Chapter 13. Mastering Soft Skills in the Tech World

> When dealing with people, remember you are not dealing with creatures of logic, but creatures of emotion.
>
> Dale Carnegie in *How to Win Friends & Influence People*

Every skill you learn has a shelf life, something you should consider when allocating your precious time. You've probably figured out by now that technologies are constantly changing; APIs evolve and are replaced, approaches that were best practices in a previous version of a language are obviated by a new feature. If it seems like something becomes irrelevant just as you start to understand it, you're not wrong. And you're not alone.

Regardless of your path to becoming a software engineer, you probably focused on developing your technical skills. After all, they are fundamental to the field; it's pretty hard to write code if you don't understand programming languages. To progress in your early career, your focus tends to be on growing your technical toolkit, learning more frameworks, becoming proficient with a cloud provider, and staying on top of the latest advancements in your programming language of choice.

However, one set of skills will last you your entire career: the soft skills many engineers tend to ignore. Learning how to work with others and communicate clearly is just as important to your success as mastering the next language or framework. Human beings don't change as quickly as technology; it takes millennia to update our operating system. That's why soft skills never go out-of-date. Learning how to communicate effectively, work with and influence others, and manage your time pays dividends for your entire career.

Time spent learning a skill that will last you 30 or 40 or even 50 years is a pretty good return on your investment. Obviously, you still need to keep up with changes to your technical toolkit; to set yourself apart, don't neglect more evergreen skills. It may seem daunting, but ultimately, developing strong soft skills mostly boils down to developing good habits. Let's dive in!

Collaborative Communication

Some people enter the technology field today with the expectation that doing so allows them to avoid some of the messier aspects of human interaction. The stereotype of developers as introverted loners can be traced back to the early days of computing when projects were smaller and often driven by individuals. Go far enough back, and the word *computer* meant a person who performs mathematical calculations. Companies didn't have legions of software engineers working on multimillion-line codebases. These days though, if you want to thrive in your career, you must master the art of communication.

That doesn't mean you must be an extrovert to succeed in software; in fact, many conference speakers are themselves introverts! While it may require more effort for some than others, software projects are team sports. With the sheer size of most applications today, the era of the "lone wolf" developer is over. No longer can one individual hold the entire codebase in their head; modern codebases require teams working together.

Even the way teams work has evolved. Instead of people toiling away in windowless cube farms, many teams work in a project room setting. Though very effective for collaboration, project rooms can be exhausting for introverts. Some developers will actually block out some alone time on their calendar. Taking a break away from the project room for some solo time gives them a chance to recharge their social batteries. You may need to have a conversation with your manager, but don't be afraid to advocate for what you need to be successful.

Software is a collaborative endeavor that requires you to utilize an array of techniques to work effectively with your team. As Kent Beck once said so eloquently, "Software design is a human process…done by humans for humans". While it may not come as naturally to you as picking up a new programming language, mastering technical communication is something you can learn. Communication involves more than just words coming out of your mouth: it means picking the right communication channel; preparing for enterprise operator; and learning how to communicate up, down, and across your organization.

Communication Channels

Of course, you won't communicate only via a programming language, and you face no shortage of communication channels. They range from warm to cold, personal to impersonal, high touch to low touch. Some produce a record of the encounter; others allow for plausible deniability. You have a veritable plethora of options, so choosing the proper approach is vital. Your challenge is to choose the right method at the right time, which is easier said than done. It can help to visualize those various channels as in Figure 13-1, the Communication Continuum.[1]

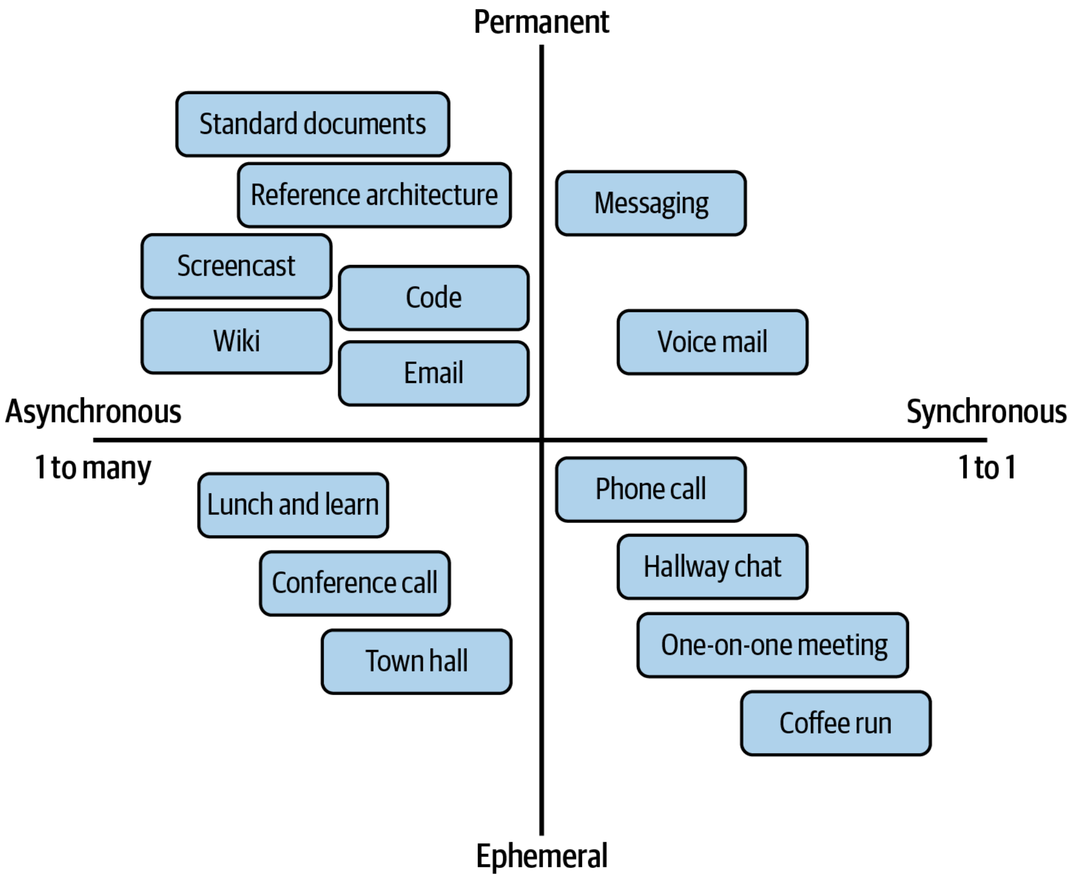

Figure 13-1. The various ways you can communicate with your teammates

There are times when having a record of a discussion is critical. Perhaps there is a critical vulnerability that requires you to upgrade a key component of your application. Ensuring that everyone is aware of the problem and the resolution can prevent your organization from facing a crippling hack.

You should also consider your audience when choosing a communication channel. Some people live on email; others haven't checked their inbox in months. Your organization may run on a corporate message tool, while others believe everything should happen in a meeting. Understanding how information flows in your world will guide your decision making. Covering each and every component of communication is beyond the scope of this book. Let's explore some of the most common options and those that are most important for your career growth.

Messaging

Regardless of your organization's stance on remote work, you will likely spend a significant amount of time dealing with a corporate messaging tool like Microsoft Teams, Slack, or Google Chat.[2] Messaging tools allow you to easily send asynchronous text messages to individuals or groups, and many allow you to include files, code, images, GIFs, or just something you can copy and paste instead of having to retype. Messaging is usually informal and excellent for quick questions or a heads-up about an issue. The asynchronous aspect of messaging is one of its primary strengths; you can ping a teammate knowing they can respond when it's convenient for them.

Depending on how flat (or not) your company's organizational chart is, messaging is generally appropriate for your immediate team. However, that may or may not extend to more senior leaders. Many companies live on messaging these days. Even if your entire team is in the same physical location, messaging someone sitting near you is incredibly common, whether to tamp down on office chatter or to avoid interrupting someone. It can also be a simple way to offload specific details, such as a function that needs to be refactored in the next commit, from your short-term memory.

Because of its asynchronous nature, your recipient may not see your message in a timely fashion. Setting team standards around messages may be useful, but be mindful of people's time. Even a "quick question" can blow up a person's morning; pay attention to people's notification settings, focus time, and meeting entries.

Again, soft skills are ultimately a set of good habits. For example, if you're swamped but someone messages you a question, try to respond in a timely fashion even if it's just to say your plate is full today but you'll have some time to chat later in the week. Remember to apply the Golden Rule (as discussed in Chapter 1). Treating others the way you'd like to be treated is one of the simplest things you can do to advance your career.[3] Promotions aren't based solely on a mathematical formula or earning a certification; they often reflect how you are perceived in an organization. What other people say about you when you aren't in the room is often a deciding factor. You want people to have a positive view of you and your work.

Despite your organization's retention policy, you should assume anything you type into a messaging tool is logged and can be forwarded or retrieved later by lawyers or executives. Corporate culture often shines through in policies and practices around a messaging tool. However, it is generally not a good idea to use it to workshop your new standup material. You should assume anything you type into a corporate messaging tool can be retrieved later.

Finally, remember that tone doesn't always transfer in a written medium, and your reader may miss the nuance you intended. You may write something to be sarcastic or as a joke, but your reader may take it literally and seriously. In other words, it may sound different outside your brain. Err on the side of caution.

Meetings

Surveys routinely show that developers spend a significant amount of time in meetings. Not coincidentally, if you ask the average software engineer what they like least about their job, the answer "too many meetings" is likely in the top five. That said, sometimes meetings are the fastest path to resolving an issue or communicating a new initiative.

Meetings are synchronous and ephemeral, though they are often recorded for later viewing, and slide decks may be shared. A meeting can have any number of purposes, ranging from a morning standup to a one-on-one with your manager to an all-hands meeting to discuss recent financial reports. They can be the fastest route to a solution, but people still may not be on the same page afterward.

An Hour Can Cost You Thousands

Nate here. For part of my career, I had a standing two-hour meeting every Thursday morning. A quick glance at the stock attendee list showed more than 200 names, all senior technologists within the company. A rough calculation tells you this meeting routinely cost the organization thousands of dollars a week. But it was probably worth the expense; it was all about changes to the environment, from applications performing routine deployments to bringing new infrastructure online.

Making sure everyone was informed minimized disruptions and allowed people to interject if a given change had a possible impact that hadn't been spotted. Luckily, this meeting was also incredibly well run. The organizer sent out a detailed agenda every week, allowing people to step in and out as needed. Just because a meeting can be costly doesn't mean it isn't worth the time and money.

Odds are you'll find yourself double or triple booked at some point. Sometimes meetings are sent to very broad distribution lists, or your role means you'll be looped in by default. In some cases, the organizer isn't specifically including you or requesting your input; you're just on the list. Your time is valuable, and it's perfectly reasonable to be deliberate with meeting attendance. Don't be afraid to reach out to the organizer and clarify your role in the meeting or consult with your manager, who should be able to provide guidance.

An agenda is a minimum courtesy in any meeting, but asking what you are there to speak to can be clarifying. Meetings have a variety of purposes: what is the goal of this one? Are you required to be there the entire time? Would an email suffice?

You will eventually find yourself in *the* meeting, the one with the CTO or the VP of Engineering, the big client, or some other "very important person." While it is often a sign of career growth, these meetings can be intimidating. Don't be afraid to ask the organizer some questions! If this is a recurring meeting, ask regular attendees as well as the other people on the invitation for background on how things work and how the meeting tends to flow. Is it a collaboration session where the boss is looking for input, or is the purpose just to pass out marching orders? Should you feel free to interject a question or comment, or is there an implicit wait-to-be-called-on rule? Some senior managers have certain *expectations* that may not be common knowledge; without insider information, you may inadvertently make a bad impression with the person who controls your next promotion.

In many contexts, recruiting allies such as your manager or a senior technical person can be incredibly helpful, especially if it is the first time you've interacted with leadership. Sometimes just having a friendly face is enough to calm the nerves. These allies can also help clarify if your remarks aren't quite landing with the boss, and they can also give you personal feedback for future meetings.

Do your homework before the meeting. If there are specific points to review, block out time to read them before the meeting. Whether you're in person or on a video call, be sure to pay attention to nonverbal cues—furrowed brows, crossed arms, smiles, etc. Try to determine the power structure for the meeting: who is the decision maker? Are they in the meeting or are they external?

Even with return-to-office mandates, most organizations have multiple sites, meaning you will spend some of your time on audio or video calls. Virtual meetings have etiquette too. Depending on your culture, virtual meetings may have implicit "rules" about turning on cameras or ensuring that your background is tidy. Invest in a good headset, and if you get a new one, take it for a test spin before you hop on a call with the CTO; many headsets allow you to adjust the sensitivity of the microphone. Adjust accordingly. It should go without saying, but if you're not speaking, mute your microphone, especially if you are eating.[4] Few things are quite as annoying as hearing a person finishing their lunch.[5]

Back-channel communication via your corporate messaging tool can be useful during many meetings. Whether it's to ask a clarifying question or inject a comment without interrupting, having a conversation about the conversation is quite common. In some instances, you may need to use messaging as a way to assert yourself in a meeting.

Make sure as well that you know exactly *who* is in a meeting. Is it just IT people? Are there customers? Senior management? Don't assume! Modern video conferencing technology has largely limited the anonymous lurker, but it still pays to know who's on the call.

You will run meetings as well. Practice good meeting hygiene. Have an agenda. Keep the audience to those required to be there. If possible, ask a colleague to take notes. Just because the typical corporate calendar client defaults to 60 minutes does not mean you have to schedule all of your meetings to 60 minutes. As you've no doubt experienced, many meetings expand to the size of their container; how many times have you heard some variation of "This shouldn't take the entire hour"?[6] Constraints can be freeing: scheduling a meeting for 30 minutes forces you to stay on task. And, if you booked an hour but you've covered everything in the first 30 minutes, give everyone some time back; don't fill the remainder with fluff.

Tip

Many calendaring systems do allow you to change the default meeting interval. Don't be afraid to try something "radical" like 50 minutes for longer meetings and 25 minutes for shorter meetings. You might also consider starting meetings 5 or 10 minutes past the typical top or bottom of the hour to allow for travel time or just the inevitable meeting overrun. This will help you focus, and your attendees will appreciate having time to hit the restroom or grab another cup of coffee before their next meeting. Some organizations have globally modified the event settings, but there's nothing that says you can't do so locally for your meetings.

Be respectful by starting the meeting on time![7] Inevitably, some meetings will run long, or people will need a bio break, but do your best to keep the trains on schedule. Be mindful of lead times: if you're asking someone to review a complicated design, make sure they have a chance to look over the materials. If you're booking rooms for people in a different location, you may want to check with them about what is actually nearby; calendaring systems aren't always up-to-date when it comes to booking physical resources.

As the organizer, you should work to keep the meeting on schedule. You may need to curtail tangents, and some discussion points may require a follow-up meeting. Also, work to keep everyone involved: circle back to people to make sure they have an opportunity to contribute. Meetings may not be your favorite activity, but unless you're a team of one, they're unavoidable. That doesn't mean they have to be pure misery.

Presentations

Ask people about their fears, and public speaking is bound to come up. Many people hate presenting, but if you want to grow in your career, you will have to deliver some talks. While you may never become a regular on the conference circuit, you need to be comfortable presenting to various audiences.

Presentations take various forms and aren't always hour-long blocks. Consider having a short, one-minute "elevator pitch" at the ready. You never know when you'll have a quick, impromptu chance to chat with a senior leader in your organization.[8] Shorter talks are often more challenging; try not to bury the lede, and focus on the key point you need to get across.

You may be asked to give a short, 5- to 10-minute presentation to an audience beyond your project team. It could be a project update to the engineering organization, the architecture team, or the CTO’s direct reports. Learning to shape your message to resonate with a given audience is key: you may have to adjust the content to meet them where they are.

There are no shortcuts to improving as a presenter. If you want to improve, you need to give a lot of presentations. Consider volunteering to present at a local user group or meetup; many of the speakers you know and admire got their start in their own backyards. Giving a presentation is also a great motivator for learning something new, as the commitment acts like an immovable wall, forcing you to be ready to present.

Practice is essential. Dry runs in front of your pet will give you a sense of timing, while enlisting a test audience can give you vital feedback.[9] Presentations, like code, evolve over time; you often discover a nugget you weren’t even looking for. You’ll discover that the real value of the talk lies somewhere you weren’t expecting. For example, Nate once created a talk comparing React and Angular, but after the third or fourth delivery, he realized that he could extract from it a more generalized talk about comparing and contrasting *any* technology.

There are any number of excellent resources for improving your speaking skills. Many cities and some large companies have one or more Toastmasters chapters. Some chapters are even focused on more technical presentations. While the Toastmaster curriculum can teach you important techniques, having a safe space to practice the craft is key to learning and improving.

There are many excellent resources on the topic as well. One of your authors co-wrote *Presentation Patterns* and also recorded several videos you can find on the O’Reilly learning platform. *Resonate* by Nancy Duarte and *Presentation Zen* by Garr Reynolds are also excellent.

Code as a communication medium

As a software engineer, code itself is one of the most common ways you will transmit information to other developers. In the technical world, having good communication skills extends to your ability to write clear and effective code. Code is the ultimate source of truth, and it is vital that you can convey your meaning to other developers.

As you learned in Chapter 3, it is crucial to craft code with the human reading it in mind. Using good naming practices, keeping code concise, and avoiding clever code are crucial for the developer who follows you into the codebase.[10] Optimize for the human reading your code.

> Any fool can write code that a computer can understand. Good programmers write code that humans can understand.
>
> Martin Fowler, British software developer, author, and international public speaker on software development

Virtually every document on a software project is outdated shortly after it is written. But not the code. The code itself is the source of truth;[11] it is the most up-to-date documentation you have. From tests to following good coding practices, take the time to ensure that your code communicates clearly and effectively.

Communication is a massive topic, and you’ve just scratched the surface. As your career progresses, you’ll learn and master different parts of the communication continuum. Being an effective communicator is important for your career. Though it may not all come easily or naturally, invest the time to learn these skills, and future you will thank you!

Enterprise Operator

At school or a party, you may have played the game Telephone or Operator. Sitting in a circle, one person whispers a phrase to the person to their right, who then repeats the phrase to the person to their right, and so on until the last person, who says the message out loud.

Unsurprisingly, the message shifts as it goes from person to person. Perhaps someone doesn't enunciate, or the listener interprets what they hear instead of repeating the message verbatim. Perhaps one person isn't familiar with the phrase and inserts something they are more used to.

The same thing occurs in organizations on a shockingly regular basis. Sometimes someone misspeaks in a meeting, and other times people just hear what they want to hear. Regardless, at some point, you may unwittingly find yourself in a game of *enterprise operator*. Even with all the best intentions and your considerable communication skills, you cannot avoid it, but you can be prepared for it.

Enterprise operator is particularly common around enterprise standards. Don't be surprised if you hear "Alice said we're out of compliance" or "I heard Foo is the corporate standard." Is there actually such a standard? Does it apply to your application? Don't be afraid to bring proverbial receipts.

What should you do if you find yourself in a game of enterprise operator? First, keep your wits about you and recognize that it's happening. There will often be a lot of noise and some people in a less than sanguine mood. Second, try to identify where the message was garbled. Did someone get only part of the story? In many instances, knowing where the communication went sideways will pinpoint where you need to focus your remediation efforts.

Third, figure out who is involved. Is there a stakeholder you weren't aware of that you need to keep in the loop? Proactively reach out and discuss the issue with them. Lastly, perform a retrospective on the situation. Is there anything you could have done differently to prevent it?

Dealing with a Hostile Room

Nate here. Several years ago, I created a best practice around client-rendered user interfaces; it was extremely common across the industry at the time. With the proliferation of smartphones, tablets, and other modalities, applications couldn't be built for one specific monitor resolution anymore. Instead, applications evolved to a series of services that acted as JSON pumps while building the UI that was appropriate for each client. Though not controversial, the best practice still required a significant amount of teaching and presenting to make sure our portfolio was following the approach.

A few months later, the chief architect of one of our divisions asked if I would come talk to his team about the best practice, as the team members had some concerns. He set up some time on my calendar, and I dutifully ran them through my presentation on the topic. When I stopped for questions, one person was clearly irate, saying that I was asking them to completely redesign their application to use Windows MVC and how unreasonable that request was.

His response surprised me as I'd made no comment on specific implementation technologies. I asked them to walk me through their architecture, and it turned out they were already about 90% of the way to complying with the best practice.

Digging further, it turns out one of their business partners had heard one of my presentations on best practices that mentioned JavaScript MVC libraries. They didn't know what that was but interpreted it as Windows MVC and were worried the application was violating a standard. Once we cleared that up, the meeting became quite positive.

Know Your Audience

As a software engineer, you will spend a significant amount of your time communicating with other technical people, be they fellow engineers, architects, testers, or product managers. As you progress in your career, though, you will have to be comfortable sharing technical concepts with less technical stakeholders. Learning to translate from developer to business speak is an important skill that requires practice. You need to know your audience and adjust your messaging accordingly.

As proud as you may be of your in-depth knowledge of arcane technical trivia, audiences outside the project room are rarely impressed by your command of jargon.[12] Learn from sales engineers: success in their jobs requires the ability to describe deeply technical products in a way that resonates with their specific audience, tailoring their message appropriately. Some vendors are very good at "speaking exec" (in other words, tying technology back to solving business problems), a skill you should learn. Decision makers are rarely swayed by "It's a cool new technology," so learn to articulate *how* that cool new technology will deliver business value.

How you communicate matters a great deal. While it may be tempting to tell someone in your management chain to "read the manual," cynical, derisive personas rarely enjoy the career progression of those with a less confrontational approach.

It's also important to understand and elucidate the business value of software. Do you understand the domain you're working in? Do you know what your business partner's top concerns are?

Practicing Influence

> Leadership is influence, nothing more, nothing less.
>
> John Maxwell, American author and speaker

Influence is the art of getting someone to do what you want them to do, ideally with them thinking it was their idea. As you learned in Chapter 9, too many think they can just issue a command, but unless you are a founder or the CEO, you won't be able to just order people to do things. Instead, you're going to have to master the subtle art of influence. From deciding which framework you should use on your next project to which snacks should populate the break room, there is no shortage of opportunities for you to practice influence. But how do you get what you want if you aren't the ultimate decision maker?

Your company has influential people, something that isn't always obvious by the org chart. If you're not sure who they are, take note of who gets the majority of the really interesting assignments. The best managers recognize the edge of their expertise and cultivate a set of experts they rely on for information and advice. While you may not have a direct line of sight to the CTO, you can likely work through someone in their orbit; influencing the influencers is very effective.

Understanding and Articulating Value

Start by articulating the benefits of your preferred option to the person you’re trying to influence. Seek out common ground. Shape your message based on the reality of your organization. Is the decision maker focused on time to market? Developer efficiency? Reducing costs? Don’t cut against the grain; show how your desired outcome helps them drive their mission home. For example, if you know the decision maker wants to reduce the drama from a release, align your arguments to how your approach delivers on that goal.

By the same token, understand the information ecosystem the decision maker lives in. What sources do they trust? Some leaders rely on a particular analyst organization, or they’re particularly receptive to articles from a given technology vendor. Cite those sources.

Strategic Approaches to Influence

It’s hard to convince people to change their minds if they disagree with you. There are two basic approaches: the hammer, where you order someone to do something, and the ninja, where you make them think what you want them to do was their idea. If you’ve ever been in a relationship with another human being, you know the former rarely works. You have to be subtle, to nudge.

Some resort to being a bully; they choose to yell louder in an effort to bend people to their will. That approach nearly always backfires. Instead, engage in a conversation! Don’t underestimate the challenge in front of you: it can be very hard to convince someone to change; it takes patience and perseverance. Don’t be afraid to start small, show success, and grow from there.

Your passion may be interpreted as aggression, so take time to listen to those you are trying to influence. What are they saying? What are their concerns? What resonates with them? Sometimes it is as simple as adjusting your phrasing to match their expectations.

Your reputation is just as important as the strength of your ideas. It speaks for you when you’re not in the proverbial room; do you know what yours is like? If you aren’t sure, don’t be afraid to ask. You may not love the answer, but it’s the only way you can change things.

Don’t be afraid to recruit allies. Reach out to others who share your stance and work together to achieve your goal. There is often strength in numbers, and another set of eyes may see an angle you don’t.

Who delivers the message can also be a key aspect of influence. Sometimes, your preferred viewpoint needs to come from, well, someone other than you.

The Messenger Matters

Years ago, one of your authors joined a new company and decided he’d “make his mark” by introducing test-driven development (TDD). He prepared and delivered an overview on TDD, how it would help reduce defects, and why it really wasn’t as hard as you might think. His message was not met with the warmth he was expecting.

However, one of his friends (let’s call him Steve) was eager to add testing so he worked closely with Steve’s team. Over the course of a few months, that part of the codebase rose out of the proverbial muck. After seeing how much of an impact it had on his team and their code, Steve essentially presented the aforementioned TDD talk again, nearly verbatim. This time though, the reception was different. It was met with near universal praise and determination to extend what Steve’s team had done to the rest of the codebase.

What was the difference? Everyone knew Steve, Steve had tenure, Steve had been at the company for years. Your author was “the new guy”; what could he possibly know? The messenger matters. Who will the decision maker turn to for advice? Is that person on the same page with you? Don’t be afraid to work through someone with more clout or influence in your company. While it may not be as satisfying as being in the spotlight, it can be incredibly effective.

How you communicate with your stakeholders matters too. You also need to be aware of what messaging works best within your organization. Some companies fixate on speed to market; others care about cost savings above all else. Shape your message accordingly. For example, if you’re working in a startup, emphasize how your approach will get features to customers faster. If you work in a more established firm and you know there are budget constraints, focus on how your idea will save money.

Wielding influence is not something taught to most software engineers. But learning how to do so can help you get things done, and it can advance your career. An important aspect of influencing people is understanding and managing stakeholders, which we’ll discuss next.

Stakeholder Management

You have to know who your stakeholders are and understand them in order to effectively communicate with them. Every project has multiple interested parties, from your teammates, to your management, to the people ultimately using the software you build. Some stakeholders are very obvious—if you’re working on the CTO’s number one initiative, they’re clearly going to be involved. However, some stakeholders may be more obscure.

Sometimes there are people who aren’t directly involved but who can absolutely help, or hurt, your project. You can often feel the presence of these level 2 players in comments like, “The VP will never go for that.” In some instances, the mere *idea* of these people can be disruptive. It is difficult to negotiate with someone who isn’t there; it may require getting them in the proverbial room to break a logjam.

Not all stakeholders are equal, and they often require different levels of your time and energy, depending on their interest level and power within your organization (see Figure 13-2). Low interest, low power won’t be your focus. But high-interest, high-power people are key to the success of the project.

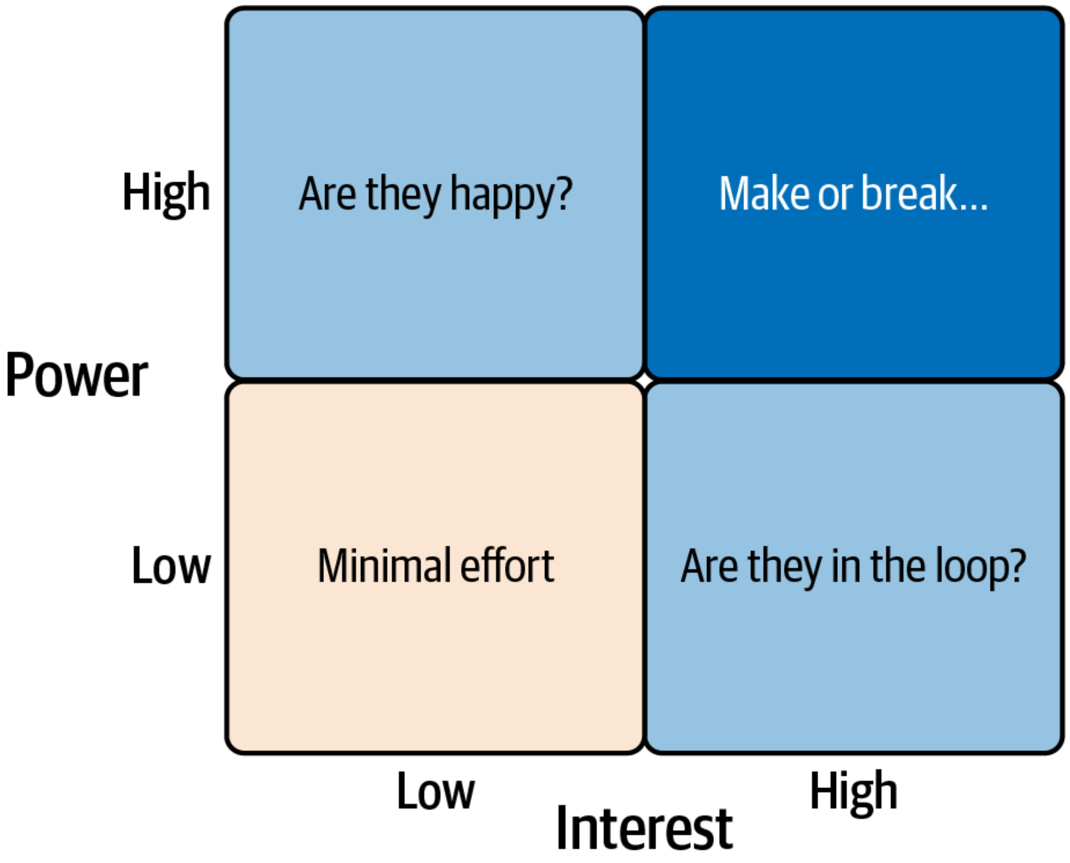

Figure 13-2. Understanding the power/interest matrix can make or break your project

Looking at the upper-left quadrant, you'll want to ensure that those stakeholders have the information they need to have peace of mind about your project. Respect their time and let them come to the information as they want or need it. Sometimes they'll quickly pivot from low to high interest. Transparency is key with high-power, low-interest stakeholders.

Moving to the upper-right quadrant, you'll find the people who are instrumental in successful projects. These are the people who can make or break an initiative. Never underestimate their voice when it comes to promotions and bonuses. High-power, high-interest people can be challenging to work with, and their jam-packed schedule can be an impediment. They cannot be ignored, and they will occasionally show up at the last moment and upend any number of things.

You must be proactive and engage with high-power, high-interest people. And you may have to adapt your working style to better fit their needs. Have a conversation about how you can best work together to ensure a successful project.

It's also important to note, unlike the command line, silence is not always golden. Keep them informed using whatever method works best for them. That could be regular demos, a high-level summary, or one-to-one meetings.[13]

Disregard those in the lower-right quadrant at your own peril. High-interest, low-power people can be extremely helpful. Use their passion to your advantage; their enthusiasm can be contagious. They will often be some of your best advocates within the organization, and they will often volunteer to help. They may have connections to some of the high-power, high-interest people as well. Whether it's testing a new feature, allowing you to shadow them for a few hours, or sitting for an interview, high-interest people are invaluable. If you ignore them, they may actively work against your project; keeping them in your camp can result in smooth sailing.

Despite all the zeros and ones, software is ultimately about people. Understanding the web of stakeholders who surround your project can be the difference between a failed initiative and a smashing success.

Time Management

Managing your time is vital. Time is a nonrenewable resource; it does not scale, and there's no way to get more. There is no shortage of demand on your schedule. From meetings, to code reviews, to dealing with a production issue, to filling out your TPS report, there's rarely enough time in the day. You must be the guardian of your own time. Again, don't be afraid to block out uninterrupted time to work of the most important things on your plate.[14] Give yourself some room to breathe.

You also need to be mindful of your personal rhythm: are you a morning person, or are you most alert in the early afternoon? As best you can, build your schedule around when you are at your best, performing more rote work when you know you're not as sharp. While you will need to be flexible, working *with* your cadence and setting reasonable expectations can make you happier and more productive.

Giving yourself some space to think can make all the difference in finishing a feature or fixing a defect. Rich Hickey gave an excellent talk on hammock-driven development, emphasizing the importance of uninterrupted time to think. It's obvious that fixing bugs found in production is the most costly approach; that's why many developers write automated tests, and software processes involve quality assurance steps. However, the cheapest solution is to never introduce the bug in the first place.

> "Most of the big problems we have with software are problems of misconception. We do not have a good idea of what we are doing before we do it. And then, go, go, go, go and we do everything."
>
> Rich Hickey, creator of the Clojure programming language

Problems of misconception are endemic in software. Terms are often overloaded and many seem like synonyms, only to be delineated by critical nuance. Blocking out your schedule can give you crucial time to think through the consequences of a design. Take the time it takes so it takes less time.[15] It's also important to understand that different roles have different types of schedules. There's a stark difference between those who spend their days making (like software engineers) and those that focus on managing.

Maker's Schedule

Software is closer to a craft than a science; it requires what Paul Graham refers to as a *maker's schedule*. A manager's schedule is just a series of hour-long blocks of meetings. Literally any hour is the same as another. Thus managers have a "find an open spot on my calendar" approach to scheduling. And that approach works for getting a consensus on a decision or working through a budget proposal. It doesn't work very well for building software, though.

As a software engineer, what unit of time do you typically think in? A full day? Half a day? Your day *isn't* a series of one-hour blocks. A 30-minute meeting at 10 a.m. could completely destroy your morning by breaking your time into blocks that are too small to do anything meaningful. There's no point starting anything hard because you'll just have to put it down and context switch out for the poorly timed meeting. You must be the guardian of your own time. That's the maker's schedule.

Software is ultimately about loading the problem into your brain, creating a mental model of the application. Think about debugging: you're setting breakpoints, rerunning the program, re-creating the issue, working through the internal state of the system, and then BANG. Someone interrupts you. You've stacked the stones in your mind, and now it's gone. How long will it take for you to return to that point?

All is not lost. You can take steps to protect your productivity. Use technology to your advantage, and don't be afraid to turn off interruptions. Close your email and messaging apps if you can't resist the song of the sirens. Tune notifications and leverage "do not disturb" settings to minimize distractions when you need to focus. Consider restructuring your week. Some organizations have set times or even days of the week that are free of meetings, allowing their engineers to focus on work. Scheduling your meetings for one or two days a week allows you to concentrate the disruptions, allowing you more focused time. If your company doesn't do something like that, propose an experiment!

Staying on Task

Part of managing your time is staying focused, which can be incredibly difficult to do. Between a never-ending flood of emails and instant messages, let alone the infinite distraction of the internet, staying on task can be a struggle. Sometimes that mental battle is an indication you're working on something that can be deferred, but don't be afraid to bring some structure to completing your to-do list! Having a plan or approach can reduce the cognitive load, ultimately turning focus into a habit. Instead of getting pulled in random directions, you give yourself a framework to get work done.

Consider adopting the Pomodoro technique. In a nutshell, it is Agile software development on a very micro scale. Pick a task to work on, set a timer for 25 minutes, and work on that task.[16] If you think of something else you need to do, jot it down and return to the task at hand. When the timer goes off, switch contexts and take a break. Rinse and repeat. At the end of four tasks, reward yourself with a longer break.

Other techniques, like the Focus Course, can give you tools to help you set and achieve your goals. Habits can be very powerful, as illustrated by the likes of James Clear. Don't be afraid to consume some high-quality content outside of the computer science section of the internet!

As a practicing software engineer, it can seem like there is never enough time in a day to accomplish everything on your agenda. Learning to proactively manage your time will help you focus on the important things and be more productive.

Wrapping Up

Soft skills may not be the first thing you learn on your journey as a software engineer. Effective communicators tend to get promoted: deploying the right techniques at the right time can make all the difference in achieving your goals. Managing your time will help you keep your head while all those around you are losing theirs. And while you may be more passionate about the latest technology, you can't afford to neglect the soft skills that will benefit you for your entire career.

Putting It into Practice

Practicing your soft skills can be done in any number of situations. Volunteer to deliver a presentation to your teammates or at a local user group. Proactively block out time on your calendar to work and think. Be more thoughtful the next time you schedule a meeting—include an agenda, invite only those required, and try scheduling it for a time frame other than 60 minutes.

Experiment with the Pomodoro technique for a few weeks. Did it improve your productivity? Before sending off yet another email, pause and consider whether a different communication channel would be more effective. Practice your influence skills by introducing a new idea, tool, or technique to your team.

Additional Resources

- Thinking Architecturally (report) by Nathaniel Schutta (O'Reilly, 2018)
- *How To Win Friends & Influence People* by Dale Carnegie (Simon & Schuster, 1936)
- *Influence, New and Expanded: The Psychology of Persuasion* by Robert B. Cialdini, PhD (Harper Business, 2021)
- "Maker's Schedule, Manager's Schedule" by Paul Graham
- "Hammock-Driven Development" by Rich Hickey

[1] There are also different communication styles.

[2] Whichever one you're required to use is probably the one you hate the most. Sorry about that.

[3] Alternatively, treat your coworkers as you would treat your manager.

[4] Or making snide comments about the meeting in question…

[5] Darth Vader–level heavy breathing might be worse.

[6] Of course, the appropriate question is "Then why did you book the entire hour?"

[7] More often than not, the simplest solution is the best solution.

[8] While you may not literally run into them in the elevator, you never know when you'll have 45 seconds at the start of a meeting to deliver your message.

[9] You may need to provide incentives for your test audience. Food works well.

[10] And never forget, you are often the person that follows you…

[11] That doesn't, of course, imply it is *correct* or meets the business needs, but that's a different problem.

[12] Most people recognize Spring as a season, not a popular Java framework.

[13] Yes, you might need to put a cover sheet on your TPS report.

[14] Some calendaring systems make it very easy to set up focus time along with core work hours.

[15] Alternatively, slow is smooth, and smooth is fast.

[16] Bonus points if it’s a pomodoro (tomato) timer.