**Chapter 2. Three Maps**

As a staff engineer, you need a broad view. Every time you react to an incident, run a meeting, or give advice to a mentee, you’ll need context about the people you’re working with and what the stakes are. When you propose a strategy or move a project along, you’ll want to understand how your organization works and the difficulties you might run into along the way. And you won’t make good choices about what to work on unless you can step outside your day to day and see where you’re all supposed to be going.

In [Chapter 1](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch01.html#what_would_you_say_you_do_herequestion), we zoomed out and took a big-picture view of what staff engineers are and why organizations need them. We defined some axioms that are helpful in understanding staff roles, and then I invited you to do a fact-finding mission to unpack some aspects of your own role: your *reporting chain*, your *scope*, your *work preference*, and your current *primary focus*. If you didn’t already have a big picture of what your job is, I hope you now do. But if you’ve ever been hiking or navigated through a new city, you’ll have seen that knowing where *you* stand is just the beginning. Getting oriented means knowing about your surroundings, too.

**Uh, Did Anyone Bring a Map?**

In this chapter, we’re going to describe the big picture of your work and your organization by drawing some maps. Maps take different forms depending on their purpose: you wouldn’t try to include elevation, voting districts, and subway navigation on a single map, for example. So rather than overlaying all of the information we have into one dense, unreadable picture, we’re going to set out to build three different maps. They won’t be perfect models, but they’re useful tools for thinking about work and asking yourself questions about where you are, how your organization works, and what you’re all trying to do.

You can approach this as a mental exercise—just a metaphor for thinking about your engineering organization—or you can actually set out to draw these maps. It can be enlightening (and fun) to compare notes with a colleague and see which landforms and points of interest you disagree on.

Here are the three maps we’re going to end up with:

**A Locator Map: You Are Here**

We’re going to start with your place in the wider organization and company. Last chapter we talked about your *scope*, but to truly understand that scope, you need to see what’s outside it. What’s along the borders? When you zoom way out, how big is your part of the world compared to everywhere else? Think of it like one of those maps that a news station throws up behind the presenter to remind you where a particular place is, and put it in context.

You need the locator map because it’s tricky to be objective about any work while you’re deep inside it. Unless you can maintain perspective, the concerns and decisions of your local group will feel more important to you than they would if you looked at them on a bigger scale. So we’ll try out some techniques for getting that perspective. You’ll be honest with yourself about which of the projects you care about would actually show up on a big map of the company, and which ones you wouldn’t see unless you zoomed all the way in.

**A Topographical Map: Learning the Terrain**

The second map is all about navigating the terrain. If you’re setting off across the landscape, you’ll go further and faster if you have a robust knowledge of what’s ahead. In this section, we’ll look at some of the hazards on the map: the canyons and ridges along the fault lines of your organization, the weird political boundaries in places nobody would predict, and the difficult people everyone’s been going out of their way to avoid. If there’s quicksand ahead, or krakens to be wary of, or an impassable desert full of the sun-bleached skeletons of previous travelers, you’ll want to mark those pretty clearly before you set out on your journey.

Despite the dangers and difficulties, you might find that there are navigable paths already in place. Discovering these paths will include understanding your organization’s “personality” and how your leaders prefer to work, clarifying how decisions are made, and uncovering both the official and the “shadow” organization charts.

**A Treasure Map: X Marks the Spot**

The third map has a destination and some points on a trail to get there. It shows where you’re going and lays out some of the stops on the journey. The voyage might be perilous, but if you have a map, you’ll be able to see whether you’re getting any closer to that huge red X.

Uncovering this map means taking a long view and evaluating the purpose of your work. Is each project a goal in itself, or is it just a milestone along the path to the actual goal? Sometimes you’ll discover that there isn’t a destination at all or that there are several incompatible ones. When nobody has declared what the treasure is, or everyone disagrees on how to get to it, a staff engineer can have a huge impact by creating a vision or strategy, making decisions, or otherwise drawing a brand-new treasure map for the organization. But I’m getting ahead of myself. For now we’re looking at uncovering the *existing* big picture. Creating a *new* one will happen in [Chapter 3](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch03.html#creating_the_big_picture).

**Clearing the Fog of War**

These three maps already exist in your organization; they’re just obscured. When you join a new company, most of the big picture is completely unknown to you. A big part of starting a new job is building context, learning how your new organization works, and uncovering everyone’s goals. Think of it like the [fog of war](https://oreil.ly/P6S9K) in a video game, where you can’t see what awaits you in the parts of the map you haven’t explored yet. As you scout around, you clear the fog and get a better picture of the terrain, learning what’s surrounding you and whether there are wolves coming to bother your villagers. You can set out to uncover the obscured parts in all three of the maps and find ways to make that information easy for other people to understand. For instance:

* Your locator map can help you make sure the teams you work with really understand their purpose in the organization, who their customers are, and how their work affects other people.
* Your topographical map can help highlight the friction and gaps between teams and open up the paths of communication.
* Your treasure map can help you make sure everyone knows exactly what they’re trying to achieve and why.

You’ll be able to clear some parts of the map through everyday learning, but you’ll need to deliberately set out to clear other parts. A core theme of this chapter is how important it is to know things: to have continual context and a sense of what’s going on. Knowing things takes both skill and opportunity, and you might need to work at it for a while before you start seeing what you’re not seeing.

Let’s start with the skill. I spent a few months in the Irish countryside during the pandemic and went for a lot of nature walks with my friends who live there. At first, I thought I was seeing everything there was to see: a bunch of foxgloves or an oak tree, things that were striking and beautiful. But my friends were seeing more than I was. They’d pause at a patch of mud that I wouldn’t have looked at twice and point out the footprint of a pine marten. They’d pick out leaves that I would have dismissed as just grass, and note that they’re delicious and peppery, a treasure for foragers. Even the kids would see little flowers or dive on a patch of wild strawberries that I’d have walked right past. Why could they see all of these things when I couldn’t? Because they had learned to pay attention and they knew what they were looking for.

Paying attention means being alert to facts that affect your projects or organization. And that means continually sifting information out of the noise around you. If you can train your brain to say “That’s interesting!” and remember facts that you might need later on, you’ll start to add detail to your maps and build skills in synthesizing new information.

What sorts of facts are useful? Anything that can help you or others have context for your work, navigate your organization, or progress toward your goals. Here are some examples:

* A company all-hands presentation about an upcoming marketing push might be a hint that huge traffic spikes you’re not ready for are coming your way.
* Your director asks you to take on a project you don’t have time to do, but you know which senior engineers in your organization are ready for opportunities to stretch their skills.
* A shift in corporate priorities could mean a platform you’d considered but backburnered has become an amazing investment.
* Your database just disappeared, and you remember getting an email about network maintenance.

Over time, you’ll get used to how news travels in your org and what you should pay attention to. You’ll know which emails you need to read and which meetings you need to go to. If your brain’s not naturally “sticky” for retaining information like this, I recommend challenging yourself to note down facts that might be useful later, just to get yourself into the habit of paying attention. Think of gathering context as a skill to build as part of your job.

But noticing only takes you so far. Paying attention doesn’t help if you don’t have access to the decisions and discussions that affect your work. While you might be privy to a daily flow of meetings, emails, and regrettable @here messages on Slack, there’s a lot of other information you won’t be able to ask for unless you know it exists. How do you get into the “room where it happens”? I’ll share some strategies in this chapter.

**The Locator Map: Getting Perspective**

As you grow in seniority, making a real impact will mean being able to put your work in a bigger context, and recognizing that your point of view is heavily influenced by where you’re standing. ([Figure 2-1](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#a_locator_map_of_the_milky_way_galaxy_l) gives maybe a little too much perspective.)



**Figure 2-1. A locator map of the Milky Way galaxy (original Milky Way image by Jean Beaufort, CC0).**

Of course, everyone else you work with will have their own point of view too: their “You are here” marker will be somewhere else on the map. If you want to make good decisions, you’ll need to be able to see from some of those other points of view.

The more time you spend absorbed in any domain and learning the nuances of the work at your scope, the richer and more complex it will become for you. As you understand the people, the problems, and the goals, you’ll become more focused on them. That focus brings depth and understanding, but it comes with some risks, especially for a staff engineer.

Let’s look at four of those risks now.

*Prioritizing badly*

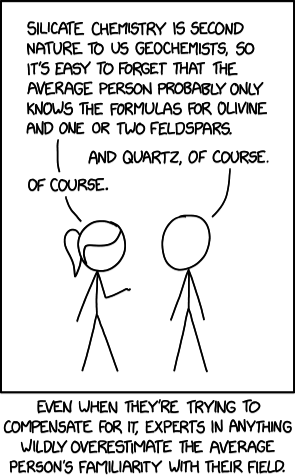
When everyone around you cares about the same set of things, it’s easy to magnify the importance of those things. The problems that exist outside your group can start to appear simple or unimportant by comparison. That’s why you see teams making those local maximum decisions I talked about in [Chapter 1](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch01.html#what_would_you_say_you_do_herequestion): the local maximum starts to feel *really* important. The more time you spend staring at your own group’s problems, the more they seem special and unique and worthy of special, unique solutions. And sometimes they are! But it’s unusual to find a problem that is genuinely brand new. If you check for prior art and preexisting solutions, you’ll spend less time reinventing wheels.

*Losing empathy*

It’s easy to overfocus and forget that the rest of the world exists, or start thinking of other technology areas as trivial compared to your rich, nuanced domain. It’s like you start looking at the world through a fish-eye lens that makes the thing right in front of you huge and squeezes everything else into the periphery. You can lose empathy for the work other teams are doing: “That problem they’re solving is easy. I could solve it in a weekend.”

The words you use, the things you choose to explain versus those you leave implicit, and the motivations you ascribe to other people will all be influenced by your perspective. That’s why it can be so difficult for engineers to communicate with nonengineers. [Figure 2-2](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#itapostrophes_easy_to_lose_perspective) tells a familiar story about how easy it is to misunderstand what other people know about your domain.

Loss of empathy shows up in incidents, too, where teams can get absorbed in the interesting technical details of the problem and forget there are users waiting for the system to be back online.



**Figure 2-2. It’s easy to lose perspective about what other people know (source:**[**https://xkcd.com/2501**](https://xkcd.com/2501)**by Randall Munroe).**

*Tuning out the background noise*

If one failure mode is your team’s concerns seeming more important than everyone else’s, another is the exact opposite: you stop noticing problems at all! If you’ve been working around the same mucky configuration file or broken deploy process for months, you might get so used to it that you stop thinking of it as something you need to fix. Similarly, you might not notice that something that started out as just slightly annoying has slowly become worse. Maybe a problem is close to becoming a crisis, but you don’t even notice it anymore, so you can’t be objective about how quickly you need to react.[**1**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn20)

*Forgetting what the work is for*

Being in your silo can mean that you lose your connection to what’s going on elsewhere in the company. If your group originally took on some project to solve a larger goal, the project might still be ongoing even though the goal no longer matters or has already been solved in some other way. If you’re working only on your own little part of a project, it’s easy to stop thinking about what the project is *for*. You can slip into a world where everyone does their own little part and nobody feels like they’re responsible for the end result. You can lose sight of the ethics of what you’re doing, too, and find yourself working on something that you wouldn’t really be OK with if you stepped back and thought about the whole picture.

**Seeing Bigger**

Open up your company’s org chart and look at where your group and others you care about connect to the rest of the organization. When you extend the amount of the map you can see, your own group might seem a lot smaller, and your “You are here” pin might feel far from where the action is. But without perspective, you can’t do impactful work. In this section we’ll look at some other techniques for seeing the bigger picture

**Taking an outsider view**

When I was the newest person on an infrastructure team years ago, my colleague Mark commented after a few weeks, “There’s this facial expression you do when I describe our systems...” Certainly I’d thought a few of the aged systems needed to be replaced, but I hadn’t realized I was wearing my opinions so clearly (and rudely!) on my face. Two years later, the team’s hard work meant that the architecture had vastly improved. We were proud of the work. I thought it was pretty good! Until a new person joined and…wore their opinions pretty clearly on their face. By then, I had become a team insider. I needed a newer “new person” to help me see the problems again.

When the new person on your team looks at an architectural tangle or a pile of technical debt, they have no historical context. As my colleague, Dan Na, [says](https://oreil.ly/GD8Gz), a new person can always see the problems. They haven’t been around for the gradual change and the boiling frogs: they’re just seeing the raw situation as it is. Without preconceptions, they’re free to look around and ask, “What’s really happening here? Is any of this working?”

**WARNING**

Being new isn’t a license to be a jerk. It’s easy with hindsight to say, “This is terrible! Why didn’t they just...” But have humility and assume there are good reasons for everything being the way it is. Amazon’s principal engineer group acknowledges this in one of its community tenets: [“Respect what came before”](https://oreil.ly/2R4ET).

Being new is the best opportunity you’ll have to get a complete outsider view, but as a staff engineer, you should try to have this perspective all the time. You need to be able to look at your own group as if you weren’t part of it and to be honest about what you see. Do your technical decisions only make sense to people who have forgotten that there’s a world outside your team? If you all stopped doing the work you’re doing, how long would it be before other people would notice or care? Have you gotten absorbed in the technology and forgotten what your original goal was? *How is everything?* The next four sections offer techniques for viewing things like an outsider.

**Escaping the echo chamber**

When you find yourself in an echo chamber where everyone you meet holds the same set of opinions, it can be a shock when you connect with peers in other groups and discover that some of their views are just…different.

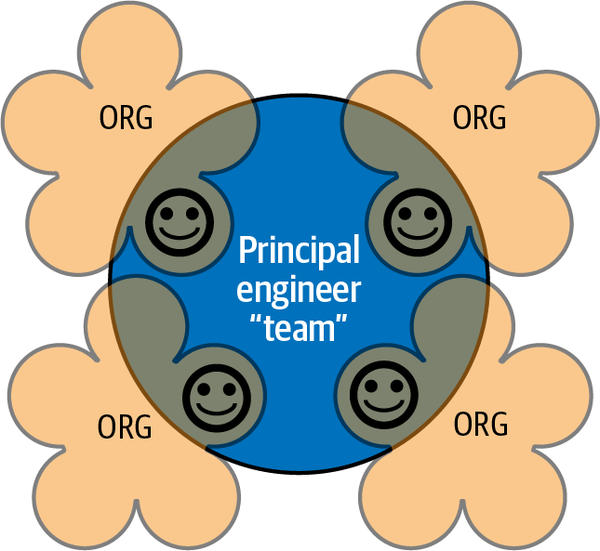
After spending more than a decade at the bottom of the stack in infrastructure roles, it was a shock to my system when I first worked with product engineering teams. They moved *fast,* took risks, and thought creating features that customers loved was *at least as important* as those features having rock-solid reliability. Our debates shook some of my firmly held beliefs and made them more nuanced.

Seeking out peers in other groups is an important part of your job. Build friendly relationships with other staff engineers. Get to a point where you can speak the truth to one another, and it won’t be contentious, because you’ve built up so much goodwill. This includes understanding any negative opinions that other teams hold about your group—if you start seeing what’s valid about their comments, you’ll do better work. Think of the other staff engineers as *your team*, just as much as any team you’re part of.

The same principle applies across organizations. In [Figure 2-3](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#an_example_software_engineering_organiz) and [Figure 2-4](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#multiple_engineering_organizations_insi), I depict each staff engineer as scoped to a single group, and each principal engineer as scoped to an organization. While the actual structure will vary, the point is to be part of something that’s bigger than your own team or group, so you can have a more objective view of what everyone is doing.



**Figure 2-3. An example software engineering organization. Each group here contains multiple teams. In this company, each staff engineer’s scope is a single group, and they consider themselves to be part of their own group, but also part of a bigger virtual “team” of staff engineers.**



**Figure 2-4. Multiple engineering organizations inside a company, each of which has a staff engineer. Every principal engineer is in their own org, but is also part of the virtual team of principal engineers.**

Go beyond engineering: build relationships with product folks, customer support, administrative staff, and more. If your work affects them or their work affects you, go be friendly and understand their point of view. It will give you a whole new way of thinking about what’s important to your department or your business.

**What’s actually important?**

Befriending nonengineers is good for your perspective in another way: As an engineer, it’s easy to get absorbed in technology. But technology is a means to some end. Ultimately you’re here to help your employer achieve its goals. You should know what those goals are. You should know what’s *important*.

A startup will have a different definition of what matters than a behemoth tech giant or a local nonprofit. A mature product will have different needs than an early one. Some goals, and thus some projects, matter more than others. [Figure 2-5](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#putting_your_project_in_perspectivedot) shows how a project that feels like the center of your universe can be much less significant when looking at a bigger picture. The ordering will change over time, so understand what matters *right now*. If your customers are leaving in droves because your product is missing core features that your competitors have, it’s probably not the time to push for a focus on technical debt. If everything is smooth sailing and you’re anticipating growth, this might be a great time to make sure your foundations are solid.



**Figure 2-5. Putting your project in perspective. That upgrade might be the most important work your organization has, but people looking at a bigger picture won’t see it as important.**

A company’s goals extend beyond its stated objectives and metrics; they include “continuing to exist,” “having enough money to pay everyone,” and “having a good reputation.” My colleague, Trish Craine, head of operations for engineering at Squarespace, calls these “the objectives that are always true.” These are the needs of your company that are so obvious, they’re only really stated if they’re in danger. The product or service that your organization provides should *work*. Its customers should want to use it. Deploying it shouldn’t be painfully slow. Know your implicit goals as well as the explicit ones.

**TIP**

As time passes, your company’s priorities will change and parts of your map will fog up again. To stay up to date with what’s important, pay attention to all-hands meetings for your group and others, ask for skip-level one-on-ones with your manager’s manager, and find face time with customers or teams that depend on you. If you don’t have business context about why (or whether) your work matters, ask for it.

Notice when the goals change, too, because that might mean your scope or focus should change. It’s OK if you’re not working on the *most* important thing, but what you’re doing should not be a waste of your time. If you can’t explain to yourself why what you’re doing needs a staff engineer, you might be doing the wrong thing.

**What do your customers care about?**

Charity Majors, CTO of Honeycomb, often hands out [stickers](https://oreil.ly/2pxrj) that say: “Nines don’t matter when users aren’t happy.” “Nines” here refers to [service level objectives (SLOs)](https://oreil.ly/9LeCI), a common mechanism for measuring system availability. “Three and a half nines of availability” means that 99.95% of the time, the service is up and running. SLOs are useful, but as Majors points out, they don’t tell the whole story. Because who defines what “available” means?

Mohit Suley, an engineering manager and former principal engineer at Microsoft, [has spoken about his team tracking down and contacting unreliable ISPs](https://oreil.ly/Fsj4k) where their search engine, Bing, wasn’t reachable. It wasn’t Bing that was broken, but as Suley says, “A user doesn’t distinguish between DNS services, ISP, your CDN, or your endpoint, whatever that might be. At the end of the day, there are a bunch of websites that work, and a bunch that don’t.” You need to measure success from your users’ point of view. (If your customers are other teams inside your company, this still applies!) If you don’t understand your customer, you don’t have real perspective on what’s important.

**Have your problems been solved before?**

Amazon’s tenet of [“respect what came before”](https://oreil.ly/2R4ET) includes a reminder that “many problems are not essentially new.” You’ll come up with better solutions if you study what other people have already done before creating some new thing. Remember that your goal is to solve the problem, not necessarily to *write code* to solve it. Take the time to understand what already exists—inside and outside your organization—before building something new.[**2**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn21)

**INDUSTRY PERSPECTIVE**

Understand how other people in the industry have solved the problems you’re working on. Your preferred publications and resources will depend on your interests, but here’s some I find valuable for architecture, technical leadership and software reliability. I love the [LeadDev](https://oreil.ly/P3SYi) and [SREcon](https://oreil.ly/S6OYy) conferences and try to make it to as many as I can. LeadDev has a new (at the time of writing) conference track called [StaffPlus](https://leaddev.com/staffplus-new-york). I’m hosting some of their events, so I can’t be entirely objective, but I think it’s excellent!

For online conversations, I like [Rands Leadership Slack](https://oreil.ly/ZheFA): the #architecture and #staff-principal-engineering channels are gold. The LeadDev Slack’s #staffplus channel is very active during events, too.

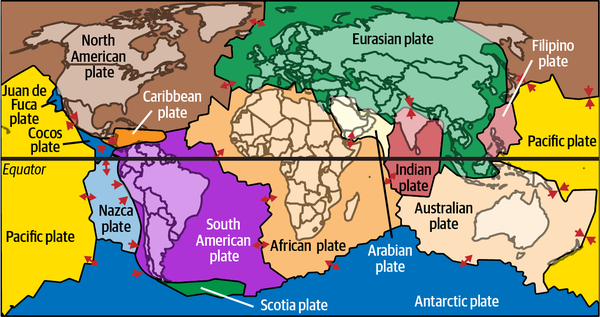
I subscribe to the monthly [InfoQ Software Architects’ Newsletter](https://oreil.ly/ReBFX), as well as [the VOID report](https://oreil.ly/wx82Q), and [SRE Weekly](https://sreweekly.com/). I read [the Raw Signal newsletter](https://oreil.ly/CwcQp) for a weekly dose of a manager point of view. I also eagerly await the quarterly [Thoughtworks Radar](https://oreil.ly/iu0Sy).

Whatever domain you’re in will have its own publications. Use them to maintain perspective and spot new ideas that you can explore when you need them. They’ll help you keep learning, too.

**The Topographical Map: Navigating the Terrain**

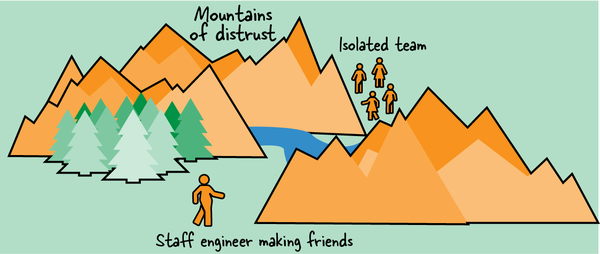
A locator map gives perspective, but you can’t navigate by it. You need another map: one that shows the terrain.

Geologists study *plate tectonics*, the way the huge pieces of the earth’s lithosphere (see [Figure 2-6](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#simplified_map_of_earthapostrophes_prin)) move against each other over time, forming mountains and trenches and creating earthquakes and volcanic activity. Team tectonics have similar properties. As domains of responsibility smash against each other, they form an organizational terrain, complete with overlaps and conflict, ridges and chasms.



**Figure 2-6. Simplified map of Earth’s principal tectonic plates. (adapted from: Scott Nash, public domain,**[***https://oreil.ly/UdeNz***](https://oreil.ly/UdeNz)**).**

Reorganizations can disrupt communication between groups that need to work closely together. Teams that are under a heavy load can entrench and put up barriers. A new senior leader can cause an earthquake that reshapes the landscape overnight. Navigating an organization (see [Figure 2-7](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#a_staff_engineer_navigating_tricky_terr)) requires a *topographical* map.



**Figure 2-7. A staff engineer navigating tricky terrain.**

**Rough Terrain**

Let’s explore some of the difficulties you’ll face if you set out on a mission without a detailed map of the terrain.

*Your good ideas don’t get traction*

Being *right* about a need for change is less than half the battle. You’ll have to convince other people that you’re right and, even more difficult, convince them to *care* that you’re right. That means knowing how to build momentum within your organization: figuring out who can sponsor your idea or help it spread, and how you can get it over the finish line and make it “real.”

*You don’t find out about the difficult parts until you get there*

Many obvious-seeming journeys have some crucial point that nobody has figured out how to get past. You may be attempting to scale a cliff that’s defeated many other people before. Staff engineers can often navigate past obstacles that less experienced engineers can’t, and it’s possible that you’ll be able to succeed where others have failed. But if you know where people got stymied in the past, you can take a different path or solve the hardest part of the problem first, so other people will be convinced the project is worth their effort.

*Everything takes longer*

Unless you know how your organization works, decisions that should be straightforward will take months or quarters. The mechanics of your organization’s planning cycles will affect you too. There are times of year when it’ll be easier to make the case for staffing a new project or to rally everyone behind some goal. If you announce an initiative immediately after the quarterly engineering OKRs have been set, you’ll have a hard fight and you may have to wait a quarter before you’ll see any progress toward your goal.

**Understanding Your Organization**

Engineers sometimes dismiss organizational skills as “politics,” but these skills are part of good engineering: considering the humans who are part of the system, being clear about the problem you’re solving, understanding long-term consequences, and making trade-offs about priorities. If you don’t know how to navigate your organization, every change will be much more difficult.

In this section, I’ll describe some ways you can clear the fog and understand your company’s terrain. That starts with evaluating some aspects of your culture, including what gets written down, how much trust there is, whether people are eager or hesitant to change, and where new initiatives come from. This knowledge will set your expectations about an average journey: will it be easy to make progress? After that, we’ll look at some of the obstacles and shortcuts that will show up on your topographical map.

**What’s the culture?**

Whenever I interview a job candidate, their first question is often, “What’s the culture like?” I used to struggle to answer; where do you even start? Tomes have been written on organizational culture. Now, though, I think most of the time people are really asking these questions:

* How much autonomy will I have?
* Will I feel included?
* Will it be safe to make mistakes?
* Will I be part of the decisions that affect me?
* How difficult will it be to make progress on my projects?
* Are people…you know…*nice?*

Your company culture is not the only factor determining the answers to these questions: individuals and leadership are part of it, too. But organizations do have their own distinct “personalities,” so let’s talk culture.

If your organization has published a statement of values or principles, that can help you see what the leaders care most about. But these values are aspirational: the real values of the company are reflected in what actually happens every day.

To understand more about the engineering culture at your company, here are a few questions you could ask yourself or discuss with a colleague. For most of them there’s no right or wrong answer. It will be difficult to maneuver if your company is all the way over on one side or another, but there’s a lot of space for success in between.

**Secret or open?**

How much does everyone know? In secret organizations, information is currency and nobody gives it away easily. Everyone’s calendars are private. Slack channels are invite-only. Often you can get access to something if you ask for it, but you have to know it exists! When all information is need-to-know, it’s harder to come up with creative solutions or really understand why something’s not working.

In open organizations, you’ll have access to *everything* (even messy first drafts!)*.* You might get decision fatigue from choosing which information to consume. You might not know which documents are official and need action, and which are just early ideas. And open information can lead to more drama: it’s harder for bad ideas to be quietly shut down.

Knowing the cultural expectations around sharing is crucial. In a culture that keeps knowledge locked down, you’ll lose your boss’s trust if you reshare something they told you in confidence. In a more open company, you’ll be considered political or untrustworthy if you withhold information or don’t make sure everyone knows what’s going on.

**Oral or written?**

What gets shared by word of mouth and what gets written down? How much writing and review is involved in decisions? In some companies, it’s typical to make a big decision during a hallway conversation or to find out your colleague has built a huge new feature after it launches (or when you get paged for it). In other workplaces, every software change comes with a formal specification, requirements, sign-offs, and an approvals checklist, and you can expect a one-line change to take a quarter.

Thankfully, most workplaces are somewhere in between. If yours prefers quick conversations, you may get pushback if you take the time to write a decision down—and a design document longer than a page just won’t get read. Bigger and more mature companies tend to be more deliberate about changes. If you’re at one of those and you *don’t* create a change management ticket or a design document, you’ll seem sloppy and irresponsible. One team I worked in had a cowboy hat that would end up on the desk of whichever team member had last done something a little too “Wild West.” It was affectionate, but it was a good reminder too.

**Top-down or bottom-up?**

Where do initiatives come from? A completely bottom-up culture is one where employees and teams feel empowered to make their own decisions and champion the initiatives they think are important. However, when those initiatives need broader support, they slow down. If teams disagree about direction or priority, the lack of a central “decider” can lead to deadlock.

On the other hand, people in a fully top-down company will find it much easier to choose initiatives and take decisive action. Those decisions won’t be the best ones, though, because they’re missing local context. The engineers likely feel controlled and won’t be empowered to react to changes as they arise.

Staff+ engineers should be fairly autonomous and self-directed, but make sure your organization agrees: if your manager expects to approve where you spend your time, it can cause conflict if you don’t check in. If you’re used to seeking permission or having work handed down, and you move to a bottom-up company, you’ll be seen as having low initiative and have trouble getting anything done.

If you know how your organization tends to work, you’ll also know whether to take your ideas to fellow grassroots practitioners and get their support first, or whether to start by trying to convince your local director.

**Fast change or deliberate change?**

Younger companies tend to make rapid decisions and pivot abruptly to try a new opportunity. As companies get larger and older, they take longer to change course. “Fast” organizations may be repelled by the idea of taking on a long-term project like a two-year migration. Slow ones will miss low-hanging opportunities to improve.

Depending on where you are, you’ll need to frame your initiatives differently. If you’re somewhere that moves like lightning, you’ll want an incremental path that shows value immediately. In a more deliberate environment, you’ll need to show that you’ve thought through the whole plan. This is tightly connected to oral and written culture, too.

**Back channels or front doors?**

How do people in different groups talk with each other? There may be formal paths for information and requests, but your social culture adds informal channels too. If people are friendly across teams, they’ll send a DM when they have a question and share ideas over coffee.[**3**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn22)

If an engineer in one group can just go chat with a counterpart in another, it’s going to be easier to make decisions that cross both teams. In some places, the only real way to get work done is to have an “in” via a back channel with someone on the team. If it’s more typical to file a ticket and wait, or to send a collaboration idea up your management chain until you and the other team have a manager in common, everything will take longer—but it will also be more predictable and fair.

Understand what’s considered typical in your organization. If everyone is strict about only using formal channels, it will be considered rude to ask questions out of turn, and people will judge you poorly for skipping the queue. If back channels are the typical way to get things done, you’ll be waiting for a response for a month when you could have just had a chat with that person who’s been admiring your cat pictures on the company pets mailing list.

**Allocated or available?**

How much time does everyone have? If teams are understaffed and overworked, you’ll have trouble finding a foothold for any new idea that isn’t on an existing product road map—the fastest and easiest response is just to say no without really looking at the request. You’ll have the most impact with any initiative that can free up time without major investment. Your most likely successes will come when you can work alone or with a little help, and don’t need to get a bunch of busy people to commit to anything new.

Teams that aren’t busy may seem easier to work with, but they have a different problem: underallocated engineers rarely stay that way for long. If there are plenty of free cycles available, chances are that a Cambrian explosion of competing novel grassroots initiatives is taking hold, each with a small number of devotees. You’ll have more impact if you choose a nascent project, help it over the finish line, and convince others to rally around it too.

**Liquid or crystallized?**

Where do power, status, and reputation come from? How do you gain trust? Some organizations, particularly in academia and in big and old companies, have a clear hierarchy: the same group of people, in the same configuration, climb the ranks together and have a fairly fixed structure for communicating, making decisions, and allocating the “good projects.” Each person is like a node in a crystal lattice: so long as the people around you are moving up, you’ll move up too. Senior people in groups like this will often say that they never looked for promotion: they stayed where they were, got a project, got support from the group, and got the promotion when it was their turn.

This sort of hierarchy is anathema to young, small, scrappy companies, which claim to be something like a meritocracy. [Let’s be realistic about that](https://en.wikipedia.org/wiki/Myth_of_meritocracy): success still depends on having access to opportunities and sponsorship, so it’s hugely affected by stereotype bias, in-group favoritism, and other cognitive biases. (See the website [Is Tech a Meritocracy?](https://istechameritocracy.com/) for more.) “Liquid” companies offer more room to change your place in the structure, but you’ll likely have to hustle a bit to get promoted. You might move from group to group to find high-impact work so you can advance at your own pace. If you sit around waiting for someone to assign you a project, you’ll be waiting a long time.

In teams with a solid crystalline lattice, it’s vital that you understand your place in the hierarchy and know when your time will come to have a project that will take you to the next level. If you suggest taking on something that’s been earmarked as a promotion project for someone else, you’ll ruffle feathers—one friend who tried this said their boss looked at them “like I’d suggested stealing the silverware.”

Try the slider diagram in [Figure 2-8](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#most_companies_will_be_somewhere_in_the) to think about how these seven attributes influence how your organization works. If you’re trying to cause a culture change, it’s often possible—with determined effort—to nudge the sliders in one direction or the other over time. At least know where they are, and you’ll avoid some of the pitfalls of working against the prevailing culture.



**Figure 2-8. Most companies will be somewhere in the middle on each of these attributes.**

**Power, rules, or mission?**

Here’s another lens to view your culture through: what do your leaders think is important? In his 2005 paper, [“A Typology of Organisational Cultures”](https://oreil.ly/rtHz5), sociologist Ron Westrum wrote:

*Through their symbolic actions, as well as rewards and punishments, leaders communicate what they feel is important. These preferences then become the preoccupation of the organization’s workforce, because rewards, punishments, and resources follow the leader’s preferences. Those who align with the preferences will be rewarded, and those who do not will be set aside. Most longtime organization members instinctively know how to read the signs of the times and those who do not soon get expensive lessons.*

Westrum classified organizations and their influence on information flows using three categories (see [Table 2-1](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#the_westrum_organizational_typology_mod)):

*Pathological*

A low-cooperation culture where power and status are the goal and people hoard information; in Westrum’s words, “a preoccupation with personal power, needs, and glory”

*Bureaucratic*

A rule-oriented culture where information moves through standard channels and change is difficult; the preoccupation here, Westrum writes, is with “rules, positions, and departmental turf”​

*Generative*

A mission-oriented, high-trust, high-cooperation culture where information flows freely; Westrum calls this ​​“concentration on the mission itself”

| **Pathological** | **Bureaucratic** | **Generative** |
| --- | --- | --- |
| Power-oriented | Rule-oriented | Performance-oriented |
| Low cooperation | Modest cooperation | High cooperation |
| Messengers shot | Messengers neglected | Messengers trained |
| Responsibilities shirked | Narrow responsibilities | Risks shared |
| Bridging discouraged | Bridging tolerated | Bridging encouraged |
| Failure → scapegoating | Failure → justice | Failure →inquiry |
| Novelty crushed | Novelty → problems | Novelty implemented |
| Table 2-1. The Westrum organizational typology model: How organizations process information (Ron Westrum, [“A typology of organisational cultures”](https://oreil.ly/tUnoa)) BMJ Quality & Safety 13, no. 2 (2004), doi:10.1136/qshc.2003.009522. | | |

The [DevOps Research and Assessment (DORA) group (now part of Google Cloud) has shown](https://oreil.ly/Epx4s) that high-trust cultures that emphasize information flow have better software delivery performance. It’s not surprising that an increasing number of software companies aim to have a generative culture. That means encouraging cooperative cross-functional teams, learning from blameless postmortems, encouraging experimentation, taking calculated risks, and breaking down silos. If your organization works like this, you’re going to have an easier time sharing information and making progress.

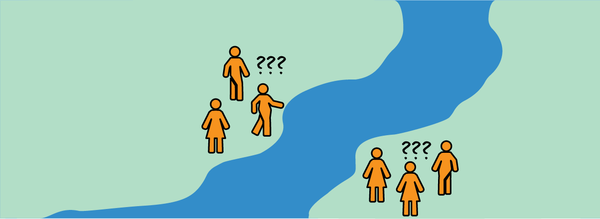
If you know whether your workplace is oriented around power, rules, or mission, you’ll find it easier to get things done. A feeling for how much people will share information, cooperate, take the time to help, and get behind new ideas will keep you safer and less frustrated as you cross the terrain. If you know you’re in a [bureaucracy](https://oreil.ly/4zT4y), you’ll have more success if you plan ahead, stay within the rules, and respect the chain of command. If you’re in a pathological organization, you’ll take fewer risks—and cover your ass when you do. Pushing a cart across cobblestones is more difficult than doing so across smooth paving. If you know the road will be rocky, you’ll budget more time, and you’re less likely to get mad at aspects of the situation that you can’t control.

**Noticing the points of interest**

This brings us back to the topographical map. Understanding your organization’s culture gives you a rough idea of how easy or difficult an average journey will be. But to navigate, you’ll also want to understand the barriers, the difficult parts of the journey, and the shortcuts. Here are a few points of interest in the terrain of organizations I have known.

**Chasms**

The plate tectonics of a company can form chasms between teams and organizations. For instance, [Figure 2-9](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#the_chasm_between_an_infrastructure_and) depicts the canyon that can form between product-focused software engineering teams and the infrastructure, platforms, or security teams providing services for them. Group cultures, norms, goals, and expectations evolve differently, causing gaps that make it difficult to communicate, make decisions, and resolve disputes.



**Figure 2-9. The chasm between an infrastructure and a product engineering organization.**

Smaller chasms can form even within an organization. The edges of each team’s defined responsibilities rarely line up perfectly, and project work and information can get lost in the gaps between teams.

**Fortresses**

Fortresses are teams or individuals who seem determined to stop anyone from getting projects done. Maybe you need their approval but can’t get time with them. Or maybe they’re gatekeepers and seem to decide your idea is bad before they even know what you’re asking. Although some fortresses are petty tyrants, the majority are well-intentioned. They gatekeep because they *care*. They’re trying to keep the quality of the code or architecture high and keep everyone safe.

To pass through the fortress gates, you might need to bring a token of sponsorship from someone the gatekeeper respects, or know the password to lower the drawbridge. (Common passwords include proving that you’ve mitigated all of the risks of your proposed change, completing lengthy checklists or capacity estimates, or replying to huge numbers of document comments with acceptable answers.) Another option is a protracted, bloody battle where you argue every point and pull other people into the fight—winning one of those can be such a Pyrrhic victory that you almost regret trying. Or you can give up and go the long way around the fortress, complicating your journey and losing access to any wisdom the gatekeeper would have shared.

**Disputed territory**

It’s very hard to draw team boundaries in a way that lets each team work autonomously. No matter how opinionated your APIs, contracts, or team charters, there will inevitably be some pieces of work that multiple teams think they own, and navigating those disputes can feel risky.

I worked on a project once that needed a critical system to be migrated from one platform to another. Migrating this particular system accounted for less than 5% of my project, so I didn’t want to spend too much time on it—but when I looked for someone to take responsibility for it, I hit a wall. Ownership of the system was smeared across three teams, each responsible for a different aspect of its behavior. Nobody could tell me whether migrating it to our new platform would be safe. Each group said, “Yes, as far as I know, but you should also ask…” and pointed to the next team. Without an owner who could speak for the whole system, I went around in circles trying to build enough context to convince myself that the migration would work. (It didn’t. Aligning the three teams around the rollback wasn’t pretty either.)

When two or more teams need to work closely together, their projects can fall into chaos if they don’t have the same clear view of where they’re trying to get to. The lack of alignment can lead to power struggles and wasted effort as both sides try to “win” the technical direction. Overlaps in team responsibilities make this worse, complicating decision making and wasting everyone’s time.

**Uncrossable deserts**

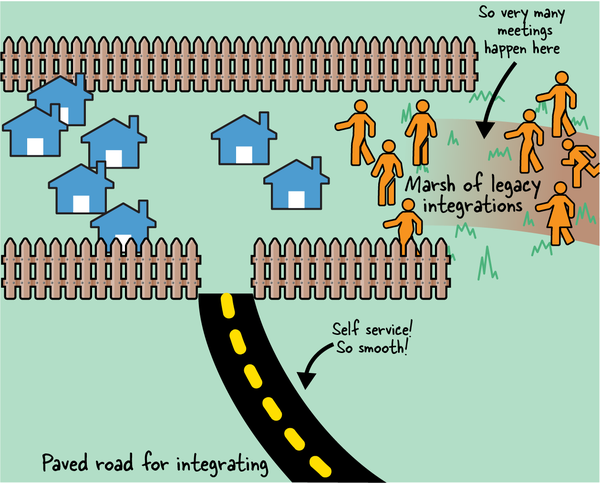
As you try to achieve your goals, you’ll sometimes run into a battle that other people consider unwinnable. This may be a project that’s just too big or a politically messy situation that always ends with a veto from some senior person. Whatever it is, people have tried it before, and any suggestion of tackling it again will be met with discouragement and ennui.

That’s not to say you shouldn’t try! But you should have enough evidence to convince yourself and others that this time will be different. It’s good to know going in if you’re picking a fight that might be unwinnable.

**Paved roads, shortcuts, and long ways around**

Companies that have worked to make engineers efficient will often set up processes to ensure that the official ways to do something are also the easiest ways. An example might be following a self-service checklist to ensure a new component is safe to put into production. If you’re lucky enough to have some of these easy, well-defined paths, know where they are and use them.

Unfortunately, not all roads are well paved. We’ve all tried to solve a problem the official way for a long time before someone told us the secret path to success: the undocumented search feature, the admin who can set up an account for you, or the one person in IT who responds to DMs. Sometimes the official way is the way that everyone learns *not* to use. [Figure 2-10](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#the_new_paved_road_is_beautifulcomma_bu) shows a paved road that doesn’t lead to most of the places people actually want to go; they take the legacy paths instead. If you don’t know these goat tracks through your org, everything takes longer. But when you learn them, teams may ask you not to document them: people *should* use the new path, they insist, and it will be better *soon*.



**Figure 2-10. The new paved road is beautiful, but most of the places people actually want to go are deep in the marsh.**

**What Points of Interest Are on Your Map?**

What else should be on your topographical map? Are there unexpected cliffs you can walk off? Are there behaviors or communication styles that would be perfectly fine in another company or team but that are considered rude in yours? Are there guardrails you expect to be in place that just aren’t? Are there areas that are prone to eruptions, or leaders who cause earthquakes (or surprise reorgs) for people who thought they were on solid ground? How about local politics—which teams are led by monarchs, and which by councils? Which ones are anarchy? Who’s at war with whom?

Try sketching your own map. Remember that [cartography is inherently political](https://oreil.ly/V5UbQ): what you choose to include says something about *you* as well. Pay attention to what you put at the center of the map and where you’re inclined to take sides.

Organizations end up with weird shapes due to reorgs, acquisitions, individual personalities, and, in some cases, people who just don’t like each other. If you come up with barriers, conduits for information, or other landforms that I haven’t thought of, I’d love to hear about them.[**4**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn23)

**How are decisions made?**

It is fascinating to watch how information and opinions flow through a company and to see how unexpectedly they can become a plan of record. Suddenly everyone’s using a new acronym or holding a particular opinion, and it can be hard to see where that came from. A project that held great hope and promise is now dismissed as likely to fail. Everyone’s excited about microservices, or they’ve moved on from microservices and they’re curious about serverless, or they think a modular monolith is just pragmatic common sense. One team has approval to hire more people this year and another doesn’t. How did all of these decisions happen? Was there a memo?

Some decisions seem to emerge from conversations without anyone really declaring that they’ve decided. Others happen more formally, but in rooms you’re not in. If you have a lot of ideas, it can be frustrating when you see other initiatives take root but not yours. Why aren’t they listening to your proposals?[**5**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn24) The truth is something that a lot of us struggle to make peace with: being technically correct about a direction is only the beginning. You need to convince other people too—and you need to convince the *right* people.

If you don’t understand how decisions are made in your organization or company, you’ll find yourself unable to anticipate or influence them. You might also find that you think you hold the same opinions as everyone else about what should happen next, and then find that suddenly everyone is advocating for a different path. If you consistently feel out of the loop, that’s a sign that you don’t understand how decisions are made and who influences them.

**Where is “the room”?**

Decisions that affect you and your scope are happening every day, and it’s uncomfortable if you keep being shocked by them. You should at least have a feeling for where they’re coming from, and you’ll likely want to have some influence on them too. Let’s start with the formal channels and the official meetings where big decisions get made.

Your access to decisions will be different depending on where in the organizational hierarchy you sit. Some of these decisions will inevitably be happening higher up in the company than you are. You can influence them by making sure relevant information reaches those rooms via your reporting chain or other channels. But decisions are also being made directly at your scope, and, as much as possible, you’ll want to be involved. If you’ve watched the musical *Hamilton*, you’ll remember Aaron Burr’s craving to be “in the room where it happens.” As Burr tells us, people who aren’t in the room [“don’t get a say in what they trade away”](https://oreil.ly/G1Csw). While there are times when an outsider perspective can help, this isn’t one of them. If you want to set technical direction or change your local culture, you need to be an insider in the group that’s making the decisions.

Figure out where decisions are happening. Perhaps there’s a weekly managers’ meeting that’s intended to make organizational decisions but that often weighs in on process or technical direction. A director might tend to make plans in their staff meeting with the people who report to them. A central architecture group might have a Slack channel where they come to consensus on the path forward. If you’re not seeing how your organization works, ask someone you trust to walk you through where a particular decision came from. (Be clear that you’re not fighting the decision, just trying to understand the inner workings of your organization.)

Beware: there might not be a “room” at all. At the most extreme ends, major technical pronouncements might get made in one-on-ones with the most senior leader, or they might be intended to be entirely bottom-up (and therefore often not made at all).[**6**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn25) But if there is a “room where it happens” for the kind of decision you’re interested in, find out what that is and who is in it.

**Asking to join in**

Once you discover a meeting where important decisions get made, it’s natural to want to be part of it. But you’ll need a compelling story for why that should happen. It seems obvious, but your reasons should be about impact to your *organization*, not to you *personally*. No matter how much your peer managers like you, framing your exclusion as being bad for your career advancement will be unlikely to change hearts and minds. Show how including you will make your organization better at achieving its goals. Show what you can bring that’s not already there. Have a clear narrative about why you need access, practice your talking points, and go ask to join.

You will probably get some resistance. Adding someone to a group is rarely free for the people who are already there. Every extra person in any meeting slows it down, extends discussions, and reduces attendees’ willingness to be vulnerable or brutally honest. If the group is used to working together, every new person resets the dynamic; to some extent, attendees have to learn to work together again.

If you do get an invitation, don’t make anyone regret inviting you. Will Larson’s article [“Getting in the Room”](https://oreil.ly/us7eX) emphasizes that as well as adding value to the room, you need to reduce the *cost* of including you: show up prepared, speak concisely, and be a collaborative, low-friction contributor. If you make the room less effective at making decisions or sharing information quickly, you won’t be invited back.

If you *don’t* get into the room, don’t take it personally, especially in orgs where people are still figuring out what their staff+ engineers are for and aren’t yet on board with it being a leadership role. While they work that out, you’ll have more influence (and will appear more of a leader) if you’re friendly and do good work than if you grouse about not being invited to things. Understand the situation, be kind, and, as I said in [Chapter 1](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch01.html#what_would_you_say_you_do_herequestion), never be a jerk.

There are also some rooms you just shouldn’t be in. If you’re decidedly on the individual contributor track, you usually shouldn’t be part of discussions about compensation, performance management, and other manager-track things. You might bring information to your manager or director that affects those decisions, but it’s up to them to act. If big technical decisions are happening in the same room as those manager conversations, you could suggest splitting the topics into separate meetings.

Finally, remember that the room you’re trying to get into may contain less power than you think. Years ago, I was shocked to discover that a group of directors didn’t think their opinions carried a lot of weight; they were frustrated at not being able to influence the decisions of the *real* movers and shakers two levels up. It turned out that there was another “room” I hadn’t ever thought about. There were probably others above that! Be realistic about what you’re asking for access to.

**The shadow org chart**

So that’s the formal decision making. If you understand that, you’ll understand a lot about how your organization sets its opinions and decides what to do. But inevitably there’s a whole lot of other influence going on, and some of it will, on the surface, make *no sense whatsoever*. Informal decision making doesn’t follow rules based on hierarchy or job title. Those things certainly carry weight, but there’s more going on.

While it’s important to understand who the official technical leaders are, it’s just as important to understand who they listen to and how they make decisions. What happens if Jan, the director of your infrastructure organization, seems to be entirely on board with your idea, then suddenly goes cold? If you’re paying attention, you’ll learn that Jan’s first move in any decision is to check in with Sam, who joined the team 10 years ago. Sam is not particularly senior, but if Sam thinks something is a bad idea, you’ll never get Jan on board. These influence lines aren’t immediately obvious when you join an organization, so a good early step is to make some friends and ask how the organization works.

In their book [*Debugging Teams: Better Productivity Through Collaboration*](https://oreil.ly/TrqIi), Brian W. Fitzpatrick and Ben Collins-Sussman describe the “shadow org chart”: the unwritten structures through which power and influence flow. The shadow org chart helps you understand who the influencers of the group are, and it’s probably not the same as the actual org chart. These influencers are the people you need to convince before a change can happen.

The authors identify “connectors” who know people all across the org, and “old-timers” who, regardless of rank or title, wield influence just from being around a long time. These folks are likely to have a good pulse on what can and can’t work, and the people who do have rank and title will likely trust them and rely on their good judgment when making decisions. If you can get their buy-in, you’re making good progress.

**Keeping Your Topographic Map Up to Date**

I talked earlier about how important it is to keep your locator map up to date. Keeping your topographic map fresh is even more important. The facts on the ground will change quickly, and things that you think you know will stop being true. On an average day, you might need to know that:

* A team you depend on has a new lead.
* A project you’ve been waiting for isn’t happening after all.
* Quarterly planning is about to start.
* A useful new platform is launching.
* Your product manager is about to go on extended leave.

There’s a lot of information to keep up with. But you need to know it all, so you need to know what to look for. Here are some ways you can stay up to date:

*Automated announcement lists and channels*

Dedicated channels for sharing new design documents, announcing outages, or linking change-management tickets give everyone an easy high-level view of what’s going on. If these kinds of channels don’t exist and you’d find them useful, consider creating them.

*Walking the floor*

The [Lean manufacturing](https://oreil.ly/zfdHj) folks talk about *gemba,* the idea of walking the manufacturing floor and seeing how things actually operate. Find some avenues to stay attached to the work that teams around you are doing. This could take the form of pairing on occasional changes, managing incidents, or doing a deploy for a system you want to know more about. Drifting too far from the technology doesn’t just reduce your context; it can reduce your technical credibility. (More on that in [Chapter 4](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch04.html#finite_time-id00004).)

*Lurking*

I asked on [Rands Leadership Slack](https://oreil.ly/O4bad) about how everyone approaches knowing things, and a common thread was paying attention to information that isn’t *secret* exactly, but isn’t necessarily for you. This included reading senior people’s calendars, skimming agendas or notes for meetings you’re not in, and—something that had never occurred to me—looking at the full list of Slack channels sorted by most recently created so you can see what new projects are happening.

*Making time for reading*

In companies with a mature documentation culture, plans and changes will often be accompanied by RFCs, design documents, product briefs, and so on. Skim for some basic context, or schedule time on your calendar to read deeply.

*Checking in with your leadership*

You need allies and sponsors who will tell you things. Check in often enough to hear behind-the-scenes updates and to make sure the way you’re thinking is still aligned with the way your leaders are.

*Talking with people*

Stepping out for coffee and a chat isn’t just pleasant relationship-building—it’s a great source of context. If you really want perspective, talk to people outside engineering: product, sales, marketing, legal, and so on. If you’re creating a product, befriend your customer support folks: they know more about what you’ve created than you do. Befriend the admin staff, too. Admins are smart, resourceful, and well connected. They know what’s going on, and they tend to be the most fascinating people in the company. Go make friends.

**“I DON’T KNOW HOW TO TALK TO PEOPLE”**

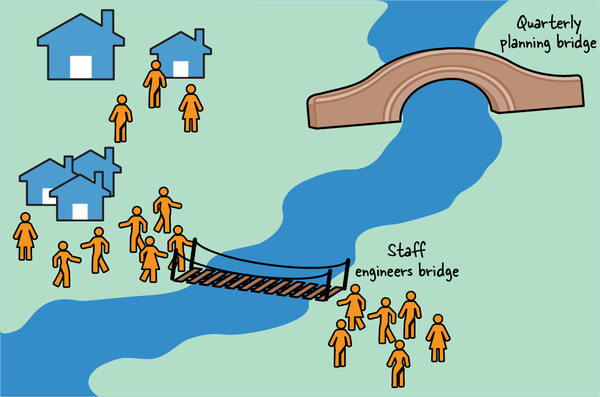
Many engineers have an aversion to anything that smells like “networking.” It makes us think of smarmy ‘80s power lunches. (Or is that just me?) But networking doesn’t have to be cynical or grubby. If you get to know people and are friendly, sharing information and helping each other will follow.

If you’re struggling to begin a conversation with someone, an easy starting point is to ask a question, take an interest in what they work on, or (genuinely) compliment something you admire about them.[**7**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn26) Most people are interested in talking about their work or their priorities, and most will be happy to explain how something they’re interested in works. Small talk is a learnable skill that will pay dividends throughout your career. (And if you’re talking with someone more junior than you, it’s kind of your responsibility to make it not awkward.)

**If the Terrain Is Still Difficult to Navigate, Be a Bridge**

The problems that slow down tech organizations are most often human ones: teams that don’t know how to talk to each other, decisions that nobody feels empowered to make, and power struggles. These are difficult problems! As you add information to your topographical map, you may find places where it’s tempting to scrawl “There be dragons” and vow to steer elsewhere. But a staff engineer can often have the most impact by going where everyone else fears to tread and making the dangerous territory easier for everyone else.

The Westrum model highlights the importance of “bridging” (see [Figure 2-11](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#when_quarterly_planning_is_a_long_way_o)), making connections between parts of the organization that otherwise would have enormous information gaps. The more you know the terrain, the easier it will be to bridge gaps by sending the email summary nobody is sending, introducing two people who should have spoken a month ago, or writing a document to show how projects connect to each other.



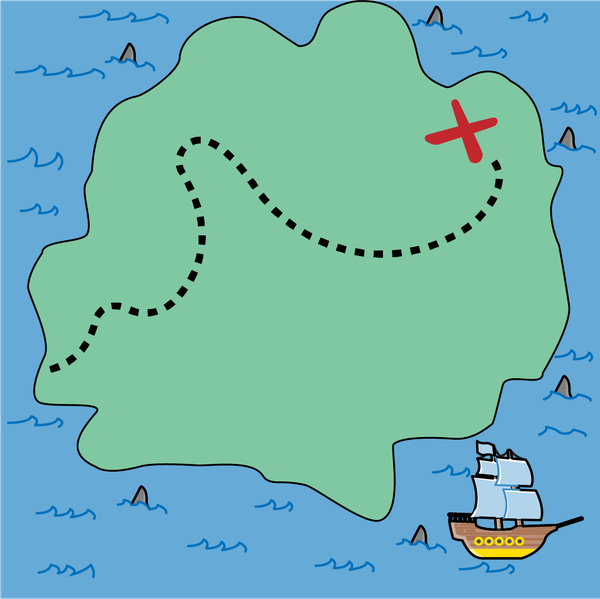
**Figure 2-11. When quarterly planning is a long way off, staff engineers can build connections to bridge the gap between two orgs.**

[Google’s DevOps site](https://oreil.ly/DU4Nc) suggests preemptively building bridges: “Identify someone in the organization whose work you don’t understand (or whose work frustrates you, like procurement) and invite them to coffee or lunch.”

When you can, define the scope of your job so that it crosses the tectonic plates and encompasses *all* of some system or problem domain, not just the part belonging to a single team. That way, you can catch work that is getting dropped, mediate conflicts, and help create a single story about what’s happening. When there are major changes proposed, you’ll have enough context to say “Yes, this migration is a good idea,” or “No, we have work to do.”

**The Treasure Map: Remind Me Where We’re Going?**

We’ve drawn two maps so far. The *locator map* shows where we are. The *topographic map* shows how we can navigate across the organization. But where are we going? That’s the purpose of our third map (see [Figure 2-12](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#x_marks_the_spot_where_the_treasure_is)).



**Figure 2-12. X marks the spot where the treasure is buried! Now you just need to get there.**

The *treasure map* gives us a compelling story of where we’re going and why we want to get there. Let’s go on an adventure!

**Chasing Shiny Things**

I talked earlier in this chapter about how you need to look past your group’s local problems and keep perspective about the world around you. You need that same perspective across *time*. It’s easy to overfocus on short-term goals like the current feature release or the latest unhappy App Store review. But think bigger. Where are you trying to get to? Why are you doing any of this? To be clear, I’m not saying you *shouldn’t* look for short-term wins. But thinking *only* about short-term goals can be limiting. If you’re only thinking short term:

*It’ll be harder to keep everyone going in the same direction*

If the team doesn’t know the big plan, either they’ll go to the wrong place, or every decision will be long, complicated, and full of discussion. Sometimes navigating around difficulties will mean taking an indirect path. Everyone should be very clear about the course correction that will need to come after that milestone.

*You won’t finish big things*

If your team keeps focusing on short-term projects to solve local problems and pain points, you won’t be able to solve bigger, long-term problems that take multiple steps. The value of your existing projects might not be clear to people outside the team either.

*You’ll accumulate cruft*

If teams don’t know where they’re going, they have two options. They can try to be flexible enough to support every future state, creating solutions that are overcomplicated and hard to maintain. Or they can make local decisions, taking the risk that their direction won’t match everyone else’s and that their solution will be a weird edge case that everyone else has to work around.

*You’ll have competing initiatives*

In an organization that relies on grassroots or bottom-up initiatives, there might be multiple people trying to rally enthusiasm around completely different directions. They’re all trying to do the right thing and get people aligned, but the end result is chaos.

*Engineers stop growing*

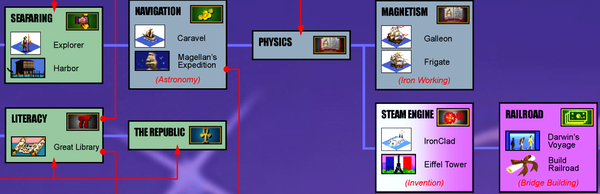
Focusing only on short-term goals limits the way you think about and frame your work, and how much ownership you take of the work that falls into the cracks between tasks. If the team is trying to achieve a big project, they’ll have to identify the gaps between the assigned tasks and figure out how to fill them, building skills in the process. A team that’s used to iterating on short, clearly specified goals won’t build muscle for bigger, more difficult projects and won’t be able to tell the story of why they did what they did.

**Taking a Longer View**

If everyone knows where they’re going, life gets easier. There’s no need to keep tight alignment along the way. Each team can be more creative in figuring out their own route, with their own narrative for the problems they’ll need to solve to get there. They’re less likely to go down wrong paths, and they’ll have enough information to make decisions, reducing the amount of hedging and technical debt they need to incur. They can celebrate the wins along the way, while remembering that there is a long-term goal and that the real celebration won’t happen until they get there.

**Why are you doing whatever you’re doing?**

An analogy I use a lot is the technology tree that you see in many strategy games, such as *Civilization*.[**8**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn27) In case you haven’t, uh, *invested* way too many hours of your life on this excellent game, I’ll explain how it works. You play as the ruler of a civilization, trying to build an empire. Your path to greatness includes amassing scientific knowledge, so as you go along you can choose to research various technologies. The set of available technologies form a directed graph (see [Figure 2-13](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#the_technologies_graph_in_civilization)). At the beginning you might research, say, pottery and hunting, but as you go through the game, your skills will build on each other. In *Civilization*, you can’t build a railroad without researching bridge building and steam engines. And you can’t build steam engines without physics and engineering. So there’s going to be a point in the game when you’re researching physics but your *actual goal* is to build a railroad. You won’t have the real win until you’ve built the bridges, researched the steam engines, and ordered little hats for your train conductors. (That last bit doesn’t really happen, unfortunately.) Unless you remember where you intended to go and keep working on it, you don’t ever get to ride the train.



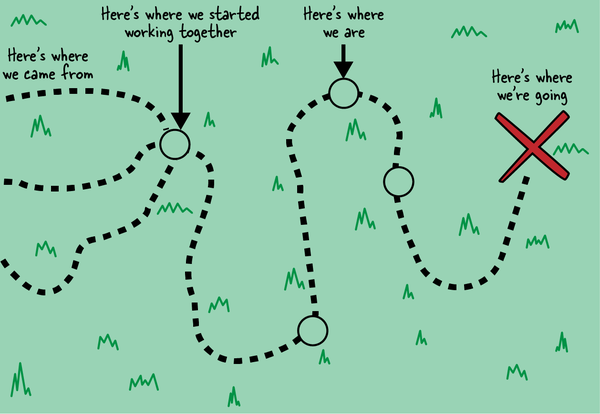
**Figure 2-13. A section of the technologies graph from Sid Meier’s Civilization II, created by Microprose/Activision (source:**[***http://www.civfanatics.com***](http://www.civfanatics.com/)**).**

When you’re choosing a technology to invest in, often that’s because it’s an unavoidable step on the path to something else. You’re not building a new service mesh for the joy of building a service mesh; you’re building it to make your microservices framework easier to use, because you want to make it easy for new services to get set up quickly, because you want teams to be able to ship features faster. The real goal is to reduce time to market. When you know the real goal, you can step back and evaluate whether any proposed work will get you closer to it.

**Sharing the map**

It may take you time to dispel the fog of war and uncover the true destination of your journey. Once you do understand it, don’t keep it to yourself. That means telling the story to other people and letting them understand why it matters. Your story should show where you are, where you’re going, and why you’re taking the steps that you are along the way. If there are sea monsters or shortcuts to know, you’ll probably want those marked—but don’t include any distractions. Make it easy to see what’s going on. The map should describe the treasure—that is, give a clear definition of success—so that everyone knows what they’re aiming for.

It’s motivational to see that you’re making progress, but it’s also surprisingly easy to forget where you were and maybe even feel like you’re getting nowhere. As the person with a map (see [Figure 2-14](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#tell_the_story_of_where_you_were_and_ho)), you’re well positioned to show that everyone is getting closer to the goals (or course-correct if not). Tell the story of where you came from as well as where you’re going.



**Figure 2-14. Tell the story of where you were and how much progress you’ve made as well as where you’re going.**

**If the Treasure Map Is Still Unclear, It Might Be Time to Draw a New One**

If everyone’s working from the same treasure map, your job here is done. But if you discover that there are multiple competing paths or no plans at all, you might need to help the group choose a destination. Sometimes all you need here is to write up a short summary of where you see confusion or misalignment. By spelling the facts out and sharing them, you’re forcing the conversation (or, perhaps, the argument) into the open. But, after asking all of the questions, tracing the *Civilization* tech tree, encouraging the people who disagree to talk with each other, and thinking really hard, you might still conclude that no one has actually chosen a long-term destination yet, or that there are multiple competing destinations.

In that case, there’s nothing more to be gained from clearing the fog of war from the map: it’s time to create a new map. That’s what the next chapter is all about.

**Your Personal Journey**

Before I close this chapter, let’s talk about *your* journey. As a staff engineer, it can take longer to see the impact of your work. That means it’s harder—but also more important—to tell the *story* of that work. When you look back, you should have a narrative of what you were trying to achieve and how it went. What did you and your group accomplish together? When you look ahead, you should have a story too: what are you trying to do? How does your current work contribute to that goal?

Once you have that narrative, even small tasks become part of a bigger story. Any given week’s work might not be elucidating, but what did you do this month, this quarter, this year? Are you getting closer to some treasure? To understand your journey, you’ll need one more map, the *trail map.* We’ll draw that in [Chapter 9](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch09.html#whatapostrophes_nextquestion_mark).

**To Recap**

* Practice the skills of intentionally looking for a bigger picture and seeing what’s happening.
* Understand your work in context: know your customers, talk with peers outside your group, understand your success metrics, and be clear on what’s actually important.
* Know how your organization works and how decisions get made within it.
* Build or discover paths to allow information you need to come to you.
* Be clear about what goals everyone is aiming for.
* Think about your own work and what your journey is.

[**1**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn20-marker) A popular metaphor, the boiling frog, says that if you drop a frog into a pot of boiling water, it will jump out, but if you put a frog into cold water and very gradually increase the temperature, you can bring the water to a boil and kill the frog. It’s often used as a cautionary tale to illustrate that gradual change can become normal and that we can slide into catastrophe without reacting to it. I was so relieved when I learned that real frogs don’t behave like this: they just jump out! Let’s leave the poor frogs alone, but the metaphor is useful.

[**2**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn21-marker) That’s also why design documents should have an “alternatives considered” section; we’ll talk more about design docs in [Chapter 5](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch05.html#leading_big_projects).

[**3**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn22-marker) Shared-interest Slack channels, social clubs, and employee resource groups (ERGs) can be fantastic ways to get to know people and make connections across the organization. Shoutout to my friends on the #crosswords channel at work who share their New York Times Crossword times every day and the #women-in-engineering channel participants who celebrate every member’s successes.

[**4**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn23-marker) I can recommend discussing landforms with a fifth grader, if you have one in your life. They’re well adapted for questions like “What would a fjord be if it was a metaphor for humans trying to work together?” (Two teams worked together to make a big project—a glacier—but they got angry with each other, the project melted, and all that’s left is the water at the bottom. Now you know.)

[**5**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn24-marker) The unspecified “they” often works as a keyword to alert you that you’re operating without enough information. If you find yourself having a thought like this, double-check who you mean by “they.” If it’s “the whole organization,” then that’s part of your problem. Understand exactly who you need to convince. I’ll talk more about the official deciders and the “shadow” org chart later in this chapter.

[**6**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn25-marker) [“Coordination Headwind: How Organizations Are Like Slime Molds”](https://oreil.ly/n2nNf) is a fantastic presentation about the failure modes of bottom-up coordination.

[**7**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn26-marker) Compliment something you admire, but please don’t tell your coworkers they’re attractive! In general, only compliment something that the person made a choice to do. A well-written RFC, a smoothly run meeting, or a cool desk toy are all fair game.

[**8**](https://learning.oreilly.com/library/view/the-staff-engineers/9781098118723/ch02.html#ch01fn27-marker) *Civilization*, now in its sixth edition, is a strategy game that has been around for decades. It uses a fog of war, and you have to make good decisions between long-term and short-term investments. I recommend playing *Civilization* to understand all things about staff engineering. Tell your boss it’s research.