FEBRUARY 17, 2023

But What Does an Engineering Manager Actually *Do*?

Alex Peattie
Senior Engineer Mgr

@ Front

When I joined Front...

The company was an emerging SaaS unicorn

We had 1000s of customers & were growing fast

We were 200+ employees

but...

I would be joining as the first externally-hired EM

and in my first weeks I got asked a lot...

A question I had asked myself back in 2015!

What does an EM actually do?

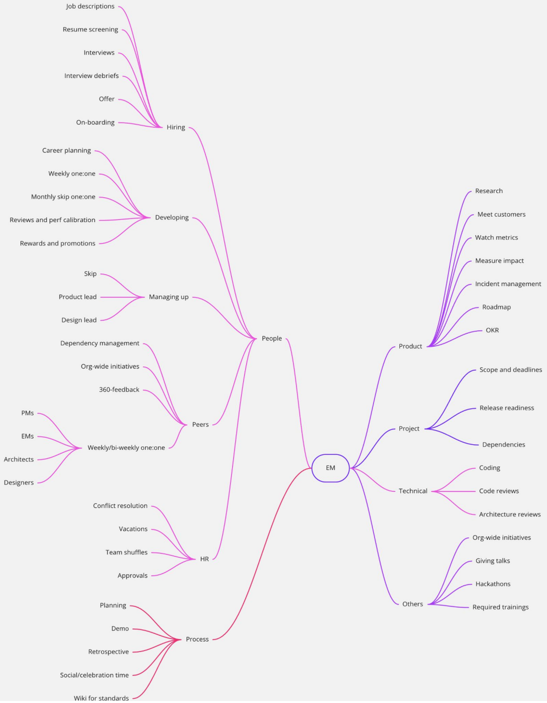
Explained in 10 ideas 💡

Agenda

- The **6 buckets** 🪣 where I spend my time as an EM
- The **3 meta-skills** 🪄 that I struggled to master
- The **1 framework** 📏 I use to evaluate if I’m being an effective EM

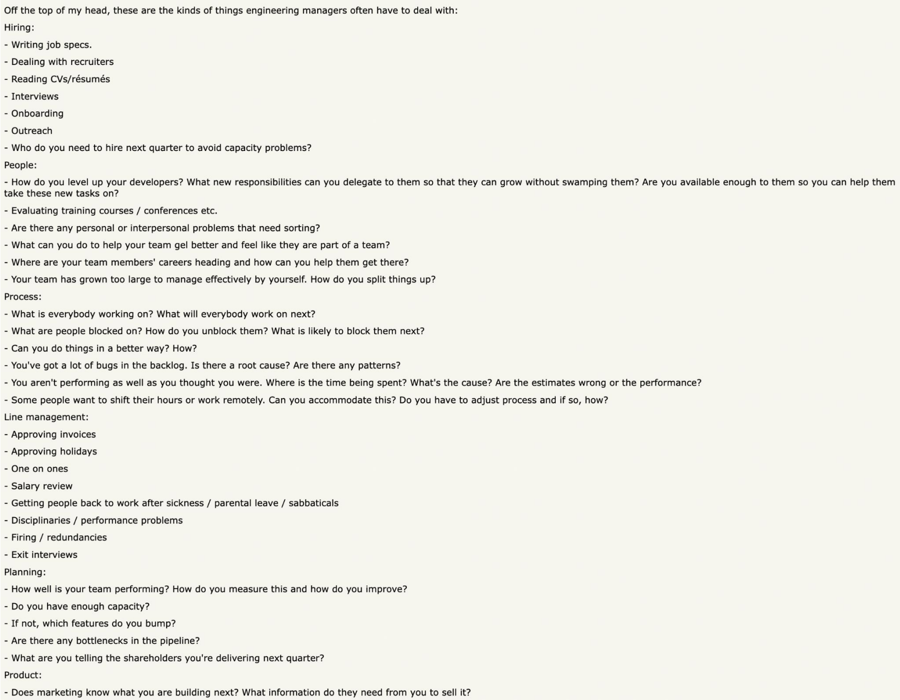
The 6 time buckets for EMs 🪣

Prior art



- EM
  - People
    - Hiring
      - Job descriptions
      - Resume screening
      - Interviews
      - Interview debriefs
      - Offer
      - On-boarding
    - Developing
      - Career planning
      - Weekly one:one
      - Monthly skip one:one
      - Reviews and perf calibration
      - Rewards and promotions
    - Managing up
      - Skip
      - Product lead
      - Design lead
    - Peers
      - Dependency management
      - Org-wide initiatives
      - 360-feedback
      - Weekly/bi-weekly one:one
        - PMs
        - EMs
        - Architects
        - Designers
    - HR
      - Conflict resolution
      - Vacations
      - Team shuffles
      - Approvals
  - Process
    - Planning
    - Demo
    - Retrospective
    - Social/celebration time
    - Wiki for standards
  - Product
    - Research
    - Meet customers
    - Watch metrics
    - Measure impact
    - Incident management
    - Roadmap
    - OKR
  - Project
    - Scope and deadlines
    - Release readiness
    - Dependencies
  - Technical
    - Coding
    - Code reviews
    - Architecture reviews
  - Others
    - Org-wide initiatives
    - Giving talks
    - Hackathons
    - Required trainings

S M Sohan / Peter Hess



Off the top of my head, these are the kinds of things engineering managers often have to deal with:

Hiring:

- Writing job specs.
- Dealing with recruiters
- Reading CVs/résumés
- Interviews
- Onboarding
- Outreach
- Who do you need to hire next quarter to avoid capacity problems?

People:

- How do you level up your developers? What new responsibilities can you delegate to them so that they can grow without swamping them? Are you available enough to them so you can help them take these new tasks on?
- Evaluating training courses / conferences etc.
- Are there any personal or interpersonal problems that need sorting?
- What can you do to help your team gel better and feel like they are part of a team?
- Where are your team members' careers heading and how can you help them get there?
- Your team has grown too large to manage effectively by yourself. How do you split things up?

Process:

- What is everybody working on? What will everybody work on next?
- What are people blocked on? How do you unblock them? What is likely to block them next?
- Can you do things in a better way? How?
- You've got a lot of bugs in the backlog. Is there a root cause? Are there any patterns?
- You aren't performing as well as you thought you were. Where is the time being spent? What's the cause? Are the estimates wrong or the performance?
- Some people want to shift their hours or work remotely. Can you accommodate this? Do you have to adjust process and if so, how?

Line management:

- Approving invoices
- Approving holidays
- One on ones
- Salary review
- Getting people back to work after sickness / parental leave / sabbaticals
- Disciplinaries / performance problems
- Firing / redundancies
- Exit interviews

Planning:

- How well is your team performing? How do you measure this and how do you improve?
- Do you have enough capacity?
- If not, which features do you bump?
- Are there any bottlenecks in the pipeline?
- What are you telling the shareholders you're delivering next quarter?

Product:

- Does marketing know what you are building next? What information do they need from you to sell it?

Jim Dabell

People

Process

Planning & execution

Hiring

Technical leadership

Interfacing

Key idea 💡: in each bucket you should be spending time on *reactive* work 🔔 and *proactive* work 💪

People

- Arguably the most important bucket, this encompasses your regular 1:1s, coaching, career development, feedback, conflict resolution, performance evaluation etc.
- Don’t over-optimize this bucket, there’s no replacement for time spent with your reports
- Tip 💡 : the way to be efficient here is to catch problems early. For example, ask probing questions in your 1:1s to identify brewing issues
- Reactive work 🔔 = 1:1s, resolving conflicts; proactive work 💪 = career planning, coaching

⚙ Process

- Defining and driving ongoing processes to support your software lifecycle, e.g. sprint planning, grooming, estimation, QA, retros etc.
- Different companies at different stages have different needs wrt. to process. In my experience: start lean and add process as needed. I.e. aim for Minimal Viable Processes!
- Tip 💡 : this is a rich area for delegation. Avoid having a set of processes purely imposed from the top-down, have your team help to define and run them
- Reactive work 🔔 = running your processes: grooming, sprint planning, retros; proactive work 💪 = evaluating and improving your processes

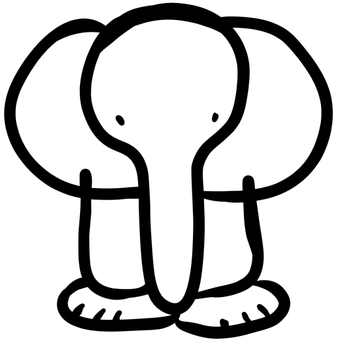
Planning & execution

- As an EM you’ll ultimately be responsible for your team delivering quality software on schedule (or as close to it as possible!)
- In this bucket you’ll be building roadmaps, identifying dependencies, balancing priorities (features, maintenance, tech debt) etc.
- Reactive work 🔔 = ensuring the delivery of immediate-term roadmap; proactive work 💪 = planning over a longer horizon, thinking about 2nd order effects, managing long-term tradeoffs

Hiring

- This bucket typically flexes the most - it could be anything from 5 - 30 % of your time depending on how aggressively your team is growing
- Tip 💡: I've found EMs bookending the hiring process is most effective here. EMs doing phone screens should be seen as a time-saver, not a time-sink
- Reactive work 🔔 = running the hiring process, outbounding efforts; proactive work 💪 = evaluating and optimizing the hiring, calibrating hiring bar, contributing to employer brand

Technical leadership



- The elephant in the room: should Engineering Managers be coding?
- If so, how much?
- (My take) The short answer: No
- The longer answer: EMs should avoid coding in the critical path

Technical leadership

- The “Engineering” part of an EM role, helping your team make the right technical decisions and intelligently weigh tradeoffs
- Tip 💡: If you work with a tech lead, clearly define your areas of responsibility. For me, my tech leads are the technical decision-makers, but I can a) consult, and b) ensure the process of decision, validation and team buy-in runs smoothly
- Reactive work 🔔 = reviewing technical designs, helping with technical decisions, being a technical tie-breaker; 💪 = investing in tooling (off-the-shelf or custom) to improve team productivity, observability etc., researching emerging technologies, mapping out areas of tech debt and risk

Interfacing

- The people work that doesn’t concern your direct reports
- Working with your cross-functional peers, managing up, broadcasting info to the wider organization etc.
- This bucket can also flex a lot depending on the stage and culture of your company
- Reactive work 🔔 = peer 1:1s, stakeholder updates; proactive work 💪 = building cross-org bridges, proactive efforts to make your team’s work more visible, anticipating future dependencies

A useful exercise: check your buckets 🪣

👥 25% 🙂	⚙ 10% 😐	🗺 15% 🙂
🤝 10% 🙁	💻 30% 🙂	💬 10% 😐

- Where am I spending too much time?
- Where am I spending too little time?
- Where do I feel least confident, and how can I move the needle there?

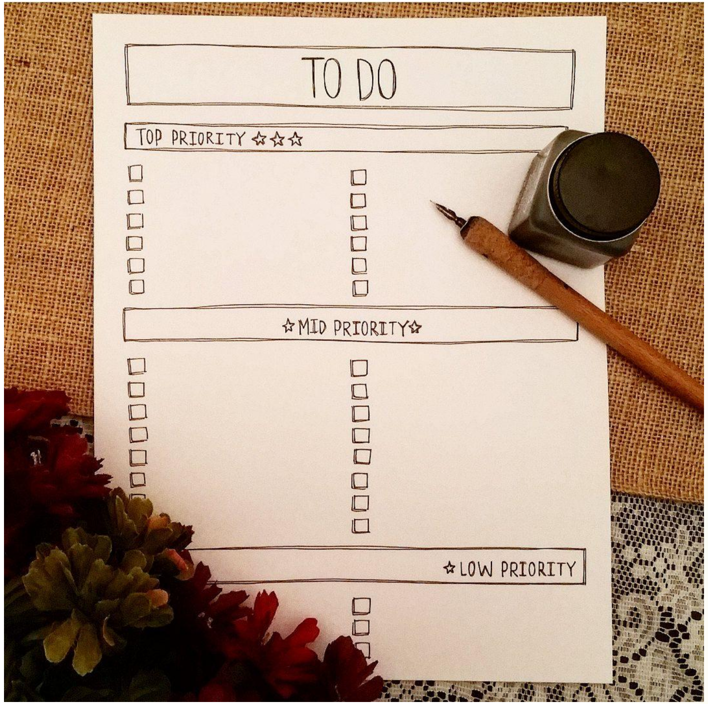
The 3 meta-skills to master 🪄

Meta-skills

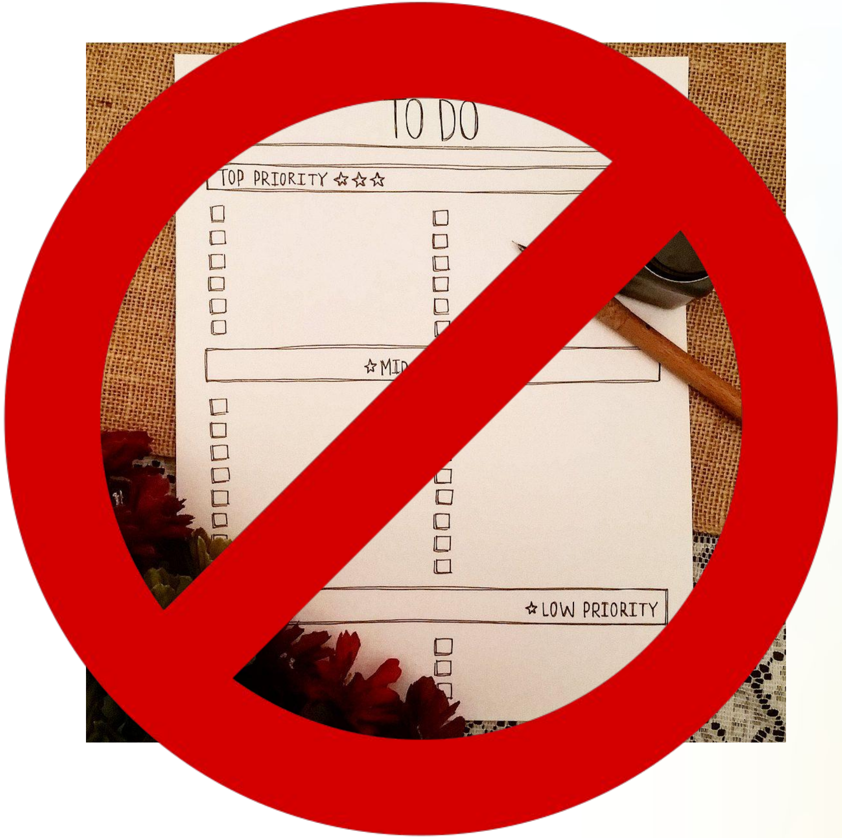
- **Aggregation**: having visibility on all the signals (internal and external) relevant to your teams work
- **Prioritization**: being able to spot the signals that need to be addressed now, and those which can be addressed later
- **Delegation**: scaling yourself by radically delegating, focusing on outcomes not tasks

Aggregation

Prioritization



Prioritization



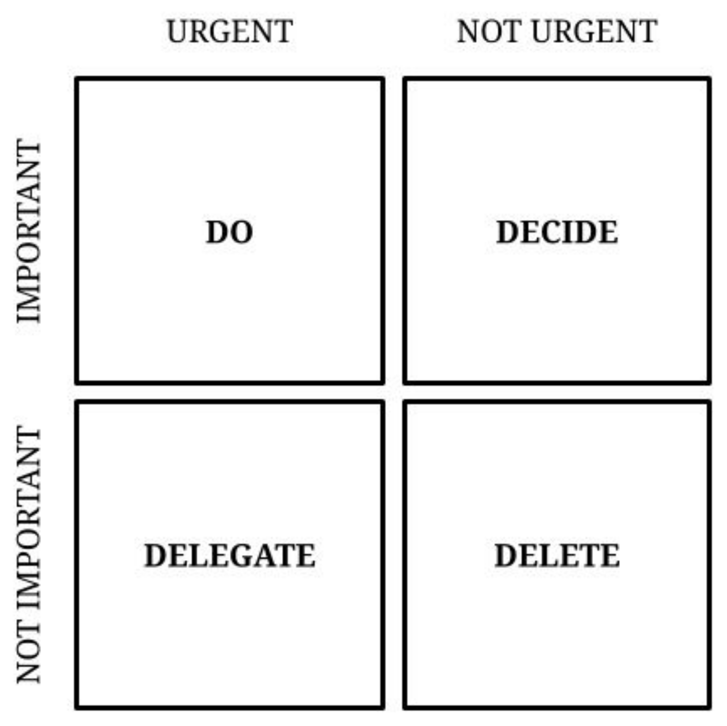
Prioritization - “Dirty dishes”

Prioritization - “Dirty dishes”

Typical “dirty dish” problems to look out for:

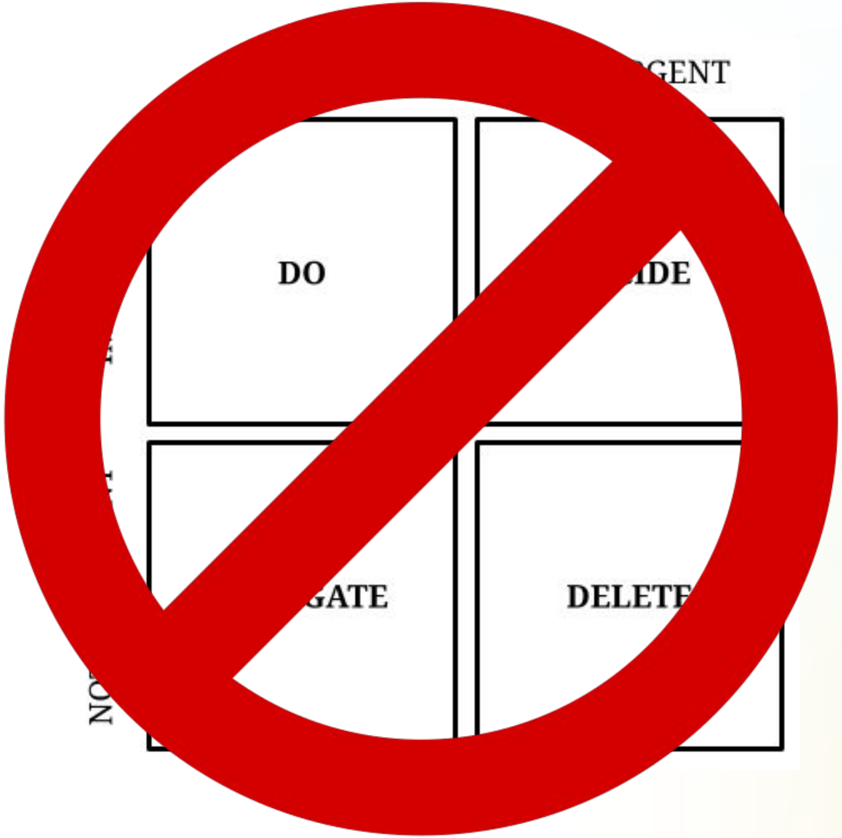
- People: burnout, emerging friction between teammates, lack of motivation/clarity/vision
  - Look out for the “doorknob effect”
- Technical: lack of observability on high value systems, things happening the team can’t explain (e.g. mysterious latency spikes), tech debt
- Project: unclear dependencies, lack of buy-in, misalignment on timelines

Delegation



	URGENT	NOT URGENT
IMPORTANT	**DO**	**DECIDE**
NOT IMPORTANT	**DELEGATE**	**DELETE**

Delegation

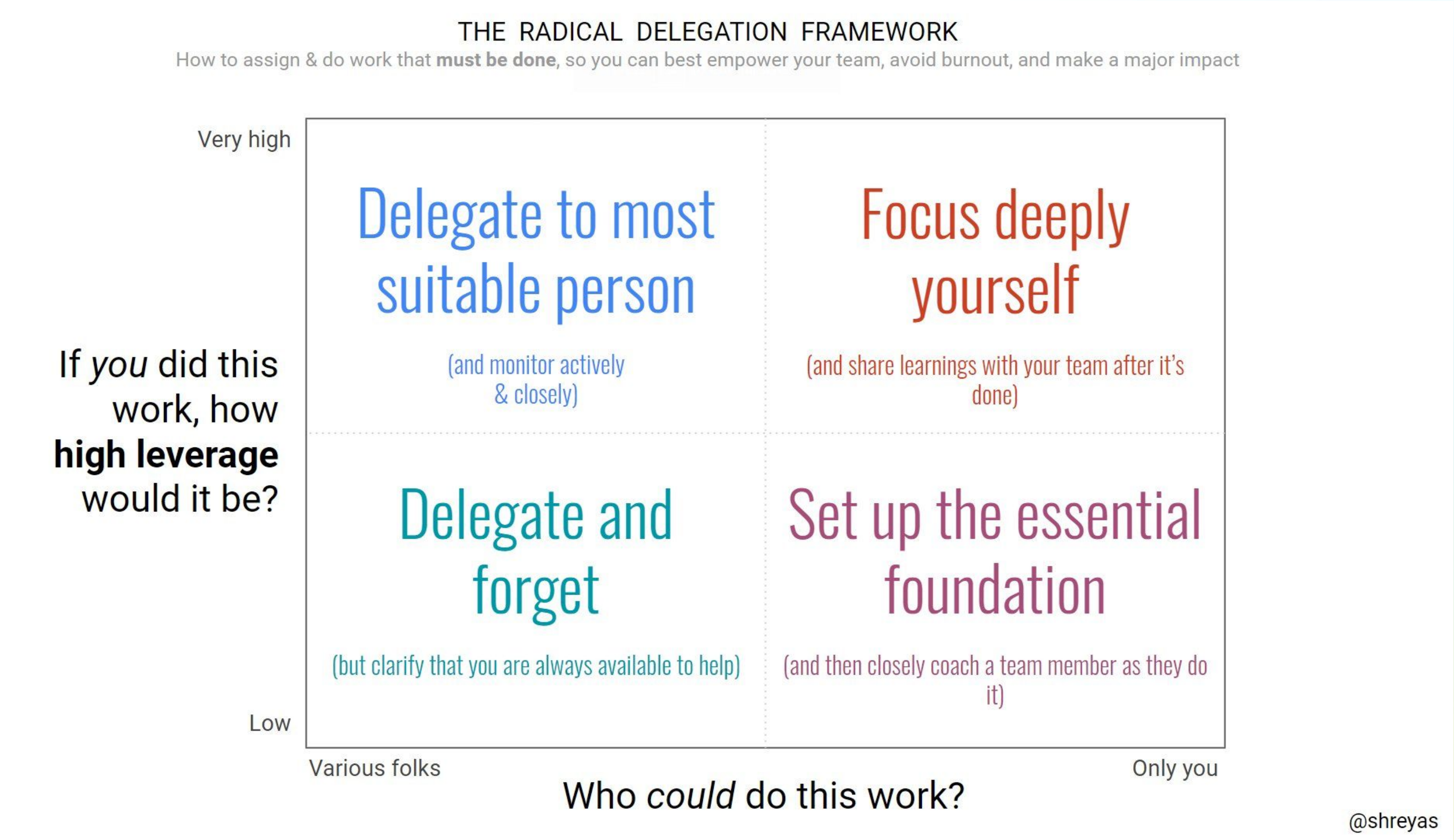


Delegate outcomes, not tasks

Prep chef

Prep chef

Sous chef



THE RADICAL DELEGATION FRAMEWORK

How to assign & do work that **must be done**, so you can best empower your team, avoid burnout, and make a major impact

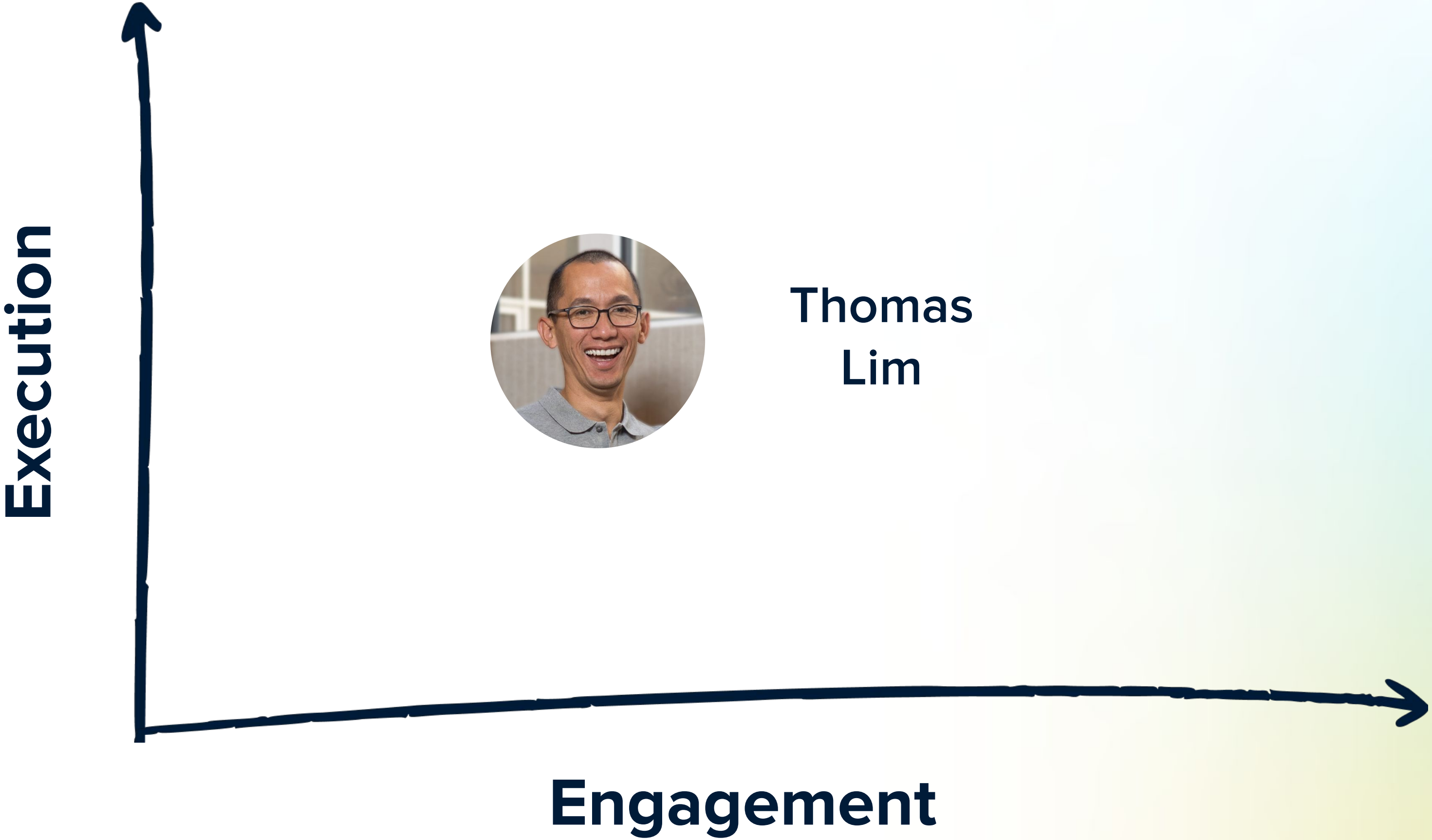
If *you* did this work, how **high leverage** would it be?

	Various folks	Only you
Very high	Delegate to most suitable person (and monitor actively & closely)	Focus deeply yourself (and share learnings with your team after it's done)
Low	Delegate and forget (but clarify that you are always available to help)	Set up the essential foundation (and then closely coach a team member as they do it)

Who *could* do this work?

@shreyas

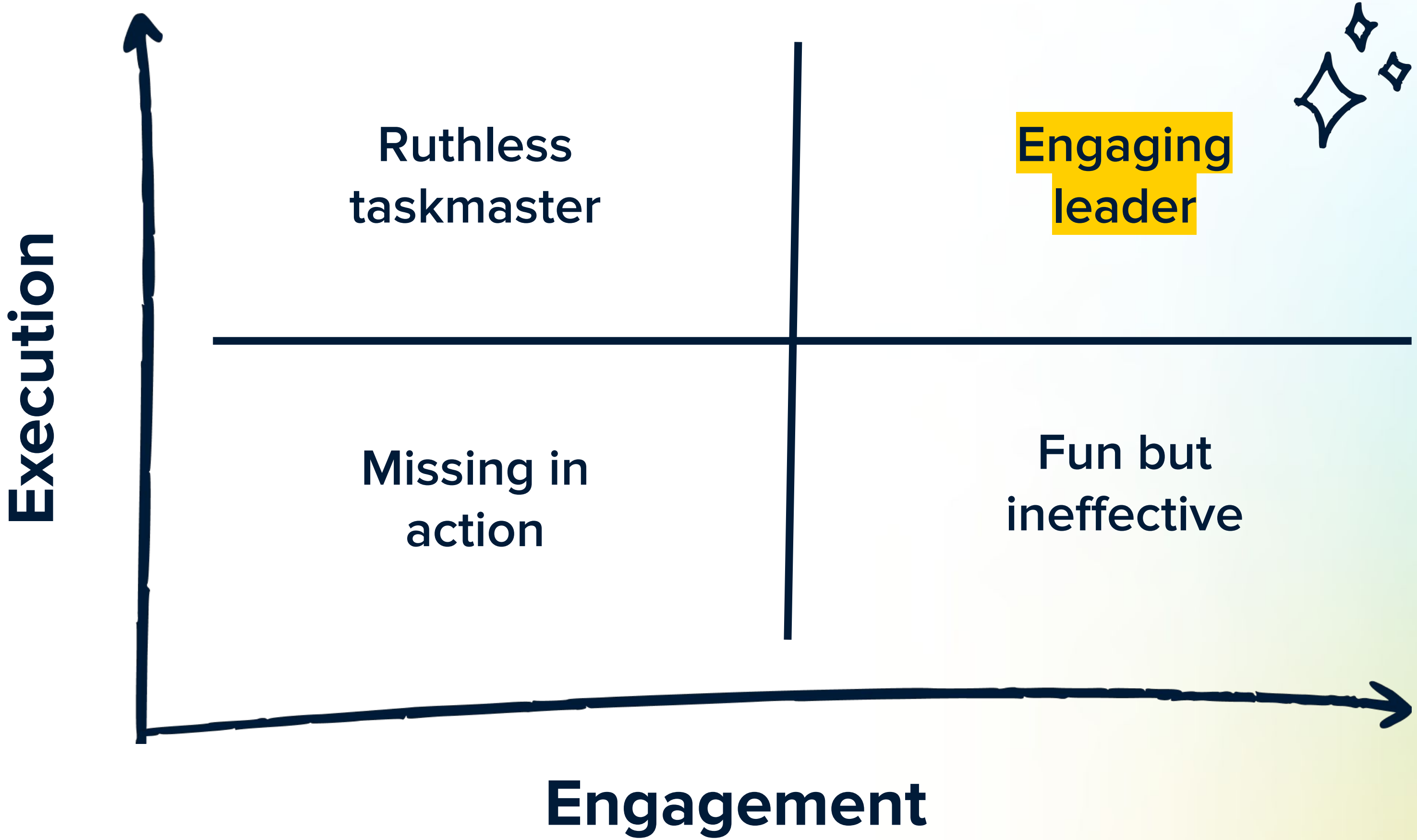
How I measure whether I’m being an effective EM 📏



Execution

Thomas
Lim

Engagement


Ruthless taskmaster
Engaging leader
Missing in action
Fun but ineffective
Execution
Engagement

Recap

- The **6 buckets** 🪣 where I spend my time as an EM
    - People, Process, Planning & execution, Hiring, Technical Leadership and Interfacing
- The **3 meta-skills** 🪄 that I struggled to master
    - Aggregation, Prioritization and Delegation
- The **1 framework** 📏 I use to evaluate if I’m being an effective EM
    - Engagement and Execution

Thank you 😊!