Notebook - An Elegant Puzzle: Systems of Engineering Management

kindle

Larson, Will

Page 26 | Highlight

> my appreciation for management, and engineering management in particular, has grown, and I've come to view the field as a series of elegant, rewarding, and important puzzles.

Page 26 | Highlight

> Organizational design gets the right people in the right places, empowers them to make decisions, and then holds them accountable for their results.

Page 31 | Highlight

> An organization is a collection of people working toward a shared goal.

Page 31 | Highlight

> truly extraordinary thing is that all organizations work.

Page 31 | Highlight

> When I have a problem that I want to solve quickly and cheaply, I start thinking about process design. A problem I want to solve permanently and we have time to go slow? That's a good time to evolve your culture. However, if process is too weak a force, and culture too slow, then organizational design lives between those two.

Page 32 | Highlight

> Managers should support six to eight engineers

Page 32 | Highlight

> Tech Lead Managers (TLMs). Managers supporting fewer than four engineers tend to function as

Page 32 | Highlight Continued

kindle

> TLMs, taking on a share of design and implementation work.

Page 32 | Highlight

> Coaches. Managers supporting more than eight or nine engineers typically act as coaches and safety nets for problems. They are too busy to actively invest in their team or their team's area of responsibility.

Page 32 | Highlight

> Managers-of-managers should support four to six managers

Page 32 | Highlight

> Ramping up. Managers supporting fewer than four other managers should be in a period of active learning on either the problem domain or on transitioning from supporting engineers to supporting managers.

Page 33 | Highlight

> Coaches. Similar to supporting a large team of engineers, supporting a large team of managers leaves you functioning purely as a problem-solving coach.

Page 33 | Highlight

> On-call rotations want eight engineers

Page 33 | Highlight

> Small teams (fewer than four members) are not teams I've sponsored quite a few teams of one or two people, and each time I've regretted it. To repeat: I have regretted it every single time. An important property of teams is that they abstract the complexities of the individuals that compose them. Teams with fewer than four individuals are a sufficiently leaky abstraction that they function indistinguishably from individuals. To reason about a small team's delivery, you'll have to know about each on-call shift, vacation, and interruption.

Page 34 | Highlight

kindle

> Keep innovation and maintenance together.

Page 34 | Highlight

> will avoid creating a two-tiered class system of innovators and maintainers.

Page 34 | Highlight

> Teams should be six to eight during steady state. To create a new team, grow an existing team to eight to ten, and then bud into two teams of four or five. Never create empty teams. Never leave managers supporting more than eight individuals.

Page 35 | Highlight

> When you talk about growing an organization, the conversation usually leads to hiring.

Page 35 | Highlight

> Teams are slotted into a continuum of four states: A team is falling behind if each week their backlog is longer than it was the week before.

Page 35 | Highlight

> A team is treading water if they're able to get their critical work done, but are not able to start paying down technical debt or begin major new projects.

Page 36 | Highlight

> A team is repaying debt when they're able to start paying down technical debt, and are beginning to benefit from the debt repayment snowball: each piece of debt you repay leads to more time to repay more debt. A team is innovating when their technical debt is sustainably low, morale is high, and the majority of work is satisfying new user needs. Teams want to climb from falling behind to innovating, while entropy drags them backward. Each state requires a different tact.

Page 36 | Highlight

kindle

> adopting the appropriate system solution for their current state.

Page 37 | Highlight

> When the team is falling behind, the system fix is to hire more people until the team moves into treading water.

Page 37 | Highlight

> Sometimes people instead attempt to capture more resources from the existing company, and I' m pretty negative on that. People are not fungible, and generally folks end up in useful places, so I'm skeptical of reassigning existing individuals to drive optimality.

Page 37 | Highlight

> When the team is treading water, the system fix is to consolidate the team's efforts to finish more things, and to reduce concurrent work until they're able to begin repaying debt (e.g., limit work in progress).

Page 37 | Highlight

> When the team is repaying debt, the system fix is to add time.

Page 37 | Highlight

> Innovating is a bit different, because you've nominally reached the end of the continuum, but there is still a system fix! In this case, it's to maintain enough slack in your team's schedule that the team can build quality into their work, operate continuously in innovation, and avoid backtracking.

Page 39 | Highlight

> their limited resources, but resist that indecision-framed-as-fairness: it's not a fair outcome if no one gets anything. For each constraint, prioritize one team at a time. If most teams are falling behind, then hire onto one team until it's staffed enough to tread water, and only then move to the next.

Page 39 | Highlight

kindle

> Adding new individuals to a team disrupts that team's gelling process, so I've found it much easier to have rapid growth periods for any given team, followed by consolidation/gelling periods during which the team gels. The organization will never stop growing, but each team will.

Page 39 | Highlight

> A case against top-down global optimization

Page 40 | Highlight

> I'm skeptical of reallocating individuals to address global priority shifts,

---

Page 40 | Highlight

> Team first Fundamentally, I believe that sustained productivity comes from high-performing teams, and that disassembling a high-performing team leads to a significant loss of productivity, even if the members are fully retained. In this worldview, high-performing teams are sacred, and I' m quite hesitant to disassemble them. Teams take a long time to gel. When a group has been working together for a few years, they understand each other and know how to set each other up for success in a truly remarkable way.

Page 40 | Highlight

> recommends rapidly hiring into teams loaded down by technical debt, not into innovating teams, which avoids incurring re-gelling costs on high-performing teams.

Page 40 | Highlight

> most teams have high fixed costs and relatively small variable costs: moving one person can shift an innovating team back into falling behind, and now neither team is doing particularly well.

Page 41 | Highlight

> My rule of thumb is that it takes eight engineers on a team to support a two-tier on-call rotation,

Page 41 | Highlight

kindle

> The expected time to complete a new task approaches infinity as a team's utilization approaches 100 percent,

---

Page 42 | Highlight

> Shift scope; rotate

Page 42 | Highlight

> I've found it most fruitful to move scope between teams, preserving the teams themselves.

Page 42 | Highlight

> if it's a choice of moving people rapidly or shifting scope rapidly, I've found that the latter is more effective and less disruptive.

Page 42 | Highlight

> The other approach that I've seen work well is to rotate individuals for a fixed period into an area that needs help. The fixed duration allows them to retain their identity and membership in their current team, giving their full focus to helping out, rather than splitting their focus between performing the work and finding membership in the new team. This is also a safe way to measure how much slack the team really has!

Page 43 | Highlight

> Productivity in the age of hypergrowth

Page 43 | Highlight

> More engineers, more problems

Page 44 | Highlight

> Productively integrating large numbers of engineers is hard.

Page 44 | Highlight

kindle

> a scenario in which untrained engineers increasingly outnumber the trained engineers, and each trained engineer is devoting much of their time to training a couple of newer engineers.

Page 46 | Highlight

> For every additional order of magnitude of engineers, you need to design and maintain a new layer of management. For every ~10 engineers, you need an additional team, which requires more coordination.

Page 46 | Highlight

> Each engineer means more commits and deployments per day, creating load on your development tools. Most outages are caused by deployments, so more deployments drive more outages, which in turn require incident management, mitigations, and postmortems. Having more engineers leads to more specialized teams and systems, which require increasingly small on-call rotations so that your on-call engineers have enough system context to debug and resolve production issues. Consequently, relative time invested in on-call goes up.

Page 46 | Highlight

> Only your trained engineers can go on-call.

Page 47 | Highlight

> Systems survive one magnitude of growth

Page 47 | Highlight

> Most system-implemented systems are designed to support one to two orders' magnitude of growth from the current load.

Page 48 | Highlight

> Ways to manage entropy My favorite observation from The Phoenix Project by Gene Kim, Kevin Behr, and George Spafford15 is that you only get value from projects when they finish: to make progress, above all else, you must ensure that some of your projects finish.

Page 48 | Highlight

kindle

> hiring and training are often a team's biggest time investment.

Page 49 | Highlight

> If your engineer is doing more than three interviews a week, it is a useful act of mercy to give them a month off every three or four months.

Page 49 | Highlight

> The strategy here is to funnel interruptions into an increasingly small area, and then automate that area as much as possible. Ask people to file tickets, create chatbots that automate filing tickets, create a service cookbook, and so on.

Page 50 | Highlight

> ownership registry,

Page 50 | Highlight

> Finally, the one thing that I've found at companies with very few interruptions and have observed almost nowhere else: really great, consistently available documentation. It's probably even harder to bootstrap documentation into a non-documenting company than it is to bootstrap unit tests into a non-testing company, but the best solution to frequent interruptions I've seen is a culture of documentation, documentation reading, and a documentation search that actually works.

Page 50 | Highlight

> if you can keep your interfaces generic, then you are able to skip the migration phase of system re-implementation, which tends to be the longest and trickiest phase,

---

Page 51 | Highlight

> a related antipattern is the gatekeeper pattern. Having humans who perform gatekeeping activities creates very odd social dynamics, and is rarely a great use of a human's time. When at all possible, build systems with sufficient isolation that you can allow most actions to go forward. And when they do occasionally fail, make sure that they fail with a limited blast radius.

Page 51 | Highlight

kindle

> treat gatekeeping as a significant implementation bug rather than as a stability feature to be emulated.

Page 51 | Highlight

> managing rapid growth is more along the lines of stacking small wins than identifying silver bullets.

Page 51 | Highlight

> learning to say no in a way that is appropriate to your company's culture.

Page 51 | Highlight

> Where to stash your organizational risk?

Page 51 | Highlight

> organizational debt. This is the organizational sibling of technical debt, and it represents things like biased interview processes and inequitable compensation mechanisms.

Page 52 | Highlight

> identify a few areas to improve, ensure you're making progress on those, and give yourself permission to do the rest poorly.

Page 52 | Highlight

> my organizational philosophy is to stabilize team-by-team and organization-by-organization. Ensuring any given area is well on the path to health before moving my focus.

Page 53 | Highlight

> Succession planning is thinking through how the organization would function without you, documenting those gaps, and starting to fill them in.

Page 53 | Highlight

> The first step in succession planning is to figure out what you do.

Page 53 | Highlight

> Take a look at your calendar and write down your role in meetings.

Page 55 | Highlight

> The first should cover the easiest gaps to close.

Page 55 | Highlight

> The latter will be the riskiest gaps.

Page 61 | Highlight

> The best changes often go unnoticed, moving from one moment of stability to another, with teams and organizations feeling stable at every step. The key tools for leading efficient change are systems thinking, metrics, and vision.

Page 62 | Highlight

> The fundamental observation of systems thinking is that the links between events are often more subtle than they appear.

Page 62 | Highlight

> few events occur in a vacuum.

Page 62 | Highlight

> Changes to stocks are called flows. These can be either inflows or outflows. Training a new manager is an inflow, and a trained manager who departs the company is an outflow.

Page 62 | Highlight

> information link. This indicates that the value of a stock is a factor in the size of a flow.

Page 63 | Highlight

> four measures of developer velocity: Delivery lead time is the time from the creation of code to its use in production. Deployment frequency is how often you deploy code. Change fail rate is how frequently changes fail. Time to restore service is the time spent recovering from defects.

Page 64 | Highlight

> Product management: exploration, selection, validation

Page 65 | Highlight

> Product management is an iterative elimination tournament, with each round consisting of problem discovery, problem selection, and solution validation. Problem discovery is uncovering possible problems to work on, problem selection is filtering those problems down to a viable subset, and solution validation is ensuring that your approach to solving those problems works as cheaply as possible.

Page 66 | Highlight

> Taking the time to evaluate which problem to solve is one of the best predictors I've found of a team's long-term performance. The themes that I've found useful for populating the problem space are: Users' pain. What are the problems that your users experience?

Page 66 | Highlight

> Users' purpose. What motivates your users to engage with your systems?

Page 66 | Highlight

> Benchmark. Look at how your company compares to competitors in the same and similar industries.

Page 66 | Highlight

> you learn the most interesting things by considering both fairly similar and rather different companies. Cohorts. What is hiding behind your clean distributions?

Page 66 | Highlight

> new kinds of users with surprising needs. Competitive advantages. By understanding the areas you're exceptionally strong in,

---

Page 66 | Highlight

> Competitive moats. Moats are a more extreme version of a competitive advantage. Moats represent a sustaining competitive advantage,

---

Page 67 | Highlight

> Compounding leverage. What are the composable blocks you could start building today that would compound into major product or technical leverage10 over time?

Page 67 | Highlight

> Problem selection

Page 67 | Highlight

> Surviving the round. Thinking back to the iterative elimination tournament, what do you need to do to survive the current round?

Page 67 | Highlight

> Surviving the next round. Where do you need to be when the next round in order to avoid getting eliminated then?

Page 68 | Highlight

> Winning rounds. It's important to survive every round, but it's also important to eventually win a

> round!

Page 68 | Highlight

> Consider different time frames. When folks disagree about which problems to work on, I find that the conflict is most frequently rooted in different assumptions about the correct time frame to optimize for.

Page 68 | Highlight

> Industry trends. Where do you think the industry is moving,

---

Page 68 | Highlight

> Return on investment. Personally, I think people often under-prioritize quick, easy wins.

Page 68 | Highlight

> Experiments to learn. What could you learn now that would make problem selection in the future much easier?

Page 68 | Highlight

> Write a customer letter. Write the launch announcement that you would send after finishing the solution.

Page 69 | Highlight

> Identify prior art. How do peers across the industry approach this problem? The

---

Page 69 | Highlight

> Find reference users. Can you find users who are willing to be the first users for the solution?

Page 69 | Highlight

kindle

> Prefer experimentation over analysis. It's far more reliable to get good at cheap validation than it is to get great at consistently picking the right solution.

Page 69 | Highlight

> Find the path more quickly traveled. The most expensive way to validate a solution is to build it in its entirety.

Page 69 | Highlight

> Justify switching costs. What will the costs of switching be for users who move to your solution?

Page 69 | Highlight

> As an aside, I've found that most aspects of running a successful technology migration12 overlap with good solution validation! This is a very general skill that will repay many times over the time you invest in learning it. Putting these three elements in place today—exploration, selection, and validation—won't make you an exceptional product manager overnight, but they will provide a solid starting place to develop those skills and perspective for the next time you find yourself donning the product manager hat.

Page 70 | Highlight

> agreeing on strategy and vision has been the most effective approach that I've found to alignment at scale.

Page 70 | Highlight

> Strategies are grounded documents which explain the trade-offs and actions that will be taken to address a specific challenge. Visions are aspirational documents that enable individuals who don' t work closely together to make decisions that fit together cleanly.

Page 71 | Highlight

> A strategy recommends specific actions that address a given challenge's constraints. A structure that I've found extremely effective13 is described in Good Strategy/Bad Strategy by Richard

Page 71 | Highlight Continued

kindle

> Rumelt,14 and has three sections: diagnosis, policies, and actions. The diagnosis is a theory describing the challenge at hand.

Page 72 | Highlight

> identify policies that you will apply to address the challenge. These describe the general approach that you'll take, and are often trade-offs between two competing goals.

Page 72 | Highlight

> allow short-term performance to dip in order to invest into long-term performance,

---

Page 72 | Highlight

> When you apply your guiding policies to your diagnosis, you get your actions. Folks

---

Page 72 | Highlight

> Because strategies are specific to a given problem, it's okay—and even encouraged—to write quite a few of them.

Page 73 | Highlight

> If strategies describe the harsh trade-offs necessary to overcome a particular challenge, then visions describe a future in which those trade-offs are no longer mutually exclusive. An effective vision helps folks think beyond the constraints of their local maxima, and lightly aligns progress without requiring tight centralized coordination.

Page 73 | Highlight

> Visions should be detailed, but the details are used to illustrate the dream vividly, not to prescriptively constrain its possibilities.

Page 73 | Highlight

> Vision statement: A one- or two-sentence aspirational statement to summarize the rest of the document.

Page 73 | Highlight

> repeat at each meeting, planning period, and strategy review.

Page 73 | Highlight

> Value proposition: How will you be valuable to your users and to your company?

Page 73 | Highlight

> Capabilities: What capabilities will the product, platform, or team need in order to deliver on your value proposition?

Page 74 | Highlight

> Solved constraints: What are the constraints that you're limited by today, but that in the future you'll no longer be constrained by?

Page 74 | Highlight

> Future constraints: What are the constraints that you expect to encounter in this wonderful future?

Page 74 | Highlight

> Reference materials: Link all the existing plans, metrics, updates, references, and documents into an appendix for those who want to understand more of the thinking that went into the vision.

Page 74 | Highlight

> Narrative: Once you've written the previous sections, the last step of writing a compelling vision is to synthesize all those details into a short—maybe one-page—narrative that serves as an easy-to-digest summary.

Page 74 | Highlight

> You'll know a vision is succeeding when people reference the document to make their own decisions,

---

Page 74 | Highlight

> Test the document! This is a core leadership tool, and your first version will almost certainly be bad. Write a draft, sit down with a few different folks to get their perspectives, then iterate.

Page 75 | Highlight

> Refresh periodically. Take some time every year to refresh the vision,

---

Page 75 | Highlight

> Use present tense. This makes the writing impactful and concise, and conveys a sense of confidence about the future. Write simply. Often, visions are saturated with buzzwords, which turns readers off.

Page 75 | Highlight

> one vision for every complete distinct area, but no more.

Page 75 | Highlight

> when top-level planning shifts from discussing specific projects to talking about goals.

Page 75 | Highlight

> goals decouple the "what" from the "how,"

Page 76 | Highlight

> You'll know a goal is just a number when you read it and aren't sure if it's ambitious or whether it matters. Good goals are a composition of four specific kinds of numbers: A target states where you want to reach. A baseline identifies where you are today. A trend describes the current velocity. A

> time frame sets bounds for the change.

Page 76 | Highlight

> “In Q3, we will reduce time to render our frontpage from 600ms (p95) to 300ms (p95). In Q2, render time increased from 500ms to 600ms.” The two tests of an effective goal are whether someone who doesn’t know much about an area can get a feel for a goal’s degree of difficulty, and whether afterward they can evaluate if it was successfully achieved.

Page 76 | Highlight

> There are two particularly interesting kinds of goals: investments and baselines. Investments describe a future state that you want to reach, and baselines describe aspects of the present that you want to preserve.

Page 77 | Highlight

> I’ve found that you should specify as few investment goals as possible, maybe three, and that those should be the focus of planning discussions.

Page 78 | Highlight

> Infrastructure cost is a great example of a baseline metric.20

Page 78 | Highlight

> “Maintain infrastructure costs at their current percentage of net revenue of 30 percent.”

Page 78 | Highlight

> Explore: The first step is to get data in an explorable format in your data warehouse, an SQL database, or even an Excel spreadsheet.

Page 79 | Highlight

kindle

> Dive: Once you know the three or four major contributors, go deep on understanding those areas and the levers that drive them.

Page 79 | Highlight

> Attribute: For most company-level metrics (cost, latency, development velocity, etc.), the first step of diving will uncover one team who are nominally accountable for the metric's performance, but they are typically a cloak.

Page 79 | Highlight

> Contextualize: Armed with the attribution data, start to build context around each team's performance.

Page 79 | Highlight

> Nudge: Once you've built context around the data so that folks can interpret it, the next step is to start nudging them to action!

Page 80 | Highlight

> What I've found effective is to send push notifications, typically email, to teams whose metric has changed recently, both in terms of absolute change and in terms of their benchmarked performance against their cohort.

Page 80 | Highlight

> Baseline: In the best case, you'll be able to drive the organizational impact you need with contextualized nudges, but in some cases that isn't quite enough.

Page 80 | Highlight

> Review: The final phase, which hopefully you won't need to reach, is running a monthly or quarterly review that looks at each team's performance, and reaching out to teams to advocate for prioritization

Page 81 | Highlight

> Migrations: the sole scalable fix to tech debt

Page 81 | Highlight

> Migrations are both essential and frustratingly frequent as your codebase ages and your business grows: most tools and processes only support about one order of magnitude of growth22 before becoming ineffective,

---

Page 82 | Highlight

> Migrations matter because they are usually the only available avenue to make meaningful progress on technical debt.

Page 82 | Highlight

> there is very little low-hanging fruit to reduce technical debt, and most remaining options require many teams working together to implement them. The result: migrations.

Page 82 | Highlight

> the awkward territory of reduced immediate contribution today in exchange for more capacity tomorrow.

Page 82 | Highlight

> Migrations are the only mechanism to effectively manage technical debt as your company and code grow.

Page 83 | Highlight

> The good news is that while migrations are hard, there is a pretty standard playbook that works remarkably well: de-risk, enable, then finish.

Page 83 | Highlight

kindle

> The first phase of a migration is de-risking it, and to do so as quickly and cheaply as possible.

Page 83 | Highlight

> Don’t start with the easiest migrations, which can lead to a false sense of security.

Page 83 | Highlight

> better to slow down and build tooling to programmatically migrate the easy 90 percent.

Page 83 | Highlight

> figure out the self-service tooling and documentation that you can provide to allow teams to make the necessary changes without getting stuck. The best migration tools are incremental and reversible:

Page 84 | Highlight

> Spending an extra two days intentionally making your documentation clean and your tools intuitive can save years in large migrations. Do it!

Page 84 | Highlight

> Finish The last phase of a migration is deprecating the legacy system that you’ve replaced. This requires getting to 100 percent adoption, and that can be quite challenging. Start by stopping the bleeding, which is ensuring that all newly written code uses the new approach.

Page 84 | Highlight

> Okay, now you should start generating tracking tickets, and set in place a mechanism which pushes migration status to teams that need to migrate and to the general management structure.

Page 84 | Highlight

> Your tool now is: finish it yourself. It’s not necessarily fun, but getting to 100 percent is going to

Page 84 | Highlight Continued

> require the team leading the migration to dig into the nooks and crannies themselves.

Page 85 | Highlight

> two managerial skills that have a disproportionate impact on your organization's success: making technical migrations cheap, and running clean reorganizations.

Page 85 | Highlight

> My approach for planning organization change: Validate that organizational change is the right tool. Project head count a year out. Set target ratio of management to individual contributors. Identify logical teams and groups of teams. Plan staffing for the teams and groups. Commit to moving forward. Roll out the change.

Page 86 | Highlight

> There is only one worst kind of reorg: the one you do because you're avoiding a people management issue.

Page 87 | Highlight

> Is the problem structural?

Page 87 | Highlight

> Are you reorganizing to work around a broken relationship?

Page 87 | Highlight

> Does the problem already exist?

Page 87 | Highlight

> Are the conditions temporary?

Page 87 | Highlight

kindle

> 1. An optimistic number based on what's barely possible. 2. A number based on the "natural size" of your organization, if every team and role was fully staffed. 3. A realistic number based on historical hiring rates. Then merge those into a single number.

Page 88 | Highlight

> If engineering managers are expected to do hands-on technical work, then their teams should likely be three to five engineers

---

Page 88 | Highlight

> Otherwise, targeting five to eight engineers,

Page 88 | Highlight

> probably in the six-to-eight range.

Page 89 | Highlight

> Can you write a crisp mission statement for each team? Would you personally be excited to be a member of each of the teams, as well as to be the manager of each of those teams? Put teams that work together (especially poorly) as close together as possible.

Page 89 | Highlight

> most poor working relationships are the by-product of information gaps,

---

Page 89 | Highlight

> Can you define clear interfaces for each team? Can you list the areas of ownership for each team? Have you created a gap-less map of ownership, such that each responsibility is owned by a team?

Page 89 | Highlight

kindle

> Are there compelling candidate pitches for each of those teams? As always, are you over-optimizing on individuals versus establishing a sensible structure?

Page 90 | Highlight

> four sources of candidates to staff them: Team members who are ready to fill the roles now. Team members who can grow into the roles in the time frame. Internal transfers from within your company. External hires who already have the skills. That is probably an ordered list of how you should try to fill the key roles.

Page 90 | Highlight

> Accidentally missing someone is the cardinal sin of reorganization.

Page 90 | Highlight

> Are the changes meaningful net positive? Will the new structure last at least six months? What problems did you discover during design? What will trigger the reorg after this one? Who is going to be impacted most?

Page 91 | Highlight

> Explanation of reasoning driving the reorganization. Documentation of how each person and team will be impacted. Availability and empathy to help bleed off frustration from impacted individuals.

Page 91 | Highlight

> Discuss with heavily impacted individuals in private first. Ensure that managers and other key individuals are prepared to explain the reasoning behind the changes. Send an email out documenting the changes. Be available for discussion. If necessary, hold an organization all-hands, but probably try not to. People don’t process well in large groups, and the best discussions take place in small rooms. Double down on doing skip-level one-on-ones.

Page 92 | Highlight

kindle

> Thanks to a great deal of feedback and reflection, I've gotten more deliberate at identifying where to engage and where to hang back, a process that I call identifying your controls.

Page 92 | Highlight

> Metrics26 align on outcomes while leaving flexibility around how the outcomes are achieved. Visions27 ensure that you agree on long-term direction while preserving short-term flexibility. Strategies28 confirm you have a shared understanding of the current constraints and how to address them. Organization design allows you to coordinate the evolution of a wider organization within the context of sub-organizations. Head count and transfers are the ultimate form of prioritization, and a good forum for validating how organizational priorities align across individual teams. Roadmaps align on problem selection and solution validation. Performance reviews coordinate culture and recognition.

Page 93 | Highlight

> I'll do it. Stuff that I will personally be responsible for doing.

Page 93 | Highlight

> Best used sparingly. Preview. I'd like to be involved early and often.

Page 93 | Highlight

> Review. I'd like to weigh in before it gets published or fully rolled out, but we're pretty aligned on the topic. Notes. Projects I'd like to follow but don't have much I can add to.

Page 93 | Highlight

> No surprises. The work that we're currently aligned on but requires updates to keep my mental model intact.

Page 93 | Highlight

> my effectiveness is evaluated based on my ability to stay on top of new problems. Let me know. We're well aligned on this, since my colleagues have done it before and done it well.

Page 93 | Highlight

> established the interface between you and the folks you support.

Page 95 | Highlight

> A prototypical head of engineering will be skilled at organizational design, process design, business strategy, recruiting, mentoring, coaching, public speaking, and written communication. They'll also have a broad personal network and a broad foundation from product engineering to infrastructure engineering.

Page 96 | Highlight

> Managers tend to have a strong sense of the business's needs,

---

Page 97 | Highlight

> Answer the question you want to be asked. If someone asks a very difficult or challenging question, reframe it into one that you're comfortable answering.

Page 97 | Highlight

> Stay positive. Negative stories can be very compelling. They are quite risky, too! As an interviewee, find a positive framing and stick to it.

Page 97 | Highlight

> Speak in threes. Narrow your message down to three concise points, make them your refrain, and continue to refer back to your three speaking points.

Page 98 | Highlight

> One of the trickiest, and most common, leadership scenarios is leading without authority,

---

Page 98 | Highlight

> Model. Start measuring your team's health (maybe using short, monthly surveys) and your team's

Page 98 | Highlight Continued

kindle

> throughput (do some lightweight form of task sizing, even if you just do it informally with a senior engineer on the team or with yourself), which will allow you to establish the baseline before your change.

Page 99 | Highlight

> Document. After you've discovered an effective approach, document the problem you set out to solve, the learning process you went through, and the details of how another team would adopt the practice for themselves.

Page 99 | Highlight

> Share. The final step is to share your documented approach, along with your experience doing the rollout, in a short email.

Page 99 | Highlight

> Mandates assume: It's better to adopt a good-enough approach quickly.

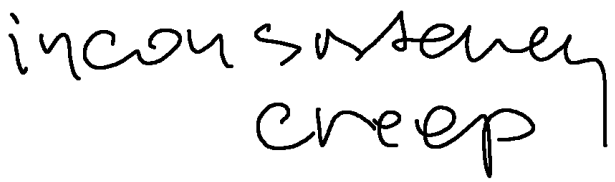
Page 100 | Highlight

> Model, Document, Share assumes: It's better to adopt a great approach slowly.

Page 100 | Highlight

> As organizations grow, there is a subtle slide into inconsistency, which is often one of the most challenging aspects of evolving from a small team into a much larger one.

Page 101 | Note



Page 101 | Highlight

> manage inconsistency creep.

Page 101 | Highlight

> when the problem becomes truly acute, folks eventually reach for the same tool: adding a centralized, accountable group. The two most common flavors of this I've seen are "product reviews" to standardize product decisions and the "architecture group" to encourage consistent technical design.

Page 101 | Highlight

> Some of these groups take on an authoritative bent, becoming rigid gatekeepers, and others become more advisory, with a focus on educating folks toward consistency.

Page 101 | Highlight

> a few words on the framing that I use for reasoning about when to create a new centralized authority.

Page 101 | Highlight

> A positive freedom is the freedom to do something, for example the freedom to pick a programming language you prefer. A negative freedom is the freedom from things happening to you, for example the freedom not to be obligated to support additional programming languages, even if others would greatly prefer them.

Page 102 | Highlight

> Influence. How do you expect this group to influence results? Will they be an authoritative group that makes binding decisions? Will you rely on the natural authority of the members you select?

Page 102 | Highlight

> Interface. How will other teams interact with this team? Will they submit tickets, send emails, attend a weekly review session?

Page 102 | Highlight

kindle

> Size. How large should the group be? If it's six or fewer individuals, it's possible for them to gel into a true team, one whose members know each other well, work together closely, and shift a significant portion of their individual identities into the team.

Page 102 | Highlight

> Time commitment. How much time will members spend working in this group? Will this be their top priority, or will they still primarily be working on other projects?

Page 102 | Highlight

> my sense is that you want time commitment to be higher for areas where folks are directly impacted by the consequences of their decisions, and to be lower for scenarios with weaker feedback loops. Identity. Should members view their role in the group as their primary identity?

Page 103 | Highlight

> Selection process. How will you select members? I've found the best method to be a structured selection process,35 in which you identify the requirements to be a member and the skills that you believe will be valuable, and then allow folks to apply. Membership in these groups often becomes an important signal of organizational status, which makes having a consistent process for selecting membership especially important. Length of term. How long will members serve? Are these permanent assignments, or are they fixed terms for, say, six months?

Page 103 | Highlight

> Representation. How representative will this group be? Will you explicitly select folks based on their teams, tenure, or seniority, or will you allow clusters?

Page 103 | Highlight

> four ways these groups consistently fail.

Page 104 | Highlight

> Domineering groups significantly reduce individuals' negative and positive freedoms, and become

Page 104 | Highlight Continued

> churn factories for members.

Page 104 | Highlight

> Bottlenecked groups tend to be very helpful, but are trying to do more than they're actually able to do.

Page 104 | Highlight

> Status-oriented groups place more emphasis on being a member of the group than on the group' s nominal purpose.

Page 104 | Highlight

> Inert groups just don't do much of anything. Typically, these are groups whose members have not gelled or are too busy. On the plus

---

Page 105 | Highlight

> Communication is company-specific. Every company has different communication styles and patterns.

Page 105 | Highlight

> Start with the conclusion. Particularly in written communication, folks skim until they get bored and then stop reading.

Page 105 | Highlight

> Frame why the topic matters. Typically, you'll be presenting on an area that you're intimately familiar with, and it's probably very obvious to you why the work matters.

Page 105 | Highlight

> Everyone loves a narrative. Another aspect of framing the topic is providing a narrative of where

Page 105 | Highlight Continued

> things are, how you got here, and where you're going now.

Page 105 | Highlight

> Prepare for detours. Many forums will allow you to lead your presentation according to plan, but that is an unreliable prediction when presenting to senior leadership.

Page 107 | Highlight

> Answer directly. Senior leaders tend to be indirectly responsible for wide areas, and frequently pierce into areas to debug problems.

Page 107 | Highlight

> Deep in the data. You should be deep enough in your data that you can use it to answer unexpected questions.

Page 107 | Highlight

> Derive actions from principles. One of your aims is to provide a mental model of how you view the topic, allowing folks to get familiar with how you make decisions.

Page 107 | Highlight

> Discuss the details. Some executives test presenters by diving into the details, trying to uncover an area the presenter is uncomfortable speaking on.

Page 107 | Highlight

> Prepare a lot; practice a little. If you're presenting to your entire company, practicing your presentation is time well spent. Leadership presentations tend to quickly detour, so practice isn't quite as useful.

Page 107 | Highlight

kindle

> Make a clear ask.

Page 108 | Highlight

> Tie topic to business value. One or two sentences to answer the question “Why should anyone care?” Establish historical narrative. Two to four sentences to help folks understand how things are going, how we got here, and what the next planned step is. Explicit ask. What are you looking for from the audience? One or two sentences. Data-driven diagnosis. Along the lines of a strategy’s diagnosis phase,39 explain the current constraints and context, primarily through data.

Page 108 | Highlight

> Decision-making principles. Explain the principles that you’re applying against the diagnosis, articulating the mental model you are using to make decisions. What’s next and when it’ll be done.

Page 108 | Highlight

> Return to explicit ask.

Page 109 | Highlight

> Quarterly time retrospective. Every quarter, I spend a few hours categorizing my calendar from the past three months to figure out how I’ve invested my time.

Page 109 | Highlight

> Prioritize long-term success over short-term quality.

Page 111 | Highlight

> Finish small, leveraged things.

Page 111 | Highlight

> Stop doing things.

Page 111 | Highlight

> it’s fine to drop things, but it’s quite bad to silently drop them. Size backward, not forward. A good example of this is scheduling skip-levels.

Page 111 | Highlight

> two hours per week.

Page 111 | Highlight

> specify the number of hours you’re able to dedicate to the activity, perhaps two per week, and perform as many skip-levels as possible within that amount of time.

Page 111 | Highlight

> Delegate working “in the system.”

Page 111 | Highlight

> Trust the system you build. Once you’ve built the system, at some point you have to learn to trust it.

Page 111 | Highlight

> Handling exceptions can easily consume all of your energy, and either delegating them or designing them out of the system is essential to scaling your time. Decouple participation from productivity.

Page 112 | Highlight

> don’t fall into the trap of assuming that attendance is valuable. Hire until you are slightly ahead of growth.

Page 112 | Highlight

> hiring capable folks, and hiring them before you get overwhelmed.

Page 112 | Highlight

> Calendar blocking.

Page 112 | Highlight

> add three or four two-hour blocks scattered across your week to support more focused work.

Page 112 | Highlight

> Getting administrative support.

Page 112 | Highlight

> I’ve always preferred learning in private.

Page 113 | Highlight

> building a community of learning with your peers.

Page 113 | Highlight

> Be a facilitator, not a lecturer. Folks want to learn from each other more than they want to learn from a single presenter. Step back and facilitate. Brief presentations, long discussions.

Page 113 | Highlight

> Small breakout groups.

Page 113 | Highlight

> Bring learnings to the full group.

Page 113 | Highlight

> Choose topics that people already know about.

Page 116 | Highlight

> Encourage tenured folks to attend. For many learning communities, you'll find that the most senior or most tenured folks opt out to focus on other work.

Page 116 | Highlight

> Optional pre-reads.

Page 116 | Highlight

> Checking in. Depending on the size of the group, it can be powerful to start by checking in with each other, having each person give a 20- or 30-second self-introduction. The format we've been using lately is your name, your team, and one sentence about what's on your mind.

Page 119 | Highlight

> You unravel most puzzles knowing they're solvable.

Page 119 | Highlight

> Work the policy, not the exceptions

Page 119 | Highlight

> consistency is a precondition of fairness.

Page 119 | Highlight

> "Work the policy, not the exceptions."

Page 120 | Highlight

kindle

> Good policy is opinionated Every policy you write is a small strategy,1 built by identifying your goals and the constraints that bring actions into alignment with those goals.

Page 120 | Highlight

> Make every office a first-tier office; there are no second-tier offices.

Page 120 | Highlight

> must own multiple critical projects,

Page 120 | Highlight

> Ensure that remote engineers remain an essential, well-supported cohort of the company.

Page 120 | Highlight

> Teams are staffed in, at most, one office.

Page 120 | Highlight

> Employees in an office must be members of a team in that office. Remote employees may work on any team. Employees within a 60 minute commute of an office must work from that office.

Page 120 | Highlight

> If you find yourself writing constraints that don’t actually constrain choice, it’s useful to check if you’re dancing around an unstated goal that’s limiting your options.

Page 121 | Highlight

> I generally don’t recommend writing policies that do little to constrain behavior.

Page 121 | Highlight

kindle

> Exception debt

Page 121 | Highlight

> Accepting reduced opportunity space. Good constraints make trade-offs that deliberately narrow your opportunity space.

Page 121 | Highlight

> Locally suboptimal. Satisfying global constraints inevitably leads to local inefficiency, sometimes forcing some teams to deal with deeply challenging circumstances in order to support a broader goal that they may experience little benefit from.

Page 122 | Highlight

> Policy success is directly dependent on how we handle requests for exception. Granting exceptions undermines people's sense of fairness, and sets a precedent that undermines future policy.

Page 122 | Highlight

> Organizations spending significant time on exceptions are experiencing exception debt. The escape is to stop working the exceptions, and instead work the policy.

Page 122 | Highlight

> Work the policy

Page 122 | Highlight

> collect every escalation as a test case for reconsidering your constraints.

Page 122 | Highlight

> escalations will only be used as inputs for updated policy, not handled in a one-off fashion.

Page 122 | Highlight

> When you roll out a policy, it’s quite helpful to declare a future time when you’ll refresh it,

Page 122 | Highlight

> At a sufficiently high rate of change, policy is indistinguishable from exception.

Page 123 | Highlight

> Commit to refreshing the policy in a month, and batch all exceptions requests until then.

Page 123 | Highlight

> Saying no

Page 123 | Highlight

> This no is explaining your team’s constraints to folks outside the team, and it’s one of the most important activities you undertake as an engineering leader.

Page 124 | Highlight

> You don’t have to switch to using a kanban system, although I’ve found it very effective for debugging team performance, you just have to populate the board once to expose your current constraints.

Page 124 | Highlight

> you have to translate the problem into something resembling data.

Page 125 | Highlight

> There are two ways to add capacity: move existing resources to the team (away from what they’re currently doing) or create new resources (typically through hiring).

Page 125 | Highlight

> a reality-based approach

Page 125 | Highlight

> expressing your priorities convincingly can be a difficult, daunting task.

Page 127 | Highlight

> When I started managing, my leadership philosophy was simple: The Golden Rule6 makes a lot of sense. Give everyone an explicit area of ownership that they are responsible for. Reward and status should derive from finishing high-quality work. Lead from the front, and never ask anyone to do something you wouldn't.

Page 127 | Highlight

> I believe that management, at its core, is an ethical profession.

Page 127 | Highlight

> how we treat a member of the team who is not succeeding.

Page 127 | Highlight

> compensation philosophy.

Page 127 | Highlight

> We have such a huge impact on the people we work with—

Page 127 | Highlight

> Strong relationships > any problem I believe that almost every internal problem can be traced back to a missing or poor relationship, and that with great relationships it is possible to come together and solve almost anything. Technical disagreements become learning opportunities for everyone. Setbacks are now a shared experience that offers the opportunity to gel together as a team.

Page 128 | Highlight

> debugging problems from the relationship angle,

Page 128 | Highlight

> “With the right people, any process works, and with the wrong people, no process works.”

Page 128 | Highlight

> Do the hard thing now

Page 128 | Highlight

> Instead of avoiding the hardest parts, double down on them. If you have a poor relationship with your manager or a member of your team, spend even more time with them.

Page 129 | Highlight

> do the right thing for the company, the right thing for the team, and the right thing for yourself, in that order.

Page 129 | Highlight

> all thinking should start from a company perspective,

Page 129 | Highlight

> For example, you’re really excited about trying out a new programming language in a project, but you should also make sure that you’ve considered the additional maintenance cost for the rest of the company.

Page 129 | Highlight

> make sure that your choices are being made on behalf of your team, not on your own behalf.

Page 129 | Highlight

kindle

> Last in the list is yourself, but while I do believe that you should generally put yourself last, it's also a reminder to "pay yourself."

Page 129 | Highlight

> interviewing, performance management, promotions, raises.

Page 130 | Highlight

> programming interviews7

Page 130 | Highlight

> Long bones have growth plates at their ends, which is where the growth happens, and the middle doesn't grow.

Page 131 | Highlight

> These new problems aren't necessarily novel (most problems are people problems),

---

Page 131 | Highlight

> This means that you can't expect to succeed by iterating on the status quo.

Page 131 | Highlight

> execution is the primary currency in the growth plates. That's because you typically have a surplus of fairly obvious ideas to try, and there is constrained bandwidth for evaluating those ideas.

Page 131 | Highlight

> What folks in the growth plates need is help reducing and executing the existing backlog of ideas, not adding more ideas that must be evaluated.

Page 131 | Highlight

kindle

> Away from the growth plates, you are mostly working on problems with known solutions. Known solutions are amenable to iterative improvement, so it would make sense for execution to be highly valued, but I find that, in practice, ideas—especially ideas that are new within your company—are most highly prized.

Page 132 | Highlight

> we make solid executors responsible for slower-growth areas—we need the innovators in the highest-growth ones—but the opposite tends to work better. As a manager, this is the environment for you to do the basics very, very well. Spend time building rich relationships, gelling your team, working with them on career development. Build up so that when innovation or external change pushes you off your local maxima, you and the team are ready and rested.

Page 132 | Highlight

> The message I'd end with is a simple one: be thoughtful about carrying your values with you from one context into another. Leadership is matching appropriate action to your current context, and it's pretty uncommon that any two situations will flourish from the same behaviors. If you're working in the growth plates—or outside of them—for the first time, treat it like a brand-new role. It is!

Page 132 | Highlight

> As a new manager, I found it useful to start each performance review season by rereading! Which also means it's an excellent time to reread Camille Fournier's "How Do Individual Contributors Get Stuck?"

Page 133 | Highlight

> Only manage down.

Page 133 | Highlight

> Only manage up.

Page 133 | Highlight

kindle

> Never manage up. Your team's success and recognition depend significantly on your relationship with your management chain.

Page 133 | Highlight

> Optimize locally. Picking technologies that the company can't support, or building a product that puts you in competition with another team. Assume that hiring never solves any problems.

Page 133 | Highlight

> Don't spend time building relationships.

Page 133 | Highlight

> Define their role too narrowly.

Page 133 | Highlight

> Forget that their manager is a human being.

Page 133 | Highlight

> To have a good relationship with your manager, you have to give them room to make mistakes.

Page 133 | Highlight

> Do what worked at their previous company.

Page 133 | Highlight

> it's important to pause to listen and foster awareness before you start "fixing" everything.

Page 134 | Highlight

> Spend too much time building relationships.

Page 134 | Highlight

kindle

> Assume that more hiring can solve every problem.

Page 134 | Highlight

> adding too many people can dilute your culture, and lead to people with unclear roles and responsibilities. Abscond rather than delegate.

Page 134 | Highlight

> Become disconnected from ground truth.

Page 134 | Highlight

> Mistake team size for impact. Managing a larger team is not a better job, it's a different job. It also won't make you important or make you happier. It's hard to unlearn a fixation on team size, but if you can, it'll change your career for the better. Mistake title for impact.

Page 134 | Highlight

> Confuse authority with truth. Authority lets you get away with weak arguments and poor justifications,

---

Page 135 | Highlight

> Don't trust the team enough to delegate.

Page 135 | Highlight

> Let other people manage their time.

Page 135 | Highlight

> Only see the problems.

Page 136 | Highlight

kindle

> Success is filling in information gaps, not reciting a mantra. At some regular point, maybe quarterly, write up a self-reflection that covers each of those aspects. (I've been experimenting with a "career narrative" format that is essentially a stack of quarterly self-reflections.) Share that with your manager, and maybe with your peers too!

Page 138 | Highlight

> Broadly, there are three types of engineering management jobs: Manager: you manage a team directly, Director: you manage a team of managers, VP: you manage an organization.

Page 138 | Highlight

> As managers looking to grow ourselves, we should really be pursuing scope: not enumerating people but taking responsibility for the success of increasingly important and complex facets of the organization and company. This is where advancing your career can veer away from a zero-sum competition to have the largest team and evolve into a virtuous cycle of empowering the organization and taking on more responsibility. There is a lot less competition for hard work.

Page 139 | Highlight

> If you've been focused on growing the size of your team as the gateway to career growth, call bullshit on all that,13 and look for a gap in your organization or company that you can try to fill. You'll be a lot happier.

Page 139 | Highlight

> Scarce feedback, vague direction

Page 140 | Highlight

> As a functional leader, you'll be expected to set your own direction with little direction from others.

Page 141 | Highlight

> This first phase is discovery without judgement. You should take ideas from everywhere and

Page 141 | Highlight Continued

> generate a pile of ideas that folks are pursuing, even if you think they're terrible.

Page 141 | Highlight

> Close out, solve, or delegate

Page 144 | Highlight

> For every problem that comes your way—

Page 144 | Highlight

> you must pick one of three options: Close out. Close it out in a way such that this specific ask is entirely resolved.

Page 144 | Highlight

> Solve. Design a solution such that you won't need to spend time on this particular kind of ask again in the next six months.

Page 144 | Highlight

> Delegate. Ideally, this is to redirect the ask to someone who is responsible for that kind of work, but sometimes it is a one-off.

Page 147 | Highlight

> The basis: an inclusive organization is one in which individuals have access to opportunity and membership. Opportunity is having access to professional success and development. Membership is participating as a version of themselves that they feel comfortable with.

Page 148 | Highlight

> When I think about having access to opportunity, I think about ensuring that folks can go home most days feeling fulfilled by challenge and growth. The most effective way to provide opportunity

kindle

> to the members of your organization is through the structured application of good process.

Page 148 | Highlight

> The key question is whether you'll continue to respect your processes when it's inconvenient to do so.

Page 148 | Highlight

> Rubrics everywhere. Every important people decision should have a rubric around how folks are evaluated.

Page 148 | Highlight

> Selecting project leaders.1 Having a structured approach to selecting project leads allows you to learn from previous selections, and to ensure that you're not concentrating opportunity on a small set of individuals. Explicit budgets.

Page 148 | Highlight

> Instead of saying that you'll pay for teams to attend a reasonable number of conferences per year, specify a fixed number.

Page 148 | Highlight

> Nudge involvement. Many people are uncomfortable applying to opportunities, using education budgets, asking for mentorship, etc. It's very effective to reach out to those folks directly and recommend they apply.

Page 149 | Highlight

> Education programs. Create ongoing training and learning programs that are available for everyone,

Page 149 | Highlight

> Retention is the most important measure of availability of opportunity, although it’s also a very lagging indicator.

Page 149 | Highlight

> Usage rate is how often folks get picked in project selection.

Page 149 | Highlight

> Level distribution is useful, in particular comparing cohorts of folks with different backgrounds.

Page 149 | Highlight

> Time at level is how long team members wait between promotions.

Page 150 | Highlight

> Recurring weekly events allow coworkers to interact socially. These are held during working hours, are open to folks from many different teams to attend, and are optional.

Page 150 | Highlight

> a paper-reading group.3 Employee Resource Groups (ERGs)

Page 150 | Highlight

> Team offsites once a quarter or so are a good chance to pause, reflect, and work together differently.

Page 150 | Highlight

> Coffee chats,

Page 150 | Highlight

> Team lunches

Page 150 | Highlight

> Held once a week or so, they can become a pleasant ritual.

Page 151 | Highlight

> Retention is once again the golden measure, and once again a long-trailing indicator. Referral rate by cohort

---

Page 151 | Highlight

> Attendance rates for recurring events and team lunches

Page 151 | Highlight

> The quantity and completion rate of coffee chats

Page 151 | Highlight

> Many activities and events don't work well for everyone—meals can be difficult for individuals with complex dietary restrictions, physical activities make some uncomfortable, activities after working hours can exclude parents—

Page 152 | Highlight

> I've come to believe that having a wide cohort of coworkers who lead critical projects is one of the most important signifiers of good organizational health.

Page 152 | Highlight

> To increase the number of folks leading this kind of project, I've iterated into a structured process that has worked quite well: Define the project's scope and goals in a short document.

Page 152 | Highlight

> Announce the project to a public email list, at an all-hands, over Slack, or however your company does persistent communication;

Page 153 | Highlight

> Allow folks to apply in private. Some individuals will be uncomfortable applying in public. Make sure that applicants don't see who else has applied.

Page 153 | Highlight

> Give at least three working days for people to apply.

Page 153 | Highlight

> Nudge folks to apply who you think would be good candidates but who might not self-select.

Page 153 | Highlight

> Select a project leader based on the selection criteria you specified.

Page 153 | Highlight

> Sponsor the project leader by finding someone who has successfully completed a similar project to be their advisor.

Page 153 | Highlight

> Notify other individuals who applied that they were not selected. It's extremely helpful if you provide them feedback on why you didn't select them. Sometimes it's because they've already done something great and you want to create room for another person to learn, and that's a totally reasonable thing to tell them! Kick off the project,

---

Page 153 | Highlight

> Record the project, who was selected, and who the sponsor is into a public spreadsheet.

Page 154 | Highlight

> These dynamics can lead to teams whose camaraderie is at best a qualified non-aggression pact, and in which collaboration is infrequent.

Page 154 | Highlight

> strange tragedy that we hold ourselves accountable for building healthy, functional teams, and yet are so rarely on them ourselves.

Page 154 | Highlight

> individuals willing to disappoint the teams they managed in order to help their peers succeed.

Page 155 | Highlight

> they balanced outcomes from a broad perspective that included their peers.

Page 155 | Highlight

> The ingredients necessary for such a team are: Awareness of each other’s work.

Page 155 | Highlight

> Evolution from character to person. When we don’t know someone well, we tend to project intentions onto them, casting them as a character in a play they themselves are unaware exists.

Page 155 | Highlight

> Spending time together

Page 155 | Highlight

> strangers into people. Refereeing defection.

Page 155 | Highlight

> Avoiding zero-sum culture. Some companies foster zero-sum cultures, in which perceived success depends on capturing scarce, metered resources, like head count.

Page 156 | Highlight

> Making it explicit.

Page 156 | Highlight

> you still have to explicitly discuss the idea of shifting folks' identities away from the team

---

Page 156 | Highlight

> treating your peers as your first team allows you to begin practicing your manager's job, without having to get promoted into the role first.

Page 156 | Highlight

> The best learning doesn't always come directly from your manager, and one of the most important things a first team does is provide a community of learning.

Page 156 | Highlight

> I believe that your career will be largely defined by getting lucky and the rate at which you learn. I have no advice about luck, but to speed up learning I have two suggestions: join a rapidly expanding company, and make your peers your first team.

Page 157 | Highlight

> hiring engineering manager-of-managers

Page 157 | Highlight

> Ensuring that internal candidates take part is essential to an inclusive culture. Fair consideration doesn't mean that we prefer internal candidates. Rather, it means that there is a structured way for

> them to apply, and for us to consider them.

Page 157 | Highlight

> Then comes the trickier part: evaluation.

Page 157 | Highlight

> Partnership. Have they been effective partners to their peers, and to the team that they've managed? Execution. Can they support the team in operational excellence? Vision. Can they present a compelling, energizing vision of the future state of their team and its scope? Strategy. Can they identify the necessary steps to transform the present into their vision? Spoken and written communication. Can they convey complicated topics in both written and verbal communication? Can they do all this while being engaging and tuning the level of detail to their audience? Stakeholder management. Can they make others, especially executives, feel heard? Can they make these stakeholders feel confident that they'll address any concerns?

Page 158 | Highlight

> Peer and team feedback. Collect written feedback from four or five coworkers.

Page 158 | Highlight

> A 90-day plan. The applicant writes a 90-day plan of how they'd transition into the role, and what they would focus on.

Page 158 | Highlight

> Vision/strategy document. The applicant writes a combined vision/strategy document.

Page 159 | Highlight

> Vision/strategy presentation. Have the applicant present their vision/strategy document to a group of three to four peers.

Page 159 | Highlight

kindle

> Executive presentation. Have the applicant present their strategy document, one-on-one, with an executive.

Page 159 | Highlight

> use a similar format for training managers.

Page 159 | Highlight

> Company culture and managing freedoms

Page 159 | Highlight

> “Freedom’s just another word for nothing left to lose.” 8

Page 160 | Highlight

> Positive freedom is your freedom to do: to vote, to wear the clothing you want,

---

Page 160 | Highlight

> Negative freedom is your freedom from things: from being forced to take an impossible literacy exam before being allowed to vote,

---

Page 160 | Highlight

> lives. Each positive freedom we enforce strips away a negative freedom, and each negative freedom we guarantee eliminates a corresponding positive freedom.

Page 160 | Highlight

> the Paradox of Positive Liberty.11

Page 161 | Highlight

kindle

> generally follow the standard operating procedure (i.e., keep doing what you're already doing, the way you're doing it), but always change exactly one thing for each new project.

Page 161 | Highlight

> Kill your heroes, stop doing it harder

Page 161 | Highlight

> Unless your problem is that people aren't trying hard, the "work harder" mantra only breeds hero programmers whose heroic ways make it difficult for nonheroes to contribute meaningfully.

Page 161 | Highlight

> You've bred a cadre of dissatisfied and burned-out heroes. You and your heroes have alienated everyone else. Your project is still totally screwed.

Page 163 | Highlight

> One of the observations from systems thinking16 is that, though humans are prone to interpreting events as causal, often problems are better described in terms of a series of stockpiles that grow and shrink based on incoming and outgoing flows.

Page 166 | Highlight

> Your options for addressing a broken system depend on whether you're in a position to set policy.

Page 166 | Highlight

> Kill your heroes and stop doing it harder. Don't trap yourself in your mistakes, learn from them and move forward.

Page 169 | Highlight

> Luck plays such an extraordinary role in each individual's career progression that sometimes the

Page 169 | Highlight Continued

> entire concept of career planning seems dubious. However, as managers, we have an outsize influence in reducing the role of luck in the careers of others.

Page 169 | Highlight

> creating fair and effective processes for interviewing, promoting, and growing the folks we work with.

Page 169 | Highlight

> Roles over rocket ships, and why hypergrowth is a weak predictor of personal growth

---

Page 170 | Highlight

> The good news is that both the stable eras and the transitions are great opportunities for growing yourself. Transitions are opportunities to raise the floor by building competency in new skills, and in stable periods you can raise the ceiling by developing mastery in the skills that the new era values.

Page 171 | Highlight

> don’t treat growth as a foregone conclusion. Growth only comes from change, and that is something you can influence.

Page 171 | Highlight

> Running a humane interview process

Page 171 | Highlight

> a prepared presentation on a technical topic instead of an impromptu presentation (this more closely replicates a real work task),

Page 171 | Highlight

> collaborative pair programming on a laptop with your editor of choice.

Page 172 | Highlight

> with enough dedicated sourcers, any process will hit your hiring goals—then it can be hard to prioritize improving your process).

Page 172 | Highlight

> while interviewing well is far from easy, it is fairly simple. Be kind to the candidate. Ensure that all interviewers agree on the role's requirements. Understand the signal your interview is checking for (and how to search that signal out). Come to your interview prepared to interview. Deliberately express interest in candidates. Create feedback loops for interviewers and the loop's designer. Instrument and optimize as you would any conversion funnel.

Page 172 | Highlight

> Start from being nice and slowly work your way through to the analytics.

Page 172 | Highlight

> When an interview runs overtime before you get to the candidate's questions, the kind thing to do is to allow the candidate a few minutes to ask questions, instead of running on to the next interview to catch up.

Page 173 | Highlight

> you can't conduct a kind, candidate-centric interview process if your interviewers are tightly time-constrained.

Page 173 | Highlight

> give them an interview sabbatical for a month or two,

kindle

> strong relationship with open communication between engineering managers and recruiters is important.

> reinforcing expectations during every candidate debrief in order to ensure interviewers are “calibrated.”

> agreeing on the expected skills for a given role can be far harder than anyone anticipates,