The Manager’s Path: A Guide for Tech Leaders Navigating Growth & Change

https://www.amazon.com/Managers-Path-Leaders-Navigating-Growth-ebook/dp/B06XP3GJ7F/

Author: Camille Fournier

Published: 13 March 2017

Overview

This is a broad and deep playbook of how to manage your technical career. Camille Fournier distilled everything she learned as she rose from individual contributor to CTO into the lessons she wished she’d known at the time.

The book is designed to be read in chunks as you progress in your career. I'd argue reading the whole thing will help you understand those above you and get you thinking about what's next. Each chapter has sections like "ask the CTO", examples of good/bad behavior as a manager, and questions for you to ask yourself at the end.

Topics include

- Management 101
- Mentoring
- Tech Lead
- Managing People
- Managing a Team
- Managing Multiple Teams
- Managing Managers
- The Big Leagues (VPs, CTO)
- Bootstrapping Culture

For anyone considering following the management track, I’d consider this book required reading.

Management 101

Common bad styles

- Micromanager
- Benign neglect -- leaving them completely alone except when needed (e.g., yearly review)
- Abusive

Ideal style

- Care about you as a person
- Actively work with you to help you grow
- Teach you skills
- Give you feedback
- Help you learn what to focus on

What to expect from your manager

1) One-on-one meetings

- Essential for a good working relationship
- Purposes
    - Create human connection between you two; the bedrock of teams is human connection which leads to trust
    - Speak privately about what needs discussing
    - Shouldn't be a status report
    - This is 2-way; both parties need to make this a worthwhile meeting

2) Feedback and workplace guidance

- Your manager should give you feedback; keep track of this (good and bad) for review time
- No feedback < behavioral feedback (or only during yearly reviews)
- Public praise -- reinforces to others what positive behavior looks like
- Managers: give timely feedback instead of waiting until convenient
- Peer review of work is a kind of feedback
- Your manager should…
    - Be your #1 ally at the company
    - Help you grow and learn new things
    - Understand the value of the work you're doing, show the larger picture, define purpose
- The more senior, the less feedback you'll get (and will be higher-level, like strategy)

3) Training and career growth

- Manager should help you find training and resources for career growth
- You are expected to figuring out what things you want
- Managers help with promotions and explaining how those work

How to be managed

1) Spend time thinking about what you want

- "…figuring out what you want to do, what you want to learn, and what will make you happy rests on *your* shoulders."

- You'll go through periods of uncertainty (jobs change, responsibilities shift, newness wears off)

2) Be responsible for yourself
- Know yourself first, then go after what you want
- Examples: bring 1:1 agendas, ask to work on projects, seek out feedback
- Figure out when to go home and when to stay late
- You won't get everything you ask for, but at least try

3) Give your manager a break
- They're human too
- Their job is to do the best thing for the company/team, not to make you happy
- You can't change them; the only part of the relationship you control is yourself
- Bring solutions and ask for your manager's advice
- Don't wait until 1:1s to speak up if something's not working

4) Choose your managers wisely
- Strong managers know how to play the game at their company (promotion, attention, feedback, networking)
- Everyone's different… one person may be great with new folks, but not with senior folks

Questions

1. Have you had a manager you considered good? What did this manager do that you found valuable?
2. How often do you meet 1:1 with your manager?
3. Do you come to 1:1s with topics to discuss?
4. If you're covering status updates in 1:1s, is there another mechanism to do this?
5. Do you feel you can tell your manager when you have a major event?
6. Do you feel that your manager knows something about you personally?
7. Has your manager delivered good feedback to you? Bad? Any?
8. Has your manager helped you set any work-related goals for this year?

Mentoring

Onboarding is a great opportunity for mentorship for both parties.

Being a mentor

This is a low-risk way to practice managing.

- Interns
    - Two goals: hopefully turn them into a future candidate, have them tell others how great your company is
    - Step 1: have a project in mind already. This task should be specific but not urgent. If it would take someone internal 5 weeks, budget 10 for the intern.
    - Step 2: onboarding is typical, but check in more often. Set as much up in advance as possible.
    - Step 3: break down and explain milestones
    - Step 4: have them present their work
    - Goals for you: practice listening, communicating what needs to happen, adjusting to what happens
        - **Listening** -- fundamental skill as a manager; what are they really saying? Are you waiting to talk next? Ask lots of questions too. Be prepared to explain complex things multiple times.
        - **Communicating** -- Are they asking you about every little thing? Show them how to research first. Have them explain what something does in their own words.
        - **Adjusting** -- anything can happen (exceeds your expectations, doesn't hit your expectations, fast but poor quality). Check in with them frequently
- New-hires
    - Opportunity to see your company through fresh eyes
    - Bits of culture, process, and jargon that are second-nature to you
    - Step-by-step guides are helpful
    - Make introductions to others
    - Getting to know people and helping them succeed will pay off; networking is key
- Technical or career mentoring
    - Seniors helping juniors -- problem is relevant to both
    - Be specific about goals and expectations around this relationship
    - You're the mentor
        - Be explicit… expectations, what they need to do, they should ask questions
        - It's okay to say no; it's worse to say yes and them fail to deliver
    - You're being mentored
        - What do you want out of it?
        - Don't waste mentor's time
        - Sometimes you need a paid professional instead (coach, therapist)

Tips for the manager of a mentor

- Figure out the purpose of the relationship (why are you mentoring?)
- Realize this is an additional responsibility; it may slow you down
- Aim for diversity (e.g., women mentor men)

- Mentoring helps everyone develop stronger external perspectives and their own networks

Key takeaways

- Be curious and open minded
- Listen and speak their language
- Make connections

Questions

1. Does your company have an internship program? Can you volunteer to mentor an intern?
2. How does your company do onboarding? Can you propose assigning mentors to new-hires?
3. Have you had a great mentor? What made it great? What did you learn?
4. Have you had a mentor that didn't work out? Why not? What would you do differently?

Tech Lead

Misconception: Give the tech lead role to the most experienced engineer

(Pages 28-29 has an example job description.)

Tech leads are strong technical project leaders whose expertise helps the whole team improve. You can't lead without engaging others, so people skills are needed. The new skill is **project management**.

All great tech leads know this one weird trick

Balance your technical commitments with the work the whole team needs.

- You must master your time and how you use it
- Balance between things you enjoy and things that need doing (that you probably haven't mastered)
- If you need to code (maker schedule), keep yourself from being pulled into meetings
- Make sure your team has a schedule that allows them long blocks of time to focus
- Help other stakeholders respect the team's focus

Being a tech lead 101 (roles)

You need a wide view of the work to keep the project moving. You may not do all of these at once, but you should know how to…

- Systems architect, business analyst
  - What needs to change to deliver the project
  - Provide structure for basing estimates and ordering tasks
  - Understand the overall architecture so you can design complex software
  - Translate business requirements into software

- Project planner
  - Break work down into rough deliverables
  - Find ways to parallelize work
  - Gather input from other experts, identify MSCW (must, should, could, won't)

- Software developer, team leader
  - Write code, communicate challenges, delegate
  - Avoid heroic bursts of coding
  - Code, but not too much
  - Communicate early and often

Managing projects

- Agile doesn't replace project management; it just means you work in smaller chunks.
- Things like infrastructure, architecture, planning which include many unknowns and relatively hard deadlines "doesn't fit so well into a standard agile process."
- Value of planning…

    - NO: perfect execution, catch every detail, predict the future
    - YES: think about the project in depth before diving in, see what happens
- What comes out
    - Find the small pieces
    - Put the pieces in the most effective order (parallel, sequence)
    - Tease out unknowns that will slow it down or cause it to fail

How to manage a project

1. **Break down the work**. Doesn't matter what tool (Excel, project, sticky notes); if you don't understand, ask. Can anything be started right now?
2. **Push through the details and the unknowns.** You will get stuck. You will get tired. Hopefully you have someone to help coach you and help you work through it. Stop spending time on things when you feel there's no more value to do so.
3. **Run the project and adjust as you go.** Things always slip. Point to milestones. Communicate.
4. **Use the insights gained in the planning process to manage requirements changes.** If things change, know what's impacted and how much it will cost. Know what to cut or simplify.
5. **Revisit the details as you get close to completion**. Focus on what could trip you up at the last minute. What's "good enough." Make a launch plan. Make a fallback plan. Don't forget to celebrate at the end!

Stay technical? Go management?

Senior individual contributor...

Imagined	Real life
Deep thinking Solving fun and novel problems Lots of choice over the work Most of the time coding Asked for advice on everything Don't deal with people much Minimal meetings Juniors hang on your every word Constant upward trajectory No evenings/weekends Write books, talks, open-source Shy and quirky, but that's okay Highly paid, influential	Selling others on your ideas Teaching people what you built Need for new shiny isn't as great You have to find the good projects yourself (usually luck) Promotion is slower Newbies want too much of your time Most other devs don't care, or find you threatening Your manager probably doesn't have time to help as much Meetings, interviews, code reviews, non-building time If you want to write books, talks, open-source, it's on your own time

Manager

Imagined	Real life
You control the team and decisions Everyone respects you and is happy with you You build the culture and fire bad apples	Getting people to do stuff is hard Many meetings Lost touch with the code; not enough time

People tell you when you're screwing up The impact of your coaching is quick and obvious Other managers are open to your criticism Your manager actively coaches you You get handed more responsibility You are encouraged to write books, blogs, talks	Your power over decisions exists, but it narrow Goals change out from under you Your team may not agree with you or even like you People may quit and you won't know why If times are tough for the company, you have to lay people off Other managers don't care about your feedback Getting ahead involves politics

The Process Czar

"They tend to be very organized and comfortable with details, and they're good at knowing the rules and following them precisely."

"They can be incredibly valuable members of a project management team because they tend to make sure that no task is forgotten and that everything is wrapped up in the way it should be."

Problems: not everyone follows process as well, failures are process problems instead of unexpected changes (which are inevitable), focused on easy-to-measure things (e.g., # hours in the office)

Process can be helpful, but ambiguity is inevitable.

Make it safe to fail.

No process is perfect.

How to be a great tech lead

- Understand the architecture
- Be a team player
  - Stop doing all the interesting work; do the annoying tricky stuff
  - Working in the less desirable places shows you where process is broken
  - Don't do all the boring work either (waste of your skills)
  - Give others a chance to grow, but don't sacrifice everything
- Lead technical decisions
- Communicate
  - Team productivity > your productivity
  - Communication skills are essential for good leaders -- speaking, writing, reading
  - Listen to what people are saying
  - Become a good note-taker

Questions

1. Does your company have tech leads?
   a. What's the job description?
   b. How would you define the role?
   c. How would other tech leads define the role?
2. If you are considering become a tech lead…

    a. Are you ready to push yourself?
    b. Are you comfortable spending time outside of code?
    c. Do you feel expert enough in your code to successfully lead others as they work in it?
3. Have you asked your manager what he/she expects from the tech lead?
4. Who's the best tech lead you've worked with? What are some behaviors that made you answer that way?
5. Have you worked with a frustrating tech lead? What are some behaviors that made you answer that way?

Managing People

This is about figuring out your management style.

Starting a new reporting relationship off right

- **Build trust and rapport**
  - Purpose: help you understand how to manage this person
  - Examples
    - How do you like praise?
    - Serious feedback written or verbal?
    - What does annoyed/bad mood look like for you?
    - Any manager behaviors you dislike?
    - Any surprises I should know about?
  - Google for 1:1 questions
- Create a 30/60/90-day plan
  - What do you expect of them after these milestones (e.g., get up to speed, fix a bug)? This sets clear expectations for both of you.
  - More senior hires should play more of a role in creating this plan
  - Be realistic based on their skills
  - If they're consistently not measuring up, you may have mishired
- Encourage participation by updating the new-hire docs
  - They're the freshest eyes on the problem
  - Gives them "skin in the game"
- Communicate your style and expectations
- Get feedback from your new hire
  - They are less blind to obvious problems
  - Take feedback with a grain of salt, as new folks don't have full context

Communicating with your team

- Having regular 1:1s
  - Every person needs different things, have different communication styles, and focus on different things
- Scheduling 1:1s
  - Start weekly for new folks, then adjust
  - Make every attempt to not cancel/reschedule
  - Respect the "maker schedule" and don't have these meetings right in the most productive part of their day
- Adjusting 1:1s
  - How often do you interact regularly?
  - How much coaching does this person need?
  - How much does this person push information to you?
  - How good is your relationship with this person?
  - How stable are things in the team/company?

Different 1:1 styles

- The to-do list
  - Doesn't waste time; checklist of things that won't get missed

    - Avoid this being something that can be done over email; choose questions that are worthy of discussion
    - This format leaves both parties time to plan responses in advance
- The catch-up
    - Let them run the show, and you listen
    - Be careful of being sucked into office drama; you may make things worse
- The feedback
    - Tune the frequency based on type of feedback
    - Document what you discuss (shows trends, can be useful if feedback is that they're not meeting expectations)
    - Don't sit on critical feedback or praise -- do it now.
- The progress report
    - Most useful if the people reporting to you don't work on your project
    - Try to mix in non-update topics
- Getting to know you
    - Great for newer folks
    - Topics like hobbies, pets, family, career goals
    - Shows investment in them as a person, not only an employee
- Mix it up
    - Get breakfast, take a walk; make sure it's still somewhat private though
    - Keep notes -- topics, next actions, context, takeaways, feedback

Micromanaging vs. delegating

- Support people to help them feel confident; help them understand
- There are times when micromanaging is helpful
    - Juniors need structure
    - Projects get off the rails and will have big repercussions
- Main issues: **trust and control**
- **Autonomy** is important, and if you take that away, people won't stay
- Delegation is not the same thing as abdication; you're still involved, but less so

Practical advice for delegating effectively

- Use the team's goals to understand which details you should dig into
    - Ask your team how they measure success, then have them make that visible to you on an ongoing basis
    - If they don't know how to measure success, there's your first task
- Gather information from the systems before going to the people
    - The team isn't going to be happy gathering info you could find yourself
- Adjust your focus depending on the stage of the projects
- Establish standards for code and systems
    - Pick what's important to you and let your team know what your expectations are
- Treat the open sharing of information (good or bad) in a positive way
    - Teach people what you want them to do
    - If you criticize people for telling you bad news, what they hear is "Don't tell me bad news"...so they'll hide things.
    - Your team deserves a manager who is willing to trust them to do things on their own

Create a culture of continuous feedback

- You have a lot of power to shape the experience your direct reports have with reviews
- **Know your people** -- goals, strengths, weaknesses, current level, next level; do their answers to these questions jibe with your answers?
- **Observe your people** -- pay attention, look for talents/achievements, notice weaknesses, find ways to praise people weekly
- **Provide lightweight feedback**
- **Bonus: Provide coaching**

Performance reviews

- 360 model -- you get feedback from peers, reports, and supervisor
- Give yourself enough time, start early
  - You need heads-down time to do these. Work from home if needed.
  - Read collected notes, digest, then summarize.
- Account for the whole year, not just the last few months
  - Having notes you keep throughout the year makes this easier
  - This is about *accomplishments* and *growth*
- Use concrete examples and excerpts from peer reviews
- Spend plenty of time on accomplishments and strengths
- For areas of improvement, keep it focused
  - Make it meaningful to the reviewee
  - Don't make up content for the sake of having something to say
  - No major improvements? This person is ready for promotion.
- Avoid big surprises
  - The review should be a formality, and not a venue for something you should have brought up in a 1:1
- Schedule enough time to discuss the review
  - Give them the text ahead of time
  - Be prepared to answer questions

About potential…

- It's not about how articulate someone is or what school they went to
- Real potential shows itself quickly -- going the extra mile, offering insights into problems, helping the team in areas that are neglected
- If someone isn't performing, maybe they're not a good fit in some way (work, team, project, etc.)

Cultivating careers

- You play a key role in getting people promoted
- Learn how promotions work in your company -- how decisions are made, limits, etc.
- Be transparent about how this works. If someone asks you for a promotion and they don't deserve it, tell them why; tell them how to get there.
- "There are fewer opportunities for people to show the kind of leadership or breadth of impact needed to get promoted as they become more senior." Solutions: refer them to other leaders in the org, may need to leave the company for better opportunities
- "If there's no growth potential on your team because there's no room for people to work at a more senior level, it may be a sign that you need to rethink the way work is done in order to let individuals take on bigger responsibilities."

Firing underperformers

- This is never easy, so have a process in place before you need it
- Most of the time a performance improvement plan is a gracious way of giving the problem employee time to exit gracefully
- Use **the rule of no surprises**; make it clear early and often if someone isn't meeting your expectations
- Don't put anyone on an improvement plan that you don't expect to lose

Coaching out

- Sometimes you and the employee are fine with their role and level of output/engagement
- Coaching out = you tell them what's expected for the next level and they haven't demonstrated it

Questions

1. Have you set up regular 1:1s with your reports?
2. When was the last time you talked to your reports about career development? (Don't go more than 3 months.)
3. Have you given feedback in the last week?
4. When was the last time you gave kudos in front of the team?
5. When was the last time someone behaved badly and needed correction?
    a. How long did it take you to give corrective feedback?
    b. Did you give feedback in private or public?
6. Have you ever been given a performance review that felt like a waste of time?
    a. What would have made it more valuable?
7. What was the most useful piece of performance feedback you ever got?
    a. How was it delivered?
8. Do you know who the process of promoting people works in your company?
    a. Can you ask someone to walk you through it?

Managing a Team

- This is a different job than being an engineer
- Responsible for identifying bottlenecks and roadblocks to success
- Keep people focused so that scope is managed and technical deliverables are met
- Recruit and hire to fill needed roles
- Communication conduit (up and down)
- Look at timeline, scope, and risk and lead initiatives
- Identify areas of technical debt and how to resolve it

Staying technical

- You may have people reporting to you that have greater technical skills than you
- Having the most technical knowledge doesn't make you a good manager. The work of supporting people is far more important.
- This level of the career ladder is not generic management; having the tech chops gives you instincts
- You need some technical skills to build credibility and respect within your team
- Risk: Pulling away from the code completely will make you lose touch of what's going on. Small bug fixes and code reviews help.
- Having an understanding of the code structure etc. will help you get a grasp of what's possible and what's difficult
- You may end up transitioning out of code entirely at some point, but doing so too soon may limit your options

Debugging dysfunctional teams: The Basics

- Not shipping
  - "Humans, by and large, feel good when they set small goals and meet them regularly."
  - You need to help the team remove bottlenecks to shipping
- People drama
  - No brilliant assholes! "You know, that person you think can't be replaced because he's just so productive and so smart, but who isn't a team player and makes everyone around him unhappy."
  - Nip people drama in the bud quickly
  - Dealing with underperformers can be easier than brilliant jerks; explain expectations
- Unhappiness due to overwork
  - Technical issues? Slow the product roadmap and deal with technical problems.
  - Time crunch? You need to cheerlead, help code, order dinner, make space after the delivery.
  - Learn why this sucks and do what you can to prevent it recurring
- Collaboration problems
  - No quick fixes here
  - Show a willingness to improve collaboration; stay positive and supportive in public
  - Make space for non-work activities -- after work, lunch, chat rooms, get to know people

The shield

- Some say managers should be a "bullshit umbrella" to protect the team from things that distract their focus (drama, politics, company changes)

    - Pro: Help them understand key important goals and keep them focused on those
    - Con: They don't get context for the organization and are siloed
    - Con: Denies that drama, politics, etc. exist which can make changes difficult
- Your team is not made of fragile children and you are not their parent

How to drive good decisions

- Roles
    - Product manager -- owns the product roadmap
    - Tech lead -- owns technical details
    - Engineering manager -- owns the team's progress through those elements; may only have authority to guide rather than dictate decisions (and still be judged by those decisions!)
- Create a data-driven culture
- Flex your own product muscles
- Look into the future -- you need to think two steps ahead (both for product and tech)
- Review the outcome of your decisions and projects -- how did your decisions work out? Were your hypotheses true?
- Run retrospectives for the processes and day-to-day

Conflict avoider vs. conflict tamer

- (Disagreement will happen, so pretending it doesn't exist isn't practical.)
- Don't rely exclusively on consensus or voting
    - Not everyone has an equal stake or expertise
    - Process is not always impartial
- Set up clear processes to depersonalize decisions
    - Establish goals, risks, questions, and standards if allowing the group to decide
    - If designating a point-person, make it clear whose feedback that person needs
- Don't turn a blind eye to simmering issues
    - It's too easy to avoid a problem while it's small
    - Don't wait until performance review time; negative feedback shouldn't be a surprise
- Address issues without courting drama -- you are not a therapist, but you need to identify problems that are causing the team to work less effectively
- Don't take it out on other teams
- Be kind. It's natural to want to be liked by others. -- There's a difference between being *nice* and being *kind*. Sometimes having uncomfortable conversations are needed to move the team forward.
- Don't be afraid -- we're scared of responsibility and messing up
- Get curious about your fear -- for example, am I avoiding talking to this person because they truly are difficult to work with or because they won't like me if I get tough

Team cohesion destroyers

- The goal is *psychological safety* -- willing to take risks and make mistakes in front of one another. Teams that gel together are friendlier and tend to produce better results. Why would you want to work with people you hate? The following make is hard for the rest of the team to feel safe…
- Brilliant jerk

    - "It's incredibly hard for a manager to justify getting rid of someone who produces great work, even though she's a drain on everyone around her -- especially if this person is only irregularly a jerk."
    - Best approach… don't hire them
    - If they behave badly in public, flip the advice of "praise in public, criticize in private" to make it clear what your standard is
- Non-communicator
    - Make it clear that hiding things is not acceptable; collaboration can't be blocked
    - Why is hiding happening? Psychological safety problem?
- Employee who lacks respect
    - Help manage their expectations or move them over/out
    - "You can't have a person working for you who doesn't respect you…"

Advanced project management

- None of this advice replaces agile project management
    - You don't need to waterfall-plan everything out. You need to focus on the big picture while keeping the team working at the lower levels.
    - Agile works best for the short-term work
- You have 10 productive engineering weeks per engineer per quarter
    - Meetings, reviews, outages, onboarding, etc.
- Budget 20% for generic sustaining engineering work
    - Debugging, cleaning up legacy code, upgrading platforms
    - If you do 100% feature development, it will eventually catch up with you
- As deadlines approach, your job is to say no
    - Cut scope at the end of the project
    - You'll have to figure out when to push back and when to cut corners
- Use doubling for quick estimates, but push for planning time to estimate longer tasks
- Be selective about what you bring to the team to estimate
    - It's stressful to ask the whole team to estimate -- let them focus on the work at hand

Questions

- What are your new responsibilities now that you're the manager of a team?
- What tasks have you stopped doing or delegated to make time?
- How well do you know the day-to-day challenges of writing, deploying, and supporting code on your team?
- How often does your team mark work as completed?
- When was the last time you wrote a feature, debugged a problem, or paired with a member of your team on code he was struggling with?
- Are there one or two members who cause the bulk of the negativity? What's the plan for solving that?
- Do your team members seem engaged with one another (smile, joke, chat, eat)?
- What's the last time you sat down with your team without talking about work?
- How does your team make decisions?
    - Who's responsible?
    - What decisions are you responsible for?
- When was the last time you reviewed a completed project to see if had achieved its goals?
- How well does your team understand why they are working on the projects they are working on?

- When was the last time you cut scope on a project?
    - How did you determine what to cut?

Managing Multiple Teams

Engineering Director

- Organizational technical competence, guiding/growing, training
- Should have technical background
- Aware of industry trends
- Should be able to review code
- Ask business and product questions during architecture/design
- Execution of complex deliverables (standards, processes)
- Recruiting, planning, career growth
- Manage some vendors
- Participate in budget process
- Growing next generation of leadership talent
- Focused on creating high-functioning, engaged, motivated orgs
- Owns retention goals
- Balanclong-term product/business goals with technical debt and strategic technical development
- Strong leader with collaboration with other areas of the business
- Helps create strategic and tactical tech roadmap for business, efficiency, revenue, tech innovation
- Strong communicator
- Positive public presence to attract candidates
- Guides the goal-setting process for teams in the org

Managing your time: What's important anyway?

- Don't plan on coding much; you're doing too much other important stuff to get large enough blocks of time to focus on code
- Code-related activities you *can* do… code reviews, debugging, production support, pairing
- Make time to do something creative (maybe 4 hours a week)… blog posts, conference talks, open source project
- Book: *Getting Things Done* by David Allen
- Note where things fall in terms of importance and urgency
    - Urgency is often easier to feel than importance
    - Focus on important but not urgent work
    - Ignore unimportant/non-urgent work
- Your manager will expect you to manage your own time
- If you skip meetings, you may miss out on clues that will help you catch problems early
- Pay attention to the morale and dynamics of your team

Decisions and delegation

- You will be exhausted your first several months -- the only way out of this situation is through it
- There will be lots of "plate spinning"; the focus should be to hone your instincts to figure out which plates to touch and when
- Delegate simple and frequent tasks
    - Examples: running standups, writing up summaries, minor code reviews
- Handle simple, infrequent tasks yourself
    - Examples: write up quarterly report, book a conference ticket
- Use complex and infrequent tasks as training opportunities for rising leaders

    - Examples: hiring plans, performance reviews
- Delegate complex and frequent tasks to develop your team
    - Make your teams capable of operating at a high level without much input from me
    - Examples: project planning, systems design, handling outages
    - This starts slowly and gets better

Warning signs

- Person that's usually chatty/happy/engaged starts leaving early, taking breaks, not hanging out, staying quiet --> something personal going on or about ready to quit
- Tech lead says everything is fine but skips your 1:1s or doesn't go into details on reports
- Team has no energy during meetings (PM and TL doing most of the talking)
- Team's project list changes every week
- Small team internally seems fragmented in understanding; engineers don't care unless it's their problem

Strategies for saying no

- "Yes, and" -- ex: yes we can do this project, and we'll have to delay the current project
- Create policies -- once you start repeating why you say "no" to things, you need a policy; this sets the expectation of what bars people will need to pass to get to "yes"
- "Help me say yes"
- Appeal to budget -- this is "not yet"
- Work as a team
- Don't lie -- you don't have the luxury of time to drag things out to appease everyone; if you have the authority and think it's a bad idea, just say no (even though you may be wrong sometimes)

Technical elements beyond code

- You're now observing and improving systems of work
    - Do people know what's expected of them?
    - Do they have the materials and equipment to do their work right?
    - Do they have the opportunity to do their best?

Measuring the health of your development team

- Frequency of releases
    - To move fast you have to break things into small chunks
    - Why don't you release more frequently? What does it look like when things go wrong?
    - TDD helps here, as it forces you to think smaller
- Frequency of code check-ins
    - This can be hard to teach people that have been coding for a while that this needs to be the norm now
    - You're (likely) not managing a research team, so they need to deliver frequently
- Frequency of incidents
    - Which is more important… features or stability?
    - Having devs work incident tickets can be good exposure, but tread carefully as this can cause burnout
    - You need to focus on reducing incidents, not getting better at reacting to them

    - Don't swing to the other side of overplanning and guarding against everything; this can be just as bad as moving too fast

Us vs. them, team player

- New managers need to create a shared team identity. WARNING: If you try to make your team feel superior, your team will be…
    - Fragile to the loss of the leader
    - Resistant to outside ideas
    - Focused on empire-building
    - Inflexible to reorgs, cancelled projects, etc.
- Don't fix what's broken; identify existing strengths and cultivate those
- Durable teams are…
    - Resilient to the loss of individuals
    - Driven to find better ways to achieve their purpose
    - First-team (peers across the company) focused
    - Open to changes that serve their purpose

The virtues of laziness and impatience

- Larry Wall -- laziness, impatience, and hubris
- Figure out what's important, then go home; this forces you to focus
- Laziness is about not brute-forcing problems just to look busy
- Questions
    - Can I do this faster?
    - Do I need to be doing this at all?
    - What value am I providing with this work?

Questions

1. When was the last time you reviewed your schedule for things you're doing but don't provide much value for you or your team?
2. If you still write code, how does this fit in with your schedule?
    a. Was it simple? Complex?
    b. What's driving you to continue spending this time?
3. What's the last task you delegated to one of your teams?
    a. Was it simple? Complex?
    b. How is the person handling things?
4. Does the process of writing, releasing, and supporting code seem to function smoothly?
    a. When was the last incident?
    b. What happened?
    c. How did the team respond?
    d. How often does this happen?
5. When was the last time you pushed the team to cut scope?
    a. Did you cut features, quality, both?
    b. How did you decide?
6. When was the last time you sent an email after 8pm or the weekend?
    a. Did the person respond?
    b. Did you need him to respond?

Managing Managers

- Higher coverage area -- more people and projects than you can handle by yourself; likely managing functions you have little expertise in
- Answers are even less available than before
- Easy to miss details because you're not engaging regularly with line-level folks
- Pulled in many directions; trust your instincts
- At first (because of lack of experience), you'll only notice problems once they're far gone
- **THIS IS A NEW JOB**, not more of what you already did
- Follow up on all the little things until you figure out what you don't need to follow up on

Open door fallacy

- Why aren't people coming to you?
- You need to ferret out problems proactively
- Take the time to look for problems; **predicting problems is part of your job**

Skip-level meetings

- Meetings with people who are two levels below you on the org chart to determine health
- This is about trust and engagement and hearing the other side of the story
- Give them prompts about what to talk about, for example…
    - What's keeping you from doing your best work right now?
    - What areas don't you understand?
    - Who on your team has been doing really well lately?
- Maybe do this every quarter, or hold skip level *lunches* to do it panel-style
- Invest time in learning how to maintain these relationships

Manager accountability

- The managers under you should make your life easier
- You must hold them accountable so they don't hide things from you
- Managers are responsible for the health and productivity of the team
- Common outcomes of mismanagement
    - Unstable product roadmap
    - Errant tech lead
    - Constant fire-fighting mode
- Managers need coaching and guidance the same way that individuals do

People pleasing

- Too much saying "yes" leads to burnout
- Signs
    - Manager is likeable but not effective; shields people and hides things
    - Things are smooth and the team avoids mistakes vs. pushing people to be better
    - Manager feels bad and brings team morale down
    - Doesn't say no to work; has excuses why work isn't getting done
    - Habitually overpromises and underdelivers
    - Says one thing to higher-ups, another to the team
    - Knows where all the problems are but has no solutions

- Solutions
  - Get them feeling safe saying "no"
  - Externalize more decisions so they don't take failure personally
  - Provide strong partners
  - Make it clear how rewards are given (prevents brown-nosers)
  - Highlight the downsides of people-pleasing even though their intent is good

Managing new managers

- Spend quality time with them; it will pay dividends
- They will be lost at first, and you need to help them
- Use skip level meetings to see if they're managing well
- Red flag: overwork
  - They're not delegating their old job
  - Person thinks he's in control now and is the taskmaster
- Beware the micromanager (needs details on everybody)
- Beware the control freak (takes decision-making power away from the team)
- Get them formal training on how to be a good manager; they're often eager to get this anyway

Managing experienced managers

- CRITICAL: culture and process fit, as managers create sub-cultures
- You need someone who can understand dev practices; prefer this over industry knowledge
- Think about what your culture values, and help your managers embody those values
- These people should be able to manage independently
- You focus less on the how, more on the larger impact stuff
- Look for programs to meet new peers to expand their network

Hiring managers

- Managers aren't typically hired from outside the company because it's difficult to demonstrate during an interview
- Management is all about communication, so it's possible for someone to interview well and get nothing done
- Role-play a 1:1 by asking a future direct report to ask for help on a problem
- Role-play difficult situations like giving negative feedback
- Ask how they debug teams (someone thinking about quitting, project behind schedule, coaching those who struggle, help great teams get better)
- Ask about management philosophy
- Ask the candidate to give a presentation to a group
- For a technical role, ask about mediating a technical debate or to discuss architecture
- Look at whether this person fits at this culture (startup vs. large org), how do they handle leadership, what do they think about power dynamics
- In Andy Grove's "High Output Managers", people in a VUCA environment go through this order until comfortable: self-interest, group-interest. You want managers to fit into your norms easily.
- CRITICAL: Do reference checks to see if they'd want to work with this person again

Debugging dysfunctional organizations

- "Managing teams is series of complex black boxes interacting with other complex black boxes."

- NOTE: The book has many examples for each of these steps.
- **Have a hypothesis** about why the team isn't working
- **Check the data** -- what behaviors are happening? What behaviors aren't happening?
- **Observe the team** -- ex: boring meetings are a red flag. Note that observing people changes their behavior.
- **Ask questions** -- do they know what to work on, how to do it?
- Check the team dynamics
- **Jump in to help** -- this could be an opportunity for managers to learn
- **Be curious** -- "the pursuit of why"

Setting expectations and delivering on schedule

- "Why are things taking so long?"
- You need to figure out why management is asking this. Sometimes projects are on-track and nothing is wrong, but the question gets asked anyway.
- Estimates help escalate complexity, and estimating is a skill
- Use agile retrospectives to learn why your estimates we're good/bad
- If you're on schedule but asked to move faster, use empathy instead of deflection.
- Work with others to cut scope. Avoid cutting quality, because this will only slow you down later.

Roadmap uncertainty

- Being in middle management sucks here because you have little ability to push back
- You can feel you're being seen as powerless, and your team will feel unfairly treated
- Tech debt is rarely on the roadmap, so how will you prioritize this?
- Be realistic about the likelihood of changing plans, especially given the cycles and size of your company
- Think about breaking work into smaller chunks so you can deliver something instead of everything/nothing
- Don't overpromise a future of technical (new shiny) projects that hasn't been written yet
- Dedicate 20% time to "sustaining engineering"
- Technical projects need rigor too -- size, importance, value it delivers, if done what does that mean
- Change happens -- surf the wave, don’t get pulled under. Help people tie up loose ends, stabilize in-flight projects, transition into new work.

Staying technically relevant

- Oversee technical investment
  - Make sure your team is placing technical bets in the right places
  - Look at the portfolio of projects to match current/future needs
  - Focus the team on areas of greatest need
- Ask informed questions
  - Guide investments by asking questions. (The book has several.)
  - Know enough about the work to sniff out misguided efforts and evaluate proposed investments.
- Analyze and explain engineering and business tradeoffs
  - You're mapping the business perspective with product roadmap
- Make specific requests

    - If you're the middleman between top management and the technical folks, you add no value.
    - Understand enough about the system to answer questions.
- Use your experience as a gut check
    - Read the code
    - Pick an area of the code and ask an engineer to explain it
    - Attend post-mortems
    - Keep up with industry trends in software dev processes
    - Foster a network of tech people outside your company
    - Never stop learning

Questions

- How often do you talk to your skip-level reports?
    - 1:1 or groups?
- How do you proactively reach out to your teams?
- How much time do you spend seeking out information instead of making it come to you?
- When did you last sit in on a team meeting?
- Write down the job description for the engineering managers that report to you…
    - What are they responsible for?
    - How do you evaluate them?
    - What areas are most important for success?
- Do the job descriptions you made vs. the company made differ?
- What areas need coaching and development?
- If outside your comfort zone, how often do you check in on that area to make sure things are going well?
    - Have you talked to that manager to see what it takes to succeed in that role?
    - What have you learned in the last 3 months to understand them better?
- Is there one team that's operating more smoothly?
    - What differences are there in their processes?
    - Interactions?
    - Their manager?
    - Their interaction with you?
- How do you interview managers?
    - Personal values?
    - Management philosophy?
    - Does the team interview that person too?
    - Do you get references?
- What are the company goals this quarter? Year?
    - How are you merging product goals with technical goals?
    - Is the org's mandate understood by the team?

The Big Leagues

- Tech leaders bring a willingness to embrace and drive change
- You can't just be a change agent -- you have to create an org that can follow through on the changes you want to push
- The company looks to you for guidance on what to do, where to go, how to act, how to think, and what to value
- Management tasks from Andy Grove's "High Output Management" (1983)
    - **Information gathering or information sharing** -- synthesize info quickly, identifying the important stuff, then sharing in ways people understand
    - **Nudging** -- keeping people on track and reminding people of their commitments (instead of giving orders)
    - **Decision making** -- conflicting perspectives, missing information, setting direction, understanding consequences
    - **Role modeling** -- show people what the values truly are

Models for Thinking about Tech Senior Leadership

- **R&D** -- experimentation and new tech generation; may just be about finding new ideas
- **Tech strategy/visionary** -- how the tech grows the business; uses business and tech trends to guide decisions
- **Organization** -- structure of the org; staffing
- **Execution** -- align roadmaps, plan work, coordinate efforts, prioritize, break roadblocks, keep things moving forward
- **Face of technology, external** -- participate in sales cycle, speak at conferences, recruit
- **Infrastructure and tech operations manager** -- cost-focused, security-focused, scaling-focused
- **Business executive** -- understand high-level business

What's a VP of Engineering?

- Experienced manager of people, projects, teams, departments
- Solid handle on processes and details, tracking several in-flight things
- Manages roadmap, hiring plan; coaches engineering management
- Quickly understands what's going on in the org
- Gains trust and shows wisdom
- Helps teams set goals; aligned with the product team
- Strong business and product instinct to deliver on projects and negotiate deliverables
- Focused on creating high-performing organizations where people work together effectively
- Identifies process gaps and manages highly complex, detailed work without getting overwhelmed

What's a CTO?

- Not an engineering role; if you want to stay technical, this is not for you
- Strategic tech executive the company needs in its current stage of evolution
- Long-term focus, make the plan real by breaking it down and directing people to execute it
- Shapes business strategy through the lens of tech; executive first, technologist second
- Needs to understand how tech lines up with business strategy
- Understands big technical opportunities and risks
- Puts people behind solving problems you think will impact the business

- Protects the tech team from becoming purely an executional arm (they need to focus on their needs and ideas, too)
- Typically needs people reporting to them (usually VPs) to show influence

Changing Priorities

- The teams close to the work have their own priorities, and whims of top management can't be executed instantly
- Everyone in the org chart needs to understand what the priorities are and why
- You'll likely need to tell people multiple times before it sinks in

Setting the Strategy

- Do a lot of research
- Combine your research and your ideas
- Draft a strategy
- Consider your board's communication style

Delivering Bad News

- Don't blast an impersonal message to a large group
- Do talk to individuals as much as possible
- Don't force yourself to deliver a message you can't stand behind
- Do be honest about the likely outcomes
- Do think about how you would like to be told

Having a Non-technical Boss

- Don't hide info behind jargon; be careful with details
- Expect that you'll run your 1:1s; come prepared with a list of topics
- Bring solutions, not problems
- Ask for advice
- Don't be afraid to repeat yourself
- Be supportive and ask how you can help them
- Actively look for coaching and skill development elsewhere

Senior Peers in Other Functions

- Senior leaders focus on the business first, and their team second
- This realm can be isolating because you may have few to no peers on your first team
- Let people own their areas; unless they ask for advice, try to stay out of it -- if needed, approach with kindness
- Approach disagreements via 1:1s
- If you don't trust your peers, the org will be dysfunctional
- There's often a clash between analytical vs. creative/intuitive. Find common ground, don't dismiss people you don't understand.
- When you have to decide, provide a unified front. Openly disagreeing with your peers makes things worse.

The Echo

- You will be watched; people want your approval and want to avoid your criticism.
- You are the person in charge, not one of the team.
- Socializing heavily with your team is a thing of the past. Show up briefly, then leave to let them be together without you.
- Detaching removes the possibility of playing favorites, and makes it less ambiguous whether your thinking out loud or asking them to do something.
- Your presence will change the tone of the meetings you attend.
- The hard decisions you make that impact the business won't be appropriate to discuss with everyone. Transparency can be damaging.
- You still need to care about your team as people. Don't treat them like cogs; people can tell when their leaders stop caring about them.

Ruling with Fear, Guiding with Trust

- You do not want to manage by fear.
- Practice relatedness. Don't focus on efficiency at the expense of getting to know people.
- Apologize. Show people you understand that your behavior has an impact on others.
- Get curious when you disagree.
- Learn how to hold people accountable without making them bad

True North

- Set the baseline of what excellence looks like in your role.
- Help people develop instincts so they can be trusted to independently follow the guidelines without much direction.
- Leaders rely on the wisdom they've developed over time to make fast decisions. This takes experience and time.

Questions

1. At this level, your coaching and mentoring are likely to come from people outside your company. You no longer have a manager, you have a boss. Do you have a professional coach, either provided by work or paid for yourself? This is a good investment even if your job doesn't pay for it. A coach can give you guidance and direct feedback, and unlike your friends, she's paid to listen to you talk.
2. Beyond a coach, how is your support network of peers outside your company? Do you know other senior managers at companies in your area? A peer group helps you see what the job looks like at other companies, and is a place where you can share experiences and get advice.
3. Do you particularly admire any technology senior managers? What is it about them that you admire? What could you be doing to be more like them, if anything?
4. Think back to the last time you needed to change priorities for part or all of your team. How did it go? What went well, and what didn't go well? How did you communicate the change to your team, and what was their reaction? If you were to do it again, what is one thing you would do differently?
5. How well do you understand where your business is going for the foreseeable future? Do you understand the technology strategy that will help you get there? What are the critical areas of focus for the team, such as feature velocity, performance, technical innovation, and hiring, that

need to evolve to reach the goals of the company? Where are the bottlenecks and opportunities for technology evolution to push the business forward?

6. How is your relationship with the other members of the company's senior leadership team? Which relationships are good and which are bad? What could you do to improve the bad relationships? How well do you understand the priorities of these team members, and how well do you think they understand your priorities?
7. If I asked your team which executives you got along with and which ones you hated, would they be able to tell me without hesitation? When the CEO or the leadership team comes to a decision that you don't agree with, are you capable of leaving that disagreement behind in supporting the decision to the rest of the company?
8. Are you behaving like a role model for your team? Would you be happy to learn that people are emulating your behavior on a daily basis? When you sit in on meetings with your team, do you dominate the conversation or are you more interested in listening and observing?
9. When was the last time you asked someone you don't talk to regularly about his life outside work? The last time someone emailed to say she was sick and couldn't come in, did you take a minute to wish her a quick recovery?

Bootstrapping Culture

- Instead of structure, talk about learning. Instead of process, talk about transparency. "We don't set up systems because structure and process have inherent value. We do it because we want to learn from our successes and our mistakes, and to share those successes and encode the lessons we learn from failures in a transparent way."
- In a startup, many decisions need to be made; sometimes those will be undone multiple times.
- Cultivate decisiveness in the face of a massive number of options. You don't need perfect; you need something.
- "Structure is how we scale, diversity, and take on more complex long-term tasks."
- If structure comes too late, you see a high degree of confusion and wasted effort as the team grows.
- An early startup feels like a race car; as you grow, it feels like a commercial flight where if you fail multiple people suffer. Later it feels like a spacecraft.

Assessing Your Role

- Things start simple, then become complex. At some point you'll experience failure – which is an opportunity to identify what needs changing.
- We often attribute success to our own actions and bad luck to our bad actions.
- Structure helps these problems… every new hire slows things down for 3 months, people leave because there's no path to advancement, production outages because people fix things on production. It's all about learning what's not working.

Creating Your Culture

- Consciously guiding the culture of your team is part of a leader's job
- Culture = unspoken shared rules of a community
- Cultural values are the glue that enables us to work as a team and make decisions when faced with uncertainty
- Not every person will fit in at every company
- You will be measured by the company values whether you like it or not; this is why it's best to make sure you and the people you hire align with the culture

Applying Core Values

- Define the culture
- Reward people for exhibiting its values in positive ways; the stories we tell as a community bond us together
- Learn to spot people who have values that conflict with yours
- Use your values as part of the interview process. This is not a "friendship test", as this increases the likelihood of you finding people like you.
- Examples of poor fit…
    - Smart engineer who values independence but has to collaborate across multiple teams
    - Someone who thinks the most analytical argument wins in an environment that values empathy and intuition

Creating Cultural Policy

- These documents are challenging to create, but more and more people are sharing templates
- You typically only need structure when things are failing
- What works for one company may not translate to your company.

Writing a Career Ladder

- Solicit participation from your team
- Look for examples
- Be detailed
- Use both long-form descriptions and summaries
- Consider how the ladder relates to salary
- Provide many early opportunities for advancement
- Use narrow salary bands for early-career stages
- Use wide salary bands when and where you have fewer levels
- Consider your breakpoint levels (i.e., up or out)
- Recognize achievement
- Split management and technical tracks
- Consider making people management skills a mid-career requirement
- Use rules of thumb for years of experience
- Don't be afraid to evolve over time

Cross-functional Teams

- Having multiple skills on your team breaks up the "us vs. them" siloes. This puts everyone who is needed to make the project successful in one group.
- There's a split management in squads/pods. The reporting structure stays the same, but the group is determined by their roadmap.
- You'll need to determine if you need stronger business skills or technical skills, because one will end up running the company.

Developing Engineering Process

- Engineers dislike process that doesn't make sense and that doesn't come with a "why"
- Without any process, your teams will struggle to scale. With the wrong process, they will be slowed down.
- Process is risk management. As things get bigger, they are too much to keep in our heads.
- A complicated process should exist only for activities that you expect to be rare, or activities where the risks are not obvious to people.
    - Process on simple stuff will slow things down. If you want something systemic (e.g., code reviews on all checkins), make that process lightweight.
    - Socialize risk to the team as a whole

Depersonalize Decision Making

- Code reviews are about quality and knowledge transfer, not a platform for criticizing people
    - Be clear about code review expectations
    - Use a linter for style issues
    - Keep an eye on the review backlog

- Outage postmortem (a.k.a. learning review)
    - Resist the urge to point fingers and blame
    - Look at the circumstances around the incident and understand the context of the events
    - Be realistic about which takeaways are important and which are worth dropping
- Architecture review
    - Be specific about the kinds of changes that need architecture reviews
    - The value of the review is in preparing for the review
    - Choose the review board wisely

Questions

1. What policies do you have now? What practices? Have you written any of them down yet? When was the last time you revisited them?
2. Do you have company values? What are they? How do you recognize them in your team?
3. Do you have a career ladder? Do you feel it accurately reflects the team today? Does it reflect the team you want to have in the future? If not, can you improve it?
4. What risks are most concerning for your team? For your company? How can you mitigate those risks without burdening your team with unnecessary processes and bureaucracy?