Product Management

Peter K. Shultz, Product Manager at Google

About me

- Hi! I'm Peter
- I graduated from UM in 2018
  - Studied CS and economics
- Things on my resume:
  - Program Manager at Microsoft (2018 - 2022)
    - Focus: high-performance computing, data science
  - Product Manager at Google (2022 - present)
    - Focus: data infrastructure, datasets, experiments (A/B tests)

About this talk

- My views, not my employer’s
- I want to teach you about:
  - Product management
    - If you like it, how to pursue it
    - If you don’t like it, how to use PMs effectively
  - General work life
    - Since many of you are eager and ambitious: how to get ahead

What is a PM?

- Lots of different perspectives:
  - “PMs make sure companies build the right things, and software engineers make sure the things are built right”
  - “PMs deal with things so that software engineers don’t have to”
  - “PMs are the voice of the customer for product development”
  - “PMs make things happen”
  - “PMs dream it; engineers build it; TPMs explain why it’s late”
    - PM = product manager; TPM = technical program manager

Understand, Prototype, Build (UPB)

A Framework for Product Development Alignment

Understand	+	Prototype	+	Build	=	Right Problem, Working Solution
~~Understand~~	+	Prototype	+	Build	=	Solving Wrong Problem
Understand	+	~~Prototype~~	+	Build	=	Backtracking When It Doesn't Work
~~Understand~~	+	~~Prototype~~	+	Build	=	Wrong Problem, Poor Solution, Train Wreck

Source: Raph Lee, VP of Engineering, Lob - bitly.com/upbsource

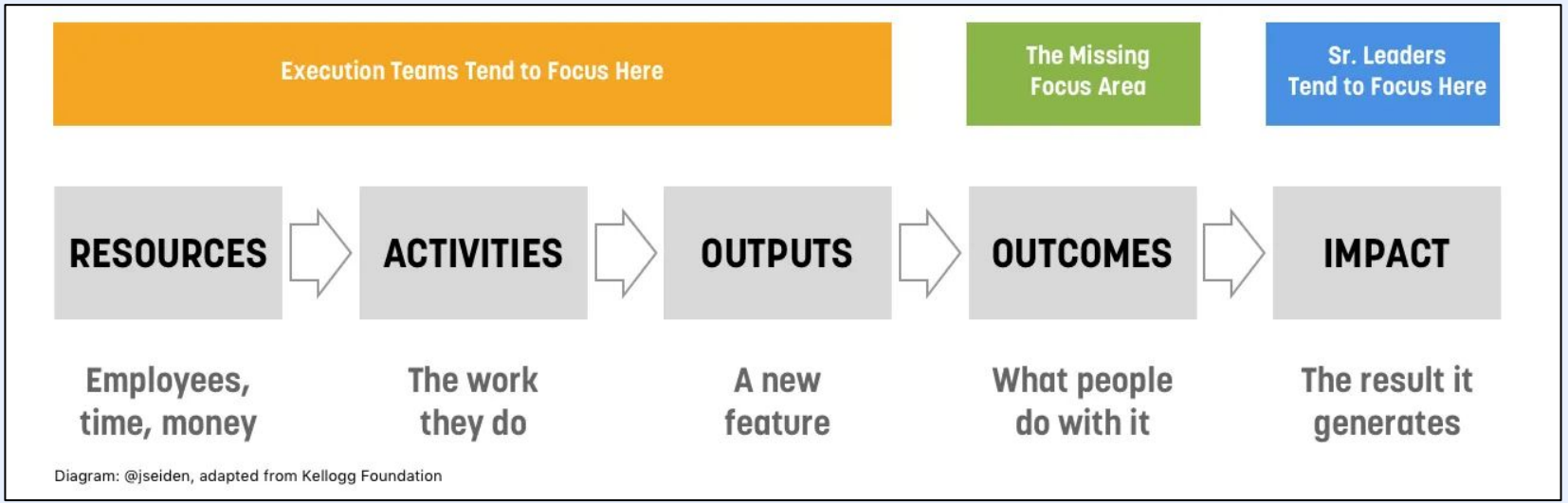
What is a PM?

- Lots of different perspectives:
  - “PMs make sure companies build the right things, and software engineers make sure the things are built right”
  - “PMs deal with things so that software engineers don’t have to”
  - “PMs are the voice of the customer for product development”
  - “PMs make things happen”
  - “PMs dream it; engineers build it; TPMs explain why it’s late”
    - PM = product manager; TPM = technical program manager
  - “PMs help understand and prototype”

What is a PM?

- Lots of different perspectives:
  - “PMs make sure companies build the right things, and software engineers make sure the things are built right”
  - “PMs deal with things so that software engineers don’t have to”
  - “PMs are the voice of the customer for product development”
  - “PMs make things happen”
  - “PMs dream it; engineers build it; TPMs explain why it’s late”
    - PM = product manager; TPM = technical program manager
  - “PMs help understand and prototype”
- Number of PMs on a team varies by how technical the product is
  - 1 PM for every 10 engineers is common for consumer products
  - My current team: 3 PMs for about 100 engineers

But what is there to understand?



A. Understand tech enough to explain ***activities*** and ***outputs***
B. Understand people enough to explain ***outputs*** and ***outcomes***
C. Understand business enough to explain ***outcomes*** and ***impact***

Okay, but what do you actually do?

A. Understand tech enough to explain ***activities*** and ***outputs***
    a. Meet with engineers to understand how a feature works
    b. Write code to prototype a feature or do some data analysis
B. Understand people enough to explain ***outputs*** and ***outcomes***
    a. Interview customers to understand their issues/pain points
    b. Write proposals to solve issues/pain points
        - Persuade leads and engineers to work on the proposal
            - “Influence without authority”
    c. Write specifics (e.g. requirements) for a new feature or product
C. Understand business enough to explain ***outcomes*** and ***impact***
    a. Write launch announcements explaining a new feature to the company
    b. Present at executive meetings and explain whether we’ll hit our goals

This sounds cool!

- Good news: you're already in the best UM course for becoming a PM
  - If software engineering were as cut and dry as, idk, building a bridge, product management as a discipline may not be necessary
- To confirm you'd like the job, evaluate:
  1. Your communication skills
     - A majority of the job is communicating your ideas and persuading others. Make sure you're comfortable writing and speaking frequently.
  2. Your conflict resolution and coordination skills
     - You need help from teams A and B to launch feature X. How do you work with both teams (whose priorities are very different from yours) to make things happen?
  3. Your comfort in and around problems without a clear answer
- Stay technical for as long as you can: (A) it's useful and (B) engineers will like you more

This sounds cool!

- Options for getting a PM job:
  A. Internships
     - apmlist.com has links to programs with statuses (near bottom of page)
     - Any experience > no experience
  B. First job out of college
     - apmlist.com also has links for this
     - Differentiate yourself by using products critically and suggesting improvements
  C. Start as a software engineer, move over to product management when convenient
     - Experience with a particular product, team, or set of customers will help
- Note: none of the PMs I work with have CS degrees, so you're already a step ahead

Source
Don't get discouraged!

No thanks, I’d rather code

- You can use PMs to make your job easier:
  - Confused about how a feature should behave?
    - ↳ Ask your PM to clarify
  - Random sales people asking for your help?
    - ↳ Forward to PM so you can get back to work
  - Worried your feature isn’t impactful?
    - ↳ Ask PM to share user data, anecdotes, relationship to team goals, etc.
  - Having trouble working with some team? Some leader?
    - ↳ Loop in the PM to help debug and fix the situation

So how is this class relevant, anyway?

- Requirements and specifications
  - Requirements as contracts
  - Difficulty of fixing missing requirements later
- Elicitation, validation, and risk
  - When talking with customers, ask questions to ensure you understand the root cause of their pain
    - “Customers can't tell us everything. They are experts in pain, not solutions” – April Dunford
  - The book *The Mom Test* talks about how you can phrase questions such that customers don’t lie to you out of politeness

Do you have examples?

- Requirements as contracts:
  - SLIs as performance requirement
  - SLO as reliability requirement

$$\underbrace{\left(\frac{\text{Good Events}}{\text{Total Valid Events}}\right)}_{\textbf{SLI}} \geq \underbrace{\textbf{Target \% over a period of time}}_{\textbf{SLO}}$$

A general definition of SLI and SLO

$$\underbrace{\left(\frac{\text{29 On-Time Pipeline Completions}}{\text{30 Total Scheduled Runs}}\right)}_{\textbf{SLI}} \geq \underbrace{\textbf{95\% over 30 days}}_{\textbf{SLO}}$$

An example of how SLIs and SLOs are applied to projects

Things to know about work generally

- School is about *finding an answer*; work is about *making things happen*
  - School assignments generally have one correct answer
  - At work, the “correct answer” is whatever gets people to agree on a plan, implement the plan, and evaluate the outcome
    - “Progress over perfection”
- You can get ahead by reading books and/or taking an interest in professional improvement
  - Really! That’s it!
  - This assumes you take what you learn and apply it
    - Even better than applying them yourself is helping others apply them

Things to know about work generally

- Short, no-nonsense books and articles that changed the way I think:
  - *The Effective Engineer: How to Leverage Your Efforts In Software Engineering to Make a Disproportionate and Meaningful Impact*
    - Learn the difference between good and great engineers
  - *Debugging Teams: Better Productivity Through Collaboration*
    - Learn how to work most effectively with others
  - *How to Make Things Faster: Lessons in Performance from Technology and Everyday Life*
    - Learn how to effectively apply the engineering mindset *and* work well with others
  - Fermi ROI: Fixing the ROI rubric
    - Learn a useful shortcut for prioritizing and evaluating impact

Things to know about work generally

- There is no secret to getting promoted–simply prove you can handle more responsibility
  - You can prove this by:
    - Being professional
    - Doing great with your assignments
    - Taking an interest in other assignments
    - Teaching others
  - “The reward for good work is more work”
    - “More work” here generally means “more scope” ≈ more responsibility

Looking back

- Q: What non-engineering classes should I take?
  - Philosophy
    - More reps of *read → understand → critique → improve*
      - This is the basis for a lot of professional work

Looking forward

- Q: Do you use AI at your job?
  - Yep! Great way to defeat blank page syndrome
    - Some things to keep in mind:
      - "Your job is to deliver code you have proven to work"
        - Nothing replaces being able to understand things
      - "A computer can never be held accountable"

> Google is this year for the first time factoring AI use into some employee performance reviews for software engineers, according to people familiar with the matter. Teams and managers have discretion to evaluate AI use as part of performance based on individual roles and responsibilities, but they aren't required to, according to a Google spokeswoman. She added that the company encourages all employees, regardless of role or level, to incorporate AI tools into their daily work.

Source

Before you leave college

- Reach out to anyone with a job/occupation you're interested in and ask them lots of questions
  - The "I'm a student" introductory line is a cheat code
  - That being said: building things is generally more useful than networking
    - Small blog post about this: Do Things, Tell People (notice that "Do Things" is first!)
- Think about what it'd be like to start your own business
  - **Best case**: you start something, learn a ton, and have a great outcome
  - **Worst case**: you become a better professional by realizing the amount of ownership/responsibility needed to run your own thing
    - Responsibility helps with promotions
  - **Good book**: *HBR Guide to Buying a Small Business* (not the same as starting a business, but gets you in the mindset of "Yeah, I could run a business by myself")
  - This is a useful exercise for aspiring PMs. Finding an underserved problem that you can charge people for is a great way to show you know your customers

Email me!

- Happy to help you with any questions: pshultz@umich.edu