product Estimation, Product Metrics...

But there's a surprisingly common set of techniques you can use to do well in all of these case types. Let's go through them.

Technique 1 - Ask for time to build a framework

In a case interview, they are looking for your ability to work through a systematic way of thinking. But just copy-pasting the CIRCLES method from Decode and Conquer doesn't work like it used to 6 years ago.

Instead, consider the following components and pick and choose what of the following 10 sections would be relevant to your situation:

1. Company mission
2. Business goal / north star metric
3. User segmentation
4. User needs
5. Key problems
6. Potential solutions
7. Prioritization
8. Outline the design
9. Discuss risks
10. Metrics to measure

The rule of thumb is that each section takes 4-10 minutes. So you can use that to size how many sections to have. Let's go through a few examples to illustrate.

Example 1: Design a vending machine for techies. You have ~15 minutes for the case.

Since you're short on time, you don't want too large of a framework. Pick a few sections that would be particularly good to cover. In this case, you can skip business stuff like company mission and business goal. User segmentation is already basically done for you. So the first section to write down is User needs. Because you don't have much time, you can combine that with user problems. Then can you add potential solutions, prioritization, and outline the design. You don't need to do the other sections for such a hypothetical case.

This leaves you with the following simple, achievable framework within 15 minutes:

1. User needs and problems
2. Potential solutions
3. Prioritization
4. Outline the design

Example 2: Help us solve this problem we had 6 months ago. You have ~50 minutes for the case.

This is a long case, so you can have 6-7 sections to your framework. You know you'll want to start with a section for investigating the problem. Then, you'll want to have some user centricity, solutioning, prioritization, and follow-up.

This leaves you with the following framework:

1. Investigate the problem (5 Whys & data)
2. User segmentation and needs
3. Potential solutions
4. Prioritization
5. Outline the design
6. Discuss risks
7. Metrics to measure

Example 3: What would you build if you were the PM for Google Maps? You have ~30 minutes for the case.

Since you have only 30 minutes, you want to shoot for less than 6 sections. Since you could easily talk about all 10 sections, it's important to pick and choose. You'll need to start with the mission and business problem, so you can combine that into one. Then you'll need to go through user segmentation & needs, so you can combine that. You'll also need to outline key problems, potential solutions, and prioritization. That's about all you'll have time for.

So that leaves with you the following framework:

1. Mission & business goal
2. User segmentation & needs
3. Key problems
4. Potential solutions
5. Prioritization

See how it feels specific to the situation, instead of something cookie-cutter you read in a book.

This framework style is called the **Unified Analysis** method, and it is by far the best way to go. Practice it before your interviews so you're comfortable with it.

Technique 2 - Write everything down

The number one mistake that PMs make in case interviews is not writing enough down. They just jump into responses. I've already included this in Tip 2 above, but it applies triply to case interviews. Write absolutely everything down!

The trouble with case interviews is that there's so much information that you have to mentally load. You just don't have the ability to do it all in your head.

The reason so many people don't write things is they are nervous. They don't want the interviewer to be waiting. Just practice this.

Every time they ask a question, say something like, “Can I take a minute to organize my thoughts on that?” and then speak.

You can even use that when you open a section of your framework. Say something like, “Okay, now let’s move on to listing potential solutions. I’ll take a minute to write down some of my thoughts and then come back to the discussion. Sound good?” This gives you time to write things down.

Technique 3 - Take time to suggest creative ideas

In the time that you ask for, be super deliberate. Don’t just rush to find the time. Once you write down a few ideas, take a look at them and ask yourself, “are these good enough?”

If the answer is no, just take time to come up with more ideas. Then ask yourself if the new ideas are good enough.

This is especially important in the potential solutions step! Creativity is everything here. When I was at Google, several amazing PMs were rejected just because their ideas weren’t creative enough.

Be creative!

Technique 4 - Find buddies to practice for months

The product case interview is a very specific skill that has very little to do with how you perform in the rest of the behavioral interview. There’s no substitute for lots of practice.

Join my Slack, or another product community, and find recruiting buddies. Just book each other for an hour on Zoom and spend 30 minutes giving each other a case interview. Record the whole thing and then go back and critique yourself.

This is an absolutely necessary step to getting better at product cases. Reading Decode and Conquer, Cracking the PM Interview, and a bunch of other people's pre-written responses isn't going to make you better. Live practice and post-practice self-critique will.

Technique 5 - Recognize the randomness

Finally, let's recognize that the product case interview is a very unique skill that has very little to do with actual Product Management skill.

In fact, success in many product cases is a factor of many things like, have you worked on a product in that area before? Did you latch onto the key detail they wanted you to?

There's a lot of randomness involved with case interviews, as opposed to behavioral interviews. If you got dinged after a case interview, don't wallow in your bad skill. It might just have been pure randomness.

Play the number game and move on.

Type 3 - What questions do you have for me?

Every PM interview ends with "what questions do you have for me?"

Yet, 9/10 of applicants don't use it for all it's worth. Their questions demonstrate lack of curiosity, and they ask too few.

I've interviewed 100s of PMs over the last few years. Here's how to rock this part of the interview:

First, shift your frame. In the interview, you are in the process of grasping for that offer. That's why you run out of questions. But, what if you have a few offers? Then you'll be looking to see which is best. Switch to the evaluative frame.

Second, preparation. Before the interview, list out all the factors you want to learn more about the role. Ask these specific questions. When you run out, these 9 generic one's work well too:

1. “How many dedicated engineers, designers, and analysts would this PM lead?”

A PM who is influencing other engineering teams has an uphill battle vs one leading a large engineering team. A PM with just engineers has a more uphill battle than one with designers and analysts.

2. “What product rituals does the team practice?”

Different product teams have different planning, review, and sprint cycles. Some practice regular discovery. Others leave that for other teams. Some review results together. Others don't.

3. “Would you describe the culture as product-led?”

Many orgs are engineering-led (even if they say they are product-led). Others are exec-led. Some are design-led (like most gaming companies).

4. “How is PM retention here compared to previous places you worked?”

Retention is one of the best all-in indicators. If PMs often leave, you can expect some negative conditions. Perhaps they ask too much of PMs, or don't empower them.

5. “What are the biggest challenges someone in this role will face?”

Each role is different. Some face challenges in getting permission to work on other's surfaces. Others' roadmaps are filled by foundational support for other teams.

6. “What are the toughest things about being a PM here?”

PMs can face a myriad of different challenges, from influencing to getting resources to graduating experiments.

7. “What do you like least about the culture here?”

Some move super fast, and PM’s roadmaps experience a lot of thrash. Even more move slowly, and PM’s struggle to ship many consequential features.

8. “Are you design, engineering, analytics, PM constrained - or something else?”

Every team is constrained somewhere, even at FAANG. Figure out what the constraint is.

9. “How is the process of working with legal and compliance?”

In highly regulated industries, bad legal teams gum up each and every feature. Others just slow them down.

Remember the ‘5 Whys’ technique. If the interviewer gives a concise response, follow up with specific questions. This shows you care about the answer and are good at getting to the “real” why as a PM.

Finally, use all the time and more. This shows you care and have options. It also demonstrates a curious, evaluative personality - great signs for a PM.

3. After the Interview

Action 1 - Debrief

I recommend you write down, after the interview:

- What went well
- What didn't go well

Then go analyze your Poised meeting summary and transcript. You'll be surprised what you notice. Revise your notes on the two topics above, and add a third topic: what questions you were asked.

Action 2 - Write A Thank You Note

68% of interviewers care about thank you notes. After you've written your diary, go ahead and draft one. For most roles, it doesn't need to be long or fancy. It just needs to reference the conversation and be sent.

But if it's your top role, make a quick Loom video where you expand on something you talked about for 60 seconds. This helps you stand out.

When should you send it? I recommend building the note right after the interview, but scheduling it to send 2 hours after the interview. You want to hit their inboxes before they submit their scorecards in their Applicant Tracking System (ATS). So, **don't wait for 24 hours** like the old advice used to be.

What if you don't have their email? Guess it based on the format of the recruiter. Eg if the recruiter first initial last name @ domain. (You can also use GPT-4 to guess the email.) If that doesn't work, use a tool like Apollo.io to get their email. If even Apollo doesn't work, send the thank you to the recruiter and ask them to forward it.

Action 3 - Build a Cherry On Top

This market is brutal. You've gone through tens of hours of interview prep and interviews for a position. Why not put in 2-3 more hours to secure the deal?

For their top positions, I have been advising my mentees to think of what work product they can make that will show quality, interest, and put them over the top.

A strategic way to do this is to think of who can be the **voice for you in the debrief meeting**. Is it the hiring manager who loved you? Or the skip-level who was impressed? Then, think about what you could create and send to them to show them you go the extra mile.

This helped 2 of my 6 recent mentees secure job offers, so I highly recommend it:

- One did a teardown of a competitor feature launch they discussed in the interview
- The other gave more detail on an area she could have been more clear in the interview

In this market, the name of the game is going above and beyond to stand out.

Action 4 - Revisit Your Diary After Results

There's no such thing as a failure in interviewing. It's all just data for learning. In this market, you're going to get a lot of No's. Don't take them personally.

Instead, head to your diary again. Write down some more learnings of how you will get better next time. Revisit your meeting recordings to analyze what went wrong.

Then, move on. Eat an ice cream, hug your wife, and take it in stride.

Bonus: Ask recruiters *and* hiring managers if they can give you feedback. Send a different email to each to ask for feedback. 50% of the time, at least one of them will give you some feedback.

Final Words

People may beat your qualifications. **Don't let them beat your preparation.**

Chapter 3: How to Negotiate the Offer

“Should I even negotiate in this down market?”

I got this question from a paid subscriber a few weeks back. And I was a bit incredulous: **of course you should!** I thought it was a flash-in-the-pan, just one person being cautious.

Then... another one of you asked! And another.

At first, I didn't get it. But as more folks asked, the more it began to make sense to me. With getting a job being so hard in this market, **most of you just don't want to mess that part up.**

The first reader, for instance, was laid off for the first time in their career. They were used to being a top performer. Many of you are in the same situation.

I understand your thinking, but I have to disagree. There are four important things negotiating gets you:

1. **Shows you're a negotiator:** Negotiating shows that you're someone who will negotiate on behalf of the company once you're in the role.
2. **Perception of being in demand:** Negotiating shows you have an abundance mentality and aren't just grasping for this job.
3. **Seniority:** Negotiating helps you come in with a more senior aperture.
4. **Higher base:** Negotiating puts you at a higher base for future raises.

Of course, there are some real downsides of negotiation if you do it badly:

- **Rescind:** Negotiating can cause them to take back the job offer.
- **Inflated Expectations:** Negotiating can cause them to expect more of you than you can deliver.
- **Negativity:** Negotiating can drive them to have a negative opinion of you once you start.

The key here is: **if you negotiate badly**. If you negotiate the right way, you don't need to experience *any* of these outcomes.

Testing It Out

So, as more of you reached about these issues, I decided to partner with 10 of you to confirm these beliefs. And it worked pretty well.

Over the past few months:

- One reader get \$60K more in per year comp
- Another reader negotiated twice as much PTO leave
- Six readers had their raise requests accepted first ask
- One reader hit a brick wall and didn't get a higher offer, but leveraged it for a raise at their current company
- The final reader *doubled their equity grant*, that is worth roughly \$1M/year if the company IPOs

After all that work testing out these strategies in the real-world of today's down-market, I'm ready to share with you all.

This Chapter

I've put all the systems and techniques into the ultimate guide to negotiating in this downturn. We'll cover:

1. The negotiation begins well before the offer stage
2. The dimensions of an offer worth negotiating
3. Assessing your negotiating power
4. Planning your ask
5. Sending your ask
6. Negotiating to meet your conditions
7. The most common pitfalls candidates make
8. An example candidate going through all the steps

I'll also share real-life tales of the 10 negotiations we worked through - plus all the negotiations I've had in my life.

This should help you earn dramatically more in the next negotiation.

The negotiation begins well before the offer stage

Negotiation isn't a single event that happens after you receive an offer. It's a process that begins the moment you interact with a potential employer. Your actions, words, and the impression you create all set the stage for the offer negotiation.

By recognizing and optimizing these early interactions, you can significantly enhance your position when the time comes to discuss the terms of your offer.

Action 1: Crush the whole application process before offer

From your application to your interviews, every step is an opportunity to present yourself as a strong candidate that a company will be willing to make extra concessions for. Here are a few things candidates tend to miss out on:

1. **How quickly you respond:** Set the tone for yourself as someone who communicates promptly and professionally.
2. **The extra work that you do:** As I highlighted in [Chapter 2](#), you want to do extra work to stand out. Develop a work product that shows you are the candidate they *need*, not just *want*.
3. **The level you present yourself:** Many people get the job, but just barely, and get down-leveled. This squashes most of your negotiating leverage. Avoid this. Seem senior.

Action 2: Set the right compensation frame early

The thing you don't want to do is scare them away with your compensation expectations. Don't just give out your numbers at the beginning.

Instead, set the tone that you're an experienced job-seeker:

1. **When they ask you what you're paid, ask them for their range:** They'll ask, in many cases, "What are your compensation expectations." You can say, "I was actually interested to understand the range for this role."
2. **If they don't discuss the range, bring up the range:** If they don't ask, bring it up. I recommend the first phone chat itself. "I'm incredibly excited about what I've heard. I wanted to ask; What is your total compensation range breakdown for this role?"
3. **If they won't share the range, ask if a range makes sense:** Sometimes, they'll be coy and not respond. Follow up with the same question, "I'd like to make sure this is worth everyone's time. What do others at this level make?"
4. **Do all of this while expressing your great interest in the role:** When you don't have an offer in hand, getting the offer is still your priority. Never come off as aggressive or not as interested as another candidate. The most interested candidate often wins.

Action 3: Get a sense of their compensation philosophy early

Try to gauge the company's compensation philosophy during the early stages. This can provide valuable insights that will shape your negotiation strategy.

- Are they a startup that **leans heavily on equity**?
- Do they **tout a high base salary** as their main attraction?
- Or perhaps they **emphasize a holistic compensation package** including bonuses, benefits, and growth opportunities.

This can be a significant hint towards where they might be more flexible when it comes to negotiations.

The dimensions of an offer worth negotiating

Now that we've got the pre-offer stuff out of the way, let's get into the offers themselves.

There are at least 7 areas worth considering in every single negotiation:

1. Your base
2. Your bonus
3. Your equity
4. Your budget
5. Your signing bonus
6. Your paid time off (PTO)
7. Your career growth plan

You should weigh what's most important to you, but **that's the ranking I recommend**. Here's why.

Insight 1: Prioritize base compensation first

You should prioritize your base salary. I know this is a hot take. Why base over bonus, equity, or signing bonus? It's because of the time period we're in.

→ **Bonus is highly dependent** on how well you do at the company. And even if you've been a high performer your whole life, a new company and job could present challenges you can't predict at the time of offer.

In addition, poor company performance could shrink the size of the bonus pool. That's can often be completely out of your control. And it's not hard to imagine, given the economy is brutal right now for tech company growth.

→ **Equity can easily go in directions you can't control.** For instance, everyone who was granted technology stock in 2021 has seen a precipitous decline in the value of such stock.

And if you think you can predict that this is a “low point in the tech stock market,” don't delude yourself. No one but the top 1% of hedge fund investors can time the market.

As a result, I recommend you consider equity as worth a small fraction of its paper value. The rules of thumb I use are about 60% for FAANG, 40% for a Mid-cap, 30% for a Small-cap, 20% for growth stage, and 10% for seed stage.

*As a result: The place where equity makes the most sense is FAANG. Let's say you go for 50% higher stock. Even if you value it at 60%, that's 30% more money. It's extremely hard to get 30% higher salary. But 50% more equity isn't unheard of. So that's the optimal strategy. **It always comes down to knowing your specific situation.***

FAANG is barely hiring right now, so this probably doesn't apply to you.

→ **Signing bonus is a one-time expense.** The power of a career in tech is the compounding you experience. Tech salaries have outpaced inflation for the past 35 years. You want to ride that train all the way up by getting the compensation into your total package.

What you'll find is that signing bonus is *always* the thing companies are *easiest* to move on for this exact reason. You shouldn't value it too highly.

Insight 2: Don't forget about things outside total compensation (TC)

If you are coming in as a leader, **you must negotiate your budget for headcount expansion and growth.** This is a vital part to your

success. You should be able to fire under-performers relatively quickly as well.

But even if you are coming in as an individual contributor (IC), consider negotiating on things like: your learning & development budget, your days off, your counterparts in vital partner functions, your phone bill etc.

Even your career growth plan is worth negotiating! Meet with your boss and set the expectation that you want to get promoted to a big next role fast. This helps get you off on the right track even before you start.

Another sneaky item that is worth negotiating is your **cross-functional peers**. If you're a PM without a dedicated engineering manager, for instance, a dedicated counterpart can greatly contribute to your success. That dedicated EM is something worthy to negotiate for.

Insight 3: PTO is a great thing to negotiate

Too many people write off PTO. I found this is most common for people coming from an unlimited time off workplace and going to a place with limited PTO.

Here's the thing: **you should value each PTO day at your total compensation.**

So, let's say your TC is \$350K, and you have 12 PTO days plus 12 company holidays. That means you have $260 - 24 = 236$ work days, and you earn $\$350K / 236 = \$1,483$ per day. If you can negotiate 3 more PTO days, you've earned yourself \$5K!

There's much more to wealth than money. Time is not just money, **time is wealth**. Each day in this world we don't live to the fullest is one more we've lost.

But I'm a work-driven person?! I don't want to take the PTO," some of you are saying.

When you leave companies that have restricted PTO, they generally pay out those days (if they don't, negotiate for that.) So you'll actually earn those days back in cash later. So from a money point of view, PTO still makes sense.

Assessing your negotiating power

Okay, so we've gotten the offer understanding out of the way. How much power do you actually have to negotiate?

Ranking scorecard

Negotiating power varies based on a ton of factors, so I've created a negotiating scorecard to help you identify what you can do based on that scorecard.

Go through these 25 questions and calculate your negotiating power.

Personal characteristics:

- Do you have more than 10 years of experience? +5
- Would you be fine if they rescinded the offer? +25
- Do you need this job (e.g. you were laid off)? -50
- Was your last full-time thing school? -20
- Are you currently in a full-time job? +5

Job search characteristics:

- Have you found the market easy? +10
- Have you found the market tough? -10
- Do you have more than one job offer? +40
- Are you in other interview processes that are going well? +5
- Is this your top choice job that you are currently in a process with? -20

Offer characteristics:

- Is this offer at a higher seniority than your last gig? -10
- Is this offer a bump from what you were previously earning? -10
- Is this offer already at the high end of their range? -10
- Is this offer at the high end of your market range? -20
- Is this offer at the low-end of the market range? +10

Job characteristics:

- Is this a people manager role? +15
- Is this a director+ role? +15
- Is this a c-suite role? +15
- Have they been looking to fill this role for more than 3 months? +25
- Have they been looking to fill this role for more than 6 months? +50

Company characteristics:

- Are you earning more than the median worker at the company? +5
- Is this a startup with a wide range of titles and people at various seniority levels? +25
- Is this an old-school stodgy company that has been around a while? -25
- Is the company doing financially well? +5
- Is the company in dire financial straits? -10

Just reading the directional impact of some of these things should help you out. Moreover, based on your score, here's what you, specifically, should do:

Looking for an example? Jump to the bottom of the article.

If: Less than 10

<10% of people

You do not want to negotiate at all. You fall into that group of people who need this job and don't have much leverage. This is a very small group - no one in my group I worked with did.

You landed in this category because you need this job. At this point, you should carefully question why you want to negotiate.

Is it really worth potentially losing the job? Probably not.

Sign and do a great job. Then, get a raise.

If: 10-50

40-50% of people

Negotiate lightly. The good news is that you have some leverage. You can definitely make a data-based ask.

Your strategy should be to make a small ask that they can accept. If they don't accept, you should be willing to sign and move on. You don't need to go duking it out for multiple rounds.

If: 50-80

20-30% of people

Negotiate with vigor. You are in a position to negotiate. You definitely have room to make this offer better.

You should come up with a carefully sized ask that is based on data. Then, stick to it for a round of negotiation (if they don't meet it).

You have market power and it'd be a shame for you not to use it.

If: 80+

<10% of people

Shoot for the moon. If you are lucky to have scored very well in the scorecard, this negotiation is heavily tilted in your favor.

→ **You shouldn't be shy about making a big ask** and following up after the first round with a second round.

→ **You can also probably push the timelines.** You can do things like call them your last allotted day with your ask to amp up the pressure. You can also go 2-3 rounds of negotiations.

→ **You should still act very professional and excited about the particular offer.** Don't let your negotiating power get to your head. People like to negotiate with people who are likely to accept, not people who are using them as a negotiation chip for another job.

Planning your ask

Now that you've gotten a sense of your type of ask based on the scorecard, it's time to plan your ask. Here's the vital research you need to do and where to do it.

Step 1 - Find the data

The first and most important thing you need to do is dig into the *right* compensation data.

There's a couple amazing places for data that didn't exist a few years ago:

1. **Levels.fyi**: Especially if you're working at a company covered by this database, this is the only one that's actually accurate. Glassdoor on other sites are completely inaccurate at this point.
2. **MostlyEquity**: My friend CJ provides an awesome service where he can help you benchmark in real-time your comp and equity if your company isn't big enough to be on Levels.fyi.
3. **H1B Salary Database**: This doesn't have total compensation information, just base, but the benefit is it's highly accurate. There is no self-report based fudging. This is the real data.

You can use the usual as well: talking to your network, Glassdoor, etc. But they aren't quite as good as the above.

What does tend to be accurate is salary benchmarking companies and studies. This is a multi 10s of thousands of dollar service provided by payroll providers to HR departments. One of the best things you can do is make friends with someone in HR at a company that is purchasing this information. If you can get this data, you are golden.

But even if you can't...

Critical Skill: Understanding Your Market Worth

Having a really strong understanding of your market worth is a critical career skill. You need to do three things:

1. Have a **realistic understanding** of your candidate-market fit.
2. **Ask recruiters early on** in each process with the total compensation range for the role.
3. Write all that down and **reference it** during your negotiation.

The key insight here is: the places you are getting interviews, you will eventually get an offer if you keep improving. So, the data from where you are getting interviews is even more relevant than some of the market data above. **It's the most accurate data you can get.**

Here's an example of this at work.

You get yourself into six interview processes with these potential offer ranges:

- \$280-480K
- \$300-450K
- \$300-400K
- \$310-420K
- \$320-430K
- \$350-400K

You can take the median of the high ends, \$420K, and take that as roughly your market worth.

Step 2 - Develop your data-based ask

If you don't have a competing offer, one of your best leverage points is going to be the data. Here's how to go from data to ask.

The best way to think about this is *relevant comparables*.

What is a relevant comparable? It's a company that:

- Has a similar **enterprise value**
- Has a similar **employee count**
- Was founded around the **same time**

You ideally want to meet all three of those criteria if you want to find a great comparable.

The next thing to consider within comparable is the percentiles.

If you were to compare Netflix to similar \$180B tech companies with 12K employees founded in 1997 (like Booking Holdings), Netflix stands out as paying much more.

That's because Netflix pays top of the market. They pay 99th percentile. If you are at a company like Netflix, Google or Meta, you should seek a 99th percentile offer.

But don't go off your skis and assume you can command top of the market compared to the Bay Area, another level, or more experienced folks.

In that case, your goal is to look at the universe of pay for:

- Your **location**
- At that **title**
- Your **years of experience**

On the other hand, smaller and less successful companies rarely tend to pay at those upper levels. So if you're at 65th percentile but they tend to pay 40th percentile, you may not have much negotiating room.

What if you're above the percentile where the company pays?

Still negotiate! You may just be that exceptional candidate they pay more. It just changes your strategy.

You need to be tactical to understand the areas of leverage within your negotiating levers the company is willing to move. I recommend finding a person at the company who can help you understand where you might be able to negotiate.

Consider your BATNA

What is your Best Alternative to a Negotiated Agreement (BATNA)?

- Do you have a **side gig** to make income? Are you currently in a job?
- Or were you **laid off**?

The higher the BATNA you have, the more you can ask for. But even if you're laid off, don't make your lack of a BATNA a hindrance to negotiating. Even in this market, you can negotiate.

Advanced Tip: Collect data well before offer stage

Salary negotiations begin from the first phone call from the company sharing your offer.

Understand your market worth well before even trying to get interviews at jobs. This can help you used the posted salary on roles (thank heavens that's more common now) to not waste your time on processes that are below your candidate-market fit based worth.

In addition, once you start to progress in interviews with a company - definitely before the final round - spend a few cycles doing the research on how the company pays so you have some target numbers ready.

Step 3 - Gut check with your network

Now that you have an initial draft ask, don't just send it! Leverage your partner, your siblings, your parents, and your trusted friends to talk through the ask.

Sometimes we can have a total blind-eye in our research, ask, or communication of the ask. Trusted confidants will help you see them and correct them.

You don't need me to tell you this. But I have to say it. Be super careful about whom you tell your TC.

- **Here's what I wouldn't do:** go around asking people who you don't know very well or have worked with about your new comp.
- **Here's what I would do:** ask your trusted inner circle before sending the ask.

But, before we go, I do have to issue a warning on your network: 9 times out of 10, if anything they will be overly conservative. This is especially true if you ask your parents, folks from a minority background, etc. They're happy for you.

So that's why asking them is not the last step. After that, here's what I recommend...

Step 4 - Think outside of the box

If at all possible, go for a long walk or sleep on your ask. You want to marinate on it and think holistically about it. (So talk to your network right away.)

I highly recommend thinking outside the box:

1. How much **power does the Hiring Manager have?**
2. Do I have **vocal support from decision-makers** like the CEO, CPO, CFO, or Board?
3. What might **this particular company** be willing to move on?

If the hiring manager has a lot of power, and you have vocal support from the CEO, you can probably make a pretty big ask. You could ask for a bigger budget - maybe some additional hires right when you start. Or, you could ask for a substantially larger chunk of equity in the company.

Don't neglect these out of the box factors. One of your fellow readers recently negotiated a doubling of his equity compensation because he had already impressed the COO in the interview process.

I also encourage you to ask a very specific question, “**what would a confident white male do in my position?**” The data shows that white men ask for more in negotiations - and get more in return. I have experienced the exact thing in my 7+ years as a hiring manager.

Be bold.

Sending your ask

Step 1 - Choose your recipient wisely

Generally, you should talk to who has been talking to you. But not always. In particular, if you are not talking to your hiring manager and their boss, you probably want to.

So let's go through each scenario briefly:

- **Talking to hiring manager:** keep talking to them. And make the ask to them.
- **Talking to hiring manager and never talked to skip level:** talk to the skip level. Then make the ask to the hiring manager.
- **Talking to recruiter:** involve the hiring manager. Make a call who to make the ask to after.

Too many people neglect to work with the person they'll be working with most and will determine the direction of their career when they join the company: the hiring manager. Keep them involved in your plan. 9 times out of 10, they'll also be the right person to make the ask to.

Why is the hiring manager so important to involve?

There's a couple key reasons:

1. **Leverages cognitive biases.** People like things they have invested in. It's the sunk cost fallacy. Involving your hiring manager is getting them to invest in you.
2. **Helps them become the voice, not you.** When there are more conversations happening later, it's your hiring manager *already* saying you deserve some level of a better offer, instead of you. This changes the dynamics of the internal conversations.

3. **If they can't get a raise done for you and they like you, do they really have much power?** It's a good first bit of insider information on how effective your hiring manager is.

So, involving the hiring manager makes you more likely to get what you want, gives you crucial information on whether you want the job, and helps you succeed in the job if you indeed start it. It's a no-brainer.

Step 2 - Create a communications plan

Once you've identified your partners, don't just wily-nily handle your negotiation. Decide not just what the optimal message is, but what the optimal message and optimal channel is.

Subpoint A: Timing

So many people let their decision-making timeframe influence their negotiation.

You should be able to make a decision almost instantly, but you should strategically use time on your side.

→ **Move fast when it makes sense:** If you're ready in seconds, you can also respond in seconds. If the moment calls for it - like if you're negotiating with a fast-moving startup - do it.

→ **Move slow when it makes sense:** When they're not moving on a number and you have a lot of leverage, wait till the last day - or even the day after to reach back out. This built up tension can win you that raise. (If you're confident enough to execute on it. Several of my recent hires have. You all got this.)

So create a communication plan immediately after every communication you receive from the company.

Subpoint B: Channel Choice

You want to talk to people in the channel that they'll want. Some people really like the phone, others prefer e-mail. The general advice is: **use the channel they are using.**

But sometimes, you might want to perform a switcheroo:

- **If they're using e-mail, you may want to use phone if:** If you are delivering a message like having another offer, or making some sort of decision the day of the deadline. You want to be professional first and foremost - and that means getting them the decision at the right time.
- **If they're using phone, you may want to use email if:** You have a very simple ask that you think they're likely to accept. If they're calling you, they may just want to avoid putting the offer in writing. But if you have a simple counter-offer, you can use e-mail. It can actually **help** you avoid needing to reduce your ask on the phone. Just be sensitive to whether the company will culturally be okay with it.

Step 3 - Make it happen with confidence

At this stage, **the last thing you want to do is get meek about your ask.** When you make the ask, whether it's in email, over the phone, on zoom, or in person, do it with confidence.

At this point, you've received an offer. You're a prized candidate. Act like one who has always been prized and knows they will be prized if this job doesn't work out. Even if you are laid off and the market has been tough, once you've decided to make an ask, make that ask with confidence.

Manifest your power. Power pose. Go for a workout before the call. Meditate. Get yourself ready to rock the call.

Don't forget to scenario plan!

One of the best ways to make the actual negotiations happen with confidence is to practice bad scenarios. I highly recommend

recording yourself answering some potential scenarios in the negotiation:

1. **They call you after you send the email to say it's too big an ask:** Get ready to handle this common occurrence. Many people still get what they want. You want to: dissect the ask, reiterate your ask, and potentially change your vector to one of the other 7 negotiation items.
2. **They start to rescind your offer:** It's a worst case scenario. But practice rescuing your offer before it potentially happens. What will you say to prevent that? If they just sent an email, you should call them. What will you say? You should reiterate your worth and why both sides wanted to work with each other. You will also want to know if you are willing to sign at the offered rate with no raise.

Don't be under prepared for the negotiation just because you sent an email.

Negotiating to meet your conditions

Tip 1 - Have a backbone

Don't just give in at the first sign of weakness. It's normal to hear things like, "OO.. that's a tough ask. I don't know about that."

Fall back to your data. Followup with something like, "The data I have on levels.fyi and H1Bs shows that this is a reasonable request."

Well-reasoned requests tend to be granted at least partly. So keep working through the conversation with respectful but firm points.

The thing you want to keep an eye is on the calendar. You want to resolve to a decision at some point in the next few weeks. So don't just keep going back and forth on e-mails with a day delay.

Have a backbone for some time, then move on. But if you truly have time, consider using it to your advantage. As the old saying goes:

He who has the most patience often wins the negotiation.

How do you know how much time you have? That's Tip 2:

Tip 2 - Understand their alternatives

As you're making maneuvers on the negotiation chess board, you want to gather as much information about your negotiating partner as you can.

In my experience, recruiters have been more forthcoming about alternatives than hiring managers. Ask them things like, "How far along in the process are you with other candidates?"

This helps you get leads and understand where their head is at. Here are three common responses and what they mean for you:

1. **They're rushing you:** And saying things like, "Please decide by tomorrow so if not we can move on to the next candidate." Then you don't have too much time on your side and you must hustle in your negotiation
2. **They're nonchalant about the whole thing:** You're likely their top choice, but there are other candidates in the process. So you're replaceable, but not quickly and guaranteed. You have good negotiating leverage.
3. **They're ecstatic about you as their top choice:** If they're saying things like, "We're so ecstatic to have the opportunity to work with you" when you ask about other people, you're the best option. You have lots of leverage in this case.

Some people have great negotiation poker faces. If you really can't assess how many other people they have in the process, ask yourself: how rare is my skillset? The more your skills are unique, specific to the company's market, and represent lots of years of specialized knowledge, the less people they likely have in the pipeline. If you're one of a kind, they don't have many alternatives.

But if you're a new college grad, you don't have unique skills. Sorry. You could've had the best school experience ever - Harvard CS and won all the hackathons - you're still not special. If you have 25 years of experience in B2B SaaS Software for HR teams, you have unique skills.

Tip 3 - Be willing and open to shifting vectors for value

As the negotiation goes on, if they're not okay with your asks. Shift them to a vector lower in importance to you. They'll appreciate your flexibility and are generally more willing to try.

Example 1: They won't move on cash issues

This is probably the most common scenario out there.

You are asking for base compensation or bonus, but they won't budge. Move on to equity, PTO, or something lower on your list. Many companies, especially FAANG, are much more able to move on equity and other factors than cash.

Those things actually have cash value to you, even if the company prefers to hide compensation in things like stock-based compensation or benefits.

Example 2: They say we won't negotiate anymore - take it or leave it

You may have alienated the person because of your attitude, and things may be lost. But if you've been handling the situation lightly and you both are still friendly, you want to dig deeper. "Can you say more? Is this because of another candidate, compensation committee, or something else?"

Often, it's because of something like salary bands. You're asking for something above what they can do. You can say something like, "I want to figure something out that can work for both of us creatively. What if..." And then suggest another vector, like more PTO.

This is also a time to consider factors that are more specific to your needs. Maybe you can have the ability to work an additional day at home. That can especially make sense if your team is already mostly remote.

What if they just say yes?

Rejoice.

You made the best decision you could. Maybe the company just prefers not to negotiate. **Too many people over-read into saying yes as 'I should have asked for more.'** Remember, the negotiation isn't a conflict or battle to be won.

More likely that not, you asked for exactly how much you should have. If you've followed all the steps up till now, I reckon you have.

6 of the people I worked with for this chapter had this outcome, and we all celebrated. One got a 15% base compensation raise! All 5 got what they wanted and accepted the job.

The most common pitfalls candidates make

Pitfall 1 - Setting themselves up poorly before offer stage

There's two major ways you can set yourself up poorly before the offer stage.

The number one pitfall folks make when negotiating is to get to offer - but just barely.

If only part of the hiring committee wanted you, and someone had to pull a 'do this for me' for you, then they will have limited ability to increase your offer. So you can try to negotiate all you want, with all the best data, but it won't matter.

It's an obvious pitfall that's hard to address once it's happened. The other pitfall is :

Getting an offer 30-40% below your target

This often happens when the team interviewed you for a level lower than you were targeting. the anchoring is almost impossible to overcome.

So the solution is: **Always clarify the level you are interviewing for at the beginning.** Then, do your best to live up to that seniority in the details of your interview responses (avoid answers from too early in your career).

Pitfall 2 - Not acting excited about the role and with the highest standard of professionalism

This pitfall often happens together, but has two subparts.

Sub-part A: Not acting excited about the role

People give roles to people who really want them. Even if the role is really just a fallback for you that need to take, act excited about it. Nobody wants to go to bat to get more salary for someone whose disinterested.

It's basic psychology of negotiations. But having been on the side of the table offering a job, you'd be surprised how many times I encountered lack of excitement. Candidates would respond like this:

“Hi, I have a counter-offer at Apple that is 15% higher than this TC. Could you beat that? (attachment)”

That short e-mail didn't make me excited to go to bat for them. They barely treated me as a human had done the work to get them an offer through hiring committee.

On the other hand, when a candidate called me to explain that they really connected with me, they think we could build great products together, but based on the data they will e-mail shortly, they want a 15% raise, I was happy to go to bat for them.

Sub-part B: Not acting with the highest standard of professionalism

Another problem with that e-mail above is that it's formatted for an internal audience or friend. But you're not chatting with a friend. You're chatting with a recruiter or hiring manager. A much better e-mail looks something like this:

Hi Aakash,

It's been amazing going through the process with Apollo. I have loved diving into your prospecting products and using them. I had 2 emails to CEO responded since we last chatted!

It would be great to help build out the next leg of growth product as a Senior Product Manager. Thank you for the offer.

My ask is on the base compensation. I request a 15-20% increase. The reason is because I have a competitive offer from Apple. (Attached)

I truly enjoyed meeting everyone at Apollo, though. So if we can make the economics work out, that would be great.

I look forward to hearing back from you.

Thanks,

James

PS. Please feel free to call me 5-7pm today at
XXX.XXX.XXXX

Notice how this treats the negotiating partner like a person, is excited about the job, and still manages to make a sizable ask seem reasoned. It also drops in more proof of work by sneaking in that you have been using the product.

That's what a golden email looks like. You'd be surprised how many people are casual and treat it nonchalantly. That's okay if you want mediocre results. But if you've read the 6,000 words so far, you want excellent results.

Pitfall 3 - Neglecting the element of time

A surprisingly common pitfall, that we've briefly touched on above, is neglecting the element of time. Let's go a bit deeper on time here.

You want to **"think fast to respond strategically."**

If there's an overriding takeaway for you from this long piece, it's that you must be prepared for the next step before it happens.

Negotiations are a game that's won by the person who can think a few steps ahead.

Of course, it's a big decision, so how can you think fast? Here's what thinking fast looks like:

When I got the offer at Affirm, for instance, it was a Monday. I asked for until the end of the week.

But I knew exactly the number I was looking for right away. I mentally wrestled with for two hours. Then I shared the request to my friends and family for feedback, sleep on it, and gave a call with the ask the next morning.

When I say instantly, that's what I mean. You should be able to make your decision overnight.

This is useful for both the low leverage and high leverage scenarios.

In a low leverage scenario: you are trying to optimize your, “**bang for your buck.**” The bang is your total compensation. The buck is **the time and attention the company has for you.**

In many cases, their willingness to negotiate over multiple weeks is low. So you need to be able to think fast so you can get a few rounds of negotiation in before they are ready to move on.

In a high leverage scenario: you can actually go later than a discussed timeline! So in my Affirm example, you could call them on the last day of the week. I didn't know this tactic existed until multiple candidates did it to me!

And it worked. They were super excited about the role but shared their ask. We got them the upgrade that day itself and they signed later that day.

Pitfall 4 - Forgetting the humans behind it all

A negotiation is not a game. It's not a competition. It's a cooperative human effort.

Most product managers aren't going to be directly responsible for the profit of the company. We have a big impact, but we don't live or die by it. (We live or die by our shipped product features.)

Same with your recruiter and compensation committee. No one in the negotiation, unless it's a CEO/Founder, lives and breathes the impact of even a 100% increase to your compensation.

So if you treat them as humans and help convince them as allies, they may go to bat to you for that. In fact - one of the people I worked with for this chapter did exactly that. He doubled his equity compensation.

It's actually a wild story:

- The candidate paid to fly across the country on his own dime and meet with the hiring manager
- In the interview process, the candidate had written a 12 page doc
- In the negotiations, the candidate sent over 2 product mocks

If you want extraordinary results like this person, you have to do extraordinary things.

We're just dealing with people after all. And people can be wowed.

So in this process, **play the game to win. Think big.**

Insider Harsh Truths From A Recruiter

Let's round this chapter out by sharing the best "harsh truths" I heard from pinging my recruiter friends.

Harsh Truth 1 - You are Replaceable

Even if you're a rockstar founder of a once-in-a-lifetime company, you are replaceable. Just ask Travis Kalanick and Uber's soaring market cap. Some candidates get all puffed up on their own candidacy that they forget this fact.

Recruiters can always find a replacement. With 8 billion people, the world is a very big place. They might not even live in your hemisphere, but there's someone out there who can do what you do.

Harsh Truth 2 - Your biggest leverage point is before the offer

Generally, when offers are made, they can't be changed more than 20%. This is especially true at large companies.

So **your biggest leverage point to increase an offer is what you do before**. The surefire way is to crush your interviews and produce extra work products. If you go miles above every other candidate, then they can justify an offer miles above. But if you perform about the same, it's harder to justify.

In addition, **make sure you are at the right level**. If you are trying to interview for a higher level, make this clear early on. If the salary you desire is above the salary bands, sometimes the company can want you all they want, but just not be able to do it.

There's also **the special case of really big companies at junior levels**. These roles usually have plenty of good data on online. In this case, you should say something like this in the initial call with a recruiter: "The data on levels.fyi shows the equity range for this

position is \$200-300K per year. I was hoping you could get me to the high end of that range.”

This helps anchor them on the high side of the largest variable, equity. But it has to happen early in the process. It makes the most sense for companies like FAANG with very good public data online. My recruiter friends tell me it's hard to take an offer that is \$100K equity to \$300K.

Harsh Truth 3 - Not everything is negotiable

In some companies, there are just limits. Here are some limits recruiters shared:

- **In some bureaucracies, you can't really negotiate after the written offer.** So you need to understand this limit and get your big ask in early.
- **In many companies, the salary bands are inviolable.** After you hit the cash limit you have to move to other elements like equity.
- **Some companies just won't budge on PTO.** There are some companies who are religious about their time-based accrual.

These limits go to show that having a call with your recruiter or hiring manager is often a great step within the negotiation to understand the levers you can move to create a great ask.

It's not that *everything* is not negotiable. Just everything is not negotiable.

Putting it all together: a hypothetical example

Let's go through a hypothetical to take this all from theory to application.

Assessing Jane's negotiating power

Our PM is Jane, a Lead PM with 11 years of experience, who is evaluating a job offer at Spotify. She was laid off from Meta two months ago.

Step 1 - Ranking Scorecard

Her TC used to be \$600K (230 base, 60 bonus, 310 equity). Her new offer is at the same level and manages 1 PM but has lower TC at \$450K (210 base, 40 bonus, 200 equity). Let's help Jane out.

Personal characteristics:

- Do you have more than 10 years of experience? +5 | Yes | Running Total: 5
- Would you be fine if they rescinded the offer? +25 | No | Running Total: 5
- Do you need this job (e.g. you were laid off)? -20 | Yes | Running Total: -15
- Was your last full-time thing school? -20 | No | Running Total: -15
- Are you currently in a full-time job? +5 | No | Running Total: -15

Job search characteristics:

- Have you found the market easy? +10 | No | Running Total: -15
- Have you found the market tough? -10 | Yes | Running Total: -25

- Do you have more than one job offer? +40 | No | Running Total: -25
- Are you in other interview processes that are going well? +10 | Yes | Running Total: -15
- Is this your top choice job that you are currently in a process with? -15 | Yes | Running Total: -30

Offer characteristics:

- Is this offer at a higher seniority than your last gig? -10 | No | Running Total: -30
- Is this offer a bump from what you were previously earning? -10 | No | Running Total: -30
- Is this offer already at the high end of their range? -10 | No | Running Total: -30
- Is this offer at the high end of your market range? -20 | No | Running Total: -30
- Is this offer at the low-end of the market range? +10 | Yes | Running Total: -20

Job characteristics:

- Is this a people manager role? +15 | Yes | Running Total: -5
- Is this a director+ role? +15 | No | Running Total: -5
- Is this a c-suite role? +15 | No | Running Total: -5
- Have they been looking to fill this role for more than 3 months? +25 | Yes | Running Total: 30
- Have they been looking to fill this role for more than 6 months? +50 | No | Running Total: 30

Company characteristics:

- Are you earning more than the median worker at the company? +5 | Yes | Running Total: 35
- Is this a startup with a wide range of titles and people at various seniority levels? +25 | No | Running Total: 35
- Is this an old-school stodgy company that has been around a while? -25 | No | Running Total: 35

- Is the company doing financially well? +5 | No | Running Total: 40
- Is the company in dire financial straits? -10 | No | Running Total: 40

Step 2 - Takeaway

Jane ends up like most of you readers probably will in this market: in the 20-50 range. She should negotiate lightly.

Planning Jane's ask

Coming from FAANG, she's looking at a great offer from a slightly smaller company but it doesn't pack the same equity punch as her prior gig, where she had gotten multiple performance equity grants. She should negotiate for equity right?

Step 1 - Look at the Data

Not so fast. Let's take a look at the data for Jane first. We'll hop to over Levels.fyi for Spotify:

Spotify Product Manager Salaries

Average Compensation By Level

[+ Add Comp](#)

[Compare Levels](#)

Level Name	Total	Base	Stock (/yr)	Bonus
Associate Product Manager	\$ --	\$ --	\$ --	\$ --
Product Manager I	\$ --	\$ --	\$ --	\$ --
Product Manager II	\$208K	\$182K	\$26K	\$0
Senior Product Manager	\$249K	\$199K	\$49K	\$1K
Group Product Manager	\$ --	\$ --	\$ --	\$ --
Director	\$ --	\$ --	\$ --	\$ --
Senior Director	\$ --	\$ --	\$ --	\$ --
VP	\$ --	\$ --	\$ --	\$ --

The data is sadly incomplete. It doesn't have Jane's level (lead PM) and it doesn't have anything beyond Senior Product Manager (SPM). Jane is leveled right above SPM though, and something immediately jumps out: her equity is already pretty large compared to that SPM comparison. At 200 equity, Spotify has come up pretty high for her already. Her bonus is also pretty sizeable compared to the SPM data.

So how should we get more information for Jane? Then, Jane should look at what a Lead Product Manager's average compensation at similar companies on Levels.fyi pay for lead product managers: similarly aged public tech companies with similar market cap.

- **Uber:** 227/230/43
- **Block:** 228/ 177

So it looks like there is some room for Jane to earn more on the base compensation. But let's learn more - this data was only okay.

Over at the H1B database. We see one entry at \$195K for an SPM. It looks like Jane's base comp, might actually be low.

Making only \$15K more than someone you manage is an unusually low gap. Surely, some people at her level would be making more.

This gives Jane enough confidence that her equity and bonus are okay, but she wants to ask for more base compensation.

Step 2 - Develop data-based ask

How much base compensation should Jane ask for?

Given Jane has some leverage, asking for the Block/Uber (227/228) base comp rounded up to \$230K so that in case the company wants to round down makes sense.

Step 3 - Gut check with network

Jane checks with her network - including some white men. They think that Jane can ask for even more. After all, it's not just her target

compensation, but also her prior compensation that Spotify has to think about.

Jane realizes they are absolutely right. She can probably ask for more...

Step 4 - Think outside the box

Jane weighs whether to ask for more base compensation, or other benefits. She reviews the policies and finds that she only has 20 PTO on her offer, but she had 30 at Meta.

This seems like the perfect additional ask in the negotiation.

Sending Jane's ask

Step 1 - Choose Jane's recipient wisely

Jane has been talking to the hiring manager since she got the offer. So she plans to talk to him.

Step 2 - Jane's communications plan

He asked to call her and they chatted on an afternoon. So she plans to do the same. She will email and ask to chat on the phone with some time blocks of her availability:

Jane talked to her family that evening and plans to send this email in the morning if she still feels that it sits right with her.

Hi John,

I'm really excited about the offer we discussed yesterday evening. Thanks for your belief in me!

Spotify creator tools would be my dream next step. I love the space, and it's a perfect continuation of my creator tooling work at Meta.

I was recently dreaming about a Spotify-powered podcast topic generator. I think it could work given the

recent advances in AI.

Could we chat about the details about the offer? I wanted to float an ask by you.

I'm free 10-12 and 1:30-2. Or I can try to work around your schedule.

Thanks,

Jane

Step 3 - Making it happen with confidence

The Hiring Manager calls Jane at 10:05AM. The crucial part of their conversation goes like this:

Jane: Based on the comparable ranges on Levels.fyi for Uber and Spotify, I request a base of \$230K and an additional 10 days of PTO. This was the base and PTO I had at Meta. I think they are reasonable asks, and I could join if you can make it happen.

Hiring Manager: I understand, but that's high for us.

Jane: But because it's based on data, I think that it's reasonable for this level. Given my 11 years of experience in creator tools, I bring the ideal background for this role.

Hiring Manager: Okay. I'll try to get this through recruiting, but it's been hard in the past.

Jane: Thank you. I will follow up with the data if it's helpful.

Like most negotiations, the hiring manager didn't just give in. Jane had to have a backbone to make the ask happen.

Negotiating Jane's ask

2 Days later, the Hiring Manager calls Jane. Here's the key part of their conversation:

Hiring Manager: Jane. Good and bad news. The Good news, the hiring committee was able to move. They often aren't. They got you 10 more days of PTO! I'm hoping this will be enough for you to join us. We really need you here!

Jane: Thank you. That's exciting. But it's not what I was looking for. I will need to think about it. I really love Spotify. And you would be great to work with. Can I have an extra 5 days to consider it?

Jane gets pretty surprised by the response to her offer. Spotify only increased her PTO but not the base compensation. But she handled the conversation well. She had a backbone, and proceeded to understand the alternatives. She also asked for 5 more days to consider. That's great - it also lengthens the negotiation window.

So what should she do?

Jane reviews the tenor of her conversation with the hiring manager. **He really wanted to get Jane.** At this point, he was willing to fight.

Since she only had some leverage, and not a lot, Jane decided to **call the HM the day the offer is due and say she still requires the base compensation increase.** The conversation went like this:

Jane: Hello! I'm so excited to be chatting. I think the role really is a perfect fit. However, the base is still not where it needs to be. Can you guys work on that? I'm sharing my screen here on the data for Spotify and Uber.

Hiring Manager: Thanks. That's interesting. I appreciate your honesty. But I don't know.

Jane: Can you try? I really want to work something out for both of us.

Hiring Manager: Okay. I'll try!

And the result? Spotify came back with upping the base compensation to 220 and keeping PTO at 30!

It was a great outcome for Jane. If we value 10 PTO days at her 460 TC over 46 weeks worked, she's earned herself the \$20K right there. Her friends were right - she managed to get a total of \$30K more, when she was originally going for just \$20K more.

And now she gets to live life more (with the PTO)! At the margins, time tends to be an even better type of wealth than money. Jane has won big time.

Many of you might have been nervous about the second round of negotiation Jane took. She only had some leverage per the scorecard! And each conversation, the hiring manager was timid.

But she knew the hiring manager was still overall excited about her. This gave her the confidence to go back for more money, and it ultimately worked out for her.

You can do the same thing and earn yourself tens of thousands. I believe in you.

Final Words

Negotiation is a life-long skill. Most of us (who aren't in negotiation jobs) are stuck negotiating at most a few times a year - for serious things that matter like the new job, the new house, and the new car.

Be gentle with yourself! Be confident too. The biggest saboteur of our best negotiations tends to be ourself.

Chapter 4: Special Case – No PM Experience

Is PM the most guarded career or what?

Especially in this market, **it seems virtually impossible to get a PM job without experience.**

Well, I'm here to tell you: You can get a PM job without any experience, without an MBA, and without a CS degree.

I first put out this roadmap 1.5 years ago. Since then, I've worked with **over 50 people** to successfully get PM jobs. Here's what some have said:

- “No one else is quite as clear with what you must do, as far as tactics and strategy.”
- “By far the most realistic advice out there. Thank you!”
- “Thank you, Aakash Bhai, for changing my life.”
- “I just doubled my TC!!!!”

What I've worked on with these 52 folks is six specific things:

1. Self-assessment
2. Long-term plan
3. Executing the plan
4. LinkedIn & Resume redesign
5. How to find the right jobs
6. Interview advice

Over time, I've refined the systems and market-tested them to perfection.

In this chapter, I break down the exact systems I have used to get people PM jobs.

So, without further ado, let's get into it.

1. Self-assessment

Your self-assessment should follow 3 steps: learn where you stand, identify an appropriate target job, and then work backwards. Let's go through each.

Step 1 - Learn Where you Stand

Your first step should be to get a realistic understanding of where your gaps are to become a PM. Let's give you a quick primer on how different backgrounds affect your chances to become a PM.

There are four major types of backgrounds that enable you to approach this process differently. I'll refer to these as 'archetypes' throughout the rest of the chapter. Let's figure out which you are:

1. CS Degree
2. Product Adjacent Experience
3. Irrelevant Degree & Experience
4. Student

Archetype 1: CS Degree

If you have a computer science degree, you're significantly ahead of the pack:

- You can build apps to actually manage a product.
- You can understand technical concepts, so you can work on technical products.

If you're in this group and have work experience, you may be able to secure an offer at one level lower than you are now. So if you are a senior individual contributor (IC), you may be able to come in as a normal IC, versus having to be a junior IC.

Archetype 2: Product Adjacent Experience

If you have been working in technology with Product Managers, you also are going to have a different path to get ahead. Here are the rules of thumb for different types of adjacent experience:

- Former CEOs can bag VP level offers.
- Developers & Designers can *often* bag offers at the same level they are at
- Marketers, BizOps, and finance folks who worked with PMs can *sometimes* bag offers at the same level they are at

If you haven't been working with PMs, this archetype doesn't really apply to you, because you don't have the knowledge required and aren't going to be able to leverage those experiences as stories in your interviews. You fall in the next bucket.

Archetype 3: Irrelevant Degree & Experience

If you have a non-CS degree and experience where you didn't work with technology product managers, you don't need to fret. You just need a completely different strategy from the above groups.

You'll have to dramatically change your pitch to make it seem relevant. And there will be a lot of extra long-term work you need to do. More on that soon.

Archetype 4: Student

If you are a student, then you have a very unique set of challenges. You have no work experience. But you also have a unique set of opportunities, including having by far the most time to work on this search. So students are the fourth and final archetype.

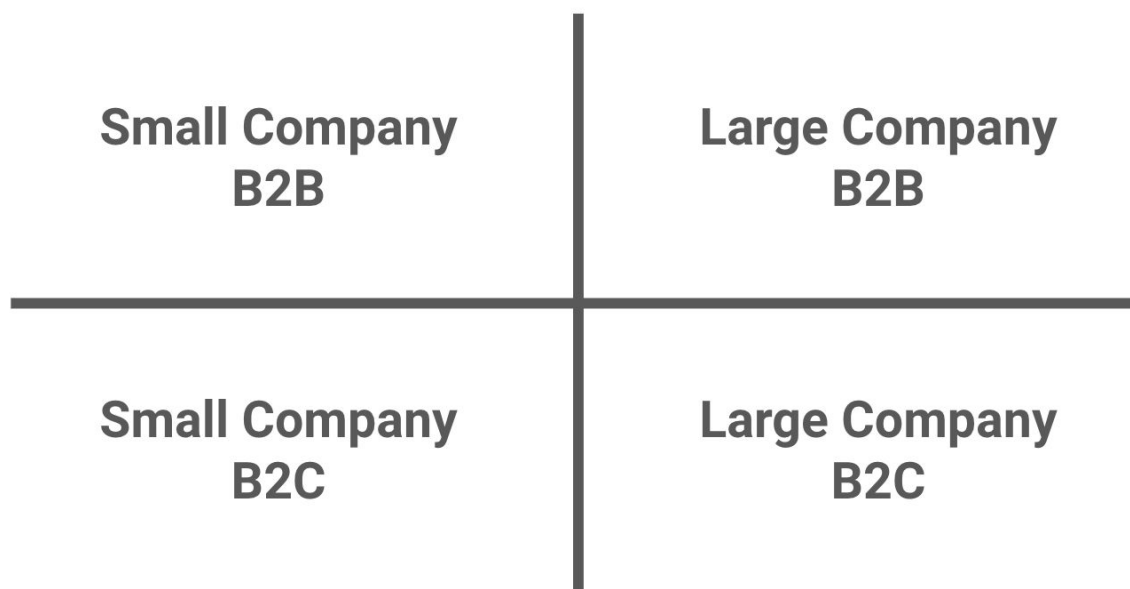
You are going to need to target an 'Associate Product Manager' position.

Step 2 - Target Job

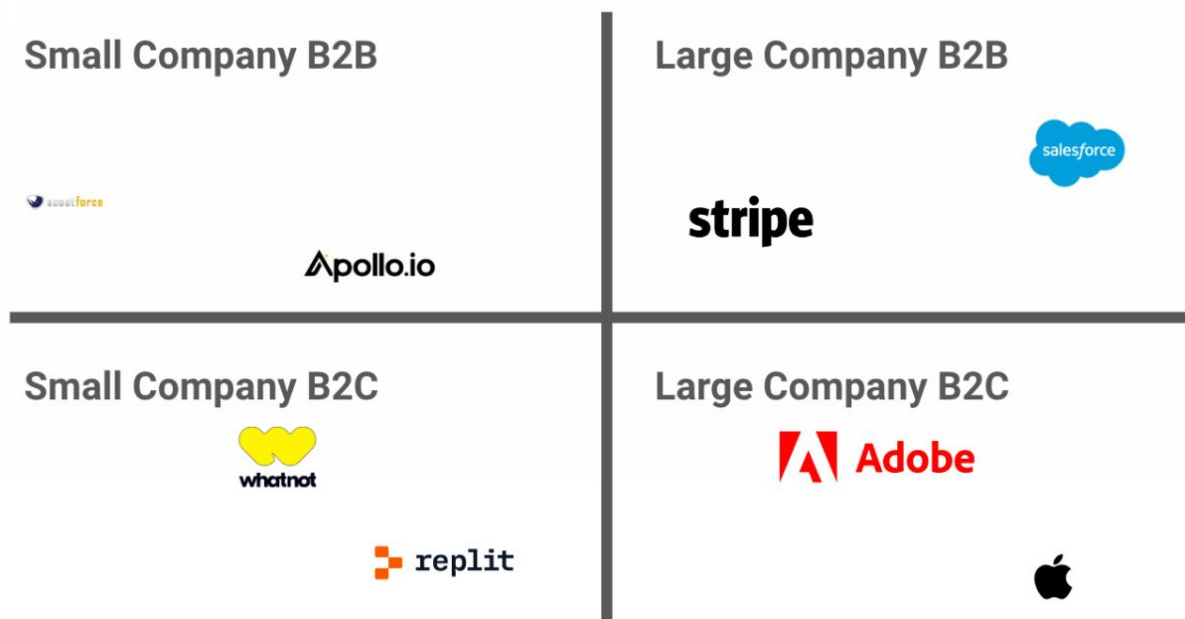
Now, you need to figure out your target job. Here is roughly where all the leveling advice I teased on the archetypes works out:

Experience	Target PM Job Level
Founder of successful company	VP/ Director
Worker in successful tech company	Current Level - 1
Everyone else:	
20-24 years of experience	Director of PM
17-20 years of experience	Group / Principal PM
13-16 years of experience	Lead PM
9-12 years of experience	Senior PM
5-8 years of experience	PM
0-4 years of experience	APM

Then, you need to think about the various types of PM jobs that exist. The 2x2 that I think is most relevant to think about is size of company and whether it's B2B or B2C. Think through where you are most qualified for and what you want to do. Then map it out:



Here's what that looks like with a couple examples in each group:



Especially with big companies like Microsoft, Amazon, and Google, they fit in the Large Company B2B and B2C categories depending on the product you work on.

Don't stress about getting pigeonholed. Choose a target, any target, to help guide your next steps. However, keep in mind the job you ultimately land may be in a different category than you initially targeted.

Step 3 - Work Backwards

The main activity in the self-assessment is to **think realistically about the gaps in your candidacy**. The hard thing to do here is to be realistic. That's why we've had so many formal steps and definitions so far.

What you need to do is to understand: what are the key skills that you need to have that you don't?

Do two things:

1. Frankly think of your main weaknesses as a PM candidate.

2. Pull up a job description of your target role, and start to work backwards.

Then converge towards your top 2-3 gaps.

Example

Let's work through an example to better illustrate this process. Let's say you are currently a Venture Capital (VC) associate with 3 years of experience in B2B SaaS, and 2 years of experience before that as an Investment Banker. You have a dual economics-CS degree from a Top 25 university in the US.

What are your strengths?

- CS Degree
- B2B SaaS knowledge
- 'Business side'

What are your gaps?

- Haven't worked on building a product at work
- No tech operator experience
- 'User side'

As a result, position-wise, you should target a B2B large company position.

Level-wise, you fall into the 'for everyone else' category. With 5 years of experience, you should pursue a PM position.

Here's why:

- B2B has a lot fewer experts in a position like you
- Big companies are going to be able to meet your compensation expectations

So you have the target role: Big Company B2B PM role. Now you can start to work backwards for what is needed for that position.

Let's hop over to a Google Cloud PM position to realistically evaluate your fit with the responsibilities:

JD Requirement	Gap
In this role, you will work cross-functionally to guide products from conception to launch by connecting the technical and business world	100% Gap - Haven't done at all
• You can break down complex problems into steps that drive product development • to help design and develop technologies that improve access to the world's information	100% Gap - Haven't done work of milestones, building PRDs.
• We're responsible for guiding products throughout the execution cycle, focusing specifically on analyzing, positioning, packaging, promoting, and tailoring our solutions to our users	10% Gap Only - Lack practical experience. But good at this from time in IB + VC
• As a Google Cloud Product Manager, you will drive product strategy and partner closely with cross-functional teams to define and deliver on the next phase of cloud services	30% Gap Only - Haven't worked with designers & engineers, but great with other groups
• Additionally, you will develop product solutions, Mergers and Acquisitions, go-to-market, and business relationships to execute on the market potential and benefits possible from applying Google's technologies in the enterprise market	30% Gap - Lack doing it as a PM, but have done in other roles

Combining the weaknesses we identified above and the gaps, we can identify the clear improvement areas in the long-term plan. Working with the cross-functional teams to ship features, and building out milestones are the top two gaps for you.

Now, it's time to build a long-term plan to flip those weaknesses into strengths.

2. Long-term plan

Your long-term plan is your secret weapon to becoming a PM. Most people only spend a few weeks in the pursuit, don't get interviews, flunk the ones they do, and then give up.

But you're in it to win it. It's the entire secret to my strategy. So let's break it down in detail - first, with principles for all; then, with specific advice for each archetype.

Principles for All

Principle 1: Internal transfer is best, but don't move for an internal transfer

The way most people get their start in PM is by an internal transfer - a study by Women in Product found it to be 52%. So consider it! It is by far the easiest way to become a PM. Of the 52 people I helped, fully 24 of them used the internal transfer technique.

But do not plan a 2-step process to become a PM. Don't move to a company and then try to become a PM there. You're much better off following the rest of my steps below instead and make the hop in one step.

If you are pursuing an internal transfer, here's how to succeed:

Take the results of your self-assessment and begin to work on a set of projects to demonstrate those strengths, and then apply for an internal transfer.

But don't be shy about sharing your intentions well before that. Let your manager, skip-level, product leaders, and PMs know your interest. Then spend extra time outside of your day job getting involved in product tasks. Also, as much as possible, steer your regular job to product tasks.

The more time you can spend with the product team, the more likely your internal transfer will be.

For everyone else, continue on with the principles:

Principle 2: Spend anywhere from 6-12 months

Making it into PM is hard. If you want to get in, it's better to spend 6-12 months developing your skills over spending 6 months interviewing to no success. As becoming a PM is a side gig to your regular thing, prioritizing is the name of the game.

And building your skills is the top priority. Not interviewing aimlessly.

Principle 3: Flip top 2-3 weaknesses into strengths

You should use this time to flip your weaknesses, and then apply what you've learned in real-world situations

It's truly crazy how much you can do in 6-12 months. Here are a couple examples for the most common gaps that I encountered while working with people for this book:

Prevalence	Gap	Top Way to Address
#1	Technical skills	Learn to code and make your own product
#2	Collaborating with designers and engineers	Start a mini-startup with designers and engineers
#3	Writing product documents	Write them for a product of your own
#4	Product sense	Read product newsletters and analyze products you use
#5	Speaking product	Listen to product podcasts and make friends with PMs
#6	Business sense	Study business and technology strategy courses
#7	Communicating with confidence	Join toastmasters and speak at several conferences

Here's why each of these is top way to address, and how to implement them:

Learn to code and make your own product

This is probably the *top advice* for every wannabe PM out there. Yes, I know it's hard! But with AI it's much easier than it was last year. And I was giving this advice last year. Pick a product idea - web or iOS is what I recommend - and begin building it.

But don't just be a developer. Use the coding to give you an appreciation for the technical aspects of building software. But focus the most on the product aspects of the job. Write out your product vision, strategy, and PRDs. Because you have to code the changes yourself, this will give you a great appreciation of finding high impact, low work projects that are the bread and butter of great PMs.

Start a mini-startup with designers and engineers

Probably one of the most powerful tactics, instead of making a solo product, which is mega hard, is to create a group product. A mini-startup where multiple people are working on a side gig is a great option to build out your product skills. Instead of being mega hard, it's just hard.

But you will still have real, on-the-ground experience of how to work with designers and engineers. Collaborating with them is very unique, and this will help flip that weakness into a strength.

Write product documents for a product of your own

This one is obvious if you've read the two above suggestions. Whether you are creating your own product or a group product, that is one of the single highest leverage things you can do to become a PM. It works.

And once you create that product, write a lot of docs.

Read product newsletters and analyze products you use

The top product newsletters are:

- The Product Compass by Pawel Huryn
- Lenny's Newsletter by Lenny Rachitsky
- Productify by Bandan Jot

Like this newsletter, going for the paid edition pays back in future earnings.

Then, analyze the products you use everyday. Develop your product sense by paying attention.

Listen to product podcasts and make friends with PMs

The top 3 product podcasts, in my mind, are:

- ProductTea by Leah Tharin
- Lenny's Podcast by Lenny Rachitsky
- The Product Podcast by Product School

But, honestly, if you're really going to be a PM, subscribe to a few more anyway. There are at least 30, and most of them are good.

As you begin to understand how PMs talk, start talking product with your PM friends! If you don't have PM friends, make some.

Study business and technology strategy courses

Flipping business sense from a weakness to a positive is by far the easiest. Do these four things:

1. Take the following courses (I would go paid but feel free to find a free version):

- Strategy
- Finance
- Accounting
- Economics

Most PMs don't need to go beyond the 101 knowledge that is in every top university's course material (go to a Harvard/Stanford 16

week, 10 hour a week course).

2. Subscribe to the following newsletters:

- How they Grow by Jaryd Hermann
- Stratechery by Ben Thompson
- First 1000 by Ali Aboueletha

3. Read the Wall Street Journal and the Economist.

4. Analyze the product features you use on a daily basis from a business lens.

These four actions will flip you from lacking business sense to having it. And that's generally the approach - do 4-5 things to flip a weakness.

Let's take our example from the self-assessment. Here are 5 ways to overcome each weakness:

Gap 1: Working with the cross-functional teams to ship features

5 Potential Solutions:

1. In her VC role, get operationally involved on the product side with a portfolio company
2. Do hackathons with her coding and designing friends
3. Building a mini-SaaS and hiring contract developers and designers
4. Starting a mini-startup side-project startup with friends
5. Working with her tech teams to redesign the VC website and jobs product for portfolio companies

These are all 5 ways she can go from having a gap to actually selling herself as one of the few with her background who do have the experience working with designers and developers to ship features.

Gap 2: Building out product docs and milestones

5 Potential Solutions:

1. Write product docs for 5 features for her top 5 companies and send them to people there
2. Write a doc for a product feature at her VC, like on the website or jobs board
3. Build her how product and write docs for it and build out milestones

Once you flip your weaknesses, your progression through the interview funnel will be much better.

Principle 4: Create the “Failed Simulation” Effect

Don't get too centered on your gaps. Yes, principle 2 matters, but it's not everything.

Everyone knows when you apply that you don't have PM experience. But what they're looking for is some wow factor related to your product qualifications.

Cal Newport, author of the best books on how to get into college, has a powerful concept that also applies to PM: the “Failed Simulation” Effect. I'll let him introduce it:

We feel that, with a high enough tolerance for pain, we too could be that busy. It's hard work. But it's not mysterious.

What happens, however, when presented with the story of a student who works with the U.N. and drafted a constitution for the Arab League? **Our simulation apparatus fails.** We don't know *how*, exactly, one becomes a player in major international organizations.

The effect of this failed simulation: **a sense of novelty and wonder.**

The basic idea of the failed simulation effect is: **accomplish something they don't know that they could do with a bunch of work.**

Here are three of the most achievable yet impressive failed simulation examples for wannabe PM candidates:

1. Code an app that is #1 on ProductHunt
2. Win a major hackathon
3. Write something featured on HackerNews

These massively impressive achievements can happen without massive time commitments (if you're focused).

A final piece of advice for everyone who wants to be a PM. Read Inspired by Marty Cagan, if you haven't yet. It as close to a seminal text as exists in the field.

Specific Advice by Archetype

Archetype 1: CS Degree

If you have a CS degree and are a developer now, you have even more power. Because you can volunteer to pick up PM work with your PM. There's no PM on the planet who wouldn't love a helping hand from his developers.

If you have a CS degree and are not currently a developer, reconnect with your technical side. And definitely build your own product. Just go back to your old CS class homework days and make something happen.

Archetype 2: Product Adjacent Experience

Many people coming from alternate backgrounds are strong in teamwork and EQ. Emphasize that in interviews! But, the long-term plan is about filling the gaps. To fill in the gaps, you have to put in the extra work.

Let's start with extra work for tech chops. You should start reading the top tech newsletters: Not Boring, The Generalist, and Stratechery. You should also start building things in code. Start with something simple you are passionate about.

Then, there is extra work for business knowledge. You should begin learning about the marketing, analytics, and user research concepts you are weakest on.

Finally, you should improve your product sense. Read product newsletters like Lenny's and mine. Then, go about asking yourself in normal life: Who is this product built for? What is the job to be done it has solved?

Archetype 3: Completely Irrelevant Degree & Experience

When you have a completely irrelevant degree and experience, it can feel like everything is a gap. You feel lost on what to prioritize. Don't prioritize everything!

Practice your PM skills by doing a prioritization exercise. The position you are in, everything in your life is a fun opportunity to pick up PM skills. Create a table of the gaps that you have, how much work it would take to overcome them, and how much impact each would have.

Let's take an example of a high school math teacher with a bachelors in Math. You're good with numbers and take interest in the stock market, so you know a decent amount about business. This might be what your prioritization looks like:

Gap	Work to Fix	Impact
Product Sense	Low	High
Collab w Design & Eng	Medium	High
Business Sense	Small	Medium
Technical Skills	High	Medium
Writing Product Docs	Low	Medium
Speaking Product	Low	Medium
Communicating with Confidence	High	Low

So, for this example, you could prioritize product sense and collaboration with design and engineering. The key is to only pick the top 2-3, not try to learn everything.

Archetype 4: Student

The ideal double major/program for PM is dual Business (Strategy) and Computer Science. (It's not a product management degree.) If you can only do one, do Computer Science.

If you're a junior or senior and the majors are already chosen, see what you can do to get a CS minor. I've even heard of seniors pulling it off with an extra summer class. A CS degree of some sorts is going to do wonders for your ability to succeed as a PM. If you don't understand how to build code and software, it's very hard to work well with engineers.

Aside from your school-work, find PM internships and side gigs ASAP. A good tactic is to e-mail 5 startups in your city with a product requirements document for a great feature idea. If they don't want to hire you, offer to work for free. What you'll learn in 6 months of work, and the experiences you'll get to talk about in your interview, are worth it.

You should hit the PM internship circuit in your country with the utmost determination. Find every company that hires PM internships, make a list, and follow the process in [Chapter 1](#).

3. Executing On Your Plan

Treat it like a product

Once you have your plan, doing the work is the hard part. This is another great opportunity to think like a product manager. I recommend writing up the milestones of your plan into a roadmap, and then track progress against that roadmap.

Many product teams use something like this stoplight methodology:

Milestones	Description	Target Date	Latest Date	Status
Milestone 1	Complete class on latest in Swift Programming	7/14		
Milestone 2	Build prototype of product and iterate	8/21		
Milestone 3	Marketing up and through launch	9/14		
Milestone 4	Write PRDs for followups and get feedback	9/28		
Milestone 5	Ship and analyze first two features	10/31	1/21	
Milestone 6	Ship and analyze next two features	12/14	3/1	Upcoming
Milestone 7	Begin applying to PM jobs	1/31	5/14	Upcoming

It often happens like this example. You create a plan to start applying for PM jobs in 7 months, but it ends taking up 10 months. That's fine.

Avoid the mistakes

I've coached so many through this that I've gotten to see the most common failure patterns. These include giving up half-way, spending way too long on the long-term plan, and getting lost doing work that isn't valuable.

Here's what I'd say to address these failure patterns:

1. **Make it a public commitment**
2. **Get accountability buddies**
3. **Don't worry if you fall off**
4. **Consistency > Ambition**

1. Make it a public commitment

Tell your family. Tell your friends. Tell your mentors. If you get all the momentum behind a plan like this, then you can actually execute over the long-term.

Strategic self-presentation only goes so far. You can tell yourself you want to become a PM. But studies show a public commitment helps make the difference.

2. Get accountability buddies

Remember as a kid when you were on a team with a shared mission? How much that motivated you? When you do things together, you can endure the ups and downs.

Find an accountability buddy. They don't even need to be in the same stage of life as you. But share your goal with them and have them hold you accountable.

3. Don't worry if (when) you fall off

Pursuing something like building your own product, creating a mini-startup, or learning to code is very difficult. There's a reason so few people do it.

So having a day or a week where you're not motivated by the end vision is almost inevitable. Consider it a needed vacation, like we do from work, and get going again.

4. Consistency > Ambition

A lot of people want to become a PM RIGHT NOW! That's okay in practice but difficult for 99% of people.

It's much better to consistently show up than think you're going to become a PM at Google in 3 months. Consistency wins.

4. Re-Designing Your LinkedIn & Resume

At this point, you have likely done a ton of work in your long-term plan to address your weaknesses. You've flipped them to strengths.

The next step: Prominently feature these on your resume and LinkedIn! We can't underestimate this enough. It's not enough to just do the work. You need to be able to sell it.

Many people I advised actually needed the most help at this step.

Step 1: Add Job

You want to list your effort as a side-job on your LinkedIn and Resume. Ideally the title is Product Manager. In the description you want to explain very clearly that it was a personal product or side project, and **actually link** to the product documents, like PRDs and strategy documents, you created.

Here's an example of what it should look like on your resume:

GoTech

Product Manager

Singapore

2022-2023

- Worked part-time on a startup featured in [TechCrunch](#) as the PM with 3 developers, 1 marketer, and 1 designer
- Led all product strategy efforts: [H1 strategy doc](#) helped us drive roadmap and ship 3 game-changing features
- Built PRDs to facilitate async collaboration as a side-project: [Boosting PRD](#) led to a feature that increased revenue 40%

It's very clear that this is a part-time gig, but it's actually going to be the thing they're going to be most interested in, because it's relevant Product Management experience. Just make sure you do a great job with those documents!

Step 2: Alter Your About Me

Now, it's time to tell the story about why you are a good PM candidate. Then, have fun writing that up into an essay.

The framework you should use is:

1. Have a quick summary and then a longer version
2. Focus the longer version on qualifying yourself and showing your value
3. Write with really engaging, well-edited copy

For your resume, I recommend just sticking with the summarized version at the top. LinkedIn is the space to go much longer.

Step 3: Fill other Gaps With Job Descriptions

The next thing to do is throw out your old resume bullet points. After looking at what you have added in your about me and for your part-time job, ask: what other gaps do I have in my PM candidacy?

Pull out stories from each of your other work experiences (or school projects/clubs) to highlight those skills.

At the end, your entire resume should speak to all the skills the PM job you want requires. I recommend putting a job description side-by-side to your resume and checking off each bullet point to make sure it is addressed.

Step 4: Create a New Headline for The New You

The last step, after all this work, is to re-invent your LinkedIn headline. You should add in product management for your side gig and have it align with your whole image.

This will finalize your branding as someone who is all-in on product management and ready to make the shift for the jobs you're about to apply to.

At this point, you don't even look like the old you who "didn't have the experience to become a PM."

Step 5: Customize For Your Top Choice Job

Then, I recommend updating your LinkedIn regularly, based on the top job you're applying for at that time. Let's say you're pursuing a large company, B2B role. Then customize for that:

- Add details and keywords around B2B and working with sales throughout your profile
- Make your headline something about B2B PM
- Add details about working with large cross-functional teams in your prior experience

Your resume can be frozen in place at that time for the specific job. Unfortunately, your LinkedIn will keep changing. But the time is well worth it. It helps you craft your story for interviews at the company down the line.

And that's it for customizing your LinkedIn & Resume. But how do you find the right jobs?

5. How to Find the Right Jobs

So, how do you go about getting the right job?

Generally, you should follow the process described in [Chapter 1](#). But from advising all these inexperienced folks for this book, I've found a couple things that are very specific to your process as someone without experience:

- Find the right manager
- Focus on a unique fit type
- The funnel is going to be rough
- There's no point in dropping resumes
- Let your work do the impressing, not your talking

Find the right manager

Some managers and companies are biased against people without a PM background. Other PMs and managers have a tradition of hiring PMs without PM experience. You're looking for that latter group.

Here's how you can do that extra level of research:

1. Use your network and ask people at companies that are hiring your target role
2. Apply for jobs on LinkedIn where the hiring manager is listed and research the people they have previously managed (these can be a bit hard to find - usually the trick is to figure out what team they lead and find the PMs on the team)
3. Ask your recruiter who the hiring manager is in the first call and then study the people they have hired

Focus on a unique fit type

You have something that makes you *you*. Maybe that's an aerospace engineering degree. Or maybe it's background as a high school

music teacher. Both of those can be translated well to specific types of PM companies.

Aerospace Engineering:

- Aerospace companies
- Engineering companies

High School Music:

- Education companies
- Music companies

You should prioritize roles in those types of companies. I call this your “unique fit type.”

But it’s not just your prior education or jobs that matter. In my experience, there are five types of unique fit:

- **Education fit:** what you studied
- **Company fit:** where you have worked
- **Role fit:** the roles you have worked in so far
- **Interest fit:** the types of things that you study in your off-time
- **Side project fit:** the areas your long-term projects or other side projects were in

Leverage at least one. Unique fit should help you prioritize what job you are applying to on a given day. If you are choosing between 5 jobs, for instance, pick the one with the strongest fit.

The funnel is going to be rough

If you start comparing your funnel to other people who have more experience, you’re going to feel unnecessarily despondent. PM is a guarded career. Accept that!

A not unusual funnel for people I worked with looked like this:

- 7 cold messages per job to get a referral

- 75% conversion from referral to interview
- 25% conversion from first to final round
- 25% conversion from final round to offer

If those are the expected values, you can expect to need to do something like:

- 116 cold messages
- 22 referrals
- 16 first rounds
- 4 final rounds
- 1 offer

And don't get despondent if your funnel is worse than that. Those people had my assistance.

Your funnel may just be broken at one step. Consult your accountability buddy and address the issues in that step.

There's no point in dropping resumes

Your default approach for all companies should be to apply via a referral. Your resume and lack of full-time PM experience is your biggest barrier. Once you're in the interviews, you can speak to your side-project more - and actually impress people.

Focus on crafting compelling cold messages that you can turn into referrals. Forget dropping resumes.

Let your work do the impressing, not your talking

Don't merely try to explain how you have good product sense. Instead, demonstrate it with specific examples from your long-term plan.

Before chats to get referrals, you should write out the 6-10 stories that answer the most skeptical parts of your candidacy. These should do a good job of selling the work you did in your long-term

plan. Structure the stories with the STAR (situation, task, action, result).

Video record yourself doing this and get critiques from friends. All that work needs to land well in your chats.

6. Interview Advice

Once you secure interviews, it can feel stressful as someone without PM experience. Generally, follow the advice in [Chapter 1](#). Here are the 6 most specific pieces of advice that apply to you and not everyone else:

- Your 'Tell Me About Yourself' Should Address All Concerns
- Be Confident with Your Stories
- Ask If They Have Any Concerns
- Represent Mature, Frank Confidence
- Be Extra Practiced
- Listen to Product Podcasts

Your 'Tell Me About Yourself' Should Address All Concerns

You should craft a 'Tell Me About Yourself' that is purpose-built to answer concerns they may have about your candidacy.

The wrong type of answer to this question is one where you generically and chronologically walk through your background.

The right type of answer starts with a bang, and then follows up with more relevant skills. It should be something like this:

I'm a creative and inspired PM. I spent the last 6 months building and shipping features to X.

This was along-side my day job where I demonstrated Z skills.

Before that, I did Y things that are relevant to product.

You want to go about 2 minutes. Don't go too long.

Be Confident with Your Stories

We already mentioned stories for the referral calls. You want to keep telling those stories in your interviews as well.

The big additional obstacle you'll face in interviews is follow-up questions. If someone starts to push on your product thinking or work, don't back down. These are crucial moments in any PM interview and signal how well you will do in the job.

Calmly and matter-of-factly defend the product work and preparation you did.

Ask If They Have Any Concerns

A pretty mind-bending opener to the “what questions do you have for me?” at the end of the interview is: “do you have any concerns about my candidacy?” If they list some, then you can use the remaining time to alleviate those concerns. If they don't, then you can feel a bit better about your interview performance.

If you don't think the interview went really well, it's a good question to ask.

Represent Mature, Frank Confidence

The strategy here is to present yourself as someone who has done extra work to prepare. You don't want to act like, just because you prepared part-time to become a PM, all of a sudden you already are a great full-time PM.

The full-time job is very different. So your interviewers are going to look for frankness and maturity when you talk about your background. Sell your skills, but don't oversell them.

Here's two examples:

- *Question:* How do you feel you would do at writing product documents?
- *Oversell:* I'm quite good at writing product documents. I have done a lot of it. I demonstrated this, for instance, when our side-startup had a crisis. I quickly whipped up a PRD that allowed us to solve the problem in a day.

- *Goodsell*: I think I would do really well. That's one of the things I focused on with my side startup. My colleagues said I did a great job when we had a crisis about X. The PRD I wrote helped us solve the problem in a day. That, combined with Z other writing experience from my past, makes me think I'd do a great job at your company.

Be Extra Practiced

The product sense and product design interviews are very specific to product management. Since you may not have experienced these types of interviews before, it's important to spend considerable time practicing.

The best way to practice for these interviews is to record yourself practicing videos on YouTube. Just pause before the respondent gives their answer, then compare your response to theirs. I also recommend a practice tool like RocketBlocks.

Listen to Product Podcasts

The last piece of advice for you is: you want to be able to speak product lingo (Chapter 7). I recommend subscribing to all of the product management podcasts you can find in your app of choice.

Listen to these when you work-out, when you drive, and make product lingo the way you think. This will help you speak like a PM in the interviews.

SECTION 2: The Learning Material

Chapter 5: Key PM Skills At Each Level

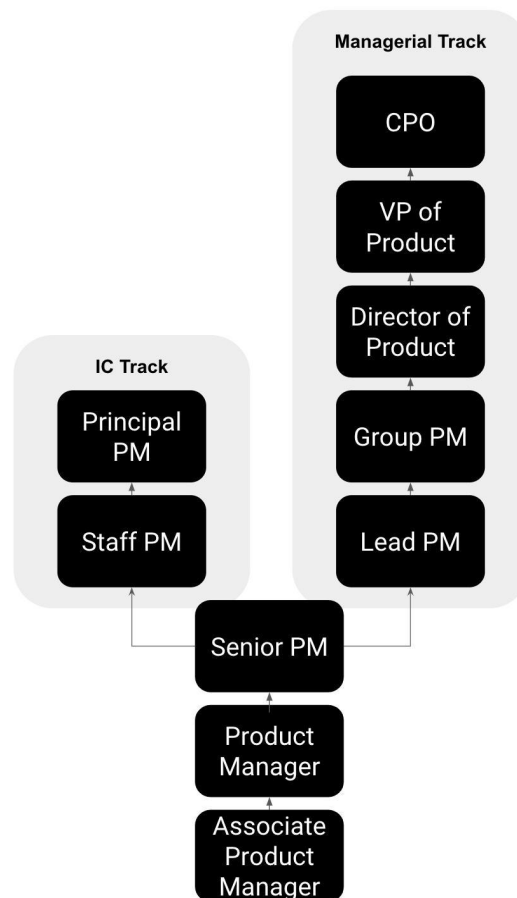
Product managers tend to be an ambitious bunch. It's rare to find a PM who doesn't want to grow in their career as fast as possible.

So, a question that's always top of mind is, "how can I move ahead in my career?"

The PM Career Ladder

There are about 5 or 9 steps depending on the track.

Lenny Rachitsky provided a great resource to the product community by collecting PM career ladders at 10s of companies in 2022. It can be distilled into two major tracks:



There's an IC track and a Managerial track. Not every company has the IC track, but more than half of the bigger one's do.

What exactly is needed at each stage?

Different Skills

As you move up this ladder, you need to master totally new skills. While there's tons of variation, there are two things to highlight.

The first is that **as you go up in the ladder, your ability to do the items below doesn't really decrease.**

It's difficult to effectively lead in product management if you don't have that base of being able to ship features, contextualize them versus past features, consider features versus other types of bets, etc.

The second takeaway is that **the big axis shift for you as you progress is from execution to vision.** If you're early in your PM career, as much as strategy and vision do matter, the value you are uniquely positioned to provide relative to others in the company is execution.

So, being able to ship features on time, with attention to detail, great design, and reach the goals they set out to is crucial. There's a reason that 'ship feature' sticky has big text. It's important at each and every level, and is the rock foundation upon which everything rests.

How might we further sharpen our pencils on this?

The Three Key Phases

A few years back, Fareed Mosavat (former Director of Product at Slack) and Casey Winters (former CPO at Eventbrite) identified that one stage is particularly tough - the jump from senior PM to product leader.

They called it the Product Leader Canyon. I couldn't agree more. Inspired by that, I think there might be one more jump to add - moving from a product leader to product executive. Because again, at this step, your aperture changes dramatically.

So we might say, really, there are three stages:

1. Progressing Seniority
2. The Product Leader Canyon
3. Manager to Executive

The progressions looks like this:

Chapter	Title	Ownership Area	Aperture
Progressing Seniority	Associate Product Manager	Enhancement	0-3 months
	Product Manager	Feature	0-6 months
	Senior Product Manager	Problem	0-9 months
Product Leader	Lead Product Manager	Problem area	0-12 months
	Group Product Manager	Group of problems	0-18 months
	Director of Product	Subset of product	3-24 months
Executive	VP of Product	Entire product	6-36 months
	CPO	Multiple products	6-36 months

As you move into the executive line, you are managing a whole product or multiple products. This makes your skill sets required far broader than just PM.

This Chapter

I'll walk through the “earned secret” tips I have gained from working through these three phases of the PM career.

Consider this the *unofficial companion* to the career ladder your company gave you. This is the advice for accelerating through your career ladder that you used to get at in the cafeteria before everyone went hybrid/ remote. We'll walk through:

- What are the **key “jumps”** from a PM role perspective?
- How does your **approach** need to **evolve** as you climb?
- How do you **learn these skills** before you need them?

Let's get into it.

The Phases of Your Product Career

What you need to do to succeed varies widely depending on what phase of your career you are in.

Phase 1 - Progressing Seniority

Associate Product Manager → Senior Product Manager

As you are moving up the PM career ladder in the early stages, your critical skills tend to be around **delivery**. To earn that promotion at each level you need to:

1. Have the product features you ship drive significant impact
2. Work well with IC engineers, designers, analysts, marketers, and legal & compliance counterparts
3. Have exquisite-quality product work products: PRDs, charters, roadmaps, etc.

Phase 2 - The Product Leader Canyon

Lead Product Manager → Director of Product

As you move from Senior Product Manager to actually managing PMs, many PMs hit a brick wall. They were great individual contributors in the product world. They could drive features to success and make a significant impact on the business.

But it all falls apart when they become managers. What's driving the failure?

1. You are no longer judged on your own success
2. Influencing up and across becomes a larger portion of your time
3. You have barely any free time outside of meetings
4. It's not just product design, but org design
5. You have to be able to provide input with less context

Phase 3 - Manager to Executive

VP of Product → Chief Product Officer

What makes phase 3 so different from phase 2?

1. Your aperture expands beyond product management
2. You have to be able to anticipate investable areas far in advance
3. You have to prioritize the top 1/10th of what you could be doing

Tried and True Techniques to Progress

I've boiled down all the millions of tips we could provide into 5 key techniques you should focus on to make quick progress through the career ladder:

Technique 1 - Understand Your Shape

Every PM I've ever worked with has improvement opportunity areas. The failure pattern is to get overly bogged down in your own weaknesses and let you hold them back.

Instead, understand your shape. Ravi Mehta, former CPO at Tinder, has a wonderful breakdown of the different shapes of product managers. The key takeaway? Double down on your shape. We'd add two vectors to think about this:

1. Try to **move into the subset of the PM jobs** that require competencies within your skillset
2. Attempt to **job sculpt** to spend more time in your centers of excellent competency rather than those where you are weak

Technique 2 - Operate across departments and teams

While I've defined the aperture for all the PM levels fairly narrowly above, there's great value in constantly flexing outside your core working group.

Even APMs should be making an effort to work with colleagues from marketing, sales, and other important disciplines in the company. The amount of that focus just increases exponentially as you advance.

Ultimately, every group in the company wants to have a say in product decisions. So getting out there and being proactive might feel unusual since it's not reciprocated, but that's the job of the PM.

Technique 3 - Make friends with others at and above your level

Advice from people within your company is invaluable. Consider making friends and talking about career topics as part of your job. Make time for it.

This seems like common-sense, since you're subscribed to a paid newsletter about product management. But, many of us find ourselves too busy by the end of our actual product work to do things like this.

But the easiest way to understand the insider tips on how to progress at your company is to talk to the insiders.

Technique 4 - “Plus one” the product craft

Your company has a certain standard for product documents, product work, prioritization, and the rest. If you can “plus one” that, you're always putting yourself in a good position.

It's tough to do at scale. You don't want to burn yourself out. Instead, try to constrain the scope of what you do very high quality work on. Go for a few very high quality things over many okay quality things.

Technique 5 - Drive focus and prioritization

It doesn't matter if you're an APM or a VP of product, there are a myriad of things you could be doing with your time. This is especially true if you're trying to “plus one” the product craft, per Technique 4. So, having an effective strategy to drive focus and prioritization is at the center of effectiveness.

The Leveraged-Neutral-Overhead (LNO) framework from Shreyas Doshi is one really good way to break down the prioritization problem as a PM. The core of it is to really focus on the 10x tasks.

As a PM, you have to get good at saying no, delegating, or simply not doing things. Picking the right spots and communicating it to your teams is the trick.

Different Operating Models

As I previewed at the top, world-over there are many different operating models for PMs. The trick is to *recognize* your situation, which is easier said than done. Many PMs make the mistake of thinking they are in one situation, but they are actually in another.

So, I'll try to typify for you the three most common operating models. Once you recognize your situation, how to optimize is relatively self-explanatory.

Model 1 - Leans to feature delivery (~40%)

The feature delivery model is common in companies where PMs tend to be considered in the "Tech" org - and technology is not the company's core business. You know you're in this situation if the main things people discuss are "launches." Instead of focusing on the impact of features, people celebrate their happening.

In this case, PMs should index less on business impact and more on feature craft. Deliver those features in the most orderly and high volume fashion possible. Create the coolest features that get the most senior internal stakeholders excited, versus driving a first principles process to drive maximum business ROI.

Model 2 - Leans to metrics movement (~30%)

The model of moving certain product metrics has become popularized with the rise of OKRs. You know you're in this situation if the main focus for PMs is moving engagement metrics like app downloads, monthly active users, or something like that.

In this case, these PMs need to focus on building really strong mental models for how to move the metrics in their control. They should carve out lots of time with their analytics counterparts and plan to be in the data a lot themselves.

Model 3 - Leans to Business ROI (~30%)

The business ROI model has been popularized recently by companies like Amazon. In this model, PMs are heavily measured upon the financial success of the product. You know you're in this situation if your leaders are held accountable for revenue and profit metrics.

In this case, it's critical to triple-click into operating across teams and disciplines. The fact is, product isn't always the biggest driver of revenue and profit. As a result, it's critical to spend time with the teams that actually drive the metrics (eg, sales or marketing).

Other Dimensions of Complexity

Beyond the core operating model, there are a couple other giant contextual differences you should account for as you build your plan to rise.

B2C vs B2B

B2B and B2C PM are different beasts. In B2C, experimentation, rolling back features, and the like are common practice. In B2B, it's much more common to have to do alpha-beta-GA rollouts, have high confidence in things shipped to enterprise, and find the Sales team has a large say in your roadmap.

There are three main ways you should make sure to manifest this in your work to progress fast:

1. **Identify the relevant stakeholders:** Focus on the stakeholders that matter based on your context. If you go into B2B PM and don't build a great relationship with sales, you're bound to fail.
2. **Focus what you learn:** Consider focusing your attention and reading on companies within your space; for B2C, focus on consumer products; for B2B, focus on business products. Pay attention to what really makes the products succeed within your domain.
3. **Consider specializing:** Doing both B2C and B2B is fun and great. We both have. But driving focus can help you progress a bit faster, if that's your goal. B2B companies, in particular, tend to value B2B experience.

Inbound vs outbound

The traditional inbound product management discipline is probably what 70-80% of product managers focus on. This is working with developers and designers to drive forward product feature success.

But about 20-30% of product managers live in an outbound world. They own many of the tasks that might be called product marketing or just marketing. These can include SEO optimization, sales materials, marketing copy, and more.

If you find yourself in an outbound role, become a student of marketing. There is a lot of content from those in the marketing world to learn from.

Common Anti-patterns

Now that we've gone through the key techniques to progress and their dimensions of complexity, what about what NOT to do?

Anti-pattern 1: Getting too focused on the ladder

Career ladder is really a misnomer. I've used the term to title the piece because it's the common term of art, but careers tend to be more path-like than ladder-like.

Take Aakash's career, for example: from Director of Product at thredUP to Senior Product Manager at Google. Sometimes, you take a step back in one area to continue to advance your skills and improve.

Anti-pattern 2 - Being limited by direct cross-functional staff mapping

Before product management, many of us had to think about our jobs in terms of the resources we have at our disposal. And once you're plopped into a product role, it's easy to think the same about our cross-functional counterparts.

Only have 4 engineers, with 3 focused on the backend? "Okay, we'll focus on backend changes" is the natural inclination. But the easy way out is not the right way in this case. Reframe the job to think that you should actually input on what staffing makes sense - at every level.

If you need to have a mix of more frontend engineers, make the case. Leaders are more open than some folks may realize to change for the sake of impact.

Anti-pattern 3: Doing what worked at the prior level

This is general career advice that could be said for any job, but it's doubly true for product managers. In fact, it's probably the most common failure pattern of PMs who are trying to progress in their careers: doing what worked at the prior level.

Even within the phases we've already highlighted, the job is constantly changing. So it's better to re-evaluate at each stage what you should be focusing your time on.

For example: whereas junior PMs should be spending tons of time with their engineering counterparts, that percentage tends to decrease dramatically as you progress.

Anti-pattern 4: Bringing your old job with you

Product is not a standard practice like engineering, design, analytics, legal, or most of the other functions PMs work with.

Whereas an engineer can easily port over their programming skills to a new job, don't assume your product management skills are quite as portable. Trying to force fit the processes, documents, presentations, and features from your old job is a very common failure pattern.

Don't be the guy or gal bringing over their old roadmap.

Chapter 6: The History of PM

Have you ever told a relative what you do, only to receive a puzzled look? Product management, as a field, is new. Our parent's generation didn't even consider it. Your average non-techie has no idea what it is even about.

How did Product Management as a job come to be? Has it always been around, but in different forms? As product managers, it would be great to understand the history of our profession. Then, we could help answer these questions we commonly get at holiday dinner tables with relatives.

Yet, when I asked my Product Management friends about it a few years ago, no one had a clue. There is shockingly little known broadly about how Product Management came to be.

This set off my years-long obsession with the history of the profession. I thought it would be easy to find a coherent history. But, everything I have read has several obvious flaws and gaps. It seems like writers are rewriting the same bad base article over and over.

The apocryphal story is that it starts with, "brand men," at Procter & Gamble. Neil McElroy, who never worked as a PM, is called the father of the profession. Excuse me, what?

Are these articles written by PMs or content people? Brand men are not PMs. Plus, obviously PM-type people in consumer goods companies existed in companies hundreds of years before. So, that is just a horrible starting place that makes no sense.

Toyota is also always thrown in there. Sure, the Toyota Product System and Kanban are interesting. But they have nothing to do with PM.

Instead of rewriting the same garbage about consumer goods brand men, I have been reading through troves of different books and old

articles to figure out the actual phases of PM's progression. Now, I think I am about ready to share an interpretation.

Strap in as we explore, "Product Management: A Brief History." I promise my interpretation of the phases of PM's history is quite different from what has been written before. Most people have not even broken out the last 5 phases I do, lumping them into one.

Pre-Role

If you want to get philosophical, the whole of human history is product management. The architects of the pyramids invented several new products, like the spiral ramps that allowed the teams of slave laborers to transport the enormous stone blocks.

That was 4,500 years ago. Before that, the architects of Stonehenge devised their own techniques to move 25 ton rocks 150 miles.

In some ways, the people calling the shots for the laborers at Stonehenge and Pyramids were like product managers. In similar ways, architects or project managers have existed across large projects for millenia.

Generally, they were accountable to a king who ordered the construction, and they had to influence the actual builders to build the far out vision incrementally in the fashion they recommend.

This mirrors the life of a product manager. Accountable to the CEO, they have to influence the engineers on a course of incremental actions to reach the ultimate product vision.

But these people went by other names. They did not develop software with the title product management. For this piece, I am focused on the history of folks who were known as product managers. That is, people developing technology products with the title product manager.

To get there, first we have to make a pit stop in the realm of consumer goods.

In 1931, Neil H. McElroy was a junior executive at Procter & Gamble. By that time, nearly 100 years old, the company was amongst the most culturally significant in the US. Neil, a future Secretary of State, was working day in, day out with the future leaders of the country.

Merely getting in, let alone rising to the level of executive, was akin to rising to executive level at Apple, Microsoft, or Google today. It was a role with a lot of pressure and visibility. The substantial pay led to substantial expectations.

The young executive was tasked with managing the Camay brand of soap. Introduced six years earlier as, “The soap for beautiful women,” the brand was growing but leadership wanted to see a new trajectory of sustainable growth.

When Neil got there, the team had many ideas of what to do. Neil had a different perspective. A traveled man, Neil had gotten an up close look at rival consumer products company Unilever in Europe.

Neil zeroed in on building out the promotions department of the Camay brand. He wanted someone to think about the brand, what customers want, and how to position properly. To sharpen his thinking and drum up support for the expensive headcount, he drafted a now famous memo.

In it, he defines several of the most important tenets of promotions management by “brand men.” It is a role where outcomes matter. Up until that point, much of the work for wrappers, promotions, advertising, and brand was managed by the sales department. Neil wanted that responsibility broken out to a new role.

But, what do brand men have to do with product managers? Well, it turns out Neil was not just a future Secretary of State, he was also an advisor to the founders of Hewlett Packard: Bill Hewlett and Dave Packard. And that is where product managers actually show up for the first time, and our history begins.

Part 1 - Hewlett Packard

As Procter & Gamble was an icon in consumer packaged goods, Hewlett Packard was an icon in technology for the half century between 1940 and 1990. It sustained unbroken 20%+ year-on-year growth.

One of the early innovations at Hewlett Packard was actually creating the product manager role. At some point in the 1940s, product manager became a title. Influenced by the success of Brand Men at Procter & Gamble, on the advice of Neil, Bill and Dave consistently supported the product management function throughout its history.

Like brand men took a responsibility from the sales department at Procter, the product managers took a responsibility from the engineering managers at Hewlett. Product managers were tasked with decision making for the product engineering teams, staying close to the customer, and advocating for the voice of the user internally.

Before product managers, the role was a mix of the CEO, the executives managing a division, and the engineering managers. Suddenly, a product manager was inserted, accountable to the CEO and executives, enabling the engineering managers to focus on the technical components of building a product.

While technology had been built for millenia with people telling workers what to do, for the first time, product managers were inserted into the scene who had to influence without authority.

In this phase, the profession remained fairly small. Outside of the consumer goods player's "brand men," and Hewlett Packard, the profession was fairly contained.

Part 2 - SCRUM Comes

In 1986, the Harvard Business Review published, “The New Product Development Game.” The article by Hirotaka Takeuchi and Ikujiro Nonaka reads as well today as it did then. In it, they outline fast and flexible processes that are a better way to do product development.

Using case studies of several of the most successful companies of the era - including Hewlett Packard - they outline a counter-intuitive way to build products. It has 5 main components:

1. **Built-in instability:** the teams pursue broad and challenging goals, but have a wide measure of freedom to accomplish them
2. **Self-organizing project teams:** the teams are autonomous, but since they are motivated by the same things, assemble on their own cross-functionally
3. **Overlapping development phases:** versus the Rugby method of slowly moving down the field, several processes close together parallelize
4. **Multilearning:** when failures or successes happen, they are shared for learning at individual, group, and company levels to foster cross-functional understanding
5. **Subtle control:** who is on teams, the work environment, evaluation, and celebration of failures are used as deliberate systems to encourage desired behavior

The article would prove to be very influential. Scrum would grow slowly in the beginning years, but the curve was exponential. Eventually, SCRUM grew to be the upper case, even though it is not an acronym and not used as such by Takeuchi and Nonaka.

In 1987, Ken Schwaber published “SCRUM Development Process.” He focused on how it is hard to plan, estimate, and successfully complete work, especially software. To deliberately move further

from overlapping, yet still waterfall spirit of Takeuchi and Nonaka, he described a new SCRUM process that includes iteration.

Scrum Methodology

■ Pregame

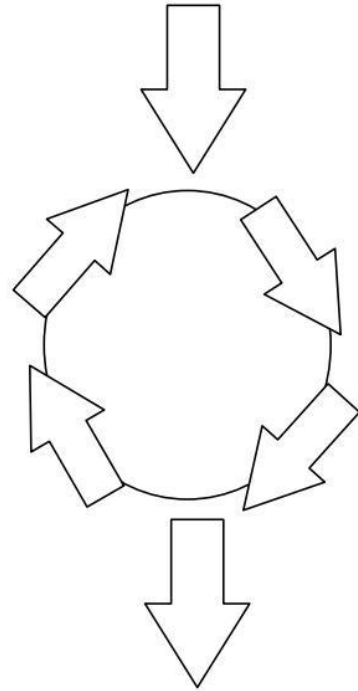
- Planning
- System Architecture/High Level Design

■ Game

- Sprints (Concurrent Engineering)
- Develop (Analysis, Design, Develop)
- Wrap
- Review
- Adjust

■ Postgame

- Closure



The 1987 SCRUM method.

The process caught on majorly. Buoyed by the writings of Jeff Sutherland, Ken Schwabeber and others, SCRUM would continue to grow in momentum over the years. The hockey-stick-like ascent snuck up on many software development professionals.

However, the role of Product Owner (PO) that many of you would expect actually did not formally get added until 2002. Instead, SCRUM was mainly practiced by developers. Engineering managers often played the roles of Scrum master and product owner.

At the same time SCRUM was rising, before the PO role was added, the 90s were happening. Tech companies were proliferating, and so, too, was product management.

Part 3 - Dot Com Era

With the rise of software, startups and venture capital extended beyond semiconductor investments and into bits. Suddenly, much smaller teams could make a much larger impact. They did not need to build an expensive manufacturing facility.

The internet vastly accelerated this process, and there was a cambrian explosion of new technology companies. With the internet, a startup at most needed to hire a few programmers and host a few servers.

Product management had a fairly mixed reputation at this point. Startups featuring early employees who had seen the role at Hewlett Packard tended to implement it.

Even at those companies however, it was not an early role hired at most companies. Those that did hire a PM tended to do so only for their biggest products. It was a later role, used to accelerate customer advocacy to engineering teams and create an accountable owner for ever-increasing executive and marketing teams.

As Rafayel Mkrtchyan notes, the big advocates at this time were companies like Intuit. Intuit was founded by a former Procter & Gamble brand man, in fact, Scott Cook. With their first product Quicken, they created a home finance software product than an intelligent stay-at-home mom could build.

It was a classically PM-build product. They identified a specific user segment, understood her user needs, and designed a product for her. Then they iterated along the way. Intuit's customer-centricity put a deep imprint on the product management career, and helped increase the number of the people with the title.

With the success of Intuit and HP, some companies were hiring PMs. But it was not a part of every software development team quite yet.

In addition, where it was practiced, it was not uniform. Companies like Microsoft built out their Program Management functions. These were product managers who were expected to do some technical program management. It was their own twist.

Part 4 - The Agile Revolution

In 2001, the Agile Manifesto was published by 17 authors - including SCRUM founder Jeff Sutherland. It ended up having even more impact than SCRUM. Suddenly, people on product teams - engineers, designers, product managers where they existed - were asked to build software differently.

It worth thinking about just how many things changed:

- Instead of waterfall, teams were asked for early and continuous delivery of valuable software.
- Instead of silos, business people and developers were encouraged to work together
- Instead of processes, projects were to be built around motivated individuals
- Instead of complex tools, simplicity became essential
- Instead of hierarchical decision-making, teams were to be self-organizing
- Instead of quarterly planning, reflection and tuning were to become regular

Despite SCRUM existing, most companies still had most, if not all of the, “instead of.” Agile helped change that. This had profound impacts on the lives of software development teams.

Those software development teams without product managers felt the pinch on their bandwidth. They were being asked to do a lot more. Instead of doing so, they hired someone. This led to continued uptick in the profession’s commonality.

In addition, for those product managers who were already practicing the profession, Agile profoundly changed the relationship between engineers and product managers. The relationship shifted, as Martin Erikson describes it, from adversarial to collaborative. It helped get rid of the barrier between research, specification, and development

phases of a project. Overall, it permeated the practices of what a product manager does into the software development process.

Agile, like SCRUM, would continue a sneaky, exponential growth trajectory. In 2007, the book, "Scaling Software Agility," served to introduce the Scaled Agile Framework (Safe). Safe ended up massively expanding the reach of SCRUM into government software development.

So while Agile started in 2001, it has been a consistent driver of the increase in product owners and product managers throughout software development teams globally. But, it was not the only important driver in that time period.

Part 5 - Google's Embrace

In the late 90s, Google did not have any product people. Larry and Sergey, along with the engineers, played the role. So, when Larry and Sergey hired Marissa Mayer as an engineer, they expected her to focus on code. But, quickly Marissa showed she was good at a lot more than code. Customer insight, attention to detail, and roadmapping were all skills of hers.

This landed her a promotion to Product Manager, a newly created role at the company. She “accidentally” became a PM. But, everyone on the team saw her value. When she wanted to hire more people to her team, she was supported.

But, Google was not a trillion dollar company in 2002. A small startup in Palo Alto, lesser known than the much bigger Yahoo, Marissa was looking for a way to attract the best young talent. Stanford grads like herself, Larry, and Sergey had all the options in the world.

Looking to brand the product manager job for new Stanford grads, Marissa created the Google Associate Product Manager (APM) program. The now-legendary program would go on to not only recruit, but also nurture, many of the most important next generation of product managers.

Over the years, these product managers have helped set the tone for the entire product management industry. Plus, their lucrative impact and extensive post-APM impact have made the role the envy of Silicon Valley aspirants around the world.

More generally, since 2002, Google has been a key driving force in the rise of product management. The company has regularly hired and paid the best, given them platforms at conferences and on blogs, and, with Sundar Pichai, elevated one to CEO. Like HP and Intuit before it, Google has been critically important to the rise of PM.

Part 6 - Product Takes a Seat

With forces like SCRUM, Agile, as well as companies like Intuit and Google, product management would continue its rise in the mid 2000s. Up until this point, the “primary job” of a product manager was considered to be writing product requirements documents (PRD). Marty Cagan was most well-known for his 27-page piece, “How to Write a Good PRD.”

But as product managers grew more ubiquitous, the notion of product managers as “CEOs of their product,” began to become ubiquitous throughout Silicon Valley. This increased the mystique and impact of the profession.

With important PMs like Marissa Mayer calling the shots, product management stopped reporting up through marketing or engineering. Instead, product became its own dedicated function that reports to the CEO. This was a profound shift for the function, one that I have not even seen written about anywhere else - which is a huge disservice to those histories.

In 2006, things were already changing. Marty wrote, “Revisiting the Product Spec.” With Marty, so did the rest of the profession follow. Product became a strategic role that helps define how to take advantage of things like SCRUM and agile to develop software best. Instead of taking directions from execs, PMs began to own vision and strategy. The shift began from output to outcomes.

Part 7 - Big Tech Darling

As Product Management became a more important role, it proliferated across the most important tech companies. Through the mid-2000's, virtually every technology company began to establish PM groups. Amazon, Netflix, and the rest of the big tech names hired and empowered PMs.

In 2007, the gospel of product management began to spread further with Dan Olsen's publishing of *The Lean Product Playbook*. With the demands of discovery and iteration, startups hired more PMs.

Then, in 2008, Marty published *Inspired*. The book provided a practical handbook for PMs and companies across the world. It went through vision, strategy, discovery, and management. This helped the big tech PMs, in particular, succeed.

Then, those big tech names massively succeeded. FAANG became an acronym. Some PMs, like Sundar Pichai, rose to become CEOs and billionaires.

With that success, PMs continued to gain more prominence. PM began growing like a weed outside the big tech companies as well. Everyone wanted to be like those tech names.

Part 8 - Here Come the Schools

As Product Management grew to become millions of practitioners, the schools arrived. It is easy to ignore these schools. But Product Management, up until recently, had no certification or degree. It was a job, unlike engineering or design, that people from many different backgrounds came into.

In 2014, Carlos Gonzalez founded Product School. It would be easy to dismiss a school, but the program has actually been pivotal for the PM profession. When you search google Trends for PM, for instance, it is one of the top topics.

The company recently raised a \$25M financing last year to cap off 7 years of pushing the profession forward. Its blog posts, podcasts, and highly attended talks (pre-Covid) have helped push the theory-crafting and professionalism of the job forward.

Theory-crafting would further be pushed by Brian Balfour and the Reforge team. Since its founding in 2016, and addition of PM courses a few years later, Reforge has push forward the thinking of what a PM should be. Bringing in strong product leaders from like Keya Patel of Dropbox, it has been setting the standard for topics like PM specialization.

Where product school has found its niche mostly for early career folks, Reforge has excelled for middle career folks. With a fresh \$21M in funding and new courses like Mastering Product Management, it continues to help PMs make more impact.

But certifications and part time programs only do so much. In the end, degrees mean a lot. In 2017, Carnegie Mellon began offering the first degree program from a highly reputable technical school, its Masters in Product Management. Programs like CMU's are pushing forward interest, and qualified candidates, into product management.

With all the new talent, there has been a thirst for PM content. In 2019, Lenny Rachitsky launched his newsletter. Now over 85,000 subscribers, his newsletter has help spread the gospel of PM. Recently, hundreds of thousands of PMs have been hired in India, as the startup tech scene explodes there. Many of those newly christened PMs read Lenny.

Part 9 - Today: The Hottest New Job

This all brings us to today. We live in the, “Golden Age of Product Management.” PM is the hottest job for MBAs at top schools. PMs are compensated well, and many of the top grads want to become one.

With the growth of the profession, we have finally seen tools even being built. Software like UserVoice, pendo.io, productboard and more are purpose-built for PMs.

The whole ecosystem is reaching a crescendo. People like myself are jumping into the content creation game to serve the growing audience.

By some estimates, there is one product manager for every seven software developers in the world today. Using that ratio, there are about 4 million practicing PMs in the world today.

Part 10 - Future of Product Management

So, where do we go from here? Predictions are hard, and experts tend to be bad at them. As a PM of 16 years, I have my biases. I saw the era of spec focus. I have seen the growth of strategy and vision. Now, I am managing a team of PMs.

I have three predictions for where PM will go:

1. **More people will become software developers, resulting in more PMs.** I expect the 7:1 ratio, roughly, to continue. In a future world where blue-collar workers are coders, I expect hundreds of millions of developers, and tens of millions of PMs.
2. **Product management will become a widely known profession.** Today, only techies know Sundar Pichai was a product manager. When I tell my home contractor, I am a PM, he has no idea what I am talking about. But in a few decades, everyone will know.
3. **PMs will grow in impact.** Many PMs are still locked into a world where they are held accountable for outcomes, but they have no authority. This has made project management the easy path. I think, in the future, PMs will be more strategic and impactful.

Chapter 7: The Languages of PM

Product managers need to be multi-lingual.

If you do not speak the language of the function you are interacting with, everything becomes more difficult. Conversations take longer. Alignment becomes tedious, if it ever happens. Stakeholders come back after a feature is launched with a list of complaints.

The best PMs prevent these messes by speaking in the language of their stakeholders. My friend Aatir Abdul Rauf calls these, “the languages of Product Management.” From my point of view, it is one of the most powerful concepts in product management today.

The worst product managers forever speak, “**Producti.**” They just speak in the language of PMs: terms like jobs to be done, user needs, empowered teams, putting the problem first, and product sense. While stakeholders and cross-functional partners have a sense of what these terms mean, they lack the deep neural net of context tied to each term product folks do.

The job of a Product Manager involves a lot of collaboration with these folks. Just like people from different countries and nationalities are most comfortable conversing in their native tongue, each department of a company prefers their own parlance. The most effective PMs pick up on this jargon quickly, to become more relatable and effective.

So, I am very excited to present a deep dive into the six most important languages for PMs to speak - Corporatish, Marketindi, Salesian, Designese, Techugu, and Analytian. The goal is to help you speak at least 10% better with your stakeholders. Let’s get into it.

The Languages to Speak

Corporatish

One of a PM's most important relationships is with the company's leadership. Leadership tends to speak a language known as, "Corporatish." This is very different from Producti.

It's not that leadership doesn't care about the state of the product. However, once the overarching goal is laid out in terms of a business metric like revenue, profitability, or market share, they need to know how product efforts translate into achieving them.

When talking with leadership, product managers need to communicate the health of the product and business metrics using terms that they care about and understand. It also helps, as a product manager, to be able to understand the lingo these folks use.

Key Terms

B2B: MRR, ARR, Logo/ Revenue Churn

B2C: Lifetime Value, Cost of Acquisition, Payback Period, Average Revenue per User

Both: Gross & Net Margin

Generally, the terms to know to speak Corporatish depend on whether your role is in B2B or B2C.

In either case, leadership is going to be interested in metrics around revenue.

MRR is monthly recurring revenue. This is the sum total of revenue the business is getting, stated on a monthly basis.

ARR is annual recurring revenue. It is the annualized version of MRR. It's not just leadership who cares about ARR and MRR.

Everyone from financial analysts to sales are always keeping a close eye on how ARR and MRR are tracking. As a result, the best PMs generally try to keep a close pulse on these rates, and, where they can, measure the impact of their features on them.

For subscription-based products especially, deciding which product features go in each package tier can have an impact on both MRR and ARR.

Thus, a conversation with leadership could look like:

“We’re beefing up our Pro tier with a workflow editor making it a more compelling proposition. We feel this will further motivate our Freemium users to upgrade, hence, bumping up our MRR.”

Logo Churn is the fraction of customers lost during the period over the number of customers at the start of the period. It is sometimes called customer churn. Leadership loves features that reduce customer churn.

The corollary to logo churn is **revenue churn**. Revenue churn is the fraction of revenue lost in the period, over the revenue at the start. Many think about a metric like ARR in the context of new revenue, existing revenue, and churned revenue.

Revenue churn biases to the biggest customers. As a result, many companies pay attention to both types of churn. But, all things equal, most companies care more about revenue churn than logo churn.

So, it’s very possible for a product to incur low logo churn in a month (say lost 1 customer out of 100) but suffer massive revenue churn if that “lost customer” was a majority revenue contributor.

Conversation Byte:

“We need to open a free trial for our product. This will help us capture more customers in the long run. Diversifying will help us reduce our logo churn. If we don’t work on this, then with just a

couple of high-paying clients on board, we'll always remain at risk of experiencing high revenue churn."

Lifetime value is the total revenue a customer will generate for the business. Leaders often want to compare this to the **cost of acquisition**, which is the total cost it takes for the business to acquire the customer.

Leadership is usually playing a delicate balance between these two metrics. They may even measure the **payback period**, which is the amount of time it takes for the lifetime value to exceed the cost of acquisition.

Conversation Byte:

"Since competition is driving our cost of acquisition up, we need to drive up our lifetime value. To do that I propose, we focus on improving our onboarding experience & prioritize personalized customer service. On the side, we should consider increase our prices in the medium run. It will help shorten our payback period."

Average revenue per user (ARPU) represents the blended rate of revenue you receive for an active customer. It is similar to lifetime value, however, it is usually calculated on a specific timeframe for financial statements, like the average revenue per user over a year. Some consider ARPU an overall business health metric, while LTV a user profitability metric.

Conversation Byte:

"By launching grocery delivery service alongside ride hailing, we'll be able to cross-sell to our existing customers. This will kickstart adoption and also lift our ARPU."

The final concept in Corporatish to understand as a PM is margins. There are two important one's to understand. First, **gross margin**. Gross margin reflects the profit ratio leftover after accounting for cost of goods sold. It does not account for things like the sales team, or

the product team. These usually fall in Sales/General/ Administrative Expenses, and Research & Development expenses, respectively.

As a result, while many high-burn companies will focus on gross margin, many profitable companies and financial analysts will focus on **net margin**. This is the lower of the two rates, which takes into account the company's other expenses.

Conversation Byte:

“Our current SMS vendor is proving to be very costly as we’re scaling. Thus, we’re considering other upcoming players in the market who might be more economical. This will help dial up our net margins too. ”

Good vs Great Corporatish

Name dropping these metrics does not impress company leadership. Just like being an MBA doesn't make you a magical PM, neither does knowing these terms. Instead, as a PM, the key is thinking through these metrics, and the impact your product will have on them.

Great Corporatish builds trust from the audience that the PM knows what moves these metrics, and has built a roadmap and features to move them.

Company leadership is often looking to harden the product roadmap. So, the conversation for good Corporatish will establish consensus on how the products are prioritized and built to achieve these metrics.

Marketindi

Product managers collaborate with marketing to ensure the product is presented in the best light. To understand their tactical and strategic performance, PMs need to be wary of the common vocabulary they use.

Marketers tend to have a specific way of thinking about things, which influences the vocabulary they tend to use.

Key Terms

B2B: Marketing/ Sales qualified leads

B2C: Landing pages

Both: Site copy, SEO, Cost per click, Impressions, Open/ click rates

Marketing qualified leads and **sales qualified leads** are the lingua franca of marketers in a B2B context. Often abbreviated MQLs and SQLs, MQLs refer to leads that have come to the website and marketing is passing over to sales. Most often, these are people who have filled out a form on the website with their contact information.

SQLs are then the leads that the sales department accepts as leads to further pursue. Most companies only convert MQLs to SQLs once the sales team has spoken with the potential customer and confirmed they are in a potential position to purchase the product.

For a variety of reasons, including being the wrong person, the wrong product, or the wrong time, many MQLs do not become SQLs. As a result, marketers are keen to measure and optimize the gap.

Conversation Byte:

“This quarter we saw a rise in inbound interest for our AI-powered CRM product for real estate agents. This can be measured by the rise of MQLs we experienced over last month (over 40%). However, the most common objection prospects are putting forward is lack of a mobile app and that’s the primary reason our MQLs are not converting into SQLs. We may not to consider prioritizing that on our roadmap.”

They often do so by optimizing **landing pages**, especially the messaging to convey the right value proposition and speak to the right audiences. The singular landing page (LP) is the first site a user

first sees when entering a website. Since many users bounce after this first impression, marketers spend a great deal of time asking PMs to optimize LPs.

Sometimes, this means working with a writer or content designer to rewrite parts of the page. **Site copy** is the term of art used to describe those words. It seems like a small thing, but changing site copy for prominent areas and buttons can have a huge impact.

The ultimate goal of the landing page, its layout and site copy is to create a compelling narrative that drives **conversions**. A conversion for a marketer is when a user takes the desired follow-up action after consuming the page. This is mostly done through signing up for a free trial, filling up a form for a demo (usually counted as a MQL) or even sharing contact details to download a whitepaper or e-book.

But before the visitors even arrive on the web page, the marketer has to get them there. With Google still the #1 site on the web, many marketers have to think about Search Engine Optimization (**SEO**).

Conversation Byte:

“Surveys show that 1 out of 3 people don’t realize that our fintech product also offers micro-loans. How about we set up a nice landing page for this? We can optimize the site copy for loan-related keywords to help with SEO. Perhaps, add in a clear GIF on the top to show how easy and quick the process is. I’m sure it will drive up our conversions.”

It’s a mixture of art and science. The technical elements of the site have to be constructed well for Google, but, more importantly, the content has to be stellar, authoritative, and with many links to it from authoritative places around the web.

SEO is contrasted from the ads on Google, referred to as Search Engine Marketing (SEM). SEO is free. SEM is paid. Marketers bid a potential **cost per click** (CPC) in Google’s auction, which is running globally and in real-time. Google does not just show the highest cost

per click. It considers the bidder's quality as well, and takes a balance of what's best for the searcher and Google's revenue.

The cost per click is a function of two things: **impressions** (views) and **click rates** (fraction of users who view that click). Google is calculating on the backend what the estimated click rate of an ad is given a particular slot, and how many people it needs to show it to.

Conversation Byte:

"The edtech market is getting competitive and naturally our cost-per-click will start rising as other players place high bids for keywords we care about. Our upcoming hyperlocal tutoring program, however, might help as we can adjust our targeting & use city names in our ads. Our impressions may remain stable but click rates will go up because students prefer localized solutions. "

Good vs Great Marketindi

To speak great Marketindi, you should have a good basic understanding of how to help your marketing colleagues. A great PM marketing partner will understand the marketer's current challenges, and help guide them on how to use technology and engineering resources to overcome them.

One challenge many marketers recently have faced is rising CPCs. With each click now costing more, they can drive less visitors to their landing pages. This makes it harder for marketers to achieve their MQL goals. Great PMs will help marketers cope with these rising CPCs by helping increase their landing page conversion rates. They can give marketers tips on better positioning based on competitive insights and suggest new landing page ideas as they expand the product.

Salesian

Product managers need to inspire sales to convince the world how great the product is. Apart from sharing new features, user stories,

and case studies, PMs need to know a bit about the internal mechanics of a sales process to better facilitate them.

Key Terms

B2B: ICP, Discovery Call, Close Rates, Win Ratio, Productivity, Lead Scoring

B2C: Channel strategy

Both: BATNA, Quota

Just like building for everyone is a fool's errand, so is selling to everyone. As a result, most teams focus on an Ideal Customer Profile, or **ICP**. The ICP describes the type of business and user within the business who would get the most out of the product.

Many sales teams will start the sales process with a **discovery call**. This is not the same discovery we PMs who follow Teresa Torres are used to doing. The sales team is usually doing a discovery call to discover if the MQL is to be converted into a SQL (jump to Marketindi if you skipped it and don't know those terms), by understanding if they are the ICP.

In discovery calls, sales teams use qualifying questions to determine good-fit. A commonly used framework is BANT:

- Budget: does the prospect have the budget for the product?
- Authority: is the prospect the decision maker or just a proxy?
- Need: is the prospect's problem solved by the product?
- Time: how urgently is the prospect looking to purchase?

If the potential customer is an SQL and negotiations progress, eventually the sales team will issue a formal quote. The conversion rate from that first formal quote to a final, signed deal is the **close rate**. But it varies widely, so be sure to check the denominator where you work.

Win ratio is another tricky rate, though typically calculated based on opportunities. It represents how efficient a sales rep is from beginning, the opportunity, to end, the signed deal. But it varies widely between companies. Some differences are just based on Salesforce, so be wary.

Conversation Byte:

“Sales has been going on a number of discovery calls where prospects cite the need for extensive customizations. I’m seeing most of these prospects are from large conglomerates trying to find a single tool to fulfill all the needs of their subsidiaries. It’s important to remember that our ICP is focused on SMBs looking for a simple social media scheduling tool. We’re not for enterprise customers yet. This means we need to tweak our marketing messages. No wonder our win ratio is down this month.”

Regardless of the exact definition, win ratio is one of the primary inputs into measuring sales **productivity**. Broadly, sales productivity is the measure of how productive a sales person is. Win rate can help determine their performance productivity.

How do organizations help sales representatives be more effective? One methodology is **lead scoring**. Lead scoring is a system that provides a score for each lead (MQL or SQL), estimating how likely they are to convert, and how much effort the sales team should put into them.

It is usually done based on things like company size, who the team is contact with, the number of people the team is in contact with, and how much time the company has devoted to the deal. But instead of calculate all this themselves, most companies will use a pre-built tool like Marketo, ZoomInfo, or Salesforce.

Conversation Byte:

“Our fitness coach app is seeing great traction from gyms with multiple outlets. Let’s bake that parameter into our lead scoring so

that we prioritize those accounts.”

The sales terms up until now are primarily used in B2B contexts. In B2C, this role might exist and use similar terms but be called partnerships or strategic partnerships. These teams often mention **channel strategy**.

For instance, in a video game context, a developer can sell online-only, they can try to partner with major retailers, they can partner with gaming websites, and more. Each of these options is considered a channel, and the channel strategy is which channels the partnership team has prioritized and why.

Both B2B and B2C sales teams make have use of terms like **BATNA**, which stands for Best Alternative to a Negotiated Agreement. A bit of a borrowed term from corporate finance and procurement, it helps the sales team understand, “if not us, then what?” By understanding what the other side’s options are, it makes negotiation easier.

Finally, when speaking sales it’s always good to know about **quota**. Sales quotas are goals or target minimums sales people need to achieve. Unlike PMs, their salaries are heavily weighted towards variable compensation tied to the quota. As a result, a sales person falling behind quota can be considerably more stressed than one above.

Conversation Byte:

“Our doctor appointment marketplace is stagnating in conversions. We need to explore channel strategies where we can partner with hospitals & clinics to increase brand awareness. At the same time, we need to sign up more doctors. To help sales hit their quota, we’re going to add a dedicated customer service rep in the sales pitch to motivate doctors to try us out.”

Good vs Great Salesian

Speaking with sales is an especially important skill for B2B PMs. It is not uncommon to see senior product leaders in B2B contexts not only regularly meeting with sales, but also regularly doing sales. Big clients like to meet with important product leaders to ensure their requests are reflected in the upcoming roadmap.

As a result, it is imperative that B2B PMs regularly communicate and proactively manage their relationship with sales. A good B2B PM might share the upcoming roadmap. A great B2B PM makes sales feel like they helped co-create the roadmap.

B2C PMs, and PMs generally, also need to have a great relationship with sales and strategic partnerships. These people are the ones who drive deals that significantly move your stock price or enterprise valuation. As a result, helping enable them can directly increase your compensation.

Designese

To get that pixel-perfect design, PMs should use design jargon that most UX/UI professionals are fluent in. Feedback given this way leads to accurate revisions, quicker time-to-market, and less back and forth.

Key Terms

UX, UI, Typography, Margin, Padding, Layout, Hero Image, Isometric, icons, Above the Fold, CTA, Line spacing, Hierarchy, Alignment, Whitespace

The language here is, perhaps thankfully, common across B2B and B2C.

UX and **UI** are different. UX refers to user experience, which is focused on the user's journey to solve a problem. UI refers to user interface, and describes how a product's surfaces look and function. Put another way, UX is about the experience, while UI focuses on visual interface elements.

Conversation Byte:

“Recent surveys for our train ticket booking system show that users love the new minimalist UI we’ve launched and the clean design of our route guides. However, they feel the UX is confusing as they’re unable to figure out how to go past the seat selection interface.”

UI designers tend to care about things like **typography**. Typography refers to the presentation of the letters and text. It’s a deep field one, one famously loved by Steve Jobs, who introduced fonts to the digital world in the 1980s after auditing a class in college. Your choice of it helps convey your brand image and feel, so designers tend to care quite a bit that it matches the design style guidelines.

UI designers also tend to care about **margin** and **padding**. They are both words used to describe white space, but refer to different elements. Padding refers to space within an element. Margin refers to space between elements. Both of them describe the ever important negative space, and it guides both UI and UX design.

Each of these negative spaces are elements of the overall **layout**. The layout is the overall structure that supports the varying visual components in the interface. It is a fundamental of UI design, so most designers will pay attention to it.

Many folks will be particularly interested and iterating on, and optimize the **hero image**. The hero image is the big, eye-catching image that is towards the top of the website. In many cases, like the one below, it takes up most of the space at the top of many websites. It usually plays a huge role in achieving the objective of the page.

Conversation Byte:

“Ling, I saw the mockups you shared for our webinar tool - totally loved the new layout. Thank you for the effort you put in. Some notes:

- *The cursive typography used in our testimonials section can be a bit hard to read, especially on smaller screens.*

- Perhaps we could use an alternative there like Open sans?*
- *The hero image is so on-point. I love the idea of this being a GIF where people from different backgrounds are chatting and collaborating.*
 - *The margin between the cards under the Benefits section look a bit tight to me with respect to the rest of the page. Can we double-check that?"*

One of the most interesting things that designers can do is depict 3D objects in 2D planes. This is called **isometric** style. This is contrasted to flat style, which does not attempt to depict the 3D nature of the image, but instead shows a cross-section.

Designers also tend to care quite a bit about **icons**. Generally, design teams will develop a set of shared icons that the app or web page regularly uses. Because icons often indicate to users what tapping on something means, getting them right matters.

Another element of design that is critical is the content that exists **above the fold**. This refers to all the content that renders before a user scrolls. The fold is the end of what's visible before scrolling, at which point the content is referred to as below the fold.

Any design discussion is also bound to have talk about the call to action, or **CTA**. There are very few informative pages in products. Most have an explicit next action, and the CTA is the button and text for that next action. It might even be closing the page. But designing it intuitively for the user is still critical.

Conversation Byte:

"Sandeep, thanks for pushing out the design in time. Phew - you're a life-saver. I had some feedback on the live page and wanted your thoughts. The icon sets we have used on the navigation bar differ from those in our action dropdowns - perhaps those can be more consistent? I'm also a bit nervous about the fact that the delicious red CTA you crafted isn't showing up above the fold on my screen - I

feel that might hurt usage. I can share my specs with you. See if you can take a look.”

Designers also frequently discuss **line spacing**. Line spacing refers to the vertical distance between text. Different apps and web pages will have different overall guidelines. What’s generally most important is consistency.

Designers also tend to focus a great deal on **hierarchy**. The visual hierarchy helps users rank what to look at. Careful arrangement of visual elements can lead visitors down the exact path a product team intends.

A final term of art to know when speaking Designese is **alignment**. Alignment refers to placing text or other elements in the page so they are aligned with something. Generally, if things are not aligned, it can look unprofessional. However, an intentional misalignment can also create emphasis.

Conversation Byte:

“Matt - the blog is now looking stellar! I think your line spacing adjustments were the secret sauce. It’s so much more readable now on smaller devices. And now that the heading font sizes and bullets have proper visual hierarchy, it’s going to make the page more skimmable. I did want to highlight the alignment of the quote blocks with the rest of the article text though. They seem to be a bit off.”

Good vs Great Designese

Designers, of all people, tend to be the least concerned with understanding their lingo. However, designers also tend to respect PMs who understand that lingo more. In fact, because designers regularly work with PMs, they often have strong opinions on how good a PM is when working with design.

One of the biggest mistakes a PM can make when trying to work well with design is doing their job for them. Designers don’t want you to

define the margin and padding. Instead, they want you to help provide the business context of what is relevant to that decision.

As a result, the best way to speak designese is to facilitate and notice the work a designer is doing. Notice that they were thoughtful about the above the fold content. Share your thoughts on the CTA during a design review. These tactics will be much better recieved than trying to pre-define them in the spec.

Techugu

Product managers talk a lot with engineering teams consisting of tech leads, developers, & QAs. Tech units prefer talking with someone who understands their challenges, comes up with feasible requirements, and writes clear specs they can build against.

Key Terms

Databases, servers, backend, frontend, version control, staging, production, estimates, technical debt, bugs/ defects, sprints

The first concept to understand is **databases**. These are the sets of data stored in some logical, organized manner. They are typically structured in terms of tables and fields. Fields [are](#) stored in a table structure. Different types of databases have pros and cons.

Relational databases support relationship mapping between different points in tables, non-relational do not. A PM will rely on engineering to decide on database design but may need to understand the implications of choices on the product.

The next term to now is **servers**. They are a crucial component of how the internet works. They are how information is served to our phones and laptops. Using protocols like HTTPS, users interact with servers, where the **backend** happens. Databases are typically on servers. In many modern mobile applications, the majority of content and computation occurs on the server. This is known as the client-server model.

The user's phone primarily focuses on **frontend** components, like how the interface is designed. So when engineers say things like, "we're a frontend team," that usually means they focus on areas like the presentation layer and how it interacts with the data from the backend.

But, it is a complex topic. Take the example of filtering. For filtering large data like Yelp, that should happen on the backend. But filtering quick data like a Google Doc would happen on the frontend. PMs should be ready to spark discussion about the pros and cons of what's better with their engineering teams.

Conversation Byte:

"Now that we're launching our own custom reports for our social media scheduling tool, we may need to add a few more fields (like reactions) to the database. We'll also need to think about adding more servers as fetching complex reports will be a heavy operation that'll take a toll on them. The backend team will need to think about architecture and how we can best structure the database for deriving such reports. I'm advising the designers to take a look at the UI of tools like Google Data Studio and Tableau to conceive the design but keep the frontend team in the loop to assess the feasibility of actually implementing them. "

The next concept for PMs to know is **version control**. It is the practice of tracking changes to code. For most teams with many engineers working on the codebase, it's necessary to have a version control system that enables them all to work on the project. Many modern teams use Git (and a company Github) for this, which enables distributed version control. Tools like Git help teams roll back to an older version without a bug or remove a feature.

In version control, engineers put their code up to a **staging** branch to check for conflicts. They test in the staging environment before going live. In the staging environment, PMs and other folks can ensure the

product behaves as intended. They often work with internal folks like QA.

Once the feature has cleared testing, it is passed to the master branch. Then the code is deployed to the **production** server. This contains the production code that goes live to users. Although external users can try the feature, companies often start with a small percentage or group of users.

Conversation Byte:

“Sure, the search filters look fine on staging and I’m seeing all the right values. However, on production, I’m seeing duplicates and the facet counts are off. This looks like a data issue. Can we get tech to look into this asap? If all else fails, let’s roll back to the old version we have on Git.”

When speaking with developers, PMs are bound to chat about timeline and work **estimates**. These are the common language between PMs and EMs in many daily conversations. PMs work with EMs to get estimates of the time required to make decisions on which work to prioritize based on impact over work ratio.

Often, conversations about priorities also center around **technical debt**. Tech debt can be deliberate, accidental, or the result of rot. Regardless, it’s often something the team needs to invest in so it can work faster and more efficiently.

Sometimes, the team also needs to prioritize **bugs** or **defects**. Even the best teams miss edge cases, or edge cases are created with future features. These create bugs the team needs to go back to address to maintain the proper success of a pre-existing feature.

Finally, to speak great Techugu, PMs should know about **sprints**. These are units that engineering teams go through, anywhere from two to four weeks. Many teams try not to disrupt the work assigned in between. Planning is instead conducted towards the end of the current sprint for the next, and so on. The term grew out of Scrum

and Agile practices but has long since sprouted its own tree, used by teams who don't practice strict Scrum or Agile.

Conversation Byte:

"I'm seeing that most of our prior sprints fail to deliver a couple of user stories that were agreed on. I guess the estimates we're getting from devs might be too optimistic? Paul had a good point that many of our issues stem from some technical debt we've accrued in our database and it's becoming unwieldy. Perhaps, we should dedicate a sprint to that to clean up the cobwebs and enable better velocity moving forward. "

Good vs Great Techugu

Speaking with engineering is a daily task for product managers. So, speaking great Techugu is essential. To do so, make sure to not overuse the terms. Instead, stick to your areas of expertise. Be rigorous about the data, use research to back up your recommendations.

Then, translate those WHAT and WHY concepts of the strategy into tactical execution. The worst product managers are those who just have their head in the clouds, and cannot help with the HOW. When speaking Techugu with engineers, help them clarify and frame pros/cons. Do not try to be the expert, but help relay the business and user context for engineering decisions, facilitate conversation, and remove blockers.

Analytian

PMs often have to venture into the land of analytics to gauge the growth, retention, and engagement of their product. This requires comfort with a lingo that product ops & data analysts fluently converse in.

Key Terms

DAUs/ MAUs, Bounce Rate, Pageviews, Sessions, Time on Site, Conversion Rates, Returning Users, Source/ Channel

Nearly all products care about usage. To quantify this usage, analysts and PMs often talk about daily active users (**DAUs**) and monthly active users (**MAUs**). DAUs represents the total unique users who used the product at least once in a 24 hour period. MAUs is the same for the last month.

Those two definitions are relatively straightforward. But analysts will also often talk about the ratio of these two metrics: **DAUs/ MAUS**. It is critical for PMs to have an intuitive understanding that this is an engagement metric. It shows how many of the users who used the product in the last month did in a day. If it's 100%, then it's a daily-use product. If it's 3%, then it's a monthly-use product.

Conversation Byte:

"Ever since we put a payment wall on the auto classifieds section, we've seen a massive drop in DAUs & MAUs. The good thing is that we were aware of the tradeoff we were making: monetizing serious sellers vs. serving everyone."

Another metric PMs are bound to hear when speaking Analytician is **bounce rate**. This is the percentage of people who land on a page and do nothing. They "bounce." Identifying pages with higher than expected bounce rates is a great exercise for PMs to do with product analytics. They can often be quickly reworked with design to change content, CTAs, and the like to decrease bounce rate.

The denominator for bounce rate can vary. Sometimes it is **pageviews**, while other times it is **sessions**. A page view is any view of a page. A user can have multiple views of the same page. A session is usually a unit of time by a user. So it might be 30 minutes. It is important to understand which metric is being divided.

Product teams also pay close attention to the **time on site**. This is another alternative to page views or sessions. Each metric has its

pros and cons, so it is good to try to choose the right metric for your specific situation in conversation with analytics.

Many features that teams release will be looking to impact **conversion rates**. This is the rate of visitors to a page that did some conversion goal: often email signup or item purchase. Many growth teams, especially, try to release experiments to improve these rates.

Conversation Byte:

“Although we’re getting a lot of pageviews on our reseller page, the bounce rates are exceptionally high. This is because the page is too text-heavy and the CTA isn’t clearly highlighted. To improve time-on-site and eventually conversions, we need to redesign the page. A well-placed video might help too.”

Often, the team will split the result by new vs **returning users**. It is critical to understand what qualifies as returning. Sometimes this is just, not being the first visit. Other times, this is being a paying customer.

A final term to know to speak great Analytician is **source/ channel**. Often, analytics will split traffic or data by the source the traffic came from. For instance, traffic from paid ads may behave differently than organic traffic. Usually source, or channel, refers to a marketing channel like paid search, Facebook, or Display ads.

Conversation Byte:

“Ever since we launched our loyalty program & flash deals, our e-commerce site is seeing a rise in returning users by 15%. The main driver of this returning traffic is Facebook where existing users are being roped in through ads for time-bound deals.”

Good vs Great Analytician

Speaking with analysts is nearly a daily activity for most PMs. So, ask questions. Analysts want PMs to be fluent in the terms.

Leverage analysts to help determine the right metrics and measure metrics.

By focusing on describing what you want to measure, and then hypothesizing the metric, PMs can help work with analysts. The great PMs do not merely understand the concepts, they work with analysts to make the analytics even better.

The Other Languages

Product managers do not interact with just 6-7 functions. They interact with 10-20, depending on the size of the company. As a result, PMs also have to be able to speak:

- Legalese - the language of legal
- Customese - the language of customers
- Managerian - the language of managers
- Projecti - the language of projects
- Writerish - the language of writers
- Supportindi - the language of customer support
- Qualuruba - the language of quality assurance
- Recruito - the language of recruiting
- Opsalog - the language of operations

While a deep treatment of those language is outside the scope of this article, one final language is worth double clicking on.

The False Idol

Producti

As PMs, we are first and foremost worried about being great product builders. We deeply analyze the products we use. We think deeply about the product strategies we propose our teams to pursue.

As a result, the language of product management, product design, and product engineering takes up most of our mental space. Indeed, meetings with these three groups also tend to dominate the actual time on our calendars.

Especially when speaking with product leadership in an attempt to prove our worth to be promoted, we are incentivized to speak great Producti. Indeed, every PM should become amazing Producti. But the idol becomes a false one when PMs forget to speak non-Producti to everyone else.

Regardless, given how important Producti is to our careers as PMs, let's quickly go over it.

Key Terms

Common PM Jargon: Jobs to be done, user needs, empowered teams, putting the problem first, product sense, roadmap, backlog, persona, mock-up

Jobs to be done is one of the most powerful PM phrases in today's modern PM jargon. It attempts to describe a rigorous understanding of what a user or customer has hired the product to do. Based on concepts from the late Harvard Business School (HBS) professor Clayton Christensen, it's continual usefulness has made it a favorite of PMs.

This video by the late Clayton Christensen can be watched infinite times, it's so valuable.

User needs is another PM term of art that means more to product people than their stakeholders and collaborators. It's another term that's arisen from the mix of user research and design thinking. It describes the overall sum of what a PM has learned through discovery and user research about what the user needs from a feature.

It is a great way to quickly summarize to product teammembers and leadership the type of motivation the group is talking about. For instance, "let's talk about user needs now, instead of business motivation or usability."

Many product folks also layer on additional theory to the user needs concept, like what the hierarchy of needs are. Most stakeholders will not know these.

Empowered teams refer to product squads who set out to achieve outcomes by solving problems as product teams, instead of executing on handed to them features. These have become a term of their own since the publication of the two landmark books by Marty Cagan: *Inspired* and *Empowered*.

Putting the problem first describes another key tenet of product management philosophy, that most people won't have as much neural circuitry tied to. For modern product leaders, this is everything. They want to help the team identify problems to solve. The worst product leaders come solution-first, pushing the team to build something before going through the product research and design process of validating the solution.

As a result, a casual mention of the phrase means a lot in product circles. But most other functions won't understand the importance of this process. It's not their fault they are coming to you with product solutions. That's what they think PMs do.

Product sense is the intuition and judgment that leads to good product decision-making. For product practitioners, we understand this typically means understanding user needs, formulating the job to

be done, and prioritizing solutions. Many outside product feel that product sense is a bit of a “dark art,” because it’s hard to pin down and define.

But great PMs exude great product sense. In brainstorming, their ideas pack a punch. The features they choose represent deep product thinking. A high percentage of their experiments hit. This is what product sense is, and many product interviews try to test for it. As a result, it is commonplace in today’s PM vocab.

The **roadmap** is a shared document that shows the direction and priorities of the product team over time. Some teams list features and dates, many have a long list of philosophical reasons that’s heresy. It’s the ultimate “PM” topic that most other functions care far less about.

A **backlog** is a list of work for the development team. Sometimes, this list is everything the team is not going to do, as in, “that’s in the backlog.” Other times, it’s the prioritized list of work the team is going to do, or has in the queue to do next. It depends on the context, but PMs love to throw this term around.

Conversation Byte:

“Our chat tool needs to consider a strategy for integrating bots in the future. Our competitors are leaning in on this and our enterprise customers have been on our case. I’m trying to clear the existing backlog riddled with defects as quickly as possible, so let’s put something on the roadmap for Q3 around bots.”

A **persona** in product is a profile of one of the types of users of the product. It often includes a description of the key characteristics, usage, and needs of that user type. Some product people quickly make these on the fly conceptually. Others deeply research them with their user research team.

A **mock-up** is usually a visual representation of the product. They are often less high fidelity than a prototype, perhaps completed by a

product manager in PowerPoint.

Good vs Great Producti

Mashing up or misusing product terms is the fastest path to your boss losing confidence in you. The worst thing a PM can do is misuse product jargon. Make sure that you know the basics of the above terms, so that can use them correctly.

Good PMs will use these terms intuitively when speaking with product leadership and fellow PMs. Great PMs will help imbue these terms with new meaning by being a thought leader on product topics in their company.

Takeaways

Politics does not need to have a negative connotation, at all. Have you ever seen Bill Clinton engage a person? He can engage anyone from any walk of life.

As a PM, it pays to study the professions of your colleagues. By being better able to speak their language, you stand more ready to influence them as well.

SECTION 3: The Advanced Skills

Chapter 8: Advanced Roadmapping

There's a hilarious video from Sanjeev NC - founder of Supermeme.ai - poking fun at Product Managers' tendency to be offended when called project managers. Comedians are often the most astute observers of reality. And one of Sanjeev's nuggets is worth double-clicking on:

There are certain things only a product manager can do

Okay, so tell me one thing only you can do that no one else does?

[After thinking]... **The roadmap**

It's true. The product roadmap is one of the few work products only PMs own. Everything else that is an actual work product, can be done by someone else: engineering, designing, talking to customers.

This is one of the big reasons that, as much as Marty Cagan and Teresa Torres would have you abandon the roadmap, it is **here to stay**.

I chat with product leaders at different companies all day long, and I haven't heard from one that doesn't write and share roadmaps. All of FAANG uses roadmaps.

And the other reality of roadmaps today is that **most roadmaps are not hitting their goals**.

In today's tech environment, product team after product team is shipping features that aren't driving the results the business wants. Tech growth is slowing to a crawl.

Product leaders are left to blame "macroeconomic pullbacks in advertising." Everyone who looked like a genius a few years ago now can't bend reality anymore.

I hear the disappointment from product leaders every day:

- “In reality, if we could increase our hit rate from roadmap to actual impact, we would be much better off than hiring another leader.”
- “Continuous discovery and tougher product reviews haven’t been some magic solution. **We still need help in the basics** of prioritizing well.”

If your company’s revenue is growing at <5% and leaders are not thinking those things, there might be a problem somewhere. Great product teams can and should be driving that growth.

So, this chapter goes over some advanced tips - a **Roadmaps 201** guide.

Here’s our roadmap for the chapter:

1. The **different types** of roadmaps **teams are using these days**
2. My favorite **advanced tips and tricks** to increase the chance you hit your goals
3. The **common mistakes** that I’ve seen teams fall into in my 15 years
4. Wrap-up and **how I suggest you deploy roadmaps**

The different types of roadmaps teams are using these days

The three main buckets of roadmaps that are widely in use at the top tech companies these days are: the classic spreadsheet; the now, next, later roadmap; or, a visual roadmapping tool.

The Classic Spreadsheet (50-60%)

The key reason it's so prevalent: everyone else in the organization has **access** to it.

Elements of an Advanced Spreadsheet Roadmap

The 201 Roadmap takes advantage of many of my favorite techniques to upgrade a roadmap:

1: Goes from vague impact to specified

The 201 roadmap asks teams to specify the **exact number increase to their OKR**. This replaces low/medium/high impact sizing in a 101 roadmap.

Having a very specific number is valuable for all sorts of reasons. But the most important is: it forces the team to be really rigorous and build a model to understand the impact.

Impact goes from vague to very specific. This helps hold the team's feet to the fire and build the muscle of really sizing what the impact of features will be.

2: Has a single field to unite teams

The 201 roadmap has incremental annualized revenue. This is one of the most powerful ways to bottom-line impact. It gets everyone trained on thinking about what their work means for the business.

And it doesn't have to be revenue. It can be profit, or GMV, or a similar universal "output" metric as a field for every team.

It's also a great reality check if a team is costing more than it's delivering. Should the team really be that big? Does focusing on that product artifact even make sense?

The one caution is: many up-funnel teams naturally get penalized with such a metric, while platform teams look ridiculously impactful. These are valid criticisms of the approach. But, on the whole, the added rigor and quantitative approach has only had added benefits for the Product org as a whole - in my experience.

3: Incorporate sizing beyond engineering

The 201 roadmap has columns for design and research sizing. This is one of the blindspots I often see product squads run into. They don't plan around the capacity constraints in design and research. They just focus on engineering.

This has the added benefit of giving design and research leadership visibility into the work. Too often, they are totally extracted from the roadmapping process.

4: Builds in dependency mapping

The larger the organization you are in, the more dependencies in product development become more or less inevitable. To not accommodate this in the planning process is a huge mistake.

In fact, I'd argue dependency mapping is one of the primary values of a "waterfall-like" process such as roadmap planning.

Making it a 301 Roadmap

There's a lot of customization you can do with the 201 roadmap, like:

- Add in the themes for each project
- Create tabs for different teams
- Organize it to line up with each OKR

That's what the 301 document does. I highly recommend that as you start working on the roadmap, you make **your own** adjustments that will make it sing for **your organization**.

The other thing I would add is: as a PM/pod leader, I also will often write up the document into a 1-pager for my area.

Now, Next, Later (10-20%)

This is one of the “sexier” ways to do things. So, the more a company is interested in improving its product craft, the more likely they are to use it.

Advantages of the Now, Next, later Roadmap

1: Helps you avoid the pitfalls of dates

About five years ago, the great enemy of product managers everywhere was putting dates on roadmap items. We were all getting in trouble promising dates to our sales and marketing partners - and then missing them.

The now, next, later roadmap really picked up adoption with product leaders around then. It's found especially good product-market fit for B2B sales product leaders with a heavy enterprise sales motion. They don't want to be caught with their hands behind their backs.

2: Emphasizes discovery

About three years ago, the great enemy of product managers was being delivery focused. Everyone wanted to “escape the build trap.” The now, next, later roadmap thus saw another uptick in adoption as teams started trying to practice more continuous discovery.

It allowed you to not churn out things on last year's to-do list and focus on the new things you are learning. And it's still good for that.

3: Great for Engineering & Design

The now, next, later roadmap is an especially good fit for working with the builders: engineering & design. In fact, I think it is actually the **majority** format for engineering sprint planning these days.

What I'll do on the product side, often, is **pair** the now, next, later roadmap with a spreadsheet roadmap. I like to have different roadmaps at different levels. A roadmap is just a communication tool, after all. The now, next later roadmap is a great place to operate with the immediate execution team.

Visual Tool Roadmaps (20-30%)

There's a litany of software you can purchase to organize your roadmap visually, some of which include:

- Aha (complete with AI writing assistant)
- Productboard (a unicorn company)
- Jira Advanced Roadmaps

Advantages of Roadmap Tools

1: Helps enforce org-wide single source of truth

While every team using their own roadmapping can be great for empowerment, it becomes *hard to manage at scale*. Mapping dependencies, for instance, becomes less of a process and more political wheeling and dealing. Nobody wants that.

In addition, senior engineering and product leaders' ability to provide feedback is seriously kneecapped, if everyone has their own format.

A roadmap tool can become a great single source of truth. It can give everyone a view into how things are progressing.

2: Wedge for digital transformation

Instituting a tool can be a great way to get the whole org onto a new process. As a result, they're very common when you want to go in

and improve some of the execution processes in a company that has been around for some time.

Just updating a process without updating the tool often leads to breakage in adoption. *Some teams adopt the process, others don't.* Making everyone use a tool that has a process helps make adoption universal.

Such a high percentage of PMs are being hired into digital transformations (think old companies like Ford or J.P. Morgan) that these tools actually represent a high percentage of how PMs think about roadmaps.

3: Drives tracking discipline

Roadmap tools are particularly useful as “systems of record.” They can help you see whatever you want to track on your team:

- When a team has updated their roadmap
- How many of them hit their OKRs
- Who used validated user insights and who didn't

If these things are lacking in your org, switching to a tool can help. This is especially common as product orgs age (think tech giants like Yahoo or eBay).

Note: The Feature Debate

People on all sides of the roadmap debate tend to get a bit worked up. In particular, much of the product internet is evangelical that feature roadmaps are the devil (Google's first result for roadmaps says, “Goal-oriented roadmaps always win.”).

I don't really see any problem with them. You're going to have to decide on your features at some point. It's better to bring roadmaps to bear as tools when you need them, and don't use them when you don't need them.

In that way, I suggest roadmaps be thought of **primarily as a communication tool.**

Other advanced tips and tricks

Now that we've gone through roadmaps of all types, here are some advanced tips and tricks to increase the chances the items on your roadmap hit your OKRs.

Tip 1: Measure analytics on sizing success

The *most powerful thing* I have seen to improve roadmaps success rates is to **make the analytics function responsible for the impact sizing**. In this setup, you actually measure and review product analysts on how *well* they estimated the impact of features. What makes this so magical is: if that's their main goal, they will be very rigorous.

This then helps the team identify when the features on their roadmap will not hit their overall goals/ OKRs. Then, they can go to leadership for help in finding higher-impact work.

Sometimes, this results in leadership helping develop higher-impact roadmap items. Other times, it triggers a reorg of the team. Both are powerful responses that can help the business.

Tip 2: Build it after you identify your OKRs

The sequence of building in product should be:

1. First, understand your **mission**
2. Use that to guide your **vision**
3. Then create your **strategy**
4. Identify your **OKRs**
5. And, **finally, create your roadmap**

Sadly, most teams skip steps. The most common failure pattern is they have a mission and vision, but then start building their roadmap (or have key items from a roadmap in mind). They then retroactively create a strategy or OKRs around their roadmap.

This way of thinking is tempting if you **feel** like you have high conviction in your roadmap items. But the reality is, a roadmap without a clear strategy or OKRs tends to set you off in the direction of features that are **good, but not the ones your business needs now**. The process of strategy and OKRs helps you figure out what the business needs now.

Tip 3: Change your format

After a while, people [tune](#) things out. Sometimes, you should just change it up.

In fact, one of the most effective things to do is actually have a “big rollout” of the “new tool.” I suspect this has helped a lot of companies like Productboard, Aha, and Roadmunk exist.

Sometimes, you just need to get sales, customer success, and executives to pay attention to what’s being built.

Tip 4: Ditch the prioritization framework for an objective function

RICE, Moscow, Kano... prioritization frameworks are **everywhere**. But ultimately, you want to exceed your OKRs within the quarter or half. So after years of using prioritization frameworks, I have begun to ditch them. The objective function I use is:

Maximize impact given the resources

So in the 201 & 301 roadmaps, you see there is design, research, and engineering sizing. This helps you estimate what exactly you can complete given your cross-functional team. And there is a **specific estimate for impact**. So I just maximize the impact given those constraints.

As I say about prioritization: if it isn’t a clear yes, then it’s a clear no. Waiting till you have some solid objective functions is one way for you to wait until you’ve built conviction for the clear yes.

Tip 5: Build a culture of updating roadmaps

There's a Goldilocks zone" of "just enough" updating roadmaps. But most companies fall on the side of not updating roadmaps enough.

If you build a culture and expectation that things will change as you learn more, it keeps you agile as a team. Many of the best features my teams have shipped were fast iterations on a learning mid-quarter.

A roadmap should be considered a 'point in time' artifact instead of a contract for the quarter or half.

The Most Common Mistakes I See in Roadmaps

So now that you know what to do, what about what not to do?

Mistake 1: Roadmap is when you start thinking about what to build

The number one failure pattern for product teams is **improper planning**. In the rush to move to agile and empowered teams, far too many product teams at fancy tech companies fall into this pattern today in April 2023.

This manifests in a situation where the product manager is racing to complete feature specifications, and designers are rushing on incomplete briefs, just to keep the engineering team busy.

The roadmap should not be the time you start to think about what to build. That should be a process that you've thought of long before. The roadmap is just a point in time during the planning process to record and share with product leaders your high conviction big bets. This ties into mistakes two and three...

Mistake 2: Lacks Conviction

When I'm reviewing a roadmap as a product leader, I like to see the team have a point of view on what will work.

If the team has a series of initiatives at every step of the funnel and on many customer problems, that's a sure-fire bet they haven't built conviction on what's high leverage and important to solve now. That's a scattershot strategy sure to fail.

I'd much rather see everything the team is doing in one part of the funnel or for one part of the problem.

Mistake 3: Big Rocks haven't validated the riskiest hypotheses

If I dive into a brief for a big rock of a team and there aren't user quotes, explanation of why the problem is the most important to solve, it's obvious there is something broken with that team.

Big rocks on roadmaps should have great briefs that have de-risked the riskiest most part of the feature. The best PMs run small experiments the half before for their big rocks the next half.

Mistake 4: Huge list of little things

The key thing to focus on with your roadmap is the big bets. When a single engineering team is sharing a half-year roadmap with 25 items, they've clearly gone off the deep end into waterfall planning mode.

The huge list of little things makes it clear the team is using this as an execution planning document. That's not what roadmaps should be used for. Use something like Jira or your task tracker for that.

Roadmaps should be **communication tools for big rock features to achieve your strategy and vision.**

Mistake 5: No Wiggle Room on OKRs

The final mistake I see all too often with roadmaps is the sum total of the items on the team's roadmap hitting their OKRs. So if the business needs the team to drive \$20M in revenue, they're driving \$20.5M.

There should be a fair amount of wiggle room. Your roadmap items should be driving at least >10% of your OKR. The reason? Things always happen in product. It's not a science. A feature fails or gets pushed back.

Always have a cushion.

How I Suggest You Deploy Roadmaps

A lot of this piece has been generalized since you are a 40K+ group with very diverse situations.

How do I suggest you deploy roadmaps if conditions are ideal? If you are in a high-performing team with capable individuals in the engineering, design, analytics, and product chairs? At a modern tech company?

1. **Use the roadmap to inform senior stakeholders:** The easiest way to product purgatory is not having product leaders believe in your decision-making. Use the roadmap to convey the quality of your product sense.
2. **Use roadmapping and planning for dependency clarification:** A key problem roadmapping solves is dependency planning. Use the roadmap/planning process to get your key dependencies resolved.
3. **Use the 301 spreadsheet** available on my website: Don't get too fancy with a roadmap tool or now, next, later. Just use a Google sheet.
4. **Treat roadmaps as a "point in time" artifact:** Publish the artifact in a big publicity tour right before the start of the quarter or half, and then move on. Don't get bothered always keeping it up to date.
5. **Enumerate your big rocks:** The roadmap should be used to highlight each engineering squad's top 3 or so initiatives. It doesn't need to go into crazy detail on everything they will do.
6. **Make roadmaps no more than 5% of your time in a quarter:** The roadmap just be something you spend a few days on per quarter. "Over-roadmapping" distracts from you the importance of execution.

Of course, not all of these tips will work for every situation. So, customize them as necessary.

Chapter 9: Why OKRs Work

I share your skepticism.

‘A management principle? What can be so powerful about the latest in-vogue version of Taylorism from Silicon Valley? Everyone sets goals.’

On the surface, OKRs (Objectives and Key Results) are just another goal-setting management tool that works for some companies better than others.

But, after working in product in a range of industries, I have found OKRs are not 'just another tool' for management. Instead, I think they are a **powerful way to organize businesses for innovation**; specifically, **product innovation**.

Why? OKRs are a powerful supplement to the product vision, help emphasize outcomes over output, and bring the benefits of qualitative and quantitative measurement.

Powerful supplement to the product vision

One of the fundamental problems of the product vision, in addition to the difficulty in creating a compelling one, is *communicating the vision is hard*. Those directly involved and responsible for the vision tend to have a nuanced understanding.

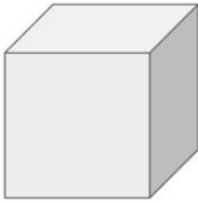
They understand the areas that are unsaid, they can fill in volumes of detail around tradeoffs, focus, and alignment with business strategy. Unfortunately, this is often only the direct product manager on the team, or the PM and the product designer!

The rest of the product team who did not have weeks and months, as well as the explicit responsibility to create the vision, often tend to have the details of what has been *explicitly said* about the vision.

They cannot speak to all of the work behind it – after all, that is not their job. They often tend to have the rough outline and understand the user stories for it.

When it comes to executives, other teams, and even senior product leadership, the vision becomes much hazier. They only understand bits and pieces of it. Often, those pieces are specifically related to large issues or wins from the past. Their understanding is a piece of the whole.

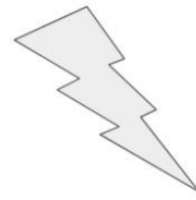
How stakeholders perceive the product vision



Product manager &
Designer



Developers &
Rest of product team



Executives &
Other teams

So, the challenge with communicating and evangelizing vision is providing a reliable set of items to give texture to the vision. OKRs are an ideal companion to the vision.

They give the rest of the product team clear goals to work towards. They give executives and other teams a clear idea of the metrics you are trying to move and the objectives you are trying to achieve.

Outcomes over output

Nearly every software team has run onto the feature treadmill without moving metrics, at some point in time. Sprint after sprint, releases fail to move the metrics. OKRs help flip the narrative.

Output, the shipping of features, ceases to be rewarded. Instead, teams & initiatives are considered to be successful when they meet and exceed business outcomes.

OKRs, properly constructed, are business outcomes – not output. OKRs allow the whole team to apply their ingenuity to move the metrics and reach business objectives. If the rest of the team understands the OKRs, then they all can contribute creative solutions to helping achieve these outcomes.

This is how product managers can get the most out of their team.

Never tell people how to do things. Tell them what to do and they will surprise you with their ingenuity.

George Patton

This also provides fertile ground to get the most out of stakeholders. Instead of debating features on the terms of the number of times sales has been asked an item, or the feeling of the HiPPo (highest paid person), the context is moving the OKRs. Everyone knows the terms of the debate – what will help us achieve our OKRs?



Qualitative and quantitative measurement

Consistent product innovation is typically not the result of launching 50 MVP features. That often results in a disjointed product just barely solving a few customer problems.

Instead, product innovation is often the result of 1) quality discovery and 2) moving steadily on certain objectives and metrics.

First, product teams need time to do product discovery. Often, the team can actually move metrics faster if they focus on lightweight prototypes and other activities before they ship a feature.

But focus on output pushes teams towards the feature treadmill, where they ship features they could have done faster discovery work on. OKRs are the antidote to this; because the team is measured in moving outcomes, they have the latitude to make the time for discovery when it is faster.

Second, conversion rate rarely doubles overnight. It doubles when the team can 10% increase conversion rate 8 quarters in a row. Having a quantitative yardstick to work towards on individual time periods helps the team work towards specific, achievable goals that are not as scary as moving a top-line metric 2x.

Certainly, achieving 10% growth each period becomes more difficult, but the product team also gains a better understanding of the levers moving metrics, and experience moving the metrics.

Consistent focus on quantitative growth helps the product team achieve the exponential outcomes that are the hallmark of great product companies. The integration of key qualitative metrics in the objectives portion also helps the team conceptualize what is involved at each stage.

Pitfalls with OKRs

Framed as output

The easiest way to separate strong and weak OKRs is any mention of output. It does not matter how often or how much the team ships. What matters is the actual business outcomes that the team moves.

OKRs must be framed in terms of business outcomes, instead of systems developed. Even a backend system can have an OKR of a better uptime, or scalability, or some other such outcome.

Refreshed at a cadence that doesn't fit the business

OKRs can be refreshed too quickly [or](#) too slowly. It is up to the management to find a cycle that makes sense given the business they are in and the capabilities of the team.

Fast moving live operations games with large engineering teams might refresh their OKRs every six weeks. Other hardware teams in semiconductors might refresh them every year.

Neither approach is wrong. But shoehorning any single approach simply because that is what one saw on TED, is. The leaders pushing OKRs should be open to adjusting the OKR frequency with the team's input regularly.

Most often, I have found OKRs set to the performance review cycle, as then OKRs can be used as a measuring stick during those periods. But different things work for different companies. Annual and quarterly are the most common.

Rolled out top down

One common mistake with OKRs happens when the high-level stakeholders are aligned, and understand the OKRs, but the low-

level teams, who do the work, were not involved in their creation, and do not fully understand them.

It is critically important for the OKRs to be established in a bottoms-up effort as well (as the top-down evaluation), with each product team developing their own OKRs.

An alternative top-down mistake is just dictating to teams who do not have experience in OKRs to establish them. In this case, it is critical for the leadership to build the OKRs hand in hand with the teams.

Since those resources are typically in high demand, one method, in this case, that helps creating this organizational change, is the use of pilot teams.

Not evaluated holistically

Another trap teams can fall into when implementing OKRs is that the sum of the OKRs does not achieve the ultimate business goals. This happens when teams are too conservative in their own OKRs, or the teams are not staffed in such a way that all the business outcomes are achieved.

It is critically important that the OKRs help be a function to have the discussions to make sure teams have ambitious enough goals, and teams are staffed so all goals can be achieved. OKRs often can help the business see the specific goals that ladder to the outcome where they have been able to make the least progress.

Teams lack the necessary skills

OKRs are generally hard to roll out if the team does not have the product managers who can own the OKRs and the analysts/data scientists who can help the product managers choose the right key results and measure ongoing progress against them.

If either of these links is missing, it becomes hard to create and enforce the OKRs. In the product manager gap situation, if the team

is just helmed by a project manager, that individual often does not have the specific charter to form and help guide the team towards OKRs.

On the other hand, in the data analytics gap situation, if the team does not have the right data understanding, then they can be left with key results they cannot achieve.

Lack of organizational value

If the organization does not take action based on achievement of OKRs, OKRs can be doomed from the start. For OKRs to be effective, the company must react. Some OKRs are moonshots, and it is okay not to achieve them; but, the team should ask how they change next time around.

Perhaps, it means reorganizing the team. On the other hand, if a team knocks out their OKRs; perhaps, it means recognizing the achievement to the rest of the company.

Do beware, this area is *hotly debated*: whether OKRs should be used in the evaluation of employees. Some people strongly believe they should not. I think that OKRs can be a useful input.

For PMs, for instance, I think it is helpful to evaluate them based on the quality of the OKRs they set. I found this often separates good PMs from great.

For the rest of the product team, teams that consistently hit their OKRs, I think, benefit from being rewarded. Without the reward, OKRs can become hollow – commitments, but not north stars, for employees.

At the same time, it is important to hear the criticism of using OKRs in employee evaluation: it can cause them to have fewer moonshot or stretch goals. To address this, evaluate teams from the perspective that it is okay to fail if the team went after big

goals. [Google](#) often mentions the 60-70% sweet spot, and teams can be evaluated on whether they were in it.

Summary

OKRs have the potential to help product teams innovate and achieve exponential growth. Structured correctly, they help empower team members to use their ingenuity to achieve business outcomes. They also help align the team and stakeholders.

But it is easy to do OKRs wrong - by framing them as output, refreshing them too often or too infrequently, rolling them out top down, not evaluating them holistically, applying them to teams without the skills, and lack of enforcement.

So, consider carefully, was it OKRs that failed or your implementation? Some of the companies with the most consistent track records of product innovation use them.

Chapter 10: How to Impact Size Features

It's mind-blowing that people still estimate the impact of a feature using something like high/medium/low (for an established product). I used to do it, too, 15 years ago.

Now the Baseline

But, nowadays, the gold standard is actually estimating the impact of a feature. Most high-performing product teams estimate the impact of a feature to their OKRs (eg, engagement metrics).

And the advanced one's also estimate the impact of a feature to their output metrics like revenue and profit. This helps you truly prioritize the features that are going to change the trajectory of your business.

Aside from the obvious benefits of helping your career due to higher impact, this also sets you up well to impress in product reviews. You can create an amazing presentation for your roadmap.

The trouble is - impact sizing is super hard. As one public company senior PM said:

It's **always rough**. I've rarely seen someone do it in a really scientific way.

How can you do it better?

So, the question is: **how can you size better?**

It's one thing to do the hand-wavy PM thing and say, "I'll estimate a 10% adoption of this feature." But that's only going to get you so far. Your impact will be much less predictable. And rigorous product leaders, engineering and analytics counterparts will question your decision-making.

It's much better to have data-driven reasoning behind your product sizing:

- Based on the actual number of users estimated to see the feature
- With a high confidence estimate of adoption and engagement impact
- And appropriate assumptions to understand the top & bottom-line impact

The problem? This type of impact sizing is something PMs just started doing in the last 10 years. As a result, there's a dearth of good knowledge or best practices.

Enter This Chapter

Uncovering this content gap, I bring you: an advanced guide to impact sizing.

We'll cover:

- **Templates** to estimate the most common metrics
- How to set up experiments to **identify and de-risk your riskiest assumptions**
- Tips & tricks to **get to the key data points**

This is a '201' level course for impact sizing. It's a great fit for PMs new to the practice or product leaders looking to adapt it.

Let's get into it.

Types of Metrics

There are three types of metrics you should be comfortable impact sizing:

- **Engagement Metrics:** Daily Active Users (DAU), Monthly Active Users (MAU), Retention Rate, Churn Rate, Time in App, etc.
- **Top-Line Metrics:** Things like Gross Merchandise Value (GMV) and Revenue
- **Bottom-Line Metrics:** Things like Contribution Margin and Net Income

Many PMs are comfortable doing some of these but not all. Let's fix that.

The High-Level 4 Key Steps

At the highest level, there are four steps in any sizing exercise:

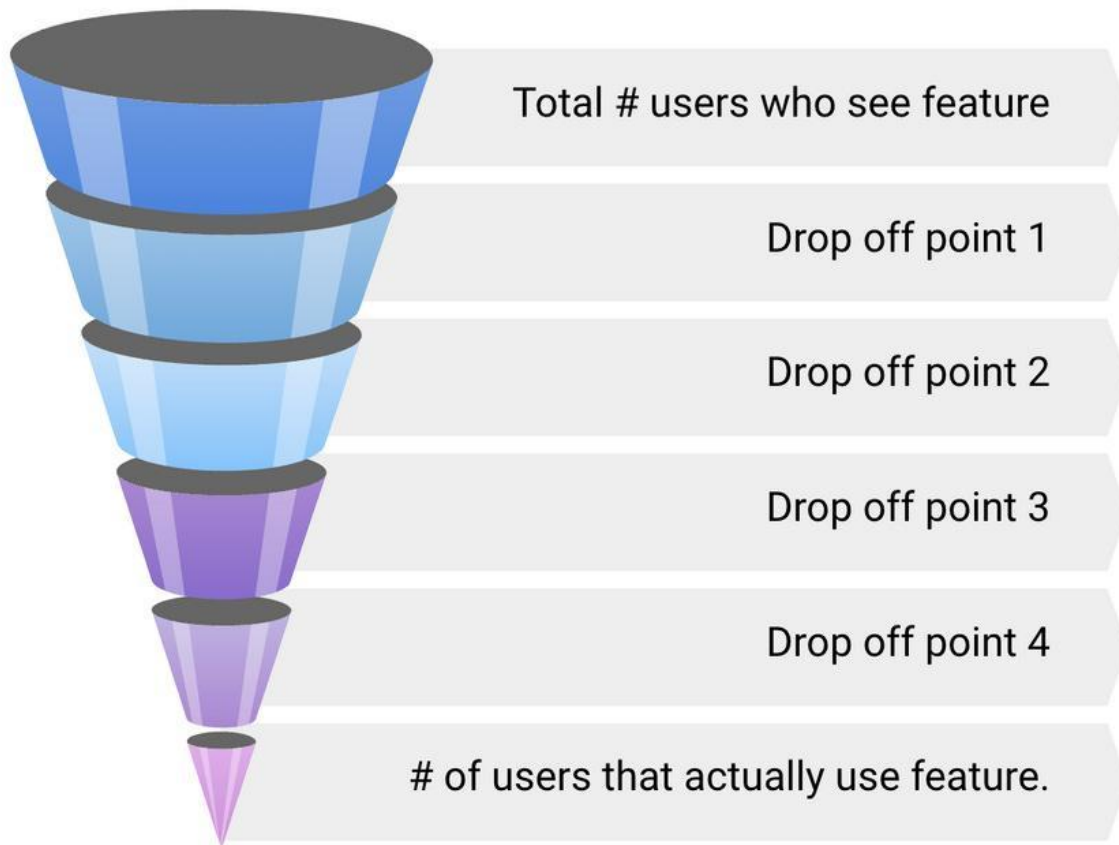
1. **Estimate usage**
2. **Calculate impact**
3. **Identify takeaways**
4. **Improve the weakest parts**

There's layers of advanced strategy to be had in each step. Let's walk through each briefly and then go through them in more detail with an example.

Step 1: Estimate usage

First, you need to get an estimate for the number of users that will use your feature. This means you create a funnel starting at the number of users exposed to the feature and estimate the drop-off at each step until you can estimate how many will actually use it.

It looks like this:



Be sure to look out for “gotchas,” like:

- How many users are actually eligible for your feature?
- How often will users be exposed to the feature?

Step 2: Calculate impact

Next, you need to estimate the impact your feature will have on these users. Generally, this usually happens in three stages:

1. Engagement impact
2. Top line impact
3. Bottom line impact

Once you understand the engagement impact, you can go to top line impact, and then bottom line. We'll go through this in much more detail in our example.

Step 3: Improve the weakest parts

After you have the first draft of your sizing, the next step is to take a look at the weakest parts. Specifically, **identify your riskiest assumptions**. You should begin to start de-risking them:

- Is it a data point you used that's very old? → Work with analytics to get a better number
- It is a usability adoption question that you can test with a prototype? → Spin up the study.
- Do you have a feature impact that's very similar to competitors? → Talk to your network.
- Is there a common industry standard technique you're implementing? → Collect benchmarks.

The impact sizing process is never truly “done.” You can always upgrade the model. So, the key is to **do the amount that fits your world**. Don't transplant best practices from your friend, or this post.

Do what fits. Do you only have a few weeks for planning? Then just address the top riskiest assumption. If you have more time, address a few more.

Step 4: Identify takeaways

Once you've gotten the model to a good place, it's time to identify the takeaways. These usually fit into three buckets:

1. **Planning**: Impact sizing is a great input for prioritization
2. **Experiment Execution**: Impact sizing can help you determine the magnitude of impact you are seeking so you know how long to run the experiment for stat sig
3. **Feature Design**: As an identifier of the levers you can move in the design to drive greater impact

Let's walk through an example to make it more real.

Example: New Item Alert

We'll talk about a feature called New Item Alert. Let's imagine the product is a mobile app for clothes shopping.

Users can set up an alert when a store posts a new item. Whenever a new item is listed, they'll receive a push notification to their device.

Here's how it works:

- **When** a user navigates to a store's page
- **If** they are there for more than 10 seconds
- **And if** they are already opted into push notifications
- **Then** a modal pops up that asks them if they want to set up the alert

The Goal: We're trying to improve our engagement metric. It's measured as Daily Active User / Monthly Active User, aka **DAU/MAU**. We're defining "Active" as opening the app at least once on any given day.

So, how do we size this feature?

Step 1: Estimate usage

Here are the things we need to know:

1. # of users that navigate to store page
2. Avg # of stores users visit
3. % of those users who remain on the page for more than 10 seconds
4. % of those users opted into push notifications
5. % of those users that will set up an alert

Multiplying all these numbers together tells us how many alerts users set up with this feature.

The simplest way to do this is in an Excel doc or Google Sheet with formulas so you can immediately see how changes impact the results. The three columns are:

1. What you're measuring
2. The data for that measurement
3. The rationale (i.e. how you got that number)

Here's an example with hypothetical numbers:

Step	Measurement	Data	Rationale
Sizing funnel	# of individual users that navigate to store pages	100K	Data
	Avg number of store pages users visit	2.6	Data
	% of those users that remain on the page for >10 seconds	32%	Data
	% of those users opted into push notifications	40%	Data
	% of those users that will set up an alert	30%	Assumption
	Est # of enrollments in new item alerts	9984	Calculation

But this doesn't tell us anything about the impact yet.

Step 2: Calculate impact

Engagement

We still need to know how many push notifications will actually be sent, and make an assumption about how often users will engage with them.

So we need to know:

1. On average, how often do stores add new items? This tells us how often users will receive push notifications.

2. How often will users engage with these notifications? This is called the click-through rate (CTR).

Multiplying these numbers together tells us how many more times per month we can expect users to come back to the feature per month.

Finally, we average the DAU/MAU of users that use the feature with the DAU/MAU of those who don't, and we get the new overall DAU/MAU

Here's what that looks like:

Step	Measurement	Data	Rationale
Engage- ment Impact	Avg number of items stores add per month	8	Data
	Total # notifications sent per enrolled user per month	20.8	Calculation
	Avg CTR of notifications per user	20%	Assumption
	Avg incremental engagements per enrolled user per month	4.16	Calculation
	Lift to DAU/MAU for enrolled users	0.139	Calculation
	Starting DUA/MAU for all users	0.2	Data
	New overall DAU/MAU for all users	0.214	Calculation
	Overall DAU/MAU lift	6.92%	Calculation

Based on these calculations, we'll see a ~6.9% lift in DAU/MAU as a result of this feature. Sanity check this against your baseline DAU/MAU. Take a look at how other features have changed it over time to make sure the change is achievable.

So now that we have the main engagement metric of DAU/MAU, let's think about how this example might apply to metrics further "down funnel."

Retention Rate

Engagement doesn't mean anything in the long run if you aren't retaining those users. So in addition to tracking adoption of New Item Alerts, it would also be helpful to estimate how it will impact metrics like overall retention.

[Preferred] Build a first principles model specifically for retention

Retention is generally the result of users visiting X days after their signup. Let's say this feature is working on 30 day retention rate.

So we need to know new users per month, existing 30 day retention rate, new users who navigate to store pages, and percent of incremental engagements from new users occurring on day 30. Here's what that looks like:

Step	Measurement	Data	Rationale
Churn/Retention Impact	Starting 30 day retention rate	24.2%	Data
	New users per month	20,000	Data
	New users who navigate to store pages	4,000	Data
	Incremental day 30 engagements	399	Calculation
	Absolute 30 day retention rate lift	2.00%	Calculation

A more complex model might think about all three levers retention comes from:

1. Have engaged users come back on that specific day
2. Bring back churned users
3. Prevent users from churning

It's about knowing how deep you need to go in your org. That advanced model requires much more data that's hard to query. Generally, as a PM, we would stop at the model above. Or even an easier one:

[Fallback/ Gut Check] Correlate the connection between engagement and retention

In this option, you take a look at how engagement (DAU/MAU) and retention rates in your app have been connected in the past. You might create a regression model or just apply a simple multiplier, depending on the strength and nature of the correlation.

If your historical data shows that a 1% increase in DAU/MAU goes hand-in-hand with a 0.5% increase in retention, then the ~6.9% lift in DAU/MAU due to the feature could lead to a 3.45% increase in retention ($6.9 * 0.5$).

Now that you have the engagement impact, it's time to move to revenue.

Top Line Metrics

Every business needs to earn revenue. As a result, even if you're the most top of funnel, platform, or back-end team, you should endeavor to connect your features to this metric.

There's a couple ways to do this.

[Preferred] Build a model from engagement to the revenue

We've already figured out the incremental visits to the item page. So we can extend that to revenue based on the add to cart, AOV, and revenue take rate.

Most of these can just be last month data, or something similar, and it's a simple calculation. Incremental engagements * add to cart rate * add to cart to purchase rate * AOV * revenue take rate of GMV.

It looks like this:

Step	Measurement	Data	Rationale
Top-Line Impact	Item page to add to cart rate	4.21%	Data
	Add to cart to purchase rate	78%	Data
	AOV	\$94	Data
	GMV	\$128K	Calculation
	Revenue take rate of GMV	11%	Data
	Monthly Revenue	\$14K	Calculation
	Annualized Incremental Revenue	\$169K	Calculation

You could get more complex by thinking through more steps of the funnel for these new users. Or, like with the retention metric, you correlate as a fallback:

[Fallback/ Gut Check] Correlate the connection between engagement and top-line metrics

This works the same as above. You analyze historical data to discover how the increase in engagement (DAU/MAU) might relate to the chosen top-line metrics. This might involve using a regression model or applying a multiplier.

EG, if your historical data shows that a 1% increase in DAU/MAU is linked to a 0.8% increase in revenue, then the ~6.9% lift in DAU/MAU could lead to a 5.52% increase in revenue ($6.9 * 0.8$).

Bottom Line Metrics

Finally, let's consider how we might tie engagement to bottom line metrics.

If you can come up with a specific profitability multiplier for what your feature encourages, that's great. In this case, it's new items. So we can check the profitability of new months last month and apply that to the annualized incremental revenue above.

It looks like this:

Step	Measurement	Data	Rationale
Bottom-Line Impact	Profitability of new items	82%	Data
	Annualized Incremental Profit	\$138K	Calculation

You could get more complex and look at the profitability of incremental items from a prior experiment, or something like that. But at least one layer is better than average profitability - in this case, profitability of new items is usually enough for PMs. The more advanced stuff can be left for analysts.

Disclaimer: Other Types of Features

Many features may not have an engagement based hook into revenue or profit. They may just directly affect those steps. That's fine. Just build a first principle model to your metrics of choice.

Step 3: Improve the weakest parts

Several of the steps in the analysis rely on assumptions. Identify the **riskiest** assumptions and begin to de-risk them. (These also make a great topic for product review.) These are two that stand out:

- Percent that will setup an alert
- Avg CTR

We could start to improve each of these estimates by collecting more data points.

1. We could look at comparable features or reach out to execs with experience on percent that will set up an alert
2. We could use Avg CTR from comparable pushes

You may even plan to run an experiment to de-risk. So for instance to look at CTR, you could send a push to a feature that is still in waitlist.

One thing to consider here is even much of the **data** has assumptions. For instance, in many of the places we looked at last month's average. But this month or next month, conditions may dramatically change. So you also need to see if that assumption makes sense.

Great product leaders tend to focus on this part - finding the riskiest assumptions - the most in evaluating PMs. So you want to be really good at showing you know what they are and have a plan to tackle them.

Step 4: Identify takeaways

At \$140K profit per year, we're not getting many executives excited. A single engineer usually costs more.

Luckily, our method of sizing easily displays levers for maximizing the impact of this feature. So, we could increase the number of users that engage with this feature by:

1. Increasing the number of users exposed to this feature.
Maybe instead of requiring them to be on the store page, we could embed this feature directly on the pages of individual items.
2. Increasing the number of stores users visit.
3. Finding ways to get a higher % of users to stay on the page (or decrease the time required.)
4. Change the design to get a higher percentage of users to set the alert.

5. Getting a higher percentage of users to opt into push notifications.

What if we could dramatically improve in each of these areas?

Step	Measurement	Data	Rationale
<div> <div></div> <div>Sizing funnel</div> </div>	# of individual users that navigate to store pages	100K	Data (add link)
	Avg number of store pages users visit	4	Data (add link)
	% of those users that remain on the page for more than 10 seconds	80%	Data (add link)
	% of those users opted into push notifications	80%	Data (add link)
	% of those users that will set up an alert	80%	Assumption
	Est # of enrollments in new item alerts	205K	Calculation
<div> <div></div> <div>Engagement Impact</div> </div>	Avg number of items stores add per month	8	Data (add link)
	Total # notifications sent per enrolled user per month	20.8	Calculation
	Avg CTR of notifications per user	20%	Assumption
	Avg incremental engagements	6.4	Calculation

	per enrolled user per month		
	Lift to DAU/MAU for enrolled users	0.213	Calculation
	Starting DUA/MAU for all users	0.2	Data (add link)
	New overall DAU/MAU for all users	0.637	Calculation
	Overall DAU/MAU lift	218%	Calculation

That would be a crazy 218% engagement lift (our OKR) - which would flow through to the other metrics. So it gives you clear direction on areas to [work](#) with design as potential levers to improve the impact.

This is the real power of this sizing methodology. It helps you understand whether your growth work is actually material. That's why teams everywhere are moving to it.

Hot Topics

When is sizing not a fit?

- **Pre-PMF** - At this point, your metrics are going to be moving too much. Focus on core user value versus specific metrics.
- **Tech Debt / Design Debt** - You should do some of this, and so in that case a framework like Kano can be helpful.

Where does the PM's role begin and the analyst's end?

Adapt to the culture of your organization. Generally, the best practice to help bring your organization towards is:

- PMs build the first draft of the model off of key datapoints the analytics team has pulled
- Analysts provide the base data for the first draft and enhance the sizing for particularly important prioritization decisions

Advanced Sizing Techniques

So, in our examples we've gone through a 101 sizing approach. How can you upgrade your sizing even further?

Sensitivities

You can start to provide ranges. It's great to visualize the sensitivity with something like this:

Impact Estimation Sensitivity

Benchmark found online	4%
Last experiment 2 years ago	3.5%
Estimated impact	3.3%
Similar experiment last month	3.1%
Statistic shared from VP product at last co	2.9%

This helps you move from a point estimate to a confidence interval.

Segmented Estimates

There are so many different types of user personas that might use a product. For instance, in our example, there might be highly engaged users and lowly engaged users who have different CTRs and conversion rates. If you apply that segmentation and calculate the numbers, you can get much better estimates.

While we tend to leave this type of deep work to analysts, if you feel it's a weak part in your analysis given your specific condition, dive into segmented estimates.

Calculate the Edge Cases

There are so many edge cases. What about people who naturally disable push notifications at the OS level? What about people who due to this push disable push and don't get the impact for other push?

There are a million such details and edge cases to any product feature. If you want to get more advanced, you can prioritize the top two-three and model them out.

The Objective Function to Prioritization

If you think of yourself has having a set of features (f) with an estimated impact size, design work required, and engineering work required, you can build an objective function to prioritize, that looks like this:

$$\max(\sum_f impact_f | designwork_f; engwork_f)$$

You maximize the impact of the features you build given limited design and engineering resources.

We find this to be amongst the most scientific and credibility-generating ways to prioritize. It takes the fluff out of methods like Rice, Moscow, and Kano. Those are better fits for pre-PMF, tech debt, design debt, and bugs.

Tips & Tricks

Tip - Identify the key inputs & go to analytics

We highly recommend building a few models, identifying the common inputs, and then putting together an analytics request. They will be happy to see you put some work into modeling and have very specific questions for them.

They're much more likely to quickly complete a request for a data-driven PM who can do most of the work on their own.

Tip - Build a database of your impact estimates

We recommend creating a spreadsheet that has a tab for each of the features you've built an impact estimate for, starting today. A year from now, this will be something like 25+ sheets deep.

Then you will have most of the data you need for future impact estimates all right there. When you run a similar feature, you can just refer to other sheets in the same workbook.

Tip - Gut-Check With A Colleague

It's easy to lose the forest for the trees when doing this work. Make friends with senior product analysts and strong data-oriented product managers. Share your sizings with them as friends to get feedback and gut check your work.

You'll be amazed how much they can help.

Trick - Get comfortable with SQL

A PM superpower is querying some of the data yourself. Here's the crazy thing. ChatGPT/Bing can teach you. Just ask it to start teaching you SQL, and it will even go through example questions and grade your code for you.

Then start using SQL to query the basic stuff for you. Just ping your favorite analytics buddy for tables and code reviews.

Trick - Use ChatGPT to build your first draft models

So we did call out BingGPT as not being capable at the top of this post. And it's sadly not so. So, there's no **free** GPT-4 to help you.

But there is a **paid** one. ChatGPT Plus GPT-4, with the right prompt, can actually first draft your impact sizing models. We played around with the prompts a bunch to get one that works well, and the trick is to give it a very specific feature and very specific metrics. Here's what that looks like:

This can help you get “unstuck” in the impact sizing process and just get started.

Trick - Use ChatGPT Code Interpreter to pull Data

The final trick just became possible this past week: as I highlighted at the top, use Code Interpreter.

What you do is you use your SQL skills to download a CSV that's less than 100MB (eg, last month's summary stats) and upload it to Code Interpreter. Then you can use natural language to do all the data analysis you need.

Remember to turn off history/contribution to data collection so you keep your chats private and don't leak any trade secrets.

5 Most Common Failure Patterns

Failure 1 - Too Much Weight on the Point Estimate

A lot of people focus on the point estimate. They will report their revenue estimate to 7 significant digits: \$1,428,342. Don't do that. Your impact estimate probably has 2-3 significant digits clarity.

Use your sensitivities and confidence intervals to create an impact band. This is especially important for prioritization exercises. \$1,428,342 seems clearly bigger than \$1,267,125. But that's false precision. If you build the bands of \$1.2-6M vs \$1.1-\$1.5M, it's not clear one feature is actually more impactful than the other.

Failure 2- Not Building a Product Mock

If we had never built the product mock in our example, we could have missed crucial details, like the number of users that navigate to store pages, or opt into push notifications.

When you go just based on a fuzzy definition - like just the problem - you tend to ignore these finer details that whittle down your potential impact.

Failure 3 - Boiling the Ocean

When teams first begin this process, PMs often tend to become overzealous and put on their Analyst hats when building these models. They overcome every weakness, do user testing for every assumption, and build complex segmented models.

Don't go too far. Know just how far to go to build confidence in your prioritization decisions at your organization.

Failure 4 - Too Much Hand-Waving

The other most common failure is not having enough precision for your big rocks. PMs get too busy with execution and overwhelmed with other tasks to build a sophisticated model.

Your model shouldn't boil the ocean, but it shouldn't be too simplistic either.

Failure 5 - Confirmation Bias

It's easy to play around with the numbers until your favorite features have the best numbers. Check your biases at the door before starting this exercise, or it will be a waste of time and when the results come back different, everyone will lose faith in your estimating abilities.

Make this exercise an actual search for truth.

Final Words

Sizing is always going to be rough. But doing sizing that is 'slightly better' than the average sizing done by PMs at your company is advisable. You don't want to be seen as the PM who is 'winging it.' Be the data-driven PM who builds a model to justify their decisions.

Chapter 11: When to A/B Test

It's one of the most hotly debated topics in product development:

Should we test it?

Some people are zealots that you must test everything. Others go with the flow. And most just don't know.

There are a couple harsh truths about A/B testing every PM needs to know:

- Most A/B tests are **inconclusive**
- A/B testing is the **best way to prove your impact**
- A/B testing has a **real and tangible cost in speed**
- Truly great changes **win in A/B tests and the bottom-line metrics**
- Many teams make **mistakes running and interpreting results of A/B tests**

Together, these harsh truths make it quite important for PMs to have the ability to have an opinion on when to test. Some execs will want you to test things you shouldn't. You need to have the knowledge to push back. Other execs will want you just ship something - and you have to be able to explain why it needs to be a test.

This Chapter

That's where this chapter comes in.

I've been involved in 100+ A/B tests in my career (thanks to PMs on my teams). And I've been involved in even more features shipped without a test. So, in this chapter, I cover:

1. The spectrum of POVs
2. Easy heuristics to start with
3. Myths to forget ~ A/B testing
4. Features of a great experiment
5. The first principles framework to organize it
6. Making the call on 5 example features for texture

Let's get into it.

The spectrum of POVs

At each end of the spectrum lie some interesting viewpoints. Let's go through those first. They help illustrate where you could end up. Then we'll address the messy middle.



One end of the spectrum: Scientific or it's Questionable (SQ)

- *Default Tendency: Experiment*
- *Prominent Companies: Spotify, Indeed, Meta*
- *~30% of PM roles*

On one end of the spectrum is SQ culture.

This tends to come from product leaders who excel at demonstrating impact. They don't claim a cent more impact than they have actually driven. When it comes to meetings with skeptical analysts, they walk out unscathed - with the same impact as originally claimed.

They tend to have come from big company backgrounds, where claiming impact based on a pre-/post-analysis doesn't pass snuff. Over years of career success, they've been trained out of the ability to ship something not as an experiment. For them, the answer to "should we test it?" is always yes.

Roughly 15% of the product leaders you encounter are SQ. But because they lean to bigger companies, SQ culture represents reality for ~20% of PMs.

The other end of the spectrum: We have Big, Global Launches (BG)

- *Default Tendency: Just Ship It*
- *Prominent Companies: Apple, Tesla*
- *~20% of PM roles*

On the other end of the spectrum is BG culture.

The product leaders who drive BG teams tend to come from highly marketing-driven environments. The big launch of product features, at an event like WWDC, is everything (why I opened with that).

So instead of focusing on testing, these leaders tend to have a high bar for the product development process itself. They want teams to validate problems are real with data, and then user test prototypes extensively before shipping to everyone.

Roughly 20% of product leaders are BG. And, as a result, about 20% of PMs live in a BG world.

The rest of folks: Different Tools for Different Times (DT)

- *Default Tendency: It Depends*
- *Prominent Companies: Google, Netflix, Amazon*
- *~50% of PM roles*

The rest live somewhere in the middle. This can be a messy space. Some people test *most* of the time. Others “just ship” *most* of the time.

But the most common model is one where leaders empower PMs: PMs have the infrastructure to test, but the decision is up to them. For these companies, you need to have a method for sorting out what to test and what to just ship.

That’s where this chapter comes in. We’re going to walk through a *first principles framework* to help you decide for *every* feature you are working on.

Even if you’re in a SQ or BG environment, this chapter will help you *rightly push back* on your leaders when taking the other approach can help you achieve more impact.

Easy Heuristics to Start With

Let's start with some easy heuristics to rule out or consider an A/B test.

Easy ways to rule out an A/B test

1. Something not expected to move the metrics
2. A fundamental change in your product
3. Results will take too long
4. Big splashy launch

1. Something not expected to move the metrics

There are many changes to a product you *need* to make. Things like: **documentation, a refactor, or a bug fix.**

Great PMs always have a mix of these initiatives in play. Here's why: a great product that does 1 thing well, in the long run, beats okay product that does 10 things well. You can almost think of these items as *protecting* the impact of your predecessors on your product shipped.

These things are valuable and make it on to the to-do list, but they aren't expected to be measurable. If that's the case, just do them and move on.

2. A fundamental change in your product

If you're re-working the entire product, it's going to break your metrics. Grounds-up rewrites are important, but they re-write baseline metrics. If you try to compare the two experiences, you'll see some metrics that are wildly better and others that are wildly worse.

Often, the "old" version of your product has years of optimizations beefing it up, so it performs pretty well in the A/B test. And that can hold you back from evolving to the next version of your product.

As a result, **if you have fundamental consumer and business insight to drive a fundamental change of your product, just do it. Don't A/B test it.**

3. Results will take too long

The most common reason I have found myself and PMs on my teams not A/B testing is if it will *take too long*. You can't let A/B tests run for too long for two different, but equally important, reasons:

1. **In product, we need to move quickly:** In the world of OKRs, we don't have the opportunity to wait a long time in product management. We're being measured on a quarterly basis.
2. **The longer a test is running, the more likely it is become invalid:** Long-running tests are susceptible to external validity threats and sample pollution.

Generally, the pressure from either of those reasons is great enough that, where I have worked, we will just ship the change and move on.

4. Big splashy launch

The final reason not to A/B test, and why Apple and others tend to A/B test less, is because of big, splashy launches. **Apple's WWDC is a phenomenal marketing moment. Taking advantage of that matters more to Apple than A/B testing.**

You can't really hold out a group of users after a marketing moment like that.

If you really want both worlds: you have to do something like the gaming companies do; you say, 'We'll roll out to select users in these countries first.' But, that marketing moment is then watered down. The press and influencers will report a beta, not a global launch.

So if you want to take full advantage of a big, splashy launch, you can't A/B test.

Easy ways to require an A/B test

1. Growth experiment to learn about future larger feature
2. There is a probability the change will cause a negative impact
3. Need to prove the value in order to get credit at your company
4. It will be easy to measure as an A/B test but hard to measure pre-/ post-

1. Growth experiment to learn about future larger feature

Having led many product growth teams, this is the reason I A/B test a lot. It's the essence of two factors we growth teams focus on a lot: the scientific method and moving fast.

We generally develop a learning agenda for each metric. We have some hypotheses, but instead of diving head-first into a big feature, we test into the hypothesis with a small change first. **If you're trying to advance a learning agenda, by all means, jump into A/B testing.**

2. There is a probability the change will cause a negative impact

Let's say you're making a change to your pricing. You want to make your monthly plan 25% more expensive. What will that do? Well, hopefully it will increase your percentage of annual plans without reducing your overall conversion rate.

But you can't be sure. That's why the change is a great candidate an A/B test. Indeed, for any change where there might be a negative impact that you want **to monitor in trade-off of another metric, A/B tests are great tools for the task.**

3. Need to prove the value in order to get credit at your company

One thing that I didn't really see any of the other content mention, which was odd to me, is adjusting to YOUR company's culture. In my

mind, there's really nothing more important to you as this. It doesn't matter what it says in the articles online. **What matters is what works at your company.**

If you work at a company where 490 PMs are running A/B tests to prove their impact, it's not a great strategy to be in the 10 PMs who don't. Don't neglect to figure this out! In your first few months at a company, always observe the other PMs. If you didn't when you joined, do it now.

4. It will be easy to measure as an A/B test but hard to measure pre-/post-

The final "easy candidate" for A/B testing that's worth mentioning are the features that would be easy to measure as an A/B test but hard to measure pre-/ post-. These are things like:

- Small growth hacks
- Quick hypothesis tests
- Small change to a high volume surface

I actually think most PMs don't have enough of these on their roadmaps. In my opinion, you should always have a few of these a quarter. They're little things that every team can squeeze in but help get you quick wins. And you'd be surprised for how many teams they end up the highest impact item of the quarter.

Now that we've gone into the "easy" heuristics, let's go through the more sinister myths.

Myths about A/B Testing

When it comes to these debates, I find that people bring varying levels of knowledge about A/B testing fundamentals. These are the four most common myths:

1. “Since we’re a small company, we can’t A/B test”
2. “There’s no cost to A/B testing with our great infrastructure”
3. “Only change one element per A/B test”
4. “Let’s launch because it doesn’t hurt”

1. “Since we’re a small company, we can’t A/B test”

Repeat after me: **Small volume products can still run A/B tests.**

This might be the most pervasive and damaging myth about A/B testing out there. Small businesses can run upstream testing programs. They just have to be on relatively high volume surfaces where the expected change is large.

For instance:

- You are going to redesign your homepage layout
- You expect a 10% change on your 3% baseline conversion rate
- You have 60,000 monthly visitors (this is baby small - my Substack has over 500K)

Then A/B testing is a great option. The key here is:

1. **You made a big change** (home page)
2. **You expect it to make a large impact** (10%)
3. **It’s one of your highest volume surfaces** (the website homepage)

You can an experiment and see statistically significant results within 15 days:

How long will your experiment take?

How many eligible visits do you have per day?

What percentage of visits will see the change?

What is your current conversion rate?

How will you change conversion?

How confident do you want to be of this?

How likely should you be to detect the change?

You should run this experiment for 15 days.

Experiment Calculator

Some people are surprised by how fast that it is. Surely, if we wanted to detect a 5% change (vs 10%) it would be much slower? At 95% confidence, yes it would take 59 days - probably too long.

But let's imagine you are willing to accept 80% confidence instead of 95% confidence (95% is totally arbitrary, 80% is Pareto). Then the experiment would only take 34 days - at least acceptable:

How long will your experiment take?

How many eligible visits do you have per day?	<input type="text" value="2,000"/>
What percentage of visits will see the change?	<input type="text" value="50%"/>
What is your current conversion rate?	<input type="text" value="10%"/>
How will you change conversion?	<input type="text" value="5%"/>
How confident do you want to be of this?	<input type="text" value="80%"/>
How likely should you be to detect the change?	<input type="text" value="80%"/>

You should run this experiment for 34 days.

My push back on this 34 day experiment is, if you're this small, why are you going after 5% gains and measuring them? Consider bigger gains.

Overall, way too many small products rule out A/B testing when it could really help them. Instead, they ship changes that actually hurt their metrics.

That's such a shame. You're making Pareto decisions all day in a startup. Consider them in your product as well.

2. "There's no cost to A/B testing with our great infrastructure"

On the flip-side of small companies launching too few A/B tests is big companies launching too many A/B tests. In many cases, there's a bit of a mistaken belief at the highest levels of these companies that A/B testing imposes no cost.

But, **A/B testing always has a cost on some skilled person's time**. If you rely just on the tool, you're liable to make mistakes as often as you make good decisions. There's a couple things a skilled person has to do - that no automated tool can do for you:

- What metrics are north-star, secondary, and guardrail
- Putting time into framing & executing the decision
- Cleaning up the data and presenting it

Those sound worthwhile to do, but the thing is: these **generally trade-off from other higher impact things a Product Manager could be doing**. Things like figuring out the next big problem to solve and cleaning up the execution on your next solution you're shipping are even more worthwhile.

Every A/B test has a cost.

3. "Only change one element per A/B test"

A/B testing is about developing a perspective on whether a change works. That means you can put a bunch of changes together. In fact, >90%% of the time you should. Some of the best A/B tests I have run made lots of changes:

- **An entirely new onboarding**: We changed number of screens, steps, and tasks to create onboarding v7 (this was a 4 year old \$5B/year game).
- **An easy mode for a product**: We created an entirely new way to experience the product, but weren't sure if it would work.
- **New credit tranches**: We changed the financing programs on a lending product.

All these changed many variables, but we wanted to test the whole new package vs the whole old package, and A/B testing allowed that.

The times you must change one element per A/B test are when:

1. **You seek to figure out a scientifically reliable growth lever:** When you're first trying to prove an area is a profitable area of product investment (eg, transactional emails), you need to know the precise impact of a small change. One-change A/B tests are perfect for this.
2. **You want to be able to attribute the impact of a specific change:** Often, with a big change failing, you have no way of knowing why. So if you're trying to move a notoriously hard to move metric at your company, it can be especially useful to know what is driving a particular change.

But in the vast majority of *practical* situations, leave the scientific A/B testing programs to marketers and other folks bound by the confines of their surface. **Product builders should be thinking bigger than growth hacks.** (Yes, even growth teams.)

4. “The standards of academia apply”

In academic science, a great experiment is:

1. Methodologically good
2. Conceptually important
3. Pedagogically important

Neither 1 nor 3 are all that important in product A/B testing.

Methodologically good usually implies a p-value of 0.05, but we don't need that level of confidence in product. 90% or 80 are fine. In fact, most PMs earn their paychecks precisely by making 80/20 Pareto optimal decisions.

Moreover, methodologically good usually implies bidirectional testing of an intermediate variable to prove causality. *We don't need that in*

product testing either.

Pedagogically important means that the experiment contributes to our knowledge about how something works, thereby making it more impactful to teach. We don't need that in product testing, either. Our goal is business metrics, not teaching.

The big difference between academic and product experiments is that: academics are generally trying to prove timeless causality. But in product, we're completely fine with time-bound causality. If a concept (like Referrals give \$20/get \$20) only works for six months, we still want to take advantage of those six months.

As a result, craft your testing plan so it helps you validate or invalidate your hypothesis, but then move on from the academic theory.

That's it for the myths! Now let's cover what makes a great experiment.

What Makes a Great Experiment

There are some situations that call for a great product experiment. Let's highlight the top three archetypes so you can pattern match to them:

Archetype 1: Helps you validate a key element of your strategy

One of my favorite types of experiments are those that advance a learning agenda - in a dramatic way. These are the classic examples of "building a milestone 1."

Too many product teams just dive head-first into building out a big change. That's fine and well if you have utmost conviction in the change, but if there are doubts, carving out an initial learning experiment is best.

Great experiments are these milestone 1's that either drive conviction in your strategy, or help you make a hard pivot.

Anti-pattern: *Experiment grenades.* These experiments create friction between teams - eg, the growth and core product team. It's much better to hold hands in kumbaya about the strategy and testing approach before.

Archetype 2: Helps you resolve a debate

Data ends arguments. In the context of multiple product teams working on a similar surface, arguments are inevitable. If you can agree with the team that the hypothesis is worth testing, an experiment can be the perfect antidote.

For example, let's say your team is arguing about the color scheme of the landing page. Some believe that a vibrant, lively scheme will drive more user engagement, while others argue for a more

minimalist, clean design. An A/B test can resolve this by directly comparing user responses to both versions.

Anti-pattern: *Data-driven*. While these types of experiments are valuable, they should not be used as a crutch to avoid productive discussion and decision-making. Data should inform our decisions, not dictate them.

Archetype 3: Wins you credibility you need for bigger things

Don't let the thought leaders get to you. **Never lose sight of the reputational value of having a validated set of results for your impact.** If your organization is not funding your team's resources the way you want, a good experiment often helps.

Because even if you deliver on your OKRs and move your business line forward, in the back of people's minds, without an experiment, they think it could be due to other things (maybe marketing or another PM). But if you ship a product experiment your team came up with and delivered amazing impact, it becomes a currency.

I've sat in many a headcount budgeting decision meeting. And while there are often table-beaters and smooth talkers who get their headcount, I have found that those with the clearest demonstration of their impact get the *most*. And in product, that means an experiment.

The takeaway: if you have a world-beating feature you have lots of conviction on and would normally just ship, but need that added oomph for more headcount in your area, run it as an experiment.

Anti-pattern: *Endless optimizations*. What you don't want to do is run everything as an experiment just for the sake of proving impact. At that point, you're playing the game defensively. Always play the game to win.

We're moving fast now! We've done myths and archetypes, it's time for top mistakes.

Top 3 Mistakes A/B Testing

The final topic before the first principles framework we want to cover is the top mistakes A/B testing. Let's go over the top three.

Mistake 1 - "Let's launch because it doesn't hurt"

There is a trap lying in wait if you make it the default pattern to launch features that have neutral results: you could be shipping something with statistically insignificant negative results.

Of course, things you have high conviction in from user testing and qualitative feedback, ship. But Because "feature bloat" is ubiquitous in products these days, I would even consider **having your default setting to be not to ship things with statistically insignificant results.**

Mistake 2 - Ratio mismatching

If the volume of your participants in each variant does not match what you planned, you may have a ratio mismatching problem. You can run a proportion test to see whether you have a bug.

Users don't just go mismatching. So go into the details to figure out your issue. The treatment may be too slow. Or you may have leakage from one experience to another. If you don't do this, your results can look artificially good.

In the search for truth, this type of bucketing skew can really lead you astray.

Mistake 3 - Not restricting analysis to exposed groups

Who would have actually have seen your experience in the A/B groups? I frequently find that automated testing software gets this part wrong. It's worth triple verifying the trigger analytics in analysis.

Because the automated testing software makes charts so readily available, many people forget this step. They make errors in their analysis, as a result.

Always verify users saw the experiment, not just that they were assigned to it.

And that's it for the educational lessons on the POV, heuristics, myths, archetypes, and mistakes. Time to put it all together.

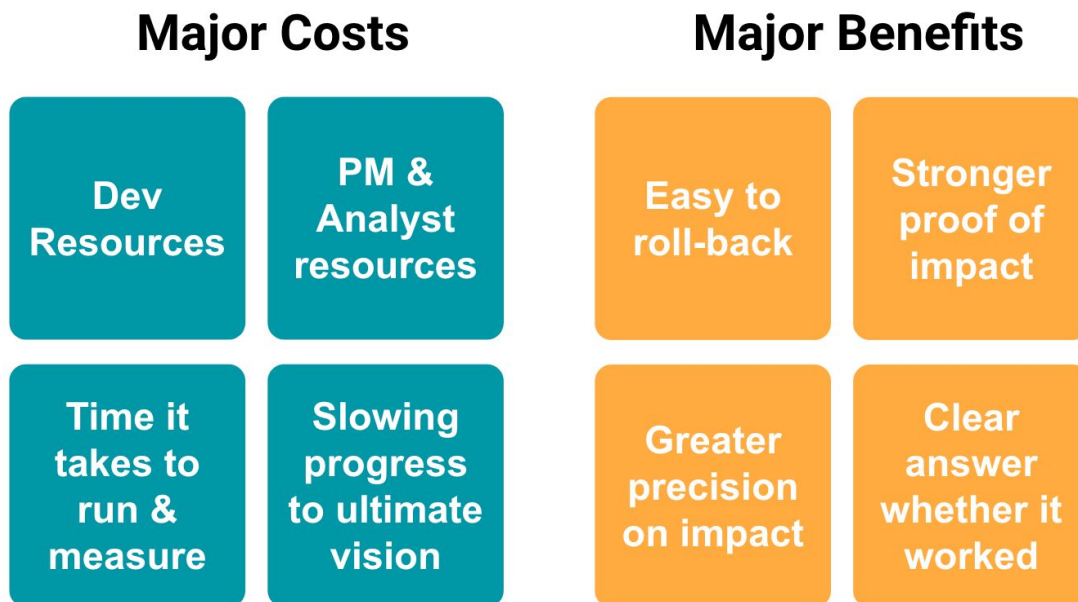
First Principles Assessment Framework

Now that we've gone through all of that, we're ready to put it together into a first principles framework to think through the hairiest decisions. It's three steps:

1. Enumerate the costs and benefits
2. Assess your company & product culture
3. Make a proposal, but highlight costs & benefits to your team

Let's go deep on each.

Step 1 - Enumerate the Costs and Benefits



Major Cost Buckets to Consider

There's a few major cost buckets you have to consider as you tip your weighing scales:

1. **Dev resources**

2. **PM and analyst resources**
3. **Time it takes to run & measure**
4. **Slowing progress towards the ultimate vision**

Let's go through each.

1. Dev resources

Dev resources are some of the most expensive at your company. Use them wisely. I have found that the additional resources required to instrument an A/B test vary tremendously from company to company. So learn how big they are at *your* company.

At Affirm, between integrating with the experiment system and adding additional data, it was often multiple days. That made us very wary of over-experimentation. On the other hand, at thredUP, it was less than an hour. So, we loved to experiment.

2. PM and analyst resources

The number one area **I would press on is not to underestimate the opportunity cost of PM and analyst resources.**

It's all well and good if your A/B testing system spits out some metrics. In my experience, there's always more work that PMs have to do beyond what's spit out there.

After all, product leaders and cross-functional partners can log into the A/B testing dashboard as well. There's **always** a pressure for high-performing PMs to do additional analysis and break-down.

This takes time and energy away from figuring out the next feature, and figuring out the most important user problems, and understanding higher leverage metrics. Those other tasks are, in many cases, more impactful for a PM to work on.

As a result, **in June 2023, I reckon too many teams underestimate the cost side of A/B testing.** 10 years ago, it was the opposite.

3. Time it takes to run & measure

Time is money. This is especially true in the world of modern product teams, where you are trying to move a certain metric. Moving a metric happens faster if you get great features out to people quickly.

It's a simple insight, but I've seen many otherwise amazing product leaders make the mistake of neglecting it. If you're barely going to hit your quarterly revenue target and you have a winning feature, consider that rolling your feature out to 50% of the population for 4 weeks means losing the incremental revenue on 50% of your population for 4 weeks.

Often, the stakes are *too high to wait* the time it takes to run & measure an experiment.

The 4 weeks part is worth double-clicking on. Why did I use that, if both the experiments sized above were about two weeks? In my experience, the time it takes to analyze an experiment is about as long as it took to run. You have to consider the time it takes to run *and measure* an experiment when enumerating the costs.

4. Slowing progress towards the ultimate vision

A fourth and final cost not to forget is that **time is really limited in product!**

Re-organizations and company strategy inevitably flow down to product, and that means your baby for 3 months that you had plans to do 5 follow-ups on is typically tossed aside for the next priority. And you never get back to those 5 follow-ups.

So if you have an ultimate vision for a product area, assume that you have very limited time to get there. As a result, an A/B test is often not a step to get there, but a step in the way.

An example is the personalized shopping feed at Affirm. We knew that it was a winner, but our intermediate milestones made it take a

year to launch instead of a mere 3 months. I had to learn that lesson the hard way.

So, I recommend you get clear on your assessment of these 4 costs for your situation. Now, we'll move on to the major benefit buckets.

Major Benefit Buckets to Consider

Just like the costs, enumerate the benefits of the A/B test. These benefits tend to all into 4 buckets:

1. **Clear answer of whether it “worked”**
2. **Greater precision on the impact**
3. **Stronger proof of the impact**
4. **Easy to roll-back**

The razor I would push you to consider before enumerating all these benefits is “what is the benefit in compared to something else we could do to learn the same thing?” There are so many other options: phased alpha-beta-GA roll-out, user interviews, session replays, analytics, surveys...

Now, let's put texture on each benefit.

1. Clear answer of whether it “worked”

A/B testing's beauty lies in its ability to provide **black and white answers in a world often dominated by grey areas**. It's a binary form of assessment. Either your feature or strategy worked, or it didn't. No guesswork, no presumptions. It's data-driven clarity at its finest. A hypothesis is generated, tested, and the results are analyzed.

When Facebook was mulling over whether to introduce reactions as an expansion of their simple "like" system, they turned to A/B testing. Rather than rely on focus groups or predictions, they tested it live, among a select user group. The results provided a clear answer - users responded positively and engaged more meaningfully with a wider range of reactions.

This clear answer becomes the guiding light for the next steps. The business impact of such clarity can be tremendous - saving not just money, but time and effort that could be better spent elsewhere.

Moreover, such clear evidence can be pivotal when presenting your project's progress and achievements to stakeholders. They might have different views, but the data from A/B testing can help you drive your point home more convincingly.

2. Greater precision on the impact

While most forms of testing can provide a high-level overview of whether a strategy worked or not, **A/B testing stands apart with its ability to provide granular insights**. It's a tool that allows you to measure the impact of your changes with utmost precision. Whether it's a significant redesign or a simple color change, A/B testing gives you a detailed understanding of how your changes impact user behavior and business metrics.

For example, Netflix uses A/B testing for various aspects of their service, including the design of their thumbnails. By running numerous A/B tests, they've determined that certain images tend to drive more user engagement than others. It's not just about whether users prefer one thumbnail over another. It's about quantifying the degree of that preference and using that knowledge to make more informed design decisions moving forward.

This precise measurement can also help identify any unexpected consequences of a change. Sometimes, a change intended to improve one aspect of a product or service can inadvertently affect another. A/B testing can help flag such unintended consequences, enabling quick corrective action.

3. Stronger proof of the impact

At times, the benefit of your feature isn't immediately clear to stakeholders. An A/B test becomes your undisputed argument. In a data-driven culture, numbers speak volumes.

For instance, there was a time at ScoutForce where we were revamping our onboarding flow. The design team was skeptical about the changes. Post A/B test, we could definitively showcase a 15% improvement in user retention. This objective proof not only earned us credibility but also boosted the morale of the team.

A validated result can increase your impact in future debates and decisions. It **can be the difference between a future feature being greenlit or red-lighted.**

4. Easy to roll-back

This is the safety net that A/B testing provides. By segmenting users, we can easily roll-back any changes that aren't having the desired effect. No harm, no foul - it's as if nothing happened.

I remember when we tried a new, edgier tone for our product notifications at thredUP. We hypothesized it would increase engagement, but in our A/B test, it significantly dropped. The beauty of the A/B test was that we could immediately revert the changes for the users in the test group.

Roll-backs are particularly critical when you're dealing with high-risk changes, like modifying pricing structures or overhauling core features. An A/B test gives you the comfort of making bold moves with a safety net attached.

An A/B test's rollback ability is **your contingency plan, a quick undo button.** But it doesn't absolve you of your responsibility to make well-thought-out decisions. It's a safeguard, not an excuse for recklessness.

Step 2 - Assess your Company & Product Culture

There are the “facts” and there’s “what works.” Even if you are the GM for the largest business in the company, there’s a point where the best operators just don’t beat their heads against the wall.

- **If you're a leader:** What does your board want? How are you going to survive the next 2 years to build your vision for the next 5 years?
- **If you're a PM:** What does your leadership want? How are you going to build ongoing confidence from your design & engineering peers in your choices.

If your company is in the “messy middle” but errs towards one of the archetypes we mentioned above, consider that.

When I was working at Epic Games, I led matchmaking on Fortnite. We had a pretty strong testing culture on that team because the system was entirely backend. It was a black box to players.

But when push came to shove, I didn't go to the COO and CEO and tell them how successful my A/B tests were. I talked to them about the north-star KPIs like playtime and revenue were. In that environment, if we had a high conviction or small feature, we would just ship it.

Use those types of subtle queues from your environment to adjust your weightings in Step 1.

That brings us to our final step.

Step 3 - Make a Proposal but Highlights Costs & Benefits to Your Team

Once you objectively assess the pros & cons of running a test, set it up for a discussion with the core team. The best way to make these decisions is to discuss the pros & cons with the people actually doing the work.

I've often found that if you get down to the brass tax of what's really behind the extra engineering work, you can find something you can build for all future experiments and remove that step entirely. So **where you are with A/B testing dev effort today doesn't need to be where it is in 3 months.**

This generally brings the debate down to two factors on each side:

- **Costs:** PM resources + time it takes to measure the impact
- **Benefits:** Greater precision of impact + stronger proof of impact

For each particular feature, your team can decide if the time is worth the precision and proof. Sometimes, it is. Other times, it is not. If you've gone through this whole process, don't sweat your decision. Just do it.

And that's it for our first principles assessment framework. We've kept the rule of three so you can actually complete it at work. Let's end with five example features to add the final layer of understanding to your toolkit before we break.

5 Examples for Texture

The theory is only helpful up to a point. These examples will add texture and help you decide further. But remember that the step 2 of our first principles framework: consider the culture you are in after you do this.

Example 1: Friction in checkout flow for fraud

- *A/B Test*

This is a clear example to A/B test. Why? Because the added friction will decrease conversion rate. Anything with a plus and a minus matches one of the 'easy heuristics' from section 1. Toss it in the testing bucket.

Example 2: Solving a critical bug

- *Just ship it*

The word 'critical' might make you pause here. Should we A/B test it? No.

When your users are facing a significant issue due to a critical bug in your product, it's a no-brainer - you need to fix the bug as soon as possible. The negative impact on user experience is already established, and delay due to A/B testing could lead to further dissatisfaction, potentially resulting in churn.

Post-launch, closely monitor user feedback and key performance metrics to ensure the issue is fully resolved and hasn't caused unintended side effects.

Example 3: Deciding whether you have more room in a lever

- *A/B Test*

Let's say you've got a lever in your product that's performing well - an email campaign with a high open rate, a feature that's used frequently, or a landing page with a good conversion rate. You wonder if you could squeeze out more performance from this lever.

This is an excellent scenario for an A/B test. You can create variations with slight modifications - a different email subject line, a new feature layout, or a revised landing page design - and compare the performance of each variation against the current version.

The goal here is not necessarily to create a drastic overhaul but to find subtle enhancements that could improve the lever's performance. Testing will provide hard data on the potential gains and guide you in making data-informed decisions.

Example 4: Evolving your product for the better

- *Just ship it*

Imagine a scenario where you've decided to enhance your product by adding new features or improving the existing ones - for example, a new design layout for better user navigation or a streamlined sign-up process.

This is a case where you're confident that the changes will result in a net positive impact and are generally improvements in the user experience. You have market research, user feedback, and data supporting your decision.

In such situations, you should go ahead and deploy the improvements without wasting time on an A/B test. You should always stay vigilant though, monitor user feedback post-launch, and be prepared to make any necessary adjustments swiftly.

Example 5: Implementing new security measures

- *Just ship it*

If the new measures will enhance user privacy or data security, it's best to implement them immediately. User trust is crucial, and improvements in this area should not be delayed.

Notice, only 2/5 had A/B tests. Sorting out features realistically, not everything has to be A/B tested! Wield the tool powerfully, not bluntly.

Advanced Topic: Bandits

This chapter has focused on A/B experimentations vs gradual or full rollout. But that's not the only universe of options. One of the most powerful new techniques is multi-armed bandits, which automatically roll-out the winning variant as they gain confidence.

Bandits balance the need for precision and the rush to outcomes that highlight the two archetypes on each end of the spectrum that we started the chapter with. If you're really interested in pursuing them, ask your data scientist.

Chapter 12: How to Deal With Tech Debt

It's so easy to focus on shipping exciting new features that sometimes we forget the importance of completing tech debt stories, and creating minimal technical (tech) debt along the way.

This is true for engineers, designers, product managers and builders of all stripes. In fact, in my experience, it is hardest for CEOs. It is so easy for CEOs to connect the dots between a feature and more growth, a stronger valuation, and greater success for the company.

For the PM specifically, one of the most challenging parts of the job is prioritizing tech debt. Then, protecting it after is nearly as difficult. As one of our primary stakeholders is the CEO, we are constantly trying to build to their vision.

This creates a perceived tradeoff between career growth, impact, and tech debt in the short term.

So, how do you deal with tech debt as a PM? In this chapter, I'll cover:

- What the Tech Debt is
- Its Impact
- Role as a PM
- Strategies to Address

Let's get into it.

What the Tech Debt is

As a Product Manager, hearing the words technical debt from an engineering lead can be scary.

Regardless of how technical you are, we all know that debt is bad right?

Well just like debt in the real world, it's not that easy and sometimes it's necessary.

Think about buying a house. Would you rather save up the full purchase price of a home or just the down payment?

On one hand you don't incur any debt but the trade off is waiting longer to save for the full purchase price. On the other hand you can move into your new home quickly if you're only putting a fraction of the purchase price down on the home, but the tradeoff is that each month you have to allocate a certain amount of your salary towards paying on your mortgage debt.

Now let's take this same concept and apply it to shipping software products.

As a Product Manager are you willing to incur technical debt to ship faster? Going back to the analogy, what if you find the perfect house in a great school district. Would you allow someone else (a competitor) to come in and swoop it up while you're stuck saving your money? But on the other hand, at what point do your debt payments take such a significant amount of your salary that you can no longer afford to do the things that you love (ship new features)?

Hopefully you see the complexity now and understand that it's all about balance. We as Product Managers have to weigh speed to market against the technical debt load that we may take on.

Now that you understand what tech debt is, let's dive into the three main types of tech debt.

How it was created

There are three main types of tech debt. It's worth understanding what drives each.

Deliberate tech debt

Many times, when building, we have a goal. Perhaps it's a quarterly OKR, or it's a date a game ships, or it's a short release window.

These goals cause us to sacrifice the long term for the short term. So, we have a discussion as a development team, “shall we go for the quick way over the right way?” And sometimes, we choose the quick way.

So tech debt is not inherently a bad thing. This is “good” tech debt that we incurred to push out an MVP. But it’s “bad” in that there is a future cost.

For those out to eliminate tech debt, the good thing about this type of tech debt is that it’s a known commodity. The team knows where it exists, what problems it might cause in the future, and when the team must absolutely fix it. This usually aids in prioritization and discovery decisions.

Unfortunately, as development teams it is exceedingly hard to document all of this. Especially when team members move on, sometimes deliberate tech debt becomes accidental.

As a result, it becomes useful to have an understanding of some of the common types of deliberate tech debt to be able to identify it:

- Not deleting dead code after an experiment result or unshipping product decision
- Not having the tests, monitoring, and alerting in place; as a result, bugs are found in production
- Deployment and build times take several hours or days
- Developers lack the proper tools to allow them to detect technical issues before they go in production
- The team prioritizes one platform over the other, so one has faulty code spots that cause bugs and crashes for corner users on the lesser platform
- A third party tool or tech stack choice no longer scales for the organization’s complexity, metrics, and number of users

These types of issues come up regularly throughout the course of moving fast to build software. A good dev and product team makes

these decisions strategically, but also comes back to address them smartly.

Accidental tech debt

The nature of accidental tech debt is that it often arises over the course of the time. Anytime you build a teal, it immediately becomes outdated. The app, and the underlying technology evolve.

Moreover, sometimes, requirements change or the competitive environment inverts. These cause the existence of either code, systems, or entire features that are no longer needed - or actively hurting the existing product.

Accidental tech debt has the insidious problem of typically being more hidden than deliberate tech debt. It might be that iOS has updated all its functions and the mobile engineers need to refactor some of the code. It may be that the website has something built on a web standard no longer supported. These things sometimes only hit the product team's radar after a user has experienced issues.

Rot tech debt

The final type of tech debt is that which arises from the constant feature treadmill. We rush to add capabilities and features to a system such that it grows to become ugly, complex, slow, and hard to update. This often happens when the original team only understood part of the future requirements, so they built a solution that solved a very specific use case. Then in future instances, maybe years later, teams bolted on other features.

The team feels like they are biking as fast as they can but eventually the systems get to a place where they biking in stationary.

All the compounding bolting on off features causes the overall gears of the system to start to turn more slowly. Eventually the circular wheels that were the initial system become squares, hard to budge without the most exceeding amount of effort.

Its Impact

Tech debt comes in all shapes and sizes. There is the small, innocuous manual config each time you build. It adds a few minutes, but causes issues if you do not. There is a system check that doesn't happen automatically so you need to do by hand.

The nastier, sinewy type is that the larger inefficient systems and processes that slow future development or make scaling difficult, if not impossible.

Slowing future development

Take the manual config and multiply by 10. Tech debt that slows future development adds hours to future changes. This might be because of choice of technology, system architecture, or decisions by prior engineering efforts.

This type of work can sometimes become accepted organizational processes. As a result, it further ensnarls itself in the culture by fooling developers it has to be that way.

Let's say tech debt makes you 10% slower at implementing features. If a team of 8 engineers deploys one feature every other two week sprint, the expense of 10% tech debt is 13 features over the year. If it only takes a few sprints to complete, the ROI for solving this type of tech is fairly high.

Making scaling difficult or impossible

Another class of strategic tech debt is that which makes scaling difficult or impossible. Sometimes, this tech debt is known, but more often than not, it needs to be encountered.

When we were running season events at Fortnite and breaking records season after season, we ran into a concurrent user limit every season. That was because we were doing something so cutting edge that the tech had literally not been built for. At certain levels of scale, scaling can be difficult unless you address tech debt.

Role as a PM

To oversimplify it, tech debt is loved by stakeholders (ship fast), hated by engineers (ship good), and on your shoulders as a Product Manager to figure out who to make happy.

So how can you possibly win in this situation? It's easy. Do. Your. Job.

You know that classic venn diagram that illustrates the role of the PM being at the intersection of UX, Business, and Technology?

Well this is where you earn your check at the crossroads of Business and Technology.

As a Product Manager you have to translate the Business needs (and speed) to your engineering team (they don't usually like Business timelines) and the current state of your tech stack to your Business stakeholders (they don't usually like technical constraints.). The more excited and aligned that you can get both groups around a shared vision the easier that your job becomes.

All of that being said, two of the most important roles for a PM to address tech debt are: 1) to prioritize it enough, and 2) to understand when to and when not to.

Prioritize Enough

As the ultimately accountable order for what engineers build, it is up to the PM to prioritize technical debt enough. But there is a goldilocks amount here. A PM that over-emphasizes tech debt becomes loved by engineers, but not quite as much by product leadership and executive stakeholders accountable for growing the business.

So how can a PM prioritize just enough, but not too much? One of the first things to do is to properly assess each tech debt element. Is it acute or systemic? Another thing to do is understand its strategic

impact. Can leaving it in place enable other, more important features. And what about the impact on eng's morale?

Also understand the team's confidence in these claims, as well as the time it will take to fix the issue. After that, try to make an informed prioritization decision.

One can probably not expect to ever get it quite right. Nevertheless, at every planning cycle, PMs should ask the question after evaluating an early draft outcome: "are we prioritizing tech debt enough?"

Then, at the end of each cycle, foster a discussion with the dev team. Ask, "did we address the right tech debt this quarter? Was our focus too much or too little? And what are the top priorities for next quarter?" These questions help teams adjust to the individual level that makes sense for them given their capabilities and business context.

Understand When to and When Not to

These prioritization questions get to the heart of the skill which is to be able to understand when, and when not to, address tech debt. There are a couple key factors to consider:

1. Developers' perspectives
2. Business impact
3. Tradeoffs on other initiatives

Part of the reason the PM role exists is to negotiate these details at the lowest level of detail. You should roll up your sleeves and write out the thinking. What do devs think? What is the business impact? What is the tradeoff? These things in paper help everyone on the team make a better decision.

The PM should also develop their own perspective. The goal should be to, as always, drive business outcomes by solving user problems, creating growth levers, and keeping the team happy. If the team is

unhappy, that might mean prioritizing tech debt more than normal. If the business outcomes are critical, then it might have to wait.

Strategies to Address

Now that we understand what tech debt is and what PMs are expected to do about it, how can PMs go about doing a better job of that? It starts with tracking and developers, and goes all the way to preventing pile up.

At each step of the PM's role in the software development process, a PM who is great with tech debt can stand out.

Track It In the Same Place

If tech debt tasks do not end up in the backlog, they will never get executed. Track tech debt issues in the same backlog you use, and hold them accountable to the same standards. In many orgs, that means having a spec with the key points in place.

This also has the benefit of constantly reminding the team of the key tech debt issues that exist. Otherwise they can easily feed into the backlog. Even if the items continually get deprioritized, at least a discussion about them is happening.

The last thing you want is to have a separate backlog of uninspiring tech debt items and an exciting set of features. That makes the tradeoff decisions too unrealistic. In the same place, they are more easily weighed.

Empower Developers

Empower developers to find, suggest, prioritize, and address tech debt. At each step, they will feel happy to be included, and feel better about the decisions being made of which tech debt to prioritize.

When finding tech debt, give developers time. Actually make it a task occasionally. Build processes around understanding which

developers are the most effective at finding tech debt so they can be rewarded and recognized.

In addition to empowering developers to find tech debt, provide tools to empower developers to add tech debt items to the backlog themselves. That way they feel as little friction as possible in suggesting ideas. Then apply the same rigor as you normally would in evaluating the estimated work, impact, and risk to prioritize the items appropriately.

Then, in the prioritization conversations, regularly ask developers to provide the impact and work estimates. Then ask questions to help them be rigorous with it. Encourage their other developer team members to push the thinking. If there is a tech lead, lean on them.

Finally, when solutioning, avoid being the answer-teller. Set the stage for developers to think through innovative solutions that are not only at the bleeding edge of what is available, but scalable to avoid significant rework in the future.

Regular Conversations

The fashionable thing to say is to make technical debt a part of every conversation you have with your eng team.

The problem with this is, it's more a tweet than a practicality. The best PMs help eng teams focus on the right thing in each meeting. Don't bring up everything in every meeting.

Instead, set up a regular forum and cadence where people come prepared to review and discuss tech debt items. This helps make tech debt a first class citizen in the ideation and discovery process, similar to features for users. Ask curious questions. Follow threads down to fine details to understand what is actually causing the problem.

When you are taking on tech debt, understand what the implications are. Ask what the shortcut saves the team now. Follow up with the

estimated impact to resolve the issue later. This information empowers you to make better decisions.

Be an Advocate

In the conversations, be the PM who cares. Dive into the details to understand them. Then follow up on the conversations by implementing the other strategies to have actions back up your words. This helps developers consider you as an “advocate” for great tech systems and minimal tech debt. Over time, developers will want to work on your team.

In addition, double click on the discovery piece. Like you interview users for their problems, interview developers. Instead of going to what developers think they should build, start with the problem. This will help them clarify their own thinking as well.

Finally, in conversations with executives and product leadership, be the advocate for tech debt work. In all the places your dev team is not, deliver on the mission of having less tech debt. Sometimes this might mean spinning up entire company wide initiatives with executives. Other times, it might mean helping carve out protected portions of developer time for this type of work.

Measure & Goal The Team On It

One of the best ways to optimize something is to measure it. By putting a KPI on tech debt, you help the team measure and improve their processes to addressing tech debt. This can not only be a unique way for the PM to add value, it also is motivating team members. Everyone likes to crush a number.

To ensure that commitment to the metric, make it a goal of the team's. Put it in official OKRs if it makes sense. This type of specific goaling helps everyone get united around the cause, and prevents derailing of projects halfway through.

Prioritize It

With a specific KPI goal in mind, prioritizing the work becomes slightly easier. But even if the work is not a committed KPI communicated to executives, make sure to be the PM who prioritizes the work.

Many PMs include a certain portion of time or sprints to focus on tech debt. This might be the right solution for your team. Consider whether they are comfortable with flexing into and out of tech debt work, or if they prefer a constant stream.

Keep it prioritized

One of the hardest challenges is not chasing after the next shiny object. Executives, sales, and other stakeholders constantly come with opinions on what the team build. Well, what should the team cut to fit the new feature? Tech debt is one of the first things to come up.

This is why it is so important, then, for the PM to use the weaponry from their regular conversations. PMs should speak to the specific details of the problems, and why the estimated impact is the way it is.

Prevent it from piling up

Finally, stop creating tech debt! Prevent it from piling up in the first place by helping the dev team to plan reasonable workloads for sprints. Every dev will need to create tech debt if they are asked to do a month's worth of work in two weeks. But if asks are reasonable, less tech debt piles up in the first place.

But, if it looks like things might be piling up, act quickly. Consider having full sprints dedicated to flush it out if necessary. At Google, it was not uncommon to see teams spend an entire month of the year on scalability and tech debt.

Customizing Your Style

None of these strategies or tactics should be applied without consideration for your context. Tech debt is a big and varied topic,

and the role of PMs in shaping varies company to company.

At some companies, tech leads and architects live to identify and prioritize the work. At others, it is more of the PM's role. In that case, the process described above can be particularly useful.

Regardless of the type of company and role of PM on paper, each PM and dev team will have a different situation in every company. To be a truly great PM, the key is to identify stakeholder motivations and work with them.

- Why are we generating tech debt?
- Do engineers feel like we are addressing tech debt enough?
- Are we prioritizing tech debt enough?

Asking first principles questions like this will help you develop a strategy for your team. Maybe your team feels like they are doing enough with tech debt, but they need to be pushed to think bigger. In other cases, maybe they are over-prioritizing infrastructure initiatives.

Determine where the team is and where the team needs to go. Some of these strategies can help along the way.

Towards Tech Wealth

The inverse of tech debt is tech wealth. This is the idea of prioritizing tech debt strategically, as well as building systems and capabilities in advance, to enable tech capabilities as a strategic advantage in the marketplace. As a PM, especially a platform PM, you might consider helping your team push towards tech wealth.

For instance, the team might decide that building an experimentation platform is a good decision for scale that it will experience over the next couple of years. A simple tool like Optimizely might work for some parts of the funnel for a small part of users in the short term. But the team decides to build an in-house platform to enable a culture of rapid experimentation. That is a tech wealth mindset.

To enable a tech wealth mindset, start by getting specific about the tech debt you are talking about. Then use the tools in this article - from track it in the same space, empower developers, and the rest - to transform your team into one that uses tech debt conversations to grow the business.

Epilogue

The learning doesn't end here! Every book purchaser gets access to a free month of the paid newsletter.

Head over to: <https://www.news.aakashg.com>.

Each week, you get something like a chapter in this book e-mailed to your inbox.

Just reply to the first email you get after signing up with the book receipt.

References

The complete references section was 30 pages, and most books have forgone printing those nowadays. (Because each page comes out of the author's profit margin.)

You can find all the detailed references used throughout the book at <https://www.aakashg.com/2023/06/24/references/>

You can also email me at aakashproductgrowth@gmail.com if you have any specific reference requests.

Acknowledgments

I couldn't have written this without my amazing wife and two kids. Thank you, Amy, Aviv, and Anjali. The zest you all bring to life inspires me daily. This is all for you.

I also have to thank my parents and wonderful brother. Thanks for providing the best environment to prepare me to write this book.

In addition, I wrote four of the chapters with four other writers:

- *Key PM Skills at Each Level*: Ibrahim Bashir - VP Product at Amplitude
- *The Languages of Product Management*: Aatir Abdul Rauf - Director of PM at vFairs
- *How to Impact Size*: Carl Vellotti - Senior PM at GoodRX
- *How to Deal with Tech Debt*: Brennan Decker - PM at Fanatics

These individuals are top-tier writers and this book is due in part to them.

I also have to thank all of my employers and former mentors. I also want to thank the teams at Epic Games, Affirm, and Apollo.io who supported my writing.

Finally, but certainly not least importantly, my high school English teacher Kirk Latimer. There is no more inspiring teacher I have ever had. "Mr. Latime," as I'll always call him, unexpectedly passed at age 40. He is sorely missed.

I couldn't have done it without all of you.