ENGINEERING MANAGER'S COMPASS

Insights for Building Effective Engineering Organizations

DÜNYA KIRKALI & MAXIM SCHEPELIN

Engineering Manager's Compass

Dünya Kırkalı, Maxim Schepelin

Version v0.3, 18.02.2026: Pre-release

Engineering Manager’s Compass

Insights for building effective engineering organizations

Dünya Kırkalı and Maxim Schepelin

This book is available at https://leanpub.com/managers-compass

This version was published on 18.02.2026

Contents

Preface

New engineering managers often find themselves learning on the job by trying to replicate the behaviors of their best managers while avoiding the pitfalls of their worst. This trial-and-error approach can be slow, frustrating, and even damaging to career and reputation. **Engineering Manager's Compass: Insights for building effective engineering organizations** aims to change that.

About this book

We're engineering leaders who led multiple teams in companies of various sizes, from small startups to large enterprises. We learned through trial and error how to navigate the challenges of managing engineering teams. To get better at our craft, we have taken numerous training courses and studied hundreds of books.

As we grew professionally, we realized that having a repertoire of tools and techniques is important. Yet even more crucial is developing strong judgment grounded in a clear understanding of reality that helps you identify the right problems to solve.

Learning management techniques is easy. You can quickly learn how to run a team retrospective: what went well, what to improve, and what action items to assign. But knowing whether your team feels psychologically safe enough to be honest, whether the retrospective targets a symptom or the root cause, and whether people are too afraid to speak up—that's the challenging part.

Through hours of discussing management challenges, we realized that success in a leadership role depends on tacit knowledge—the nuanced understanding grounded in domain expertise and knowledge of company and team context. We set off to write a book about the subtleties of managing engineering teams.

The engineering management role demands technical expertise and the ability to navigate interpersonal dynamics, align technical and business goals, and foster sustainable team growth. If you've ever felt stuck, overwhelmed, or unsure how to move forward as a leader, you're not alone.

As an engineering leader, you may often wonder why your manager or skip-level manager prioritizes certain issues while ignoring others. Understanding their priorities feels like solving a puzzle with missing pieces. This book will help you bridge that gap by uncovering the motivations and priorities of senior leadership, so you can align your efforts with organizational goals while staying true to your team's needs.

Your team might also rely too heavily on your presence to function well. Perhaps you're constantly mediating conflicts, stepping in to make decisions, or trying to enforce collaboration where none naturally exists. This book will guide you in building a self-sustaining team—one that thrives on collective agreements, shared ownership, and empowered decision-making. You'll learn how to create an environment where your team members feel confident taking initiative and resolving challenges independently.

Collaboration with business counterparts presents another challenge. The pressure to deliver while juggling maintenance work leaves you stuck in a cycle of reactive decision-making. Justifying technical investments or pushing back on initiatives that could harm your team in the long run proves difficult.

This book will equip you with practical strategies to advocate for your team's needs, communicate the value of technical work effectively, and build stronger partnerships with stakeholders.

Manager's paradox

One of the authors tracked how he spent his time at work. Here's a day in the life of an engineering manager:

I open my email in the morning, there are 74 new emails.
Many status updates for ongoing tickets.
An outage occurred during the night.
I'm adding "schedule a postmortem session" to my to-do list.
Next, a company-wide communication for managers about upcoming changes.
I'll need to read it carefully and think about how it affects my team.
Another to-do item.
The rest of the emails are not important, just updates from various systems.

Once I'm done with emails, I move on to the next routine—looking through the dashboards.
I wonder why the graphs show nothing that would indicate the last night outage.
It's definitely an opportunity to improve our monitoring.
Another to-do item.

Then, I open up my calendar: a couple of one-on-ones, a status sync for an ongoing project, a requirements review session for a new feature.
Oh yes, I need to schedule the postmortem for the outage.
As I do so, I realize that I won't have any time for focused work today.

Luckily, I already spent the first hour in the morning on what matters in the long term.
I made some progress on defining objectives for the next quarter and planning headcount distribution across key projects.
Tomorrow, I'll have another block of time to focus on strategic work: reviewing the gaps in the current architecture and planning development opportunities for engineers.

The rest of the day, I will be replying to messages in between meetings and adding more items to my to-do list.
At the end of the day, I'll have a 20-minute block to wrap up: document progress on ongoing tasks, write down priorities for the next day, reflect on what went well and what to improve.

We believe that managers add value by focusing on long-term aspects: defining direction, setting objectives, assessing team composition and the performance of individuals. But we spend most of our time on non-consequential activities. Quite often recent events require immediate attention and force us to adjust our schedule.

Only a small fraction of our time goes to what matters in the long term. Hence the manager's paradox:

> *The work that matters most is the work that feels least urgent.*

It's difficult to resist daily demands and prioritize high-impact work. Saving the day and solving problems feels like an accomplishment. But zooming out, you realize it's a run in a hamster wheel—you'll never run out of topics to jump on. There's always more work available to fill working hours.

Ignoring daily events is also not viable. In our careers, we've seen leaders who put too much emphasis on strategic work. They crafted exciting visions of the future, detailed roadmaps, and target states for the organizational structure. Their work was highly visible by leadership.

On the other hand, such leaders barely knew what the business-critical services were in their area or the main friction points of their teams. Those were "implementation details." As a result, the vision and strategy existed only in the documents and didn't affect the day-to-day operations of the engineering team.

Building a high-performing team takes balance. You look at daily events to identify patterns and understand sources of friction. You also secure time in your calendar to reflect and plan for the future. As the leader, you add value to the company by focusing on three pillars:

- Building alignment.
- Driving results.
- Shaping the team's culture.

In each of the pillars you must carefully choose what to focus on. To do so you need to deeply understand the context of your team and the company.

About this book

A big part of a manager's job is to resolve dependencies and deal with requests from various stakeholders.

In Chapter 1, we'll discuss how the way your company operates impacts your team. Factors like organizational structure, age and size of the company can significantly influence decision-making and prioritization. You'll learn typical ways of organizing reporting lines in a company and how to tailor your leadership style for each of them.

In Chapter 2, we'll discuss key data points to gauge your team's ability to deliver, as well as ways of measuring the impact your team drives. We'll end the chapter with key steps to run long-term planning. This will help you define objectives for your team, a great tool to reach alignment between product and tech.

As a leader, you're responsible for the outcome of your team. How you approach complex projects might be the difference between success and failure. In Chapter 3 we'll walk you through the crucial steps of planning and executing projects. We'll discuss how to measure the value of projects and tactics to deal with uncertainty. This chapter will help you set your team up for success and deliver on commitments.

In [ways-of-working], we'll talk about setting communication channels and organizing information. We'll discuss ways to improve efficiency of work by streamlining communication, tailoring processes and optimizing the flow of information in the team. Equipped with this knowledge, you'll be able to enable your team to reach its full potential.

In Chapter 5, we'll discuss your role in shaping your team's culture. As a leader, you build a team according to your vision. Having a clear vision of the future and communicating it effectively is crucial. Less inspiring but equally important is acting deliberately to set and maintain your team's culture.

Establishing a team charter, building working agreements, and providing feedback may not feel natural, yet these actions are essential to foster a positive culture in the team. This day-to-day aspect of leadership is well captured by a quote from Lieutenant General David Morrison, Chief of the Australian Army from 2011 to 2015: > The standard you walk past is the standard you accept.

We'll end the book with a chapter about managing your own well-being. A burned out and disengaged leader harms the team. It's common to think, "it won't happen to me."

Unfortunately, it might. Information overload, competing priorities, difficult conversations and constant context switches are common for managers. In Chapter 7, we'll discuss techniques to maintain your productivity and develop the mental toughness needed to weather difficult times.

We hope that in the chapters that follow you'll find insights that will help you become a better leader.

Let's go.

Chapter 1. Understanding your role

Every company is unique, with its own ways of working, culture, and expectations. What works for a manager in one organization might fail in another. To thrive as an engineering manager, you must align your behaviors with your company's specific context.

This chapter focuses on understanding the company's big picture. By the end, you'll have a clearer grasp of why your organization operates as it does, how engineering fits into its goals, and how factors like size, industry, and structure shape your role. This understanding is essential not only for excelling in your current role but also for evaluating whether your current company aligns with your career aspirations.

1.1. Company characteristics impacting your role

The job title "Engineering Manager" (EM) can vary widely between companies. In one company, an EM might oversee a large team of thirty people. In others, the same role may lead a small team and focus heavily on hands-on work.

It's important to ensure your expectations for the role align with the company's definition. Certain company characteristics significantly shape the scope of an EM's role and you might need to adjust your leadership style. Looking at the company's big picture will help you focus on what matters most for the business.

Industry

The industry might impose constraints that shape how the company operates. In a regulated area, delivering faster might not be a first priority and the path to production requires security and privacy audits. Safety-critical areas like automotive are required to follow a pre-defined development lifecycle and use only approved technologies. Working with government might limit vendor selection and geographies for deploying workloads.

Such specifics might significantly influence how you spend your time. You might be required to participate in audits, ensure compliance with regulations during the development process, or follow company-wide processes for legal reviews.

Industry specifics might shape how a company balances speed with risk when defining success. Businesses commonly push for shorter time to market. But cutting corners to ship features faster might backfire. In 2023 Meta Platforms Inc. was fined for 1.2 billion euros for the violation of the privacy regulation in the European Union.[1]

We encourage you to learn more about the industry of your company and reflect on how it impacts your team.

Size

The number of employees in the company reveals much about its operations—more people mean more communication, more stakeholders, and a greater need for alignment.

A company's headcount indicates the management hierarchy. For a small business, it’s normal when individual contributors have direct contact with the company leaders. This is unlikely to be the case in a company with thousands of employees.

In a smaller company, you might have direct access to information about the priorities and might even influence them. In a bigger company you might be given a direction and asked to execute. This can be a major factor in how satisfied you feel in your role. Managers tend to underestimate their need for autonomy and find themselves frustrated with their role.

The company’s current size matters, but so does what happened in the past and plans for the future. Are they actively growing now? Did they just come out of a hyper-growth phase? If so, don’t expect many processes or a consistent culture.

Profitability

There’s a huge difference between working in a company that’s profitable compared to one that relies on funding. A profitable company can afford long-term investments and innovation. Whereas companies that rely on external funding focus on short-term results.

The state of the balance sheet defines the planning horizon of the company, which impacts how you balance between long-term and immediate needs. It doesn’t make sense to work on quality strategy and test automation if the company might run out of money in three months.

On the other hand, a profitable company might be willing to take initiatives with longer payoff periods. Things like planning architecture evolution, improving customer experience, building internal tooling to streamline operations might be valid initiatives in a profitable company.

Consider the company's planning horizon when developing roadmaps and prioritizing technical improvements. Ask yourself whether the outcomes you're targeting are too far in the future to yield meaningful benefits.

Age

A company's age reveals much about its technology landscape. A thirty-year-old company will inevitably have legacy systems and in-house built technologies. Although those home-baked tools might be inferior compared to modern alternatives, the cost of migration might be too high.

Long-standing companies often struggle with legacy systems and outdated technologies. It's common to see plans for modernizing tech stacks that span multiple years. The tricky part is understanding whether those plans support the business or they're built by architects sitting in an ivory tower.

"The single worst strategic mistake that any software company can make: They decided to rewrite the code from scratch," said Joel Spolsky, creator of Trello and Stack Overflow.[2] He wrote this in 2000, way before public cloud and microservices. And yet, futile rewrites are still common.

We've seen very few companies that went through the modernization successfully. More often, attempts to modernize tech stack turn into a rabbit hole of never-ending work and questionable outcomes.

Working on tech modernization projects might be a challenging but rewarding experience where you get a chance to shape the future of the company. But to succeed in this task, you need to plan carefully, build alignment and ensure executive support for these initiatives. In Chapter 3 we'll talk more about planning and executing projects.

Engineering: Cost or investment?

How a company's leadership perceives the engineering function shapes everything from budgets to decision-making authority. In some companies, engineering is a key function and their work directly contributes to the company's revenue. Common examples are big technical companies, SaaS platforms, game development, and so on.

In other companies, engineering is a small function that plays a support role for other roles. Think of banking, insurance, accounting where many engineers build and run complex systems, but they're not the ones who make money for the business.

Company leaders might see the engineering function either as a cost center or an investment:

Investment Perception where engineering is a driver of long-term value. Spending more resources on engineering yields bigger returns in the future. From this point of view, buying tools that increase productivity, upgrading laptops, and providing learning budgets for engineers makes sense.

Cost Center Perception views engineering as an operational expense to be minimized rather than a strategic asset. From this standpoint, every dollar spent on engineering must be justified by immediate returns. Investments in better tools, hardware, or training are often seen as unnecessary costs rather than enablers of future growth. This results in more focus on short-term savings and strict budget control, which can ultimately limit innovation.

Gregor Hohpe in his talk *Enterprise Architecture = Architecting the Enterprise?* gives an overview of how different executive roles perceive the IT function.[3] Reflect on which point of view is common in your company. Keep in mind that different leaders might have different opinions and adjust how you pitch ideas based on their perception. Especially when you ask for additional resources or headcount.

Cost vs. investment perception also defines how much engineering leaders are present in the room where important decisions are made—Do they have a say on the company strategy? Are they able to influence other functions?

We've seen companies where a salesperson signs a new client by promising tailored customization of software and then engineers are presented with a list of urgent features and a tight deadline to deliver. We've also seen the opposite where the engineering team is empowered to say "no" to unrealistic promises and then together with sales they co-define a feasible offering.

We encourage you to reflect on how your company's context influences engineering teams. Consider how size, industry and other factors apply to your team.

1.2. Your place in the organizational structure

This section is available in the full version of the book.

1.3. Your team's context

This section is available in the full version of the book.

[1] https://www.statista.com/statistics/1133337/largest-fines-issued-gdpr/

[2] https://www.joelonsoftware.com/2000/04/06/things-you-should-never-do-part-i/

[3] https://youtu.be/hhlxFtV_tZo?si=E951QpkUN7HEdtRH

Chapter 2. Defining your team's direction

In the previous chapter, we have reviewed the key factors that set the context of your team. Now, we will talk about defining direction for the team.

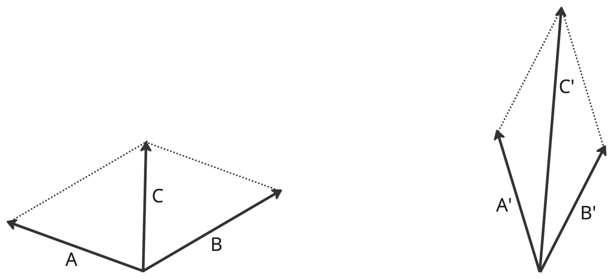
We will start with discussing the importance of alignment and how you can create alignment inside the team, with other teams, and with the company's strategy. Then, we focus on collecting data to assess the current state of your team. This is a crucial step to plan for the future. We're convinced that the direction of the team must be grounded in the understanding of what the team can and cannot do. Next, we discuss how to interpret data with care—a metric that looks great on paper might mask underlying issues, and a seemingly bad number might be explained by the team's specific context. Finally, we cover setting clear objectives for the team, ensuring that everyone shares a definition of success and that the team's work aligns with broader organizational goals.

After reading this chapter, you'll be able to:

- Deploy qualitative and quantitative data to assess your team.
- Influence non-technical stakeholders.
- Run a planning process to set objectives for your team.

2.1. Start creating alignment

We all have an intuitive understanding of how a well-operating team looks: shared understanding of priorities, timely communication, smooth collaboration, and informed decision-making. But how does a team reach that state? It’s the manager’s job to resolve the friction points, ensure alignment and build shared understanding. In this section we will talk about building alignment.



We can compare organizational alignment with a vector sum in mathematics. The length of the resulting vector, C, depends on not only the length of A and B, but also how the two vectors are positioned to each other. In fact, the position of two vectors plays an even bigger role than their length. The length of A' and B' equals A and B, but the length of their sum, C', is twice as much as C.

You play a key role in ensuring that your team’s work produces the desired outcomes. You constantly identify and address impediments that consume the team’s time but do not result in measurable impact. Below, we will share common sources of friction we’ve seen in our careers that will help you recognize similar patterns. After reading this section, you’ll know what to look for and what tools are available to you to build alignment in your team.

Unclear scope of the team

When there is no clear expectation about the team’s impact, everything becomes a matter of opinion. A team without a purpose will jump from one topic to another: an executive asks for an urgent bug fix, an orphaned service needs owners, a long-standing migration that other teams are too busy to complete. The team’s purpose doesn’t have to be inspiring or visionary, but it has to be specific, tangible, and measurable. It could be launching a product, building an internal platform, improving a part of the user journey, or any time-bound project with clear expected outcomes.

To drive prioritization, the team’s impact must be measurable: revenue of the product, adoption by internal users, conversion rate on the page. If your team lacks purpose, you must find an answer to the question "What metric is the team supposed to drive?" Discuss it with your manager, your product counterpart, and other stakeholders. Find the person who decided to staff the team. What was their intent? What did they want to change?

Sometimes you might not find answers: leadership changed, people left the company, or the original problem ceased to exist. When that happens you’ll need to define the team’s scope. Work with your senior managers to define the topics your team will own and how you will measure the impact. Remember that you don’t need to come up with a long-term vision for the team. It’s already a success if you can define the team’s focus for the next six months. At least, for half a year, the team will be able to say "no" to work that distracts from the main goal.

Lack of documented decisions

We've seen teams running in circles discussing past decisions. A project that was cancelled—is it the right time to revive it? Which test framework to adopt? Should the team switch to Kanban or stay with SCRUM?

These and many other matters have been discussed in the past, but nobody remembers the context and the rationale of the decision. As a result, the same topics pop up again and again, dragging the team into the same debates. Such debates consume time but make little difference to the team's productivity.

To prevent this infinite loop, start documenting decisions. Write down the context around the problem and the circumstances at the moment of the decision. Document the decision and its rationale and store the document in a shared location. Next time someone brings up the topic, refer them to the document. Unless there is a compelling reason to change the decision, stick to it. Don't make changes for marginal gains and only change for order-of-magnitude improvements.

> We've found that documenting architectural decisions helps establish common approaches to technical problems and preserve knowledge. Look at Architecture Decision Records Process for practical steps. You can also adopt this process for business decisions.

Incomplete prioritization

You might have seen teams where all prioritization happens by a single person. Typically it's a product or engineering manager. That person is often biased toward their own background. A PM will naturally put higher priority on business features, rather than getting into the nuances of architecture, quality or reliability of systems. Similarly an engineering leader might prioritize tech stack modernization, and overlook the opportunity-cost of such work. In the long run overindexing on one aspect skews the balance and hampers the team's productivity. It works best when prioritization is based on team discussion, rather than on one person's opinion. Through dialog, the team finds the balance between delivering business value, maintaining quality, evolving architecture, automating manual work, and other aspects of building and operating software.

Your role as an engineering leader is to inform prioritization by providing a holistic overview of the systems and teams in your area:

- How often do systems break in production?
- How much time does the team spend on non-value adding work?
- How many defects does your software have?
- How many of them are found after a release?
- How long does it take to deploy changes to production?
- How often do deployments backfire?

These questions, if ignored, will diminish the team's impact. Adding new features to services that aren't reliable for their users won't make customers happy. Buggy software shipped on time creates serious reputation risk for the company. Make sure your stakeholders are aware of such issues, and they are addressed in prioritization.

It's important to have a clear decision-maker who is

> accountable for prioritization. However, being accountable doesn’t automatically mean that they will be sufficiently informed. We found that agile practices like Three Amigos help to build holistic understanding of the work.

Another factor that might lead to skewed priorities is planning work for the full team’s capacity. We’ve seen multiple times how a team plans work for a quarter and they try to squeeze as many projects as possible into this timeframe. But some work is not visible at the moment of planning—production incidents, critical bugs, security and infrastructure updates. When they pop up, the team can’t say "no." Most of the time, this work is non-negotiable.

To account for unexpected work, we recommend setting a buffer of 20–30% of the team’s capacity. Remember that it takes effort to operate software systems in production. Plan work for reduced team capacity, and use the buffer for unexpected work.

Dependencies

We’ve seen situations where a team works on a project, all hands on deck to deliver on time. Very close to the completion date, a previously unknown dependency pops up: a new requirement, breaking change in API the team depends on, a mandatory legal review. The project halts and significant time goes into resolving these issues.

You can mitigate some of these situations before they become a problem. No team works in isolation and that’s why a big part of a manager’s role is to oversee the team’s surroundings. You need to have a good overview outside of the immediate team’s scope:

- What are the systems that your team depends on?
- Which teams consume your team's output?
- Who might block or override the team's decisions?
- Which situations require external review or approval and from whom?

Knowing these dependencies is a crucial part of creating alignment in your team. Once you identify the key stakeholders, start defining how you work with them. Agree how a downstream dependency will notify you about API changes. Discuss how much advance notice to give teams that depend on you when things get delayed. Clarify when to involve people for reviews and approvals.

We recommend investing time in building relationships with key stakeholders. Do that in advance. Asking for extra work or negotiating priorities is easier when you already know the people.

However, agreement is not always possible. Two teams often cannot agree on priorities. Team A's project requires work from Team B, but the latter is fully booked and can't deliver the work. In such situations escalation is the right approach. Team A can't decide to stop their project because it was promised to company leadership. Team B cannot accommodate A's work, because they're focused on a time-sensitive project. And neither team has the political power to change the other team's backlog. The word "escalation" has a negative connotation—It might seem like teams opposing each other or their leaders struggle to work together. But done right, it's a powerful technique to reach alignment.

It often backfires when each manager independently escalates the issue up their reporting line. This approach spreads incomplete information. On the opposite side, a clean and transparent escalation is more likely to resolve the disagreement and create alignment. This will be for the benefit of both teams. To escalate efficiently, you need to pay attention to two things:

- Identify the decision maker who has authority to resolve the conflict.
- Provide them with all the information available about the issue.

A clean escalation requires managers of both teams working together on documenting context, possible alternatives, and informing the decision maker about the situation. Approached this way, it shows that both leaders are competent, committed, and collaborate together efficiently. Don't be afraid of escalating issues if the situation demands it.

Interpersonal conflicts

Conflicts in the workspace are common and they aren't necessarily bad. In a moderated discussion, conflicting opinions ensure rigorous evaluation of alternatives and improve decision making. The focus of healthy conflict is on finding out the truth—Both parties are open to changing their minds in light of new evidence.

But when focus shifts from finding the truth to defending one's opinion, conflict becomes unhealthy. You, as a leader, have a huge influence on conflicts in your area. But before looking at the team level, we encourage you to look at yourself. Do you have unhealthy friction with your colleagues? Have you noticed yourself being annoyed by someone or unable to work with them? If so, this is the very first point to address.

Most conflicts at work happen due to misunderstandings. One person interprets someone's actions in a way that isn't intended. The tricky part of navigating such conflicts is to acknowledge their existence and clarify intention without blaming the other side. Carefully crafted phrases can help with starting the conversation:

- I feel like we got off on the wrong foot, and I'm sorry about that—I don't have anything against you at all. Let's discuss how we can work together?
- I've been feeling a low-level tension between us in a few meetings, like maybe we're quietly annoyed at each other but trying to stay polite. Does that feel true?
- I noticed it's been difficult for us to agree about the topic. I'd like to understand your thinking behind it and define together how we can work together?

Talking through the tension points will help, but it might not be enough to completely resolve it. Consider scheduling a regular one-on-one meeting with the person—meeting frequently helps to get closer and feel comfortable talking to each other about the friction. It might be helpful to have these meetings in a pleasant environment: go for lunch together or take a walk.

When you're part of the conflict, the only way to solve it is to avoid blaming the other side and genuinely invest time in improving relationships. But you have a broader toolkit when you're helping to resolve conflicts between people in your team:

- Provide feedback. People in conflict are often unaware of the ramifications of the conflict and its impact on the team.

- Mediate the discussion. You can ensure the message lands and isn't distorted by emotional judgments.
- Isolate people in conflict. Assign them to different projects and minimize their interactions to reduce friction.
- Transfer to another team. One of the people might be more productive outside your team.
- Call out when conflict goes outside professional boundaries and involve HR.

Resolving interpersonal conflicts is a crucial part of building alignment in the team. Your team is unlikely to be productive if you or your team members struggle to work with colleagues.

Mismatch with company's direction

Working in a big company, we've seen organizational islands that operate quite differently from the rest of the company. Teams living on those islands are happy and productive, until they're forced to collaborate with the rest of the company. The whole company uses Java, except engineers in the Marketing department, who use Ruby. They have adopted their own CI and deployment tooling and it works perfectly for their needs. Until one day the company goes through security audit and it turns out that while all Java systems are covered by a centralized dev experience team, the Ruby stack is lagging behind. Closing the gap in security requires significant time investment and a difficult decision: whether to maintain two tech stacks or start a big migration to a single stack.

This example is a bit extreme, but the point still stands: avoid creating organizational islands that go against the company’s direction. This advice is not limited to technologies, but also applies to business tooling. If the company uses Jira, make sure to use it in your team as well. Don’t try to create alternative documentation storage if there’s already a company-wide wiki solution in place. This only makes it harder to discover information, coordinate work across several teams, and introduces operational overhead. It might be the case that the tool you use is way better and provides order-of-magnitude productivity improvement. In this case try to influence the company leadership to adopt the tool, but consider the cost of migration.

Creating alignment is a crucial part of a manager’s job. Be aware of the levers available to you: metrics, documentation, agreements, escalations, and feedback. This type of work might not be visible and doesn’t pay off immediately. Yet, it compounds over time and leads to a well-functioning organization. Look out for potential friction points and ensure alignment inside the team, between teams, and with the company.

2.2. Understand the current state of your team

This section is available in the full version of the book.

2.3. Making decisions with data

This section is available in the full version of the book.

2.4. Setting objectives for the team

This section is available in the full version of the book.

Chapter 3. Planning and executing projects

In the previous chapter, we spoke about defining the direction for the team. It is no less important to be able to lead the execution of major initiatives toward the desired conclusion. In this chapter, we'll give an overview of project management practices and focus on practical advice for planning and executing projects. After reading this chapter, you'll be able to:

- Define project goals and measure the value of the project.
- Ensure required support for your team's projects.
- Identify early major risks and assumptions that might impact the project.
- Create an instrumental project plan to track the scope and communicate status updates.

Before we go into the tools and techniques, let's spend some time reviewing common misconceptions and the goals of the planning process.

3.1. Plan for uncertainty

The meaning of the word "plan" is murky; it might relate to different things. Just look at a few definitions from the Merriam-Webster dictionary:

- A method for achieving an end.

- An often customary method of doing something (same as procedure).
- A detailed formulation of a program of action.
- Goal, aim.
- An orderly arrangement of parts of an overall design or objective.

These definitions give an idea of the concept, but don't help distinguish a plan from what is not a plan. It's no surprise that we use this word loosely. There is a South Park episode where gnomes created a "business plan": collect underpants > ??? > profit!.[1] We have no other means but our judgment to distinguish a good plan from a nonsensical one. In this chapter, we'll focus on specifics. We'll give you the required knowledge and practical steps to build viable plans and evaluate the ones you're presented with.

Let's start with a couple of hypothetical examples:

1. A sales team plans to increase sales year over year by 10%.
2. A research team plans to conduct research for a new medicine.

Both are legitimate uses of the word "plan." For the sales team, a plan might include hiring one more salesperson, attending a conference with potential customers, and tracking the number of sales calls. For the research team, a plan might include reviewing existing literature, reproducing previous experiments, and running new experiments. Yet, these two plans still feel different.

Put complexity on a scale

In our view, the key difference is the level of uncertainty surrounding the end goal. Increasing sales is, in general, perceived to be more predictable. On the other hand, developing a new medicine is more unpredictable. Depending on the level of uncertainty, the details, as well as the meaning, of the planning process change. As uncertainty increases, the planning process shifts from "a detailed formulation of a program of action" to developing "a method for achieving an end."

In 1999, Dave Snowden introduced the **Cynefin Framework** (pronounced *kuh-NEV-in*) that describes how decision-making changes in response to the level of uncertainty.

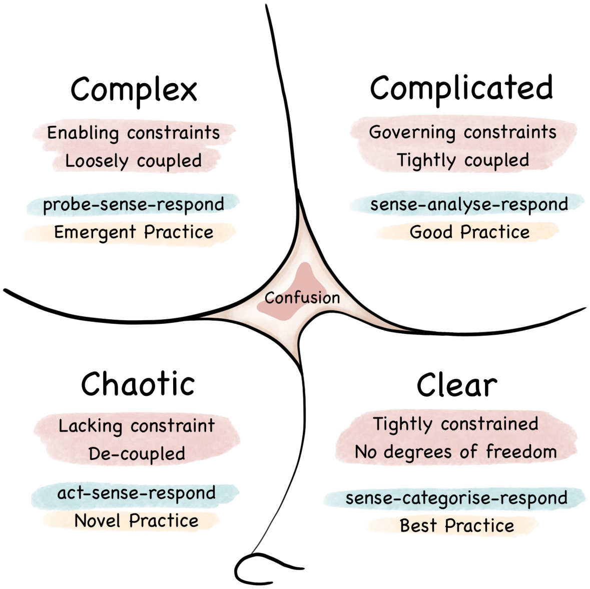


Figure 1. Image source: https://commons.wikimedia.org/wiki/File:Cynefin_framework_2022.jpg

This is what Wikipedia says about the framework:[2]

> *Cynefin offers five decision-making contexts or "domains": clear, complicated, complex, chaotic, and a center of confusion. The domains on the right, clear and complicated, are "ordered": cause and effect are known or can be discovered. The domains on the left, complex and chaotic, are "unordered": cause and effect can be deduced only with hindsight or not at all.*

Approach uncertainty with the right mindset

Cynefin provides terminology to describe complexity. Building on top of it, let's distinguish between **operational** and **innovative** planning.

Operational planning is a type of planning where a known activity leads to the desired outcome. More sales calls to potential clients will result in more sales. The only unknown here is how much more sales calls are required to reach the target.

In contrast, for innovative planning, there is no activity that leads to the desired outcome with high certainty. In other words, operational planning is applicable where a problem has a known solution (clear and complicated domains), and the plan ensures that the solution is applied correctly.

Innovative planning tackles open-ended problems (complex and chaotic domains) where the existence of a solution or the path to it is unknown. For our example, developing a medicine might not be possible with current technology, or economically unjustifiable. And this is something that cannot be known at the beginning of the research.

We've considered examples of different activities: one is mainly operational (sales), and the other is more innovative (research). The question relevant to our discussion is where software engineering resides on this spectrum.

Before we jump into this discussion, let's explicitly mention that Cynefin is not a hard classification; rather, let's treat it as a spectrum of complexity. Few knowledge work activities, if any, reside entirely in one domain. Cutting-edge research labs still write reports, procure office supplies, and do other trivial activities to keep the lab running.

With that in mind, we claim that software engineering tends to reside in the complex domain. We think this is so because when we start a software project, we rarely know where we will end. Software teams think that they know what customers need and want, but they are often wrong with their assumptions. Also, software development often requires us to balance technical feasibility, value for its customers, and economic viability. This is not something that can be known at the beginning. If software engineering were simple, we would see a much higher success rate among the start-ups.[3]

Applying the wrong mentality undermines the value of planning

Operational and innovative planning address challenges at different levels of complexity. Operational planning is effective when the end goal can be measured and activities directly influence the metric. It's like traveling to a known destination with a map.

In contrast, innovative planning validates strategies for solving open-ended problems—more akin to exploring uncharted territory. Innovative planning generates the best current approach for reaching the goal. While it doesn't guarantee success, it helps eliminate paths that are certain to fail. In technical terms, it functions much like a Bloom filter,[4] which can indicate if an element *might be* in a set or is *definitely* not.

Problems arise when we apply an operational planning mindset to situations that require innovative planning. This mentality mismatch causes two major flaws:

1. Pretending certainty in the face of inherent complexity.

2. Replacing the value with a proxy like hitting a deadline or delivering pre-agreed scope.

First flaw: Pretending certainty in the face of inherent complexity

This issue is often summed up by the question, "Here are the requirements, when can you deliver them?" Engineers hear such questions frequently, and it's remarkable how often it leads to frustration for those asking. We must also admit that engineers sometimes make matters worse by using the complexity of software engineering as an excuse to sidestep bureaucracy. Stakeholders, often nontechnical, expect predictability from engineers, but engineering often fails to communicate the underlying complexity.

In extreme cases, this conflict turns the plan into a façade, concealing loosely coordinated activities behind the promise of delivering something by a specified date. The plan that consists of a few bars on the Gantt chart that are all supposed to finish before the deadline is not much better than the one from the Underpants gnomes.

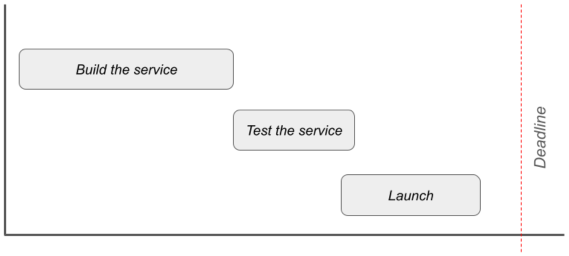


Figure 2. A façade-style plan that ignores internal complexity, but pretends certainty.

Such a plan pretends there is no underlying complexity: the new platform to handle e-commerce inventory will be launched by Q3, cloud migration will be completed in six months, etc. Such "plans" are essentially disconnected from reality and serve only to soothe stakeholders by making them believe that everything will work out by the deadline.

Using the plan in such a way diminishes the value of the planning process. We skip the most arduous, but rewarding steps of it:

- Pinpointing key requirements.
- Identifying and validating assumptions.
- Managing risks.

We'll talk about these parts in the following sections. For now, we want to convey the core idea behind the innovative planning—acknowledge uncertainty. It's unlikely that by doing something for the first time, you would produce an excellent result. Similarly, at the start of a new project, we know less about it than we ever will. Imagine a video game in the strategy genre: at the start of the game, you know nothing about the terrain of the map, resources available nearby, or the actions of your opponents. What do you do then? You scout for information to guide your actions. This scouting mentality is essential to approach innovative planning.

Let's look at a prominent instance of a project with high uncertainty. Back in 1997, when we had no iPhones and buying stuff on the internet wasn't common, Amazon launched one-click purchase on their website. A logged-in user could press a button and the order would be shipped to them. No adding to cart, no confirmation popups. Just one click.

To enable one-click purchases, the team had to redesign the entire order flow. Instead of asking customers to confirm their orders to prevent accidental clicks, they focused on making it easy to cancel orders after purchase. Initially, the process required four clicks. After further refinement, customers could complete a purchase with just two clicks. Finally, after significant engineering efforts, they successfully launched the one-click purchase.[5]

This is a great example of operating under uncertainty: the team could define a clear outcome (make purchasing radically simpler), but they couldn't fully know upfront what trade-offs, risks, and technical changes would be required to get there. They had to explore the problem space, test assumptions (for example, whether preventing mistakes should happen before or after purchase), and learn their way toward a workable design through iteration.

Second flaw: Replacing the end goal with an easy to measure proxy

What is a successfully completed project? A common answer is: one that was delivered on time, with a complete scope, under the given constraints (people, budget, etc.). This concept is also known as the **Golden Triangle** [6]: time, scope, and cost. The common saying goes that you can choose two out of three, but if you constrain all three, the project is doomed to fail.

The Golden Triangle model provides useful insight about staying flexible on constraints, but it's vague about the project's value. When we ignore value and focus too much on meeting deadlines, we get misleading feedback. A project can appear to be succeeding when it's actually futile.

FBI's attempts to build a new case management system are a prominent example of how replacing success with a proxy might backfire.[7] In the 2000s, FBI agents were relying on an old mainframe application and a lot of paperwork to manage cases. Between 2001 and 2004, the bureau worked with a contracted vendor to build a Virtual Case File system to automate the work of agents.

This project cost $171 million and never went live, but that's not the end of the story. With lessons learned, the agency took a second attempt. In 2005, they created a Request for Proposal (RFP) with all the capabilities of the legacy system. To apply learnings from the first try, the agency split work into phases, and required the contracted party to deliver a predefined set of capabilities by a specific date. To prevent overspending, the steering committee had an opportunity to kill the project at any time if they weren't happy with the progress.

Fast-forward five years, and after having burned another $335 million, the system was still in development in 2010 and had more than 10,000 bugs.[8] At this point, the FBI decided to remove the contracted party and complete the development in-house.

This issue is not limited to government projects; it is also common in the business environment. Unless we have a way to preserve the goal and the value of the project, it’s easy to get lost. At Amazon, teams write a Press Release (PR) for the product before development begins. This document helps clarify the value of the product for users and what it aims to achieve. In 2024, Amazon’s CTO published the original PR/FAQ document for AWS Lambda that started the era of serverless software back in 2014.[9]

Crafting a document for end users forces us to address the uncertainty. It helps scrutinize requirements and ensures that every product idea is delivering meaningful benefits. It helps teams identify potential issues, risks, and constraints before significant resources are committed. These steps are essential parts of what we previously called "innovative planning."

The first step of project planning is to clarify meaningful goals and document them. In the presence of uncertainty, we can’t know the path to the goal, but we need clear goals to guide our actions. Once the goals are clear, we can start planning the project.

3.2. Prepare to plan

This section is available in the full version of the book.

3.3. Make the plan

This section is available in the full version of the book.

3.4. Execute the plan

This section is available in the full version of the book.

[1] South Park gnomes

[2] https://en.wikipedia.org/wiki/Cynefin_framework

[3] https://stripe.com/en-hr/resources/more/startup-statistics-you-should-know?utm_source=chatgpt.com

[4] https://en.wikipedia.org/wiki/Bloom_filter

[5] Joel Spolsky's talk about one-click purchase: https://www.youtube.com/watch?v=-QqIyICyXbU&t=34s

[6] https://en.wikipedia.org/wiki/Project_management_triangle

[7] Why the FBI Can't Build a Case Management System: https://ieeexplore.ieee.org/document/6127846

[8] https://oig.justice.gov/reports/FBI/a1022.pdf

[9] https://www.allthingsdistributed.com/2024/11/aws-lambda-turns-10-a-rare-look-at-the-doc-that-started-it.html

Index

ENGINEERING MANAGER'S COMPASS

Insights for Building Effective Engineering Organizations

The responsibilities of engineering managers (EMs) are vast. We set our team's direction while staying hands-on and up to date with ongoing projects. We drive technical excellence, partner with the business to deliver value, uphold high-performance standards, foster a productive culture, and help individuals grow. But how can we fit all of that into our schedules and still maintain our own well-being?
This book offers an answer to that challenge: understand the context, identify key focus areas, and apply the right tools from a broad toolkit.
Each chapter addresses a core aspect of an EM's role: setting goals, managing projects, organizing team workflows, developing people, shaping team culture, and maintaining personal well-being. The tools and techniques presented help new and aspiring managers become effective in their roles.
However, the book's ambition goes further. It aims to give readers not just a set of stories, but a navigational tool, a compass, to help find the right path in any situation. We balance practical advice with theoretical grounding, emphasizing data over anecdotes.

From 2018 to 2024, the global number of software engineers grew by 25%, from 23 million to 28.7 million. This rapid growth increases the need for strong leaders and capable managers. To keep pace, we need scalable ways to share knowledge and develop new leaders. Reading many specialized books and applying their ideas is a solid long-term approach, but trial and error can be slow and stressful, especially when facing unfamiliar challenges like the massive shift to hybrid and remote work. This book emphasizes data-driven assessment and decision-making. Its chapters are designed to be accessible to managers at different experience levels, so readers can use it either as a step-by-step guide or as a reference for specific challenges.

According to the authors, the most critical skill new managers can develop is the ability to identify the right problems to solve—a core principle of strong leadership. Each chapter helps new and aspiring managers build a clearer understanding of the engineering manager's role and the fundamentals of leading teams.

This is the missing manual for engineering managers ready to move from intuition-driven leadership to deliberate, evidence-based impact.