Notebook - The Manager's Path

kindle

Fournier, Camille

Loc 92 | Highlight

> What engineering managers do, though, is not pure people management.

Loc 92 | Highlight

> We are managing groups of technical people, and most of us come into the role from a position of hands-on expertise. I wouldn't recommend trying to do it any other way! Hands-on expertise is what gives you credibility and what helps you make decisions and lead your team effectively. There are many parts of this book dedicated to the particular challenges of management as a technical discipline.

Page 1 | Highlight

> The secret of managing is keeping the people who hate you away from the ones who haven't made up their minds.

Page 2 | Highlight

> There are, however, other options. Managers who care about you as a person, and who actively work to help you grow in your career. Managers who teach you important skills and give you valuable feedback. Managers who help you navigate difficult situations, who help you figure out what you need to learn. Managers who want you to take their job someday. And most importantly, managers who help you understand what is important to focus on, and enable you to have that focus.

Page 2 | Highlight

> 1-1s serve two purposes. First, they create human connection between you and your manager.

Page 2 | Highlight

> But letting your manager into your life a little bit is important, because when there are stressful things happening (a death in the family, a new child, a breakup, housing woes), it will be much

Page 2 | Highlight Continued

> easier to ask your manager for time off or tell him what you need if he has context on you as a person. Great managers notice when your normal energy level changes, and will hopefully care enough to ask you about it.

Page 2 | Highlight

> Being an introvert is not an excuse for making no effort to treat people like real human beings, however. The bedrock of strong teams is human connection, which leads to trust.

Page 3 | Highlight

> The second purpose of a 1-1 is a regular opportunity for you to speak privately with your manager about whatever needs discussing.

Page 4 | Highlight

> Ideally, the feedback you get from your manager will be somewhat public if it’s praise, and private if it’s criticism.

Page 4 | Highlight

> Good managers know that delivering feedback quickly is more valuable than waiting for a convenient time to say something. Praising in public is considered to be a best practice because it helps the manager let everyone know that someone has done something laudable, and reinforces what positive behavior looks like.

Page 4 | Highlight

> Asking your manager for advice is also a good way to show that you respect her. People like to feel helpful, and managers are not immune to this sort of flattery.

Page 4 | Highlight

> This usually requires you to say something, though. If you don’t ask your manager about a promotion, do not expect her to just give you one magically.

Page 4 | Highlight

kindle

> It's great when managers can identify and assign stretch projects that will help us grow and learn new things.

Page 5 | Highlight

> Your manager should be the person who shows you the larger picture of how your work fits into the team's goals, and helps you feel a sense of purpose in the day-to-day work. The most mundane work can turn into a source of pride when you understand how it contributes to the overall success of the company.

Page 5 | Highlight

> As the main liaison between you and the bureaucracy of the company, your manager holds some responsibility for helping you find training and other resources for career growth. This may be helping you find a conference to attend or a class to take, helping you get a book you need, or pointing you to an expert somewhere else in the company who can help you learn something.

Page 5 | Highlight

> Whatever kind of company you work for, expect that you are responsible, for the most part, for figuring out what types of training you want.

Page 7 | Highlight

> Developing a sense of ownership and authority for your own experiences at work, and not relying on your manager to set the entire tone for your relationship, is an important step in owning your career and workplace happiness.

Page 7 | Highlight

> Your manager can point out opportunities for growth. She can show you projects. She can provide feedback on your areas of learning and development. But she cannot read your mind, and she cannot tell you what will make you happy.

Page 8 | Highlight

kindle

> In all of this uncertainty, the only person you can rely on to pull through it is yourself. Your manager cannot do that for you. Use your manager to discover what's possible where you are, but look to understand yourself in order to figure out where you want to go next.

Page 9 | Highlight

> Your relationship with your manager is like any other close interpersonal relationship. The only person you can change is yourself.

Page 9 | Highlight

> Especially as you become more senior, remember that your manager expects you to bring solutions, not problems. Try not to make every 1-1 about how you need something, how something is wrong, or how you want something more. When you have a problem, instead of demanding that your manager solve it for you, try asking her for advice on how she might approach the problem. Asking for advice is always a good way to show respect and trust.

Page 9 | Highlight

> Strong managers know how to play the game at their company. They can get you promoted; they can get you attention and feedback from important people. Strong managers have strong networks, and they can get you jobs even after you stop working for them.

Page 9 | Highlight

> Plenty of great engineers make ineffective managers because they don't know or want to deal with the politics of leadership in their companies. A strong engineer may make a great mentor-manager to someone early in his career, but a terrible advocate-manager for someone who is more senior.

Page 11 | Highlight

> Many organizations use mentors as part of their onboarding process for all new hires.

Page 12 | Highlight

kindle

> Being a Mentor If you find yourself in the mentor's seat, congratulations! This is an experience that not everyone will get: an opportunity to learn in a fairly safe way about the job of management, and the feeling of being responsible for another person.

Page 13 | Highlight

> Without a project, your intern has a good chance of being completely lost and bored the entire summer.

Page 13 | Highlight

> Listen carefully Listening is the first and most basic skill of managing people. Listening is a precursor to empathy, which is one of the core skills of a quality manager.

Page 17 | Highlight

> Effective teams have good onboarding documents they provide to new hires. Things like step-by-step guides to setting up their development environments, learning how tracking systems work, and familiarizing themselves with the tools they will need for the job are crucial for new hires. These documents should constantly evolve to meet the changes of the workplace itself. Mentoring a new hire by helping her work through the documents, and having her modify those documents with any surprises she encounters during onboarding, provides a powerful message of commitment to her. It shows her that she has the power and obligation to learn, and to share what she's learned for the benefit of your whole team.

Page 18 | Highlight

> You may be an introvert, or someone who does not find socializing easy, but

---

Page 18 | Highlight

> conscious effort and practice in getting to know new people and helping them succeed will pay off.

Page 20 | Highlight

kindle

> In some offices, whether in a mentoring relationship or outside of one, you'll encounter an "alpha geek." The alpha geek is driven to be the best engineer on the team, to always have the right answer, and to be the person who solves all the hard problems.

Page 20 | Highlight

> At their worst, alpha geeks can't let anyone else get any glory without claiming some of it for themselves. They are the origin of any good ideas but had no part in creating the bad ideas, except that he knew they would fail.

Page 21 | Highlight

> Alpha geeks who believe that their value comes from knowing more than others can also hide information in order to maintain their edge, which makes everyone on the team less effective.

Page 22 | Highlight

> When you need to assign a mentor for your new hire or intern, figure out what you're hoping to achieve by creating the relationship. Then, find the person who can help meet those goals.

Page 22 | Highlight

> First of all, figure out why you are setting up this mentoring relationship in the first place.

Page 22 | Highlight

> If the mentee doesn't know how to ask for help or what to do with the mentoring relationship, it often feels like forced socializing and a waste of time for both parties.

Page 22 | Highlight

> Emotional labor is a way to think about traditionally feminine "soft skills" — that is, skills that address the emotional needs of people and teams. Because the outcome can be hard to quantitatively measure, emotional labor is often dismissed as less important work than writing software. It's assumed to be something that should just be provided without financial recognition.

Page 25 | Highlight

kindle

> Mentoring provides a great opportunity to cultivate curiosity and see the world through fresh eyes. When faced with a mentee's questions, you can start to observe what about your organization is not so obvious to a new person.

Page 25 | Highlight

> While many people think creativity is about seeing new things, it's also about seeing patterns that are hidden to others. It's hard to see patterns when the only data points you have are your own experiences.

Page 25 | Highlight

> mentoring forces you to hone your communication skills. It

Page 25 | Highlight

> requires you to practice listening, in particular, because if you can't hear the questions you're being asked, you'll never be able to provide good answers.

Page 28 | Highlight

> My job as tech lead was to continue to write code, but with the added responsibilities of representing the group to management, vetting our plans for feature delivery, and dealing with a lot of the details of the project management process.

Page 28 | Highlight

> The tech lead may not have exactly the same role from company to company, or even from team to team within a company, but we know from the title that it is expected to be both a technical position and a leadership role, and that it is often a temporary set of responsibilities rather than a permanent title.

Page 29 | Highlight

> Perhaps a better shorthand for this is the description used by Patrick Kua in his book, Talking with Tech Leads: A leader, responsible for a (software) development team, who spends at least 30

Page 29 | Highlight Continued

> percent of their time writing code with the team.

Page 29 | Highlight

> You can't lead without engaging other people, and people skills are what we're asking the new tech lead to stretch, much more than pure technical expertise. However, tech leads will be working on one major new technical skill: project management.

Page 30 | Highlight

> Being a tech lead is an exercise in influencing without authority.

Page 31 | Highlight

> the willingness to step away from the code and figure out how to balance your technical commitments with the work the whole team needs.

Page 31 | Highlight

> From now on, wherever you go in your career, balancing is likely to be one of your core challenges.

Page 31 | Highlight

> The worst scheduling mistake is allowing yourself to get pulled randomly into meetings.

Page 31 | Highlight

> Part of your leadership is helping the other stakeholders, such as your boss and the product manager, respect the team's focus and set up meeting calendars that are not overwhelming for individual contributors.

Page 32 | Highlight

> In the systems architect and business analyst roles, you identify the critical systems that need to

> change and the critical features that need to be built in order to deliver upcoming projects. The goal here is to provide some structure for basing estimates and ordering work.

Page 32 | Highlight

> Project planners break work down into rough deliverables. With this hat on, you're learning to find efficient ways of breaking down the work so that the team can work quickly. Part of the challenge here is getting as much productive work done in parallel as possible.

Page 33 | Highlight

> Software developers and team leaders write code, communicate challenges, and delegate. As projects move forward, unexpected obstacles arise. Sometimes tech leads are tempted to go to heroics and push through these obstacles themselves, working excessive overtime to get it all done. In your position as tech lead, you should continue writing code, but not too much.

Page 33 | Highlight

> Even if you are tempted to pull a rabbit out of the hat yourself, you must communicate this obstacle first.

Page 33 | Highlight

> As you can see from these descriptions, in the process of being a tech lead, you have to act as a software developer, a systems architect, a business analyst, and a team leader who knows when to do something single-handedly, and when to delegate the work to others.

Page 35 | Highlight

> Doesn't agile software development get rid of the need for project management? No. Agile software development is a great way to think about work because it forces you to focus on breaking tasks down into smaller chunks, planning those smaller chunks out, and delivering value incrementally instead of all at once. None of this means that you don't need to understand how to do project management. You'll have projects that for whatever reason can't be completed in a single sprint, or even two small sprints. You'll need to estimate project length for your management team, and give some detail on why you believe things will take that long. There are some projects, usually described by words like infrastructure, platform, or system, that require architecture or

Page 35 | Highlight Continued

> significant advanced planning. When faced with this kind of project, which includes many unknowns and relatively hard deadlines, you will find it doesn't fit so well into the standard agile process.

Page 35 | Highlight

> Project management for a long-running, team-based project is not what most people consider fun. I find it tedious and sometimes kind of scary.

Page 36 | Highlight

> Ultimately, the value of planning isn't that you execute the plan perfectly, that you catch every detail beforehand, or that you predict the future; it's that you enforce the self-discipline to think about the project in some depth before diving in and seeing what happens. A degree of forethought, in places where you can reasonably make predictions and plans, is the goal. The plan itself, however accurate it turns out, is less important than spending time on the act of planning.

Page 37 | Highlight

> I've always been surprised how grateful senior technical managers have been when I can explain some very basic modern ideas (e.g., what's this NoSQL stuff all about, and why should I care?) to them in a nonthreatening and noncondescending way.

Page 38 | Highlight

> Project management is the act of breaking a complex end goal down into smaller pieces, putting those pieces in roughly the most effective order they should be done, identifying which pieces can be done in parallel and which must be done in sequence, and attempting to tease out the unknowns of the project that may cause it to slow down or fail completely.

Page 38 | Highlight

> Break down the work. Get out a spreadsheet, or a Gantt chart, or whatever works for you, and start breaking down your big deliverable (say, rewriting your billing system) into tasks.

Page 39 | Highlight

kindle

> Run a premortem, an exercise where you go through all the things that could fail on the launch of this big project.

Page 39 | Highlight

> Make a launch plan; make a rollback plan. And at the end of it, don't forget to celebrate!

Page 40 | Highlight

> Your days are spent in a mix of deep thinking, solving hard problems that challenge you intellectually but are still fun and novel, and collaborating with other deep thinkers.

Page 41 | Highlight

> In short, you have the perfect balance of engaging work, fame, and accumulated expertise that makes you invaluable and respected, highly paid, and influential.

Page 45 | Highlight

> The process czar believes that there is one true process that, if implemented correctly and followed as designed, will solve all of the team's biggest problems. Process czars may be obsessed with agile, Kanban, scrum, lean, or even waterfall methods.

Page 45 | Highlight

> Engineers who believe in the "right tool for the job" sometimes turn into process czars when they become tech leads, seeking out the right tool to solve all issues with planning, focus, time management, and prioritization. They try to stop all work while they search for the perfect process, or constantly push new tools and processes on the team as solutions to the messier problems of human interactions.

Page 45 | Highlight

> The opposite of the process czar is not a manager who gives up on process completely, but rather someone who understands that processes must meet the needs of the team and the work. Ironically, while "agile" is often implemented in a rigid way, the principles of the Agile Manifesto are

Page 45 | Highlight Continued

> a great summary of healthy process leadership: Individuals and interactions over processes and tools Working software over comprehensive documentation Customer collaboration over contract negotiation Responding to change over following a plan

---

Page 46 | Highlight

> My other piece of advice is to look for self-regulating processes. If you find yourself playing the role of taskmaster — criticizing people who break the rules or don't follow the process — see if the process itself can be changed to be easier to follow. It's a waste of your time to play rules cop, and automation can often make the rules more obvious.

Page 46 | Highlight

> Understand the Architecture If you go into a tech lead role and you don't feel that you fully understand the architecture you are supporting, take the time to understand it. Learn it. Get a sense for it. Visualize it. Understand its connections, where the data lives, how it flows between systems.

Page 47 | Highlight

> If you start to make all of the technical decisions without soliciting the input of your team, they'll resent you and blame you when things go wrong. On the other hand, if you make no technical decisions and leave everything up to the team, decisions that could have been made quickly can drag on without resolution.

Page 49 | Highlight

> figure out your own management style.

Page 50 | Highlight

> We'll talk about the main tasks required to manage people: Taking on a new report Holding regular 1-1s Giving feedback on career growth, progression toward goals, areas for improvement, and praise as warranted Working with reports to identify areas for learning and helping them grow in these areas via project work, external learning, or additional mentoring

Page 50 | Highlight

kindle

> Build Trust and Rapport One strategy is to ask a series of questions that are intended to help you get to know the aspects of the person that impact your ability to manage him well. These questions might include: How do you like to be praised, in public or in private?
>
> Some people really hate to be praised in public. You want to know this. What is your preferred method of communication for serious feedback? Do you prefer to get such feedback in writing so you have time to digest it, or are you comfortable with less formal verbal feedback? Why did you decide to work here? What are you excited about? How do I know when you're in a bad mood or annoyed? Are there things that always put you in a bad mood that I should be aware of?
>
> Maybe a direct report fasts for religious reasons, which sometimes makes him cranky. Maybe he always gets stressed out while on-call. Maybe he hates reviews season. Are there any manager behaviors that you know you hate?
>
> If you asked me this question, my answer would be: skipping or rescheduling 1-1s, neglecting to give me feedback, and avoiding difficult conversations. Do you have any clear career goals that I should know about so I can help you achieve them? Any surprises since you've joined, good or bad, that I should know about?
>
> Things like: Where are my stock options? You promised me a relocation bonus and I haven't gotten it yet. Why are we using SVN and not Git? I didn't expect to be so productive already! For more ideas, see Lara Hogan's excellent blog post on the topic. Create a 30/60/90-Day Plan

---

Page 51 | Highlight

> The more senior the hire, the more he should participate in creating this plan.

Page 51 | Highlight

> For early- to mid-career hires, one aspect of onboarding will likely include contributing to the team's onboarding documentation. A best practice in many engineering teams is to create a set of onboarding documents that are edited by every new hire as he gets up to speed. He edits the documentation to reflect processes or tools that have changed since the last hire, or points that he found confusing.

Page 52 | Highlight

> Communicate Your Style and Expectations Your new hire needs to understand your expectations

> and your style just as much as you need to understand his.

Page 52 | Highlight

> Get Feedback from Your New Hire One final piece of advice: get as much feedback as you can about the new hire's perspective on the team in that first 90 days. This is a rare period, where a new person comes in with fresh eyes and often sees things that are hard for the established team members to see. On the other hand, remember that people in their first 90 days lack the context that the overall team possesses, so take their observations with the requisite grain of salt, and definitely don't encourage people in this period to criticize the established processes or systems in a way that makes the existing team feel attacked.

Page 52 | Highlight

> Regular 1-1s are like oil changes; if you skip them, plan to get stranded on the side of the highway at the worst possible time. Marc Hedlund

---

Page 53 | Highlight

> When you meet less frequently, any missed 1-1s must be rescheduled, which is usually a drag on both of you.

Page 54 | Highlight

> Some people assume that good relationships require very little attention, and spend all of their time on their bad relationships. But there are plenty of people, myself included, who feel a strong need for regular 1-1 time even in good relationships.

Page 55 | Highlight

> My goal in a 1-1 is first to listen to anything my direct reports want to discuss. I want the meeting to be driven by them, and I want to give them space to bring up whatever they feel is important. I view a 1-1 session as much as a creative discussion as a planning meeting.

Page 55 | Highlight

> As much as possible, when someone does something that needs immediate corrective feedback (insulting a colleague, missing a critical meeting, using inappropriate language), don't wait for the 1-1 to provide that feedback.

Page 56 | Highlight

> When you get to the stage where you're managing managers, a lot of your 1-1 meetings will be diving into details of projects they're overseeing that you don't have time to dig into on your own.

Page 56 | Highlight

> Getting to Know You Whatever type of 1-1 you do, leave room to get to know the person reporting to you as a human being.

Page 57 | Highlight

> Good Manager, Bad Manager: Micromanager, Delegator

Page 57 | Highlight

> With this support, Beth feels confident running the project, but also knows that Sharell has her back, and when things get stressful toward the end of it, Beth enlists Sharell's help to cut scope and get the project out on time.

Page 58 | Highlight

> The hardest thing about micromanagement is that there are times when you need to do it. Junior engineers often thrive under detailed oversight because they want that specific direction.

Page 58 | Highlight

> Trust and control are the main issues around micromanagement.

Page 58 | Highlight

> Autonomy, the ability to have control over some part of your work, is an important element of motivation. This is why micromanagers find it so difficult to retain great teams. When you strip creative and talented people of their autonomy, they lose motivation very quickly.

Page 58 | Highlight

> On the other hand, delegation is not the same thing as abdication. When you're delegating responsibility, you're still expected to be involved as much as is necessary to help the project succeed.

Page 59 | Highlight

> It's important to remember that being a good leader means being good at delegating.

Page 59 | Highlight

> When you feel like you want to micromanage, ask the team how they're measuring their success and ask them to make that visible to you on an ongoing basis.

Page 59 | Highlight

> When you are managing a team that doesn't have a clear plan, use the details you'd want to monitor to help them create one. What are you holding them accountable against this month, this quarter, or this year? If you can't answer that question, the first step is to help the team create those goals.

Page 59 | Highlight

> Gather Information from the Systems Before Going to the People As engineers, we have an advantage because systems can push valuable information up without the team needing to do much of anything. If you want to know the status of work, look at the version control system and the ticketing system.

Page 59 | Highlight

> The worst micromanagers are those who constantly ask for information they could easily get

Page 59 | Highlight Continued

> themselves.

Page 59 | Highlight

> The team will not be productive or happy spending half their time gathering information for you that you could easily find yourself.

Page 59 | Highlight

> And remember, this information is just a piece of the context, not the whole picture, and it means nothing without the goals just discussed.

Page 60 | Highlight

> Different details are important at different project stages.

Page 60 | Highlight

> Establish Standards for Code and Systems I'm one of those deeply technical managers, and I have opinions about the way systems should be built and operated. Letting go has been hard for me, so I developed some guidelines to help me feel better about the structure around these issues. Developing basic standards as a team helps everyone communicate with one another in code and design reviews, and it depersonalizes the process of providing technical feedback.

Page 60 | Highlight

> For me, basic standards meant things like how much unit testing we expected to happen with each change (generally speaking, as some tests were always required), and at what point technical decisions should be reviewed by a larger group (like when someone wants to add a new language or framework to the stack). As with goal setting, putting standards in place here helps people know which details are important to think about when they're creating the technology.

Page 60 | Highlight

> Treat the Open Sharing of Information, Good or Bad, in a Neutral to Positive Way

Page 60 | Highlight

kindle

> The goal here isn't to punish him with micromanagement for his failure to communicate status, because all you're doing is punishing yourself and hindering his ability to be held accountable for his own work. Instead, your goal is to teach Jack what he needs to communicate, when, and how.

Page 61 | Highlight

> Hiding important information intentionally is a failure, and getting stuck on a problem or making a mistake is often just an opportunity for learning.

Page 61 | Highlight

> In the long run, if you don't figure out how to let go of details, delegate, and trust your team, you're likely to suffer personally. Even if your team doesn't quit, you'll end up working longer and longer hours as your responsibilities increase.

Page 61 | Highlight

> Your time is too valuable to waste, and your team deserves a manager who is willing to trust them to do things on their own.

Page 61 | Highlight

> If performance reviews make you shudder, you're not alone. Unfortunately, the review process is not something that every manager takes seriously or handles in a mature way.

Page 61 | Highlight

> That experience starts long before the reviews are written. It starts with continuous feedback.

Page 61 | Highlight

> Continuous feedback is, more than anything, a commitment to regularly sharing both positive and corrective feedback.

Page 61 | Highlight

kindle

> the most important thing is that the team has adopted a culture of providing feedback frequently.

Page 61 | Highlight

> For you, as a new manager, getting into the habit of continuous feedback is training you to pay attention to individuals, which in turn makes it easier to recognize and foster talent. You're also practicing the art of having small and occasionally tricky conversations with individuals about their performance. Few people are comfortable with providing one-on-one praise or correction, and this helps you get over the feeling of awkwardness.

Page 62 | Highlight

> Know your people. The first required part of successfully giving continuous feedback is a basic understanding of the individuals on your team. What are their goals, if any? What are their strengths and weaknesses?

Page 62 | Highlight

> Observe your people. You can't give feedback if you aren't paying attention. If anything, I think the best outcome of attempting a continuous feedback cycle is not necessarily the actual feedback generated, but rather that the effort forces you to start paying attention to the individuals on your team. Starting this habit early in your management career, while you still may have only a few people to manage, helps you build up those observational muscles. Practice looking for talents and achievements on your team, first and foremost.

Page 62 | Highlight

> Provide lightweight, regular feedback. Start with positive feedback. It's both easier and more fun to give positive feedback than it is to give corrective feedback.

Page 63 | Highlight

> Positive feedback also makes your reports more likely to listen to you when you need to give them critical feedback. When they believe that their manager sees the good things they do, they'll be more open to hearing about the areas where they might improve.

Page 63 | Highlight

kindle

> Bonus: Provide coaching. Ultimately, continuous feedback works best when you, as a manager, pair that feedback with coaching. As situations arise, use coaching to ask people what they might have done differently.

Page 63 | Highlight

> The 360 model is a performance review that includes feedback from, in addition to a person's manager, his teammates, anyone who reports to him, and coworkers he regularly interacts with, as well as a self-review.

Page 64 | Highlight

> Give yourself enough time, and start early

Page 64 | Highlight

> This process isn't something you can knock out in an hour and do well. You have a million things on your plate, but plan to spend solid, uninterrupted time working on reviews. Work from home if you need to. You owe your team enough time to read the collected feedback, digest it, and summarize it well.

Page 65 | Highlight

> Try to account for the whole year, not just the past couple of months

---

Page 65 | Highlight

> Use concrete examples, and excerpts from peer reviews

Page 65 | Highlight

> Spend plenty of time on accomplishments and strengths

Page 65 | Highlight

> When it comes to areas for improvement, keep it focused

Page 65 | Highlight

> Here are some examples of themes that I have seen. There are people who: Struggle with saying no to distractions and end up helping with other projects instead of finishing their own Do good work but are hard for others to work with, tending to be overly critical or rude in meetings, code reviews, or other collaborative activities

---

Page 66 | Highlight

> Struggle to break their work up into intermediate deliverables, and don't balance planning and design with getting things done Work well with other engineers but do not work well with other departments or teams Struggle to follow the accepted best practices of the team, cut corners, or otherwise do sloppy work

---

Page 66 | Highlight

> Avoid big surprises Set expectations appropriately before reviews are delivered.

Page 67 | Highlight

> Schedule enough time to discuss the review I usually give people a printed copy of the review as they're leaving on the evening before the review is scheduled.

Page 67 | Highlight

> Biases lead us to assume potential, and moreover, to give people the benefit of the doubt long past the point at which they've shown that their "potential" is an illusion.

Page 68 | Highlight

> Real potential shows itself quickly. It shows itself as working hard to go the extra mile, offering insightful suggestions on problems, and helping the team in areas that were previously neglected.

Page 69 | Highlight

kindle

> If you're a manager, you are going to play a key role in getting people on your team promoted. Sometimes it will simply be up to you to determine who gets promoted, but more commonly promotions will be reviewed by your management, or a committee. So you'll not only need to have a good idea about who deserves to be promoted, but you'll need to make a case for their promotion as well.

Page 69 | Highlight

> The important thing for you to start doing now that you're in management is to learn how the game is played at your company. Every company has its own variation of the promotion process, and you're probably in this role because you survived it.

Page 71 | Highlight

> One of the basic rules of management is the rule of no surprises, particularly negative ones. You need to understand what a person is supposed to be giving you, and if that isn't happening, make it clear to her early and often that she is not meeting expectations.

Page 72 | Highlight

> You'll always need to have a record of negative feedback to fire someone in any environment where HR is active and a standard performance improvement plan is required.

Page 73 | Highlight

> This is what is meant by "coaching out." Make the situation clear to him. You have told him repeatedly what the next level looks like, and he has not been able to show that he can work at that level, so you don't think that your team is the right place for him to grow his career. You aren' t firing him,

---

Page 73 | Highlight

> but you are telling him that he needs to move on if he wants to progress.

Page 75 | Highlight

kindle

> managing a team is more than just doing the job of managing the individuals.

Page 77 | Highlight

> contrite, he returned to our team and agreed to work with me. It turns out, being a good manager isn't about having the most technical knowledge. The work of supporting people was far more important to management success.

Page 77 | Highlight

> Engineering management is a technical discipline, not just a set of people skills.

Page 77 | Highlight

> your job will require that you guide technical decision making.

Page 77 | Highlight

> Furthermore, if you truly wish to command the respect of an engineering team, they must see you as technically credible. Without technical credibility you face an uphill battle, and even though you may be able to get into a position of leadership in one company, your options will be limited. Don't underestimate the value of your technical skills as you work to become a successful engineering manager.

Page 79 | Highlight

> Infrequent releases can hide pain points such as poor tooling around releases, heavily manual testing, features that are too big, or developers who don't know how to break their work down.

Page 80 | Highlight

> Making the code-shipping resource far less scarce immediately improved team morale.

Page 80 | Highlight

kindle

> A less critical version of this situation is the person who just stirs up drama, who dwells on negative experiences, or who spends a bit too much time on gossip and playing games of us-against-them.

Page 80 | Highlight

> Sometimes the negative person is just unhappy and the best thing to do is to help him leave the team on good terms;

Page 80 | Highlight

> Be careful that vocally negative people don't stay in that mindset on your team for long. The kind of toxic drama that is created by these energy vampires is hard for even the best manager to combat. The best defense is a good offense in this case, and quick action is essential.

Page 80 | Highlight

> My advice is to dedicate 20% of your time in every planning session to system sustainability work ("sustainability" instead of the more common "technical debt").

Page 81 | Highlight

> But they'll remember whether their

Page 81 | Highlight

> manager was with them during the stressful period, or off somewhere else, doing her own thing.

Page 81 | Highlight

> There's no quick fix here, but showing a willingness to improve collaboration goes a long way.

Page 81 | Highlight

> You can make the situation worse by undermining your peers in front of your team, so even when

Page 81 | Highlight Continued

> you are frustrated with them, try to stay positive and supportive of their efforts in public.

Page 82 | Highlight

> This experience can be deeply awkward, so the first thing to do is to acknowledge that. If you're now managing someone who was truly your peer, acknowledge the weirdness of the transition.

Page 82 | Highlight

> Using your managerial power to override technical decisions is usually a bad idea.

Page 83 | Highlight

> Your team won't be successful if your former peers all quit because they can't stand working for you. They're going to be extra-sensitive to any disagreements or perceived power grabs on your part. They may even do things to try to undermine you. Pick your battles. In the long run, handling this transition with maturity will pay off.

Page 83 | Highlight

> It's valuable for everyone to realize that they can and should focus on the things they can impact and change, and ignore the things they can't. Drama in the workplace is usually little more than an ego-entertaining drain.

Page 83 | Highlight

> However, it's unrealistic to expect that you can or should shield your team from everything. Sometimes it's appropriate to let some of the stress through to the team.

Page 83 | Highlight

> The extreme shielders think they can best focus and motivate their teams by giving clear goals. But humans usually need some sort of context into why these goals have been set, and thereby into what problems they're working to solve.

Page 84 | Highlight

kindle

> If you instead communicate information about such events in a straightforward, low-emotion way, you alleviate the gossip and quickly neutralize the impact on your team.

Page 84 | Highlight

> You have more responsibility than you may expect. While the product manager is responsible for the product roadmap, and the tech lead is responsible for the technical details, you are usually accountable for the team's progress through each of these elements. The nature of leadership is that, while you may only have the authority to guide decisions rather than dictate them, you'll still be judged by how well those decisions turn out.

Page 85 | Highlight

> Review the Outcome of Your Decisions and Projects Talk about whether the hypotheses you used to motivate projects actually turned out to be true.

Page 85 | Highlight

> It's easy to forget to review assumptions after the project is done, but if you make this a habit for yourself and your team, you'll always learn from your decisions.

Page 87 | Highlight

> While it seems like Jason's democratic style should lead to an empowered team, his inability to say no or to take the responsibility for any decisions means that no one feels very secure.

Page 87 | Highlight

> Creating a safe environment for disagreement to work itself out is far better than pretending that all disagreement does not exist.

Page 87 | Highlight

> The Dos and Don'ts of Managing Conflict Don't rely exclusively on consensus or voting.

Page 87 | Highlight

kindle

> Consensus can appear morally authoritative, but that assumes that everyone involved in the voting process is impartial, has an equal stake in the various outcomes, and has equal knowledge of the context.

Page 87 | Highlight

> Don’t set people up for votes that you know will fail instead of taking the responsibility as a manager of delivering that bad news yourself.

Page 87 | Highlight

> Do set up clear processes to depersonalize decisions.

Page 87 | Highlight

> When you want to allow for group decision making, the group needs to have a clear set of standards that they use to evaluate decisions. Start with a shared understanding of the goals, risks, and the questions to answer before making a decision.

Page 87 | Highlight

> Don’t turn a blind eye to simmering issues.

Page 88 | Highlight

> Do address issues without courting drama. There’s a difference between addressing conflict and cultivating dysfunction.

Page 88 | Highlight

> The goal is to identify problems that are causing the team to work less effectively together and resolve them, not to become the team’s therapist.

Page 88 | Highlight

> Don’t take it out on other teams. Ironically, conflict-avoidant managers often seek conflict when it comes to other teams.

Page 88 | Highlight

> When something goes wrong, like an incident that spans across teams, the manager turns into a bully and demands justice for his team, or blames the problems on the other team.

Page 88 | Highlight

> Do remember to be kind. It’s natural and perfectly human to want to be liked by other people. Many of us believe that the way to be liked is to be seen as nice — that niceness is itself the goal. Your goal as a manager, however, should not be to be nice, it should be to be kind.

Page 89 | Highlight

> Don’t be afraid. Conflict avoidance often arises from fear. We’re scared of the responsibility of making the decision. We’re afraid of seeming too demanding. We’re afraid people will quit if we give them uncomfortable feedback. We’re afraid people won’t like us, or that we’ll fail when we take this risk. Some fear is natural, and being sensitive to the outcomes of conflict is a wise habit.

Page 90 | Highlight

> The real goal here is psychological safety — that is, a team whose members are willing to take risks and make mistakes in front of one another.

Page 90 | Highlight

> When companies talk about hiring for “culture fit,” they often mean they want to hire people they can be friendly with.

Page 90 | Highlight

> This is why those who undermine team cohesion are so problematic. They almost always behave in a way that makes it hard for the rest of the team to feel safe around them. We refer to these employees as “toxic” because they tend to make everyone who comes into contact with them less

Page 90 | Highlight Continued

> effective. Dealing with them quickly is an important part of managing well.

Page 90 | Highlight

> One variant of the toxic employee is the brilliant jerk, who, as we discussed earlier, produces individually outsized results, but is so ego-driven that she creates a mixture of fear and dislike in almost everyone around her.

Page 91 | Highlight

> The best way to avoid brilliant jerk syndrome is to simply not hire one. Once they're hired, getting rid of brilliant jerks takes a level of management confidence that I think is uncommon.

Page 91 | Highlight

> The best thing you can do for your team, in the context of having a brilliant jerk, is to simply and openly refuse to tolerate bad behavior. This may be one of the few instances where "praise in public, criticize in private" is upended.

Page 91 | Highlight

> Your first goal is to protect your team as a whole, the second is to protect each individual on the team, and your last priority is protecting yourself.

Page 91 | Highlight

> Another very common problem team member is the noncommunicator — the person who hides information from you, from his teammates, from his product manager.

Page 92 | Highlight

> Whatever the cause, this person disrupts team cohesion because he isn't being collaborative with the rest of his teammates; he doesn't feel safe sharing his work in progress, and his fear often sets an example for the rest of the team.

Page 92 | Highlight

> The Employee Who Lacks Respect The third type of toxic individual is the person who simply doesn't respect you as a manager, or who doesn't respect her teammates.

Page 92 | Highlight

> Simply put, if your team member doesn't respect you or her peers, why is she working there?

Page 92 | Highlight

> That's it? That's it. You can't have a person working for you who doesn't respect you, or doesn't respect your team.

Page 93 | Highlight

> may have to decide which projects to take on, and when to push back on accepting projects.

Page 93 | Highlight

> Project Management Rules of Thumb

Page 93 | Highlight

> None of this is a replacement for agile project management

Page 93 | Highlight

> When it comes

Page 93 | Highlight

> to actually planning the details of the smaller pieces, an agile process where the team collaborates to divide and roughly estimate work is very effective at smoothing and organizing the day-to-day.

Page 94 | Highlight

> You have 10 productive engineering weeks per engineer per quarter

---

Page 94 | Highlight

> Budget 20% of time for generic sustaining engineering work across the board

---

Page 94 | Highlight

> By “generic sustaining engineering work,” I mean testing, debugging, cleaning up legacy code, migrating language or platform versions, and doing other work that has to happen.

Page 94 | Highlight

> if you fill the schedule to 100% with feature development, expect that the feature development will quickly slow down as a result of this overscheduling.

Page 94 | Highlight

> As you approach deadlines, it is your job to say no

Page 94 | Highlight

> you will need to figure out when to push for a hack implementation and when to hold back for the right implementation.

Page 95 | Highlight

> Use the doubling rule for quick estimates, but push for planning time to estimate longer tasks

---

Page 95 | Highlight

> Be selective about what you bring to the team to estimate

Page 95 | Highlight

> it’s distracting and stressful for engineers to have a manager who’s constantly asking them for random project estimates.

Page 95 | Highlight

> Joining a Small Team

Page 95 | Highlight

> First, get someone to walk you through the systems and architecture, as well as the process for testing and releasing the software.

Page 96 | Highlight

> Plan to work on at least a couple of features in your first 60 days.

Page 96 | Highlight

> By getting to know the code, the processes for writing code, and the tools and systems your team use for their day-to-day, you will gain the understanding necessary for managing the team,

---

Page 96 | Highlight

> and the technical credibility necessary for them to see you as a capable leader.

Page 101 | Highlight

> So often we diminish these small efforts as not worthwhile, but they’re very good at keeping you in tune with the feeling of software development and showing your teams that you are willing and able to help out with the day-to-day in a valuable way.

Page 102 | Highlight

> The tech industry is filled with people who despise management, thinking it’s not as important a job as writing code. But management is a job, it is a necessary and important job, and in particular, it’s

Page 102 | Highlight Continued

> your job right now.

Page 102 | Highlight

> Writing code is full of quick wins, especially for the experienced developer.

Page 103 | Highlight

> Managing your time comes down to one important thing: understanding the difference between importance and urgency.

Page 104 | Highlight

> Urgency

Page 104 | Highlight

> is often more clearly felt than importance.

Page 105 | Highlight

> Any sort of standard meeting that involves a group of people, whether it's planning, retrospective, or postmortem, should have a clear procedure and expected outcomes.

Page 109 | Highlight

> Tasks like project planning, systems design, or being the key person during an outage are the biggest opportunity you have to grow talent on your team while also making the team run better. Strong managers spend a lot of their time developing members of their teams in these areas.

Page 109 | Highlight

> It is your responsibility, as a manager, to build up the talent in your organization, and to help your people learn new skills they'll need for the next stages of their careers.

Page 111 | Highlight

> Instead, it looks like the “yes, and” technique of improvisational comedy. “Yes, we can do that project, and all we will need to do is delay the start of this other project that is currently on the roadmap.” Responding with positivity while still articulating the boundaries of reality will get you into the major leagues of senior leadership.

Page 112 | Highlight

> “Help me say yes” means you ask questions and dig in on the elements that seem so questionable to you.

Page 112 | Highlight

> “Not right now” implies that you might agree with the idea but can’t do it at this moment, so maybe you’ll get to it in the future.

Page 112 | Highlight

> But as I discussed earlier, when you give an implied promise that “not right now” means that you’ ll seriously do something “later,” you need to be sure that later can actually happen.

Page 112 | Highlight

> Sometimes you will appeal to the finance department to help you say no to certain budget excesses. Playing good cop/bad cop can be a little bit dishonest, so use this sparingly, but it can be useful

---

Page 113 | Highlight

> to lend your authority to a no and then be able to call in the favor when you need support for your own no in the future.

Page 115 | Highlight

> These health signals — frequency of code releases, frequency of code check-ins, and infrequency of incidents — are the key indicators of a team that knows what to do, has the tools to do it, and has the time to do it every day.

Page 116 | Highlight

> Here's the thing. Is your team working to its full capacity? Are your engineers challenged and growing? Is your product team excited by the progress you're making? Are people able to spend most of their time on writing new code and evolving the systems? If so, great. Ignore me. You've got it under control. If not, you have a problem, and you're ignoring that problem at your peril.

Page 116 | Highlight

> The beauty of pushing for more frequent releases is that it often uncovers a host of interesting challenges.

Page 117 | Highlight

> The goal here is to balance risk in such a way that neither incident frequency

---

Page 117 | Highlight

> nor incident prevention turns into a job that takes developers away from writing code for days at a time.

Page 118 | Highlight

> When risk reduction turns into weeks of manual QA, excessive and slow code reviews, infrequent releases, or a drawn-out planning process, the increased analysis can leave developers idle and restless, without necessarily reducing the risk of incidents.

Page 118 | Highlight

> They unite the team by emphasizing how this identity is special as compared to other teams. When they go too far, this identity is used to make the team feel superior to the rest of the company, and the team is more interested in its superiority than the company's goals. Rallying a team in this way is a shallow binding that is vulnerable to many dysfunctions: Fragile to the loss of the leader.

Page 119 | Highlight

> Resistant to outside ideas.

Page 119 | Highlight

> This means that they miss opportunities to learn and grow.

Page 119 | Highlight

> Empire building. Leaders who favor an us-versus-them style tend to be empire builders, seeking out opportunities to grow their teams and their mandates

---

Page 119 | Highlight

> without concern for what is best for the overall organization. This often results in competition with other leaders for headcount and control of projects. Inflexibility. These groups tend to struggle against change that comes from outside the group. Reorganizations, cancelled projects, and shifting focus all can cause breaks in the core parts of their identity.

Page 119 | Highlight

> how you’re going to create a team that works well with this culture, not against it.

Page 120 | Highlight

> this purpose-based binding makes teams: Resilient to loss of individuals.

Page 120 | Highlight

> Driven to find better ways to achieve their purpose.

Page 120 | Highlight

> First-team focused. Leaders who are strong team players understand that the people who report to them are not their first team. Instead, their first team is their peers across the company. This first-team focus helps them make decisions that consider the needs of the company as a whole before focusing on the needs of their team.

Page 120 | Highlight

> Open to changes that serve their purpose.

Page 121 | Highlight

> The Virtues of Laziness and Impatience

Page 121 | Highlight

> Impatience can be rude if it’s directed at individuals.

Page 121 | Highlight

> But impatience paired with laziness is wonderful when you direct it at processes and decisions. Impatience and laziness, applied to process, are the key elements to focus.

Page 121 | Highlight

> As you grow more into leadership positions, people will look to you for behavioral guidance. What you want to teach them is how to focus. To that end, there are two areas I encourage you to practice modeling, right now: figuring out what’s important, and going home.

Page 121 | Highlight

> What is the value in automation if you don’t use it to make your job easier?

Page 121 | Highlight

> So be impatient to figure out the nut of what’s important. As a leader, any time you see something being done that feels inefficient, question it: Why does this feel inefficient to me? What is the value in the thing we are doing? Can we deliver that value faster? Can we strip down this project into something simpler and get it done more quickly?

Page 122 | Highlight

> Laziness and impatience. We focus so we can go home, and we encourage going home because it

> forces us to constantly focus. This is how great teams scale.

Page 126 | Highlight

> People who are good at managing a single team, or even a couple of related teams, fall apart when asked to manage managers, or teams that are outside of their skill set. They're unable to balance the ambiguities inherent in their new role, and fall back into things that they find easy.

Page 126 | Highlight

> Here, you need to follow up on all the little things until you figure out what you don't need to follow up on. Is recruiting happening? Are your managers coaching their teams? Has everyone written up their goals for the quarter? Have you reviewed them? What is the status of that project that should be finishing up? That production incident that happened the other day — did the postmortem happen? Did you read the report?

Page 126 | Highlight

> In this chapter, we'll discuss some of the keys to successfully overseeing an entire division, including: How to get information from your skip-level reports What it means to hold your managers accountable Managing first-time and experienced managers

---

Page 127 | Highlight

> Hiring new managers Figuring out the root of organizational dysfunction Cultivating your teams' technical strategy

---

Page 127 | Highlight

> The open-door policy is nice in theory, but it takes an extremely brave engineer to willingly take the risk of going to her boss (or especially her boss's boss, etc.) to tell him about problems.

Page 128 | Highlight

> So, part of the job is simply to make sure that your 1-1s have room for real conversation beyond a script or a set of to-dos, as I mentioned earlier. But beyond that, you must make the time to

Page 128 | Highlight Continued

> proactively hold skip-level meetings with the people who report to your direct reports.

Page 128 | Highlight

> Skip-level meetings are one of the critical keys to successful management at levels of remove.

Page 128 | Highlight

> One form of skip-level meeting is a short 1-1 meeting, held perhaps once a quarter, between the head of an organization and each person in that organization.

Page 128 | Highlight

> Some suggested prompts to provide the person you are holding the skip-level 1-1 with include: What do you like best/worst about the project you are working on? Who on your team has been doing really well recently? Do you have any feedback about your manager — what's going well, what isn't? What changes do you think we could make to the product? Are there any opportunities you think we might be missing? How do you think the organization is doing overall? Anything we could be doing better/more/less? Are there any areas of the business strategy you don't understand? What's keeping you from doing your best work right now? How happy (or not) are you working at the company? What could we do to make working at the company more fun?

Page 130 | Highlight

> In the group setting, these questions can be used to draw out information: What can I, your manager's manager, provide for you or your team? Anything I should be helping with?

Page 130 | Highlight

> Is this team working poorly with any other teams, from your perspective? Are there any questions about the larger organization that I can answer?

Page 130 | Highlight

> The purpose of this skip-level process, beyond maintaining trust and engagement, is to help you

> detect places in which you're being "managed up" well, to the detriment of the team under that manager. Having people who manage up well in your organization is always a hard situation to detect and respond to. These individuals get to you first, so you hear their perspective before you hear anything else, and you're predestined to think they're in the right and to support their decisions.

Page 131 | Highlight

> Whether you have experienced managers or first-timers reporting to you, there is one universal goal for these relationships: they should make your life easier.

Page 131 | Highlight

> They're more than just people who take some 1-1 meetings off your hands; they are responsible for taking a team of people and helping that team succeed. When they repeatedly fail to do this, they're failing to do their job.

Page 131 | Highlight

> Well, that sounds good, except for one little thing: sometimes managers make your life easier by hiding problems and telling you what you want to hear, until months later you see things falling apart and wonder where you went wrong.

Page 131 | Highlight

> With all of this responsibility split into different roles, when can you hold a manager accountable?

Page 131 | Highlight

> Unstable product roadmap The team doesn't feel very productive, the systems are unstable, and there is some attrition happening, but the product organization keeps changing the goals for the team and everything is always an urgent mandate. Is the manager accountable? Errant tech lead

---

Page 131 | Highlight

> The tech lead has been down a rabbit hole trying to redesign one of the core systems. The design

> doc is still barely started, and work is piling up, but the tech lead insists that this is a big problem that can't be rushed. Is the manager accountable? Full-time firefighting mode The manager inherited a team with a bunch of legacy systems that are constantly broken, and the team seems to spend all their time fighting fires. They also support other teams who are using those systems, and the other teams are constantly asking for help and distracting the team with requests. There' s a roadmap to move out of these systems, but you haven't heard any reports on the progress against this roadmap, and you know the team is killing themselves to keep things stable and manage the support requests. Is the manager accountable? The answer to all of these questions is yes. Yes, despite the mitigating circumstances in each case, the manager ultimately needs to take responsibility for pulling the team out of these situations and getting them moving forward, because the manager is accountable for the health and productivity of the team.

Page 132 | Highlight

> When the tech lead is down a rabbit hole, the manager has to bring that person out and work with him to figure out how to make the design process more transparent,

---

Page 132 | Highlight

> bringing in other senior people from other teams if necessary as mentors or collaborators to help him deconstruct the problem and make forward progress.

Page 133 | Highlight

> Managers need coaching and guidance in the same way that individual contributors need coaching and guidance.

Page 133 | Highlight

> Marcus is a people pleaser. He has a deep aversion to ever directly making people he cares about unhappy.

Page 133 | Highlight

> By trying to make everyone happy, people pleasers often burn themselves out.

Page 134 | Highlight

> Unfortunately, this can mean he amplifies drama and negativity, and disappoints his team by making promises to them that he cannot possibly keep. On the other side, there is the external pleaser. She very much wants to make her boss and her external partners happy, and is terrified of revealing problems on her team. As a result, she spends a great deal of energy managing upward and outward, and in particular tends to significantly overcommit her team.

Page 135 | Highlight

> In both cases, the people pleaser struggles to say no, and sends contradictory messages to both the team and the external parties.

Page 135 | Highlight

> You might think people pleasers create teams that feel safe to be vulnerable and fail, but in fact the opposite is true. These managers make it hard for the team to fail in a healthy way, because of the manager's own fears of failure and possible rejection.

Page 136 | Highlight

> When you're managing a people pleaser, one of the best things you can do is show the person that he's exhibiting the behavior, and highlight the downsides.

Page 137 | Highlight

> Managers who neglect the job are bad, but managers who take to the job with gusto because they believe it's the key to realizing authority are sometimes even worse.

Page 137 | Highlight

> If your new manager is skipping your 1-1s or evading questions about what the team is working on, you may have a control freak on your hands.

Page 138 | Highlight

> Making the wrong person a manager is a mistake, but keeping her in that position once you've realize she's wrong for it is a critical error. I am hugely in favor of making engineers who wish to go

Page 138 | Highlight Continued

> into management take baby steps of mentoring and managing very small teams, but this is not always possible and doesn't always shake out problems that come with scale.

Page 139 | Highlight

> Of course, there can be major downsides. Management tends to be a very culture-specific task in a company. I can give you best practices all day, but if you either work as a manager or hire a manager for a company that's not a good culture fit, you'll have problems.

Page 139 | Highlight

> We talk a lot about culture fit for all hiring, but managers create subcultures, and a manager who creates an incompatible subculture can be a problem if you want your teams to work together well.

Page 139 | Highlight

> If you're building a dynamic, product-centric engineering team, you need managers who understand how to work with teams who ship software frequently, who are comfortable with modern development process best practices, and who can inspire creative product-centric engineers.

Page 139 | Highlight

> These skills are so much more important than industry-specific knowledge. It's easier to gain access to industry information than it is to retrain someone who doesn't know how to work in your culture. Don't compromise on culture fit, especially when hiring managers.

Page 139 | Highlight

> Experienced managers will have different ideas about management than you do, and you'll have to work out the differences.

Page 140 | Highlight

> How do you inspire experienced managers? The difference between an experienced manager and

> a new manager is that the experienced person should be capable of managing independently. This means that a lot of the coaching you provide will be less about the nuts and bolts of management and more about how he can have a larger impact on strategy and direction setting for his area.

Page 140 | Highlight

> Many people are very reluctant to hire in management from outside, and for good reason.

Page 140 | Highlight

> Hiring for managers is a multipart exercise, and those parts are actually very similar to those of a good engineering interview process. First, make sure that the person has the skills you need. Second, make sure that she's a culture match for your organization. The biggest difference between a management interview and an engineering interview is that managers can, theoretically, bullshit you more easily.

Page 141 | Highlight

> Someone who communicates well in a management interview, who talks a good game, can still come in and get nothing done.

Page 141 | Highlight

> Let's start with 1-1s. As we've discussed, 1-1s are an essential tool for a manager to determine the health of her team and gather and impart valuable information. Any manager you hire should role-play a few 1-1s as part of the interview process. One of the best ways to do this is by asking the people who would report to the new manager to interview her by asking her to help with problems they have right now, or have had in the recent past.

Page 141 | Highlight

> You can take it a step further and actually role-play other types of difficult situations, like dealing with an employee who is underperforming, or delivering a negative performance review.

Page 141 | Highlight

> manager must also be able to debug teams.

Page 141 | Highlight

> Ask the manager to describe a time when she ran a project that was behind schedule, and what she did in that scenario. Or ask her to role-play with an employee who is thinking about quitting.

Page 141 | Highlight

> Ask the manager to describe how she's coached employees who were struggling, and helped great employees grow to new levels. Ask her about her management philosophy. If she doesn't have one at all, that might be a red flag. While a new manager may not be able to answer this question well, an experienced manager who has no clear philosophy is a cause for concern. What does she think the job of a manager is? How does she stay hands-on, and how does she delegate?

Page 142 | Highlight

> Plenty of otherwise excellent managers are uncomfortable speaking in front of strange audiences.

Page 142 | Highlight

> about technical skills? You want to make sure that you get enough of a sense of a candidate's technical skills that she'll be able to establish credibility to the team she'll be managing.

Page 142 | Highlight

> You might also have her mediate a technical debate between engineers who disagree on the solution to a problem.

Page 142 | Highlight

> I've seen managers from big companies who treated their peers well but their underlings and other lower-level staff members like they were less than human, which caused massive friction in the startup space.

Page 143 | Highlight

> Culture fit is so important in managers because they shape their teams to their culture, and they hire new people based on their cultural ideas.

Page 143 | Highlight

> In his book High Output Management,1 Andy Grove talks about cultural values as one of the ways that people make decisions inside of highly complex, uncertain, or ambiguous circumstances where they value the group interest above their own.

Page 143 | Highlight

> Finally, I would be remiss if I did not point out one of the critical elements to hiring in new managers: the reference check. Do thorough reference checks for anyone you're planning to bring on board, even if you've worked with that person before. Ask the references to describe the ways that the person succeeds as well as the ways she fails.

Page 144 | Highlight

> Reference checks, even ones chosen carefully by the candidate, often reveal a lot about what you can expect to get when hiring her. Don't leave out this crucial step.

Page 144 | Highlight

> An interesting way to combat this problem is to use the same mindset that I advised when we talked about mentorship — namely, being very curious. Remember that you're not expected to know everything just because you're a manager. Use this to your advantage. Ask the person to teach you about the work she does. Sit down with her and treat her as if she were your mentor, the person to teach you the ropes for this job. Whether it's QA, design, product management, or technical operations, ask lots of questions, but in an open way.

Page 145 | Highlight

> Grit your teeth and make the time to get comfortable with each area; take time to get to know the manager and employees in the team, and practice asking for details about the area, so that you can start to learn and develop a sense for what the people in that team are actually doing.

Page 145 | Highlight

kindle

> Debugging Dysfunctional Organizations I believe that the best engineering managers are often great debuggers.

Page 147 | Highlight

> Ask the team what their goals are. Can they tell you? Do they understand why those are the goals? If they don't understand the goals of their work, their leaders (manager, tech lead, product manager) aren't doing a good job engaging the team in the purpose of the work. In almost every model of motivation, people need to feel an understanding and connection with the purpose of their work.

Page 149 | Highlight

> Sadly, we are often asked this question in times when things are not taking any longer than the estimate. We are often asked this question when our leadership, for whatever reason, either didn' t like the original estimate or didn't ask for it at all, and now they're upset, despite nothing going wrong.

Page 149 | Highlight

> Therefore, you must always be aggressive about sharing estimates and updates to estimates, even when people don't ask, especially if you believe that the project is critical or likely to take longer than a few weeks.

Page 150 | Highlight

> Another core element of agile software development is the emphasis on learning from the past. When estimates are wrong, what are we learning about unknown complexity? What are we learning about what is worth estimating, when? What are we learning about how we communicated those estimates and who was disappointed by the miss?

Page 151 | Highlight

> Strategies for Handling Roadmap Uncertainty

Page 151 | Highlight

kindle

> Be realistic about the likelihood of changing plans given the size and stage of the company you work for.

Page 151 | Highlight

> Think about how to break down big projects into a series of smaller deliverables so that you can achieve

---

Page 151 | Highlight

> some of the results, even if you don’t necessarily complete the grand vision.

Page 152 | Highlight

> Don’t overpromise a future of technical projects.

Page 152 | Highlight

> If the project is not urgently important, you can put it on the backlog, but you should be realistic that once “later” rolls around, there will be a long list of competing priorities from other parts of the business.

Page 152 | Highlight

> Dedicate 20% of your team’s schedule to “sustaining engineering.”

Page 152 | Highlight

> Understand how important various engineering projects really are.

Page 152 | Highlight

> How big is that project? How important is it? Can you articulate the value of that project to anyone who asks? What would successful completion of the project mean for the team?

Page 152 | Highlight

kindle

> The value of these questions is that you start to treat big technical projects the same way as product initiatives.

Page 152 | Highlight

> These projects have advocates and goals, they have schedules, and they are managed like other big initiatives.

Page 153 | Highlight

> So, back to our uncertain roadmap. Projects change. Teams may even be disbanded or moved around in ways that you don't understand or agree with. As a manager, the best thing you can do is help people feel capable of tying up loose ends, stabilizing the current in-flight projects, and easing into their new work in a controlled fashion. This is an area where you can and should push back. Make sure that your teams get adequate time to finish up current work. Furthermore, push for engineering involvement in the early planning for the new work so that people can get excited about the projects they are moving on to.

Page 154 | Highlight

> To move forward, systems need constant technical work: new languages, frameworks, infrastructure, and features. There's only a limited amount of development time and energy that can go into improving these systems, and you're accountable for making sure the team is placing its technical bets in the right places. You oversee these investments by matching the proposed technical projects and improvements to the future of the product or customer needs.

Page 154 | Highlight

> You need to know enough about the work to sniff out misguided efforts and evaluate proposed investments.

Page 154 | Highlight

> You ensure that engineers make decisions with an understanding of the business perspective and the future of the product roadmap.

Page 154 | Highlight

kindle

> By knowing enough about the progress of your teams, the projects, and bottlenecks, you can filter out technically infeasible ideas and map new initiatives onto ongoing projects.

Page 155 | Highlight

> Managers who don’t stay technical enough sometimes find themselves in the bad habit of acting as a go-between for senior management and their teams. Instead of filtering requests, they relay them to the team and then relay

---

Page 155 | Highlight

> the team’s response back up to management. This is not a value-add role.

Page 155 | Highlight

> how should you invest your time in order to stay technically relevant? Read the code.

Page 155 | Highlight

> Pick an unknown area, and ask an engineer to explain it to you.

Page 156 | Highlight

> Attend postmortems.

Page 156 | Highlight

> Keep up with industry trends in software development processes.

Page 156 | Highlight

> Foster a network of technical people outside of your company.

Page 156 | Highlight

> Never stop learning.

Page 159 | Highlight

> Your first job is to be a leader. The company looks to you for guidance on what to do, where to go, how to act, how to think, and what to value. You help set the tone for interactions. People join the company because they believe in you, in the people you hired, and in the mission you helped to craft.

Page 160 | Highlight

> management tasks into four general categories:

Page 160 | Highlight

> Information gathering or information sharing

Page 160 | Highlight

> The strong senior leader is capable of synthesizing large quantities of information quickly, identifying critical elements of that information, and sharing the information with the appropriate third parties in a way they will be able to understand. Nudging

---

Page 161 | Highlight

> Decision making

Page 161 | Highlight

> Role modeling

Page 162 | Highlight

> The common roles include: Research and development (R&D)

Page 162 | Highlight

> experimentation, research, and new technology generation.

Page 162 | Highlight

> Technology strategy/visionary

Page 163 | Highlight

> he uses business and technology trends to guide his decisions. Organization

Page 163 | Highlight

> Execution

Page 163 | Highlight

> Face of technology, external

Page 163 | Highlight

> Infrastructure and technical operations manager

Page 163 | Highlight

> Business executive

Page 164 | Highlight

> The VP is usually at the top of the management career ladder for engineers. This generally means that the VP is expected to be an experienced manager of people, projects, teams, and departments.

Page 165 | Highlight

> The VP of Engineering job is both a big one and a detail-oriented one.

Page 165 | Highlight

kindle

> This is one of the reasons it's such a hard job to hire for, even though most companies will need to hire this role in from outside. This person has to be good at quickly understanding what's going on in an organization. She must be able to gain people's trust and show wisdom in management and leadership. Unfortunately, most engineers are reluctant to trust people without technical credibility, but many managers at this level of seniority won't be interested in going through a hard technical interview process only to take on a role that's mostly focused on organizational management.

Page 166 | Highlight

> CTO is not an engineering role. CTO is not the top of the technical ladder, and it is not the natural progression engineers should strive to achieve over the course of their careers. It's not a role most people who love coding, architecture, and deep technical design would enjoy doing. It follows that the CTO is not necessarily the best engineer in the company.

Page 166 | Highlight

> CTO is the technical leader at the company's current stage of evolution.

Page 166 | Highlight

> the CTO should be the

Page 166 | Highlight

> strategic technical executive the company needs in its current stage of evolution.

Page 167 | Highlight

> First and foremost, a CTO must care about and understand the business, and be able to shape business strategy through the lens of technology.

Page 167 | Highlight

> Strong CTOs also have significant management responsibility and influence.

Page 167 | Highlight

kindle

> Other executives will have ideas and needs for technology. The CTO must protect the technology team from becoming a pure execution arm for ideas without tending to its own needs and its own ideas.

Page 167 | Highlight

> It’s incredibly difficult to maintain influence and effectiveness as an executive with no reporting power.

Page 168 | Highlight

> You can’t give up the responsibility of management without giving up the power that comes with it.

Page 168 | Highlight

> My advice for aspiring CTOs is to remember that it’s a business strategy job first and foremost. It’s also a management job. If you don’t care about the business your company is running — if you’re not willing to take ultimate responsibility for having a large team of people effectively attacking that business — then CTO is not the job for you.

Page 170 | Highlight

> You didn’t explicitly go through the list of things in flight and kill or postpone work in order to make room for this priority. You need to do that, if it’s truly urgent. Saying something is top priority is one thing, but making

---

Page 170 | Highlight

> the actual tradeoffs on the schedule to get people moving on it is completely different.

Page 170 | Highlight

> However, when you’re taken to task for not focusing on the right priority, it’s a sign that you and the CEO have a misaligned understanding of reality, and you need to get on the same page.

Page 171 | Highlight

> the more the job becomes making sure that the organization moves in the direction it needs to move in, and that includes changing direction when needed.

Page 171 | Highlight

> never underestimate how many times and how many ways something needs to be said before it sinks in.

Page 171 | Highlight

> You'll also need to repeat information when you're communicating up. When you want your boss to act on

---

Page 171 | Highlight

> something, expect that you'll need to tell him the same thing three times before he actually listens.

Page 172 | Highlight

> During that process, I went from having only a vague idea of what setting strategy meant to having a concrete, forward-thinking strategy encompassing a way to think about both the technical architecture and the engineering team's structure, which in turn ultimately influenced the way the company itself thought about its overall structure.

Page 173 | Highlight

> plan out actionable

Page 173 | Highlight

> ideas to improve operational efficiency, expand features, and grow the business.

Page 174 | Highlight

> As you can see from this tale, good technology strategy here meant several things. It meant

> technology architecture, yes. It also meant team structure. It meant understanding the underpinnings of the business and the directions in which it was headed. I like to describe technology strategy for product-focused companies as something that "enables the many potential futures of the business." It's not just a reactive document that tries to account for current problems, but it anticipates and enables future growth. If you're in a product-focused business, this is the heart of your technology strategy. It's not about actually deciding the product's direction, but about enabling the larger roadmap to play out successfully.

Page 175 | Highlight

> Don't blast an impersonal message to a large group. The worst way to communicate bad news is via impersonal mediums like email and chat, especially mediums with commenting abilities. Your team deserves to hear the message coming from your mouth directly, and without you to guide the message, you can expect some misunderstandings and bubbling animosity.

Page 175 | Highlight

> That being said, the second-worst way to deliver this message, especially to a large group that you know won't be happy, is with them all in a room at once.

Page 175 | Highlight

> Do talk to individuals as much as possible.

Page 175 | Highlight

> Think about the people who are going to have the strongest reaction, and try to tailor the news to them.

Page 175 | Highlight

> Don't force yourself to deliver a message you can't stand behind.

Page 175 | Highlight

> As someone in senior leadership, you have to learn how to maturely handle decisions you don't

> agree with, but that doesn’t mean you have to go it alone.

Page 176 | Highlight

> Do be honest about the likely outcomes.

Page 176 | Highlight

> Do think about how you would like to be told.

Page 176 | Highlight

> Ask the CTO: I Have a Nontechnical Boss

Page 176 | Highlight

> Don’t hide information behind jargon, and be careful with details.

Page 177 | Highlight

> Expect that you will need to run your 1-1s with your new boss, so come prepared with a list of topics.

Page 177 | Highlight

> Try to bring solutions, not problems to be solved.

Page 177 | Highlight

> Ask for advice.

Page 177 | Highlight

> Don’t be afraid to repeat yourself.

Page 177 | Highlight

> Be supportive.

Page 178 | Highlight

> Actively look for coaching and skill development in other places.

Page 178 | Highlight

> To start with, you let them own their areas, and they let you own yours.

Page 178 | Highlight

> Giving her respectful deference when it comes to her turf is fundamental. If you disagree with her management style or application of her skill set in places where it isn't directly affecting your team, you treat that disagreement like you would treat a good friend who happens to date people you don't love. Unless she asks for your advice, try to stay out of it as much as possible, and certainly approach any disagreement you choose to discuss with kindness. Be willing to let those differences lie.

Page 180 | Highlight

> When there's a happy hour, you go for a drink and then leave the team to socialize.

Page 180 | Highlight

> Socializing heavily with your team outside of working hours is a thing of the past.

Page 180 | Highlight

> You need to detach for a few reasons. First, if you don't detach, you're likely to be accused of playing favorites.

Page 180 | Highlight

> Second, you need to detach because you need to learn how to lead effectively, and leading

Page 180 | Highlight Continued

> effectively requires people to take your words seriously.

Page 181 | Highlight

> If you try to maintain a “buddy” image, your reports are going to have a hard time distinguishing

---

Page 181 | Highlight

> between their buddy thinking out loud and their boss asking them to focus on something.

Page 181 | Highlight

> Transparency that may have been harmless or even possibly helpful at lower levels of management can become incredibly damaging to the stability of your team at this level.

Page 182 | Highlight

> She’s also sometimes short-tempered, and when people don’t live up to her expectations or things go wrong, she can be visibly annoyed and openly angry.

Page 182 | Highlight

> His first instinct is to ask questions, and these questions often cause the team to come to their own realizations about what’s going wrong.

Page 183 | Highlight

> How do you know if you’re creating a culture of fear? It can come from placing a high value on being correct and following the rules, and having a strong affinity for hierarchy-based leadership.

Page 183 | Highlight

> Correcting a Culture of Fear Practice relatedness.

Page 183 | Highlight

kindle

> If you want a team that feels comfortable taking risks and making mistakes, one of the core requirements is a sense of belonging and safety.

Page 183 | Highlight

> Most people are scared to take risks in front of people they think will reject them if they fail.

Page 183 | Highlight

> by neglecting to create even basic personal relationships with many of my team members, I made them afraid of how I would react to mistakes, questions, and failures.

Page 184 | Highlight

> Apologize. When you screw up, apologize. Practice apologizing honestly and briefly.

Page 184 | Highlight

> apologizing doesn't make you weaker — it makes the whole team stronger.

Page 184 | Highlight

> Get curious. When you disagree with something, stop to ask why. Not every disagreement is an undermining of your authority.

Page 184 | Highlight

> Learn how to hold people accountable without making them bad.

Page 185 | Highlight

> The culture of fear is pretty common in technology, and it survives best in environments where things are otherwise going well.

Page 185 | Highlight

> the baseline of what excellence looks like in this function. I call it “True North.”

Page 186 | Highlight

> For a technical leader, True North means making sure that you’ve done your job getting things ready to go into production. It means you have honored your agreed-upon policies for review, operational oversight, and testing. It means that you won’t put something into production that you don’t believe is ready for your users to experience. It means you’re creating software and systems you’re proud of.

Page 186 | Highlight

> Another way to think of this is through the lens of risk analysis. Risk analysis doesn’t mean that we don’t take risks. Some things that are generally considered “bad” can be OK under certain circumstances. These include: Having a single point of failure Having known bugs and issues Being unable to tolerate high load Losing data Putting out code that is undertested Having slow performance

---

Page 187 | Highlight

> True North leaders rely on the wisdom they’ve developed over time to make fast decisions when they don’t have time to delve into all the details.

Page 187 | Highlight

> Leadership and Self-Deception: Getting Out of the Box

Page 187 | Highlight

> Daring Greatly: How the Courage to Be Vulnerable Transforms the Way We Live, Love, Parent, and Lead (New

---

Page 187 | Highlight

> The Effective Executive

Page 191 | Highlight

kindle

> A common failing of first-time CTOs is to underestimate the importance of being clear and thoughtful about the culture of the engineering team.

Page 191 | Highlight

> When talking about structure with skeptics, I try to reframe the discussion. Instead of talking about structure, I talk about learning. Instead of talking about process, I 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. This learning and sharing is how organizations become more stable and more scalable over time.

Page 193 | Highlight

> One of the greatest writings about organizational politics is a piece called "The Tyranny of Structurelessness" by Jo Freeman. While the article is about early feminist/anarchist collectives, Freeman's insights apply equally well to startup culture. Pretending to lack structure tends to create hidden power structures resulting from the nature of human communication and the challenges of trying to scale that communication.

Page 193 | Highlight

> It is task oriented. Its function is very narrow and very specific, like putting on a conference or putting out a newspaper. It is the task that basically structures the group.

Page 193 | Highlight

> It is relatively small and homogeneous. Homogeneity is necessary to insure that participants have a "common language" for interaction.

Page 193 | Highlight

> There is a high degree of communication. Information must be passed on to everyone, opinions checked, work divided up, and participation assured in the relevant decisions.

Page 194 | Highlight

> There is a low degree of skill specialization.

Page 194 | Highlight

> Be that as it may, the unstructured organization either displays characteristics that ultimately make it less self-directed than the members might wish to believe, or is run by hidden hierarchies and power dynamics. In many cases both things are true to some extent.

Page 194 | Highlight

> It's no surprise that we usually end up refactoring spaghetti code when we want to make it scalable, because refactoring usually involves identifying and explicitly drawing out structure in order to make the code base easier to read and work in. That, in short, is the value of structure. Structure is how we scale, diversify, and take on more complex long-term tasks.

Page 195 | Highlight

> Nothing is more ridiculous than a small team with a rigid hierarchy.

Page 195 | Highlight

> it's more common in small companies to see structure come too late.

Page 195 | Highlight

> a high degree of confusion and wasted effort.

Page 196 | Highlight

> The more people you have, the more thoughtful structure you need to get everyone moving in the right direction.

Page 196 | Highlight

> The longer a company is around, the more habits become entrenched.

Page 196 | Highlight

> the more existing business rules and infrastructure you have, the more you'll need clarity on how to handle them.

Page 196 | Highlight

> you in a highly regulated industry? Do you have a lot to lose if certain types of mistakes are made? Or are you in an unregulated industry, with little on the line? Your structures and processes should reflect this.

Page 196 | Highlight

> John Gall's book Systemantics:1 A complex system that works is invariably found to have evolved from a simple system that worked. A complex system designed from scratch never works and cannot be patched up to make it work. You have to start over with a working simple system.

Page 197 | Highlight

> at some point you'll start to experience failure, and failure is the best place to investigate and identify where your structure needs to change.

Page 197 | Highlight

> Using failure to guide evolution lets you apply structure at the right level.

Page 198 | Highlight

> the risk involved in shipping a bug is much lower than that of not expanding features quickly enough to keep up with competition.

Page 198 | Highlight

> When every new hire slows the team down for months because there is no onboarding process, that is a failure due to lack of structure. When people regularly leave the company because they have no path to advancement or

Page 198 | Highlight

> career growth, that is a failure due to lack of structure.

Page 198 | Highlight

> I prefer to talk about learning and transparency rather than using the word structure,

Page 198 | Highlight

> Culture is how things get done, without people having to think about it. Frederick Laloux, Reinventing Organizations: A Guide to Creating Organizations Inspired by the Next Stage of Human Consciousness

Page 199 | Highlight

> Consciously guiding the culture of your team is part of a leader's job, and to do this well, you need to understand what it means in the first place.

Page 199 | Highlight

> Culture is the generally unspoken shared rules of a community.

Page 199 | Highlight

> in complex environments where the needs of the group must override the needs of the individual, cultural values are the glue that enables us to work as a team and make decisions when faced with uncertainty.

Page 200 | Highlight

> employees who truly embrace and exhibit all of the core values of a company tend to do well naturally.

Page 200 | Highlight

> First, define your culture.

Page 201 | Highlight

kindle

> Second, reinforce your culture by rewarding people for exhibiting its values in positive ways.

Page 201 | Highlight

> One of the most important uses of performance reviews is to evaluate the alignment between team members' values and the company's values, and therefore what values should be part of your performance review process.

Page 201 | Highlight

> "Happiness and positivity is a choice"

Page 202 | Highlight

> cultural fit is not about hiring friends.

Page 202 | Highlight

> culture fit as determined by friendship tests is almost certain to be discriminatory in some way.

Page 202 | Highlight

> don't be vague when discussing fit. Be specific. What are the values of this team, and where have you noticed any match or mismatch?

Page 203 | Highlight

> I realized that we had no salary structure at all. Because of that lack of structure, most people were paid based on a combination of their previous jobs' salary and their negotiating skills.

Page 204 | Highlight

> Here are some important issues to consider when writing a career ladder for your organization: Solicit participation from your team.

Page 204 | Highlight

> I enlisted the support of the senior managers and engineers on the team to provide feedback and details.

Page 204 | Highlight

> Look for examples. Second, I got more examples of ladders from friends at other companies to help provide some ideas for the details.

Page 205 | Highlight

> Be detailed. One of the biggest challenges you'll face when writing a good ladder is sketching out the details.

Page 205 | Highlight

> Use both long-form descriptions and summaries. I broke the ladder out into two documents.

Page 205 | Highlight

> The first was a shorthand spreadsheet version that allowed me to see the various level attributes side-by-side and see how they evolved through increasing levels.

Page 205 | Highlight

> The second document was the long-form version.

Page 205 | Highlight

> the long-form ladder reads a bit like a performance review of a person operating well at each level.

Page 205 | Highlight

> Consider how the ladder relates to salary. Your HR department will want to use the career ladder to help set salary expectations. Usually, each level will have a salary band, or a range between a

> minimum and maximum base salary that a person in the level can earn.

Page 205 | Highlight

> Provide many early opportunities for advancement. Some people advise having a lot of levels toward the beginning of the ladder to account for the fact that early-career engineers expect frequent raises and promotions. You may want to be able to promote someone every year for the first two to three years of her career.

Page 206 | Highlight

> Use narrow salary bands for early-career stages. Lots of levels and narrow salary bands mean that you can promote people quickly and justify giving them raises while keeping your pay for all people at a certain level close to the same.

Page 206 | Highlight

> Use wide salary bands when and where you have fewer levels. Wide salary bands and few levels make a clearer distinction between the skills at each level, and should make it easier to tell who is operating at which level.

Page 206 | Highlight

> Consider your breakpoint levels. It is common for companies to have certain levels that they consider “up or out.”

Page 206 | Highlight

> What is the lowest level at which people can sit forever, never getting promoted but also not underperforming? This is your breakpoint level.

Page 207 | Highlight

> I do encourage you to have at least some of your levels be keystone promotions, which are shared and celebrated. I think that the promotion to senior engineer is a big deal, as well as the promotion to staff engineer and, if you have such a role, principal engineer. On the management track, a

> promotion to director is worth celebrating, as is a promotion to VP.

Page 207 | Highlight

> Split management and technical tracks. It's pretty obvious in this day and age that you need separate tracks for management and individual contribution.

Page 207 | Highlight

> Consider making people management skills a mid-career requirement. Encourage everyone to have some sort of management or mentorship experience before they are eligible to be promoted above the level of the track split.

Page 208 | Highlight

> Great senior individual contributors still know how to manage projects and mentor more junior members of their team, so consider making leadership experience (usually via acting as a tech lead) a requirement for promotion to senior individual contributor levels.

Page 208 | Highlight

> Years of experience. No one likes to put artificial barriers onto people, and years of experience can feel like the most artificial barrier.

Page 208 | Highlight

> Don't be afraid to evolve over time. When you write a ladder like this, you're creating a living document that will need to evolve as your company grows.

Page 208 | Highlight

> A good ladder is a critical element to use in hiring, in writing performance reviews, and of course in the promotion process.

Page 210 | Highlight

kindle

> Of course, every function has its own focused needs. Usually someone in engineering needs to oversee critical core systems, and you probably need a few specialists around for things like the core web platform, mobile, or data engineering. I kept these functions in a small infrastructure organization that was not generally assigned to product development.

Page 211 | Highlight

> Now the engineers who have the best product sense, the engineers who are capable of getting features done quickly and efficiently, and the engineers who communicate the best with the other functions will start to emerge as the leaders of the team.

Page 212 | Highlight

> Without any process, your teams will struggle to scale. With the wrong process, they will be slowed down.

Page 212 | Highlight

> Think of process as risk management. As your teams and systems grow, it’s almost impossible for any one person to keep the systems in her head. Because we have a bunch of people coordinating work, we evolve processes around that work coordination in order to make risks obvious.

Page 212 | Highlight

> complicated process should exist only for activities that you expect to be rare, or activities where the risks are not obvious to people.

Page 213 | Highlight

> The first is that you should not put a complicated process on any activity where you want people to move quickly and where you believe the

---

Page 213 | Highlight

> risk for change in that activity is low or that the risks themselves are obvious to the whole team.

Page 213 | Highlight

> The second implication is that you need to be on the lookout for places where there is hidden risk, and draw those hidden risks out into the open. There's a saying in politics that "a good political idea is one that works well in half-baked form," and the same goes for engineering processes. The processes should have value even when they are not followed perfectly, and that value should largely lie in the act of socializing change or risk to the team as a whole.

Page 213 | Highlight

> Code Review Code review is, for better or worse, a modern standard. Once you have a team of a certain size with a certain number of people working on a code base, code review can be a valuable tool for ensuring the stability and long-term quality of that code base.

Page 213 | Highlight

> Code review is largely a socialization exercise, so that multiple team members have seen and are aware of the changed code.

Page 214 | Highlight

> Use a linter for style issues. Engineers can waste absurd amounts of time on questions of style, specifically formatting.

Page 214 | Highlight

> Keep an eye on the review backlog. Some companies implement a limit on how many outstanding review requests a person can have assigned to him, and they block that person from requesting review when he has too many outstanding requests.

Page 214 | Highlight

> The Outage Postmortem I'm not going to talk about the details of incident management, but the "postmortem" process is a critical element of good engineering.

Page 214 | Highlight

> Resist the urge to point fingers and blame.

Page 214 | Highlight

> Look at the circumstances around the incident and understand the context of the events.

Page 215 | Highlight

> Be realistic about which takeaways are important and which are worth dropping. Be careful not to give the impression that people need to solve every problem they identify in the course of the exercise.

Page 215 | Highlight

> Architecture Review I'm going to roll into architecture review all major systems and tools changes that the team may wish to make. The goal of architecture review is to help socialize big changes to the appropriate group, and to make the risks for those changes clear.

Page 215 | Highlight

> Be specific about the kinds of changes that need architecture review.

Page 215 | Highlight

> The value of architecture review is in preparing for the review. Asking for review of big changes or additions to the systems forces people to think about why they want to make these changes. Again, one value of these processes is to help make people aware of risks that they may not have considered.

Page 216 | Highlight

> Choose the review board wisely. You want the review board to include the people who will be most affected by the change, not just a static chosen group of gurus.

Page 216 | Highlight

> The scope of the deciding group is best kept to the people who will be closely impacted by the decision.

Page 217 | Highlight

> you have to be able to manage yourself if you want to be good at managing others.

Page 217 | Highlight

> Great managers are masters of working through conflict. Getting good at working through conflict means getting good at taking your ego out of the conversation. To find a clear view of a complex situation, you must see past your interpretations and the stories you're telling yourself.

Page 217 | Highlight

> Learning to recognize the voice of your ego is one of the benefits of meditation, and when I wrote the first draft of

---

Page 217 | Highlight

> this book it included a series of meditations at each level. For me, having a meditation practice has been essential to developing self-management and self-awareness. Meditation isn't a cure-all, but it can be a useful exercise to practice that awareness of your own reactions, and for that reason I recommend trying it for a while if you are interested. Some of my favorite resources include the podcasts on tarabrach.com and the writings of Pema Chödrön.

Page 218 | Highlight

> Inevitably, when I told my coach about these situations, she would advise me to think about things from the other person's perspective. What are they trying to do? What do they value? What do they want and need? Her advice, always, was to get curious.

Page 218 | Highlight

> So I leave you with that thought. Look for the other side of the story. Think about the other perspectives at play. Investigate your emotional reactions, and observe when those reactions make it hard to see clearly what's going on around you, what needs to be said. Apply that curiosity to people. Apply it to process. Apply it to technology, and strategy, and business. Ask questions, and be willing to have your notions proven wrong.