The Tech Executive Operating System

Creating an R&D Organization That Moves the Needle

Aviv Ben-Yosef

Apress®

THE TECH EXECUTIVE OPERATING SYSTEM

CREATING AN R&D ORGANIZATION THAT MOVES THE NEEDLE

Aviv Ben-Yosef

Apress®

The Tech Executive Operating System: Creating an R&D Organization That Moves the Needle

Aviv Ben-Yosef
Hadera, Israel

ISBN-13 (pbk): 978-1-4842-6894-0 ISBN-13 (electronic): 978-1-4842-6895-7
https://doi.org/10.1007/978-1-4842-6895-7

Managing Director, Apress Media LLC: Welmoed Spahr
Acquisitions Editor: Shiva Ramachandran
Development Editor: Matthew Moodie
Coordinating Editor: Nancy Chen

Cover designed by eStudioCalamar

Distributed to the book trade worldwide by Springer Science+Business Media New York, 1 New York Plaza, New York, NY 10004. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail booktranslations@springernature.com; for reprint, paperback, or audio rights, please e-mail bookpermissions@springernature.com.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484268940. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

To Bar, my wife, best friend, partner, and personal life coach, for being there for me along this adventure.

Contents

About the Author

Aviv Ben-Yosef is an advisor, coach, and consultant for executives and leaders throughout the tech industry. In his consulting business, he helped companies worldwide, ranging from day-old startups to Fortune 100 companies. By coaching, speaking, and facilitating workshops, roundtables, and trainings, he helps his clients forge teams and cultures they are proud of leading.

In his work as a consultant, Aviv has developed a unique approach to aid the leadership of software organizations. Coming from a technical background allows him to "talk shop," yet maintain a business impact–driven mindset. Aviv's online writing has reached over six million readers, and his publishing includes multiple blogs, podcasts, videos, and online courses.

When not working with clients, Aviv is likely to be spending time with his three children or digging into one of his many hobbies, including reading, wine, chess, learning Italian, piano, fountain pens, and, of course, coding.

Acknowledgments

Writing a book is never straightforward. Nevertheless, it was the destiny of this book to come into being during a global pandemic inducing great stress and lack of certainty for everyone involved while we were going in and out of lockdowns. That has made me appreciate that much more and be thankful to everyone who has helped me make this a reality.

First, my thanks go to Bar, my wife, without whom there is no chance I could have written this book or even started this journey to help tech executives. She—along with our children, Romy, Zoe, and Leo—did her best to support and provide a semi-normal environment for writing while sheltering in place.

I will be forever grateful to my parents, Uri and Naomi, for going to great lengths to support my "computer hobby" since I was six years old. No matter what book, modem, or course I asked for, they made sure it was there when I needed it, even though they had no clue why "using Linux" requires a modem with thrice the cost.

I owe the ideation of this book to my mentor and coach, Alan Weiss, who initially sparked the idea of writing in me and whose guidance helped me formulate it along the way. The daily grind was always under the watchful eye of my accountability partner and friend, Rebecca Morgan. She and all fellow mojo makers, Alan Willett, Martyn Drake, and Natalie Nixon, were founts of creativity, novelty, and energy.

While many have helped with the content, a special and unique *thank you* goes to those who shaped it the most, Nimrod Priell and Yonatan Bergman. Not many would take on such a task during the first months of starting a new business, but they did and did so remarkably.

My thanks go to Shiva Ramachandran for accepting this book and the rest of the fantastic Apress team that worked on making this sound decipherable to people other than me. Thank you to Sharon Holmes as well for creating beautiful illustrations from my crude sketches.

Lastly, my greatest appreciation and gratitude goes to all my clients over the years: thank you for allowing me to join you along the ride and be a part of it.

Introduction

Software is a fantastic contribution to our lives. Creating software is even more remarkable. It's the ability to create value that improves the world without consuming natural resources. Many executives can recall "the good old days" when they typed at their keyboards and coded away. Things got much more complicated as they moved up the organizational ladder. They soon realized that understanding code does not mean one will understand how to assist people in creating it effectively.

I was once brought in to help a struggling startup. The founders had a team of talented and bright people, and all were motivated and cared about their work. However, they lacked the proper guidance and enabling culture to focus their efforts and strive for impact. That led them to spend years without significant changes to the company's bottom line. Week after week, sprint after sprint, they churned out more features, adjustments, and improvements. They used cutting-edge tools. None of those mattered, and they were quickly running out of money.

I helped them establish a skunkworks team that was effectively less than two full-time engineers. The team focused on iterations, open feedback, adequate amounts of chutzpah across all members, and, most of all, impact. Every feature was discussed openly, and the right balance was struck for each, given the work estimates and certainty of improvement that it had. We launched a product two months later and, with meticulous improvements, achieved traction the company hadn't seen before. Inch by inch, test by test, iterations were made to hone assumptions. We weren't discouraged by a failed test—to learn and invent, you have to fail spectacularly. The product amassed millions of installations and got the biggest players in the space to react. Fast-forward a few months, and the company was successfully acquired, with the founders attributing a sizable chunk of their success to this task force.

The thing is, the team didn't invent anything never seen before. It was putting together technology the company already had but doing so much faster and focusing on results. The people spoke up when a feature didn't make sense—because they *got* the product. Having worked with dozens of teams of all sizes—the tiniest startups, Fortune 100 companies, government agencies, and everything in between—I saw the same pattern emerging again and again. We could create high-impact teams that excel, with the right processes, values, and attention. In this book, I've collected the best tools I've used successfully

with companies worldwide, and it contains methods and concepts proven to work for organizations of all sizes. You will find valuable advice no matter whether you are a first-timer or an experienced executive taking over a new team.

We will start with defining your role, covering the way to successfully kick-start your role, and positioning yourself for maximal leverage in the company. Then, we dive into tools for managing yourself, aligning your team on impact, and how to best introduce changes without harming your momentum. We will finish with covering the road to transforming your team from a cost center to an innovation center, guidelines for growing your team to become more robust, and making your entire team masters of the product to help them become force multipliers for the business.

I wrote this book to make these lessons accessible to everyone. Together, we can raise the bar for the entire profession. No matter whether you are in charge of a startup team or the R&D division in a large company, you can cultivate people who collectively move the needle. The framework provided in this book will immediately help you shape a remarkable team. Let's stop reminiscing over "the good old days" and proactively create a brighter future.

CHAPTER

1

What Does the Company Need You For?

You cannot lead without a purpose

Your journey begins by establishing your destination. You want to become a great executive and lead an extraordinary team. We will start by understanding why that is important.

What Is a Tech Executive, Anyway?

You might have expected it would wear off after a few years, but I am still giddy whenever I start working with a new executive. That is because no single company is the same as another. At every company, I have to tackle a different business, different values, diverse cultures, and different challenges. Moreover, if you'd forgive a cliché, I find that every executive is a unique snowflake. Due to this, we start the path for excellence with executives by understanding what their specific role in the company is *at this moment*.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_1

A tech executive is a person who is in charge of certain aspects of the company's technology and also has an *executive* role—a leadership presence. I find that most people in these positions tend to have the "tech" part nailed down and require help with the "executive" part.

In this chapter, we will cover the most common archetypes and responsibilities of tech executives. Keep in mind that you will find your role is a weighted average of these and no single responsibility is the only "right one." Then, we will delve into the journey you are setting off on and the way you should view your role as a force multiplier. But first, let's discuss why this matters.

The Effective Tech Executive

What's the big difference between a manager and an executive? You're still managing part of the tech organization, only one level higher up the organizational chart, right? Many treat these roles by default as just glorified managers.

The significance of this role stems from the boundless impact it can have on the tech team specifically and the entire business. Good managers create a well-running team that delivers tasks on time and does as it is told. Remarkable executives leverage their organizations to become a hotbed of innovation and novelty that generates clear business value. There's a limit to how many lines of code a team of X engineers can write in a month. However, there's no limit to how much impact the right code, projects, and ideas can generate.

The best leaders I've worked with cultivated teams that were dead set on propelling the business forward and that got their satisfaction from seeing delighted customers. They break silos and turn every rock for novelty and innovation. Their teams are usually those where the employees feel self-actualized and enjoy taking responsibility, ownership, and initiatives. These tech executives foster invaluable teams, not because they are able to hire the best talent in town. They do it because these impact-driven teams tend to have experience elasticity—a year in such a group is not equal to a year in a mediocre organization, thus accelerating everyone's growth.

Achieving such success is never easy or straightforward. It will be tough. There will be ups and downs. You will try things that don't work, and you will be frustrated at times. But, if you're reading this book, you've already shown your commitment to achieving greatness. After all, if you are going to be investing millions of dollars and decades of human effort, don't settle for average.

The Good, the Bad, and the Great

When I first started consulting, I expected most of the help I would deliver to be around very technical aspects: architecture, choosing tech stacks, the right way to do code reviews, and so on. To my surprise, it turned out that my clients needed help with the "softer" parts most of all. Here are some real stories (with changed names), all recent to the time of this writing, that I've seen at clients and that helped shape my thinking about the influence tech executives have on the entire company.

The Momentum Detractors

Great executives help the business as a whole gain more and more speed. Unfortunately, not all executives move the company forward. Some stop progress in its tracks. Others even seem to be trying to push the company backward.

I've worked with CTOs (Chief Technology Officers) who created a negative force field around them. Kind of like the Dementors from *Harry Potter*, they sucked all the energy out of a room. When they felt disempowered, their team started acting powerless as well. Candidates left interviews saying they felt like the CTO didn't even want to be there.

Unmotivated executives, or those driven by the wrong objectives, can also derail progress. They lower the bar by making the entire company used to a very slow, lackadaisical pace, they create silos, or they evoke the need to use politics instead of speaking directly.

I'm certain you're not one of these, dear reader, as you are investing the time to read and become better. They weren't like this when they started, though. In some cases, I had to recommend to the CEO to jettison these executives and find replacements. I am proud to say that all those companies went on to succeed and emerged stronger than ever. I was always happy to hear that the executives often ended up in a place where they had a better fit and were good employees. Everyone can improve, but sometimes a change of scenery is required.

The Glorified Managers

The most prevalent first phase for new executives is the glorified manager phase. These are well-meaning executives, who are often trying out a bunch of different processes and approaches. They read a couple of books or browse Medium and pick up ideas for things to try. They have great potential with some guidance, but without it they often end up doing a lot of busywork that doesn't result in any real progress.

Consider Joe, a first-time VP. He had good instincts when it came to managing individual contributors and put in place some good processes. At some point, he created teams and managers below him without thinking about the way these teams would deliver value and defining what a successful manager would do. That ended up with a lot of back-and-forth, some frustrated employees, and wasted effort. As I started working with him, we realized the need to redo the whole structure four months later.

Those Who Get It

It's easiest to work with clients who are having trouble, but the most rewarding work is helping those who are truly good at what they do become even better. The best executives I work with have a thirst to improve and are eager to get feedback. They invest ample time to thoroughly think about long-term plans. They reinvent themselves and their team as the company grows.

I loved working with both Sam and Danielle because of this. They had completely different backstories. One is a cofounder and past executive, with decades of experience. The other a first-timer and employee. Still, they were so motivated and open to improvement that we rapidly put in place the values, processes, and leadership practices that pulled their teams forward. Both are currently running terrific teams, on track to become world-leading at what they do.

The principles in this book will help you make a similar transition, if you accept the mission and ownership your role entails.

Tech Executives: A Balanced Scale

As far as I am aware, the term "tech executives" is not (yet) in common use. If you've been in the high-tech industry for a while, then surely you've come across CTO groups, VPE (VP Engineering) circles, and general "tech leadership" or "developer leadership" conferences. I started using the term when I saw patterns emerging in my work with these leaders that didn't apply in other layers of the organization.

One thing I appreciate about this term is the fact that it puts squarely into focus the two areas that a person who wants to be successful in these positions should hone. Virtually all tech executives I work with come from a pure tech background—at some point in their near or distant past, they were slinging code for a living. Having a strong tech basis means that it is what these executives then tend to rely on for their instincts and hunches.

Having an intuition based on years of professional experience is priceless. However, letting that intuition drive your decisions when deeper, more thoughtful action is required is to your detriment. I like to use Kahneman's "System 1" and "System 2" concepts from *Thinking, Fast and Slow* to express this. "System 1" is in charge of the fast and automatic thinking we do, those things that you have learned to do by instinct or intuition. "System 2" is used for the slow and logical thinking that's sometimes required.

Years of work in an engineering capacity have trained many first-time executives to rely on their instincts. While this might work for the preponderance of the technological problems and decisions they have to make on a daily basis—their "System 1" is working great—it tends to fall apart when used for leadership concerns and long-term thinking.

Deciding on a promotion, organizational restructure, or your quarterly goals cannot be done wearing the same, technological, hat. When your "System 2" is required, you should think "let's put on my executive hat" and then address the matter at hand. Restructuring your teams is not as easy and clean as refactoring a microservice. Some quarterly goals are not going to be easily defined in a unit test.

In the first few months in a new executive role (even if it is not your first such role), I advise my clients to overcompensate and default to executive thinking. We work together to make them aware of the instances where an executive approach is likely to be required and to reframe the situation in the right manner. This is often done by using their tech background to their benefit.

Sometimes I will describe the different steps for considering a situation as a pseudo-algorithm. In other cases, I make use of analogies to technical situations. Take a second and reconsider the "refactoring an organization" example from the previous paragraph or the title of this book. These are tools I've used hundreds of times, and I saw how they allowed us to leapfrog issues and get to the meat of matters swiftly.

The LEAP Responsibilities

When it comes to leading tech at an organization, there are a few key responsibilities that are manifest in every team in one way or another: Leadership, Evangelism, Architecture, and People (LEAP). Sometimes, they are all the responsibility of a single person (especially at young startups). At others, each might be the role of different executives. And, you may find it surprising, a sole responsibility could be shared between several executives. No matter its manifestation in the organizational chart, the areas of leadership remain the same. See Figure 1-1, which shows the relationship between these responsibilities.

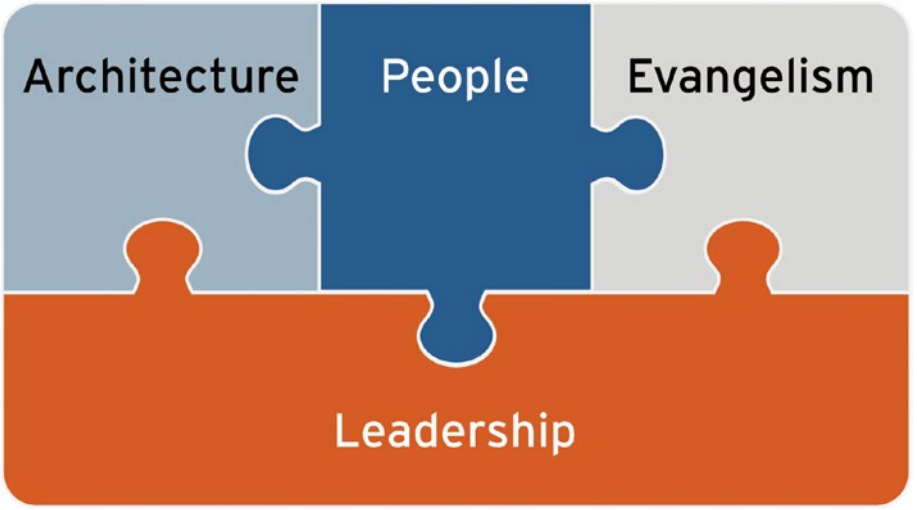

Figure 1-1. LEAP Responsibilities

To ensure clarity, let us quickly go over these responsibilities.

Leadership

All executive roles share a common subset of leadership and ownership, as you can see in Figure 1-1. Being an executive means that whether they are responsible for a part of responsibility or all of them, tech executives break silos and push the company forward. They lead the organization toward a better future by taking part in compiling the company's strategy, defining the shared values and culture, and creating an extraordinary team that is an asset and an innovation catalyst.

Later on, I will address the different aspects that make leadership different from "merely" managing people. For now, keep in mind that leaders should *lead*. They invest the time and effort in defining a wanted vision and future state—even when that currently seems like science fiction—and then show the way forward to their teams. Responding to changes and requirements isn't leading—taking the initiative *is*.

Evangelism: Outward-Facing Tech Delivery

Not all tech leadership should be facing internally to the company and the engineering team. In certain companies, it is an important part of the job description. You must have seen VPs from Apple showcase their new software on WWDC or Facebook executives interviewing about it. Essentially every company has a need to communicate its tech strategy and capabilities outside and sometimes communicate the feedback back to the team.

The most basic example would be explaining the tech, how it works, and its advantages to investors, clients, and prospects. Concerns about the solutions and their viability will come up regularly and should be addressed. Consider the board meeting where a significant tech investment has to be relayed in a manner that will help the board members reach the right decision or the need to explain to a prospective client the safety mechanisms inherent to your approach as part of a security review process.

At other companies, this role is even more prominent, and that is where the company has a decidedly technical part of its business. Typical examples are business-to-developer companies (such as GitHub or Twilio) or platforms whose product includes SDKs or APIs that are consumed by developers outside of the company (e.g., Stripe and Slack).

Architecture: Tech Guidance and Innovation

Complementing the responsibility of execution management, the tech team requires higher-level vision and path-setting to regularly bring novelty into the business, reinvent itself as the business changes, and innovate. Without it, many groups face the common pitfalls of excellent execution, running off toward an unknown goal or losing their position in the market to a disruptive and innovative competitor.

Guidance here refers to setting a strategy for the technological choices in a way that helps the team use the right tool for the right job and striking a balance between bleeding-edge and tried-and-true solutions. Teams solely focused on execution tend to do "more of the same," never trying anything new. They only take the time to readjust their existing tech and solutions when things finally grind to a halt. The onus is on the tech leadership to shine a light on the right way forward and ensure that the existing solutions and processes are being revised regularly.

Unsurprisingly, innovation is another major value mine for engineering organizations. The best teams I have had the privilege of working with are those teams that bring something different to the table. They don't merely execute flawlessly on what they're told. Rather than being siloed and focusing on "doing our job," they strive for understanding the business, market, and customers, so they can come up with innovation: processes, knowledge, solutions, and tools that become assets for the company.

People: Managing Execution

Whatever business you're in, if you're creating tech, then it needs to be managed. This responsibility is vast and has a tremendous impact on the company due to two main factors. First, engineering teams tend to be responsible for one of the biggest line items—if not the lion's share—on the company's budget. Second, tech can make or break the company. These, of course, are two sides of the same coin. The investment only makes sense if it results in rapid value being generated to propel the company forward.

Managing execution has under its umbrella several areas, such as process management, culture, hiring, talent cultivation, and so forth. When you're thinking about tech executives who are "people and process managers," they often fall into this category.

I will refer to these responsibilities in the following by their short names and together as the LEAP responsibilities.

What Would You Say You Do Here?

You might have heard of or even used the term Definition of Done before. The concept behind it is that one should not merely assign a task, but also clearly state how that task will be declared complete—what test should one use to know the work is finished. Taking the extra time to envision the manifestation of what you set out to do has a clear benefit. The same applies for jobs and roles.

When you hire people to your team, you likely have in mind the role and problems that you want them to fill. You have the intention of them taking something "off the plate." After all, it would be a bit silly to hire someone without clearly knowing what is the value that you expect to get from making that addition to the team, right?

Considering Your Role

Whenever I start working with a new client, I ask, "What do you *do* here?" The first answer usually consists of a few ughs, umms, and a blank stare. As you are reading this, some of you are surely realizing that indeed you have never given this question ample thought. Following the introspection you performed in the previous section, think now about your role and how it should manifest in the company. Here are some trigger questions to get your juices flowing:

- What would you define as personal success in 12 months? Imagine the state of the company and your team.
- In a world where all of your direct reports are all extraordinary in their roles, what would you be focused on?
- What kind of impact do you think you are best suited to have on the company?
- If you were to get stranded on a deserted island with no Internet connection for a couple of weeks, which processes are justified in getting delayed? What are the matters and decisions that you would still be the best person to make when you get back?

In the next chapter, we will also go through the steps of collecting your most pressing responsibilities and value-generating activities from your boss and peers. In the meantime, consider the preceding questions to paint a picture of the kind of leader you would like to be.

Axioms of Effective Leadership

Here are some of the principles for tech leadership that you should be using to consider your role. Whenever you feel at a loss, I commend you to review them again and work your way from the basics.

Executive Leadership Is Long Term

The first difference between merely managing a team and being an executive is that leadership at this level requires investing time in strategy and vision. While even team leaders should not be completely obsessed about short-term work, doing so at an executive level is doing a disservice to your company, and it might be fatal for your success. Tech executives, no matter which of the responsibilities they hold, are in charge of steering the ship and routinely lifting their head to gaze at the distance and decide on the path forward.

Many tech people default to "gamifying" themselves by focusing on getting tasks done as fast as possible, a remnant of what likely made them good engineers to begin with. However, leadership is about the vigilant pursuit of the "why." Why this task and not another? Why now and not next quarter? Why is this the best bet currently to help us achieve our vision?

Technology as Innovative Asset

I don't know about you, but I fell in love with coding when I was nine years old because I was enamored with the idea of creating something out of nothing. If you can imagine it, you can create it. Technology teams should be constant providers of innovation to the business, regularly creating more and more assets that yield a positive return.

Your role as an executive is to take care of this business-innovation bond and ensure that it produces ongoing value, rather than becoming a liability. Your peers in the company are not likely to be able to envision what can be made possible with your team's capabilities. It is up to you to make it known and "take the initiative" as chess players say.

I was facilitating a workshop for CTOs about the tech executive operating system. When discussing this principle, I provided an example of a client. The engineering team had taken the initiative to suggest a basic change: changing an old system to do its processing in a parallel way as opposed to the current serial way. No one in the product team knew this was possible. It resulted in a 10× improvement of processing times and allowed more to be done when processing, thus improving the company's bottom line and creating enough margin to provide clients with more offerings. On top of all of that, this gave a solid business case to make the investment in an architectural shift the engineers were craving for.

After describing this, a bright CTO objected. "How can you expect me to be able to come up with these ideas? I cannot be aware of every single line of code and its implications." He missed the point! As you're considering this principle, do not think that it puts the onus on you to be an oracle that comes up with a new trick every leadership meeting. This is about you creating an environment where such ideas are a regular occurrence and helping the team prioritize, experiment, and filter these ideas to quickly execute on the successful ones.

Coders Without Borders

The best teams are those that work well together. Rather than having a cadre of individuals all working on their isolated tasks, a *team* has teamwork where each person helps the other. The right mentality is that of winning (or failing) together. It doesn't matter if the mobile engineer got the app ready on time if their teammate, the backend engineer, wouldn't be ready. Teams have ownership of the entire stack of their product. Team members help one another and, for example, fix bugs not directly under their purview simply because that's the right thing to do.

You should take this mindset a step further. The best tech leadership is directed at enabling success for the entire business. Don't dismiss what other departments are doing as less critical, easier, or voodoo (a particularly common way for engineers to describe sales and marketing). Instead, use your position as an executive to create a culture where ownership cuts across departments—coders without borders. Your team can create rapid momentum when it doesn't point fingers and lends a hand.

Great Teams Are Made of Layers of Force Multipliers

I think I first picked up the term "force multiplier" during my service in the Israel Defense Forces (IDF). Military jargon ran so deep we started describing new tech and capabilities as force multipliers that would allow us to retain the crucial edge we needed to be able to supply life-saving intelligence regularly.

Later, I realized that every group is a living organism filled with more and more force multipliers. You've heard of the fabled "10× engineer" in the past. While some engineers might be able to bang out code rapidly, the real 10× (or even 20× or 100×) engineers are those who can strengthen the others around them. Similar to Andy Grove's concept of leverage in *High Output Management*, I believe that each person has the responsibility to act as a force multiplier for their colleagues.

Individual contributors can help one another by speaking up, taking ownership, mentoring, and so on. Managers help their direct reports grow and reach self-actualization and work along their peers in different departments to generate rapid impact. Executives like yourself cultivate cultures of excellence, compile a strategy and a vision, and act as a positive force (see "Executive Mindset" in Chapter 5).

This mental model, where your organizational chart has a multiplier written next to each person and these are all combined together, can help you understand the vast opportunity that lays ahead if you set out to create a world-class team.

Naming Things Is Hard

You probably have already realized by now that there's a plethora of titles for tech executives. Microsoft has a CTO, and so do day-old startups—they are not the same. To make matters worse, I've seen people using the exact same title for entirely different roles. CTOs at small startups often are those in charge of the *People* responsibility and not much more. Others might have a CTO and VP from day one, with a not so clear distinction regarding who does what.

You can find a detailed breakdown of the different titles in the Appendix. For now, I will briefly mention the three most common titles to create a shared vocabulary for us:

- **CTO**: The most common and vaguest of all titles. Often refers to an executive holding all LEAP responsibilities. If there is a VP or Chief Architect in the company, CTOs tend to hold the responsibilities that will be left over.
- **VP Engineering/R&D**: The executive in charge of the *People* responsibility: putting processes in place, deciding on team structure, and growing and hiring the talent.
- **Chief Architect**: An executive who's in charge of the *Architecture* responsibility. Sometimes has a team of senior architects and at other companies has no direct reports.

This will help you understand how to name the position you're currently doing (if you are free to choose your title), as well as communicate more efficiently with potential hires, investors, and so on, where you describe your organizational structure and responsibilities. As you realize that your responsibilities are shifting, you might need to revise your title.

For brevity, I will refer to a "tech executive" as any role of the preceding titles that holds executive responsibility unless specified otherwise. Further, VP Engineering (VPE for short) will be used to discuss a person mainly in charge of the *People* responsibility.

Your Personal Upgrade

You now know the typical responsibilities of tech executives, understand where you are currently situated, and should be able to envision why the journey to become a higher force multiplier is worthwhile. You might be just starting a new company, doing your first steps as an executive in an existing team, or looking to improve your game. These are the processes I've used with executives of day-old startups and unicorns.

Your personal knowledge, habits, and processes obscure critical issues or fantastic opportunities if not maintained regularly. In the next chapter, you will create a crash course for bolstering your control of the product. Hang on tight!

CHAPTER

2

Executive Product Mastery

In which you put aside the tech and learn the business

In this chapter, we start by uncovering what it takes to be an extraordinary tech executive in *your* company. As every company is different, you should find a path that best fits where you are currently at and where you want to get to. First, you will gather objectives from your boss. Then, you will design a personal crash course to become proficient in the product and business.

Product Mastery at the Executive Level

Product mastery is a term I coined for the skill set that indicates a professional whose expertise and knowledge does not merely cover their professional domain (e.g., a backend engineer or business analyst), but also specific areas regarding the business and company that they are in right now. Cultivating

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_2

high product mastery is one of the ways to accelerate the talent growth of your team, and we will cover that in depth in Chapter 10. In this chapter, though, we will focus on how this same concept manifests at your level in the organization.

We will go through the various aspects that executives should become familiar with, detail some of the mindset shifts that might be necessary both for you and your boss, and help you enrich the definition of your role.

How Will You Be Measured?

As with every project and task you take on, it is beneficial to take a second at the outset to envision what success looks like and how would you tell that you've accomplished what you set out to do.

A Successful Project

When we debrief on a project, we often consider different aspects. After it is done, we might discuss whether our quality bar was adhered to or not. Maybe the team will point out problematic milestones that were defined earlier or areas where bringing external experts might have been better off than doing things in-house. Talking about the satisfaction, growth, and burnout of the team is of great importance as well. Otherwise, you risk this being your last project.

And still, the first thing such a retrospective should address is whether the project has achieved its objectives or not. After all, the most beautiful, efficient, and clever piece of code is worthless if it solves the wrong problem. The same, dear reader, applies to you. Setting out those objectives clearly beforehand is the key for your success. Think of it as a mini premortem.

The CEO Meeting

In the previous chapter, you started thinking about the ways you will measure yourself, your goals, and your objectives. Those are terrific to have in mind and are essential. No one else is likely to tell you that your team has to be satisfied and growing personally, but you know you are not likely to succeed without achieving that. Despite that, the objective of the position—your own position—is to achieve or enable one or several business goals. No one employs dozens of engineers for the sake of having code be written, but for the business outcomes that code enables.

That is why I advise the kick-start of every job or promotion to entail a one-on-one meeting with your direct manager to openly discuss how the role will be measured. I will refer to this manager as the CEO, even though you might be reporting to someone else (e.g., the COO). In this meeting, you should sketch together the most important aspects of your position. I recommend doing this even if you and your boss are cofounders or if you think you've discussed this prior to accepting the job.

Here are some questions to help you tease out the gist of it, phrased in the language you might use:

- What will you consider as my personal success in this position in 12 months (or 6 or 18, whatever makes sense for you)?
- Why did you state those objectives as the most crucial?
- How can I take things off your plate?
- How should the company values manifest in the executive team?
- What is the most pressing matter to start addressing?
- If we got free money tomorrow, where should I invest it in my organization?
- What, in your opinion, are the bright spots in my organization—those exemplars who are working remarkably well?
- Which of my peer executives will get the most significant improvement from my team helping them directly?
- Is there any problem or project that you are aware of and think I should ignore for the time being or shut down completely?
- What has been my team's biggest or most surprising success in the past year?

These kinds of open and candid discussions with your manager are essential for you to focus on the most important places of the business effectively and then move upstream.

Grokking the Business

The next phase toward executive product mastery is, well, to master the product, the market, and the business. You have an executive role in your specific company, and thus you should possess the relevant background and knowledge for it. The actions we will describe now should be mandatory for all executives in a new position, regardless if this is their first time in such a role or the fifth.

This section will cover the topics that you need to learn to gain the background required. You can see it as the syllabus for your personal onboarding. In my experience, executives rarely get thorough guidance regarding their onboarding plan and are left to create it by themselves. Consider this your crash course, with the intent of getting the proficient background to get up and running within a few days.

The crash course will first go through understanding the company itself: the company's values and vision, the business model, and preliminary company history. Next, it will cover the users, their problem, and the offered solution.

Do not expect to become fluent with all the relevant terms and the industry's standards. Also, you will not become an expert in all these areas immediately. That comes with time. The purpose of the crash course is to expose you, make you aware of what you don't know, and provide you with the tools to research certain aspects deeper as necessary.

The Company's Gestalt

If you are joining a company, some of these areas might already be known as part of your interview process and the due diligence both parties did to ensure you have a cultural fit. Still, there is value in repeating this now that you are part of the team. These conversations will likely take place with the founders, CEO, executives, and the company's founding members.

If you're starting up a company, congratulations! It is (partly) up to you to come up with these answers and definitions. The whole picture won't become evident within a few days, but rather years. These first few days should be used to sketch answers and guidance to keep you on the right path for the next three to six months.

Values, Vision, Mission

Some companies have these terms already nicely defined and laid out with fancy graphics for everyone to see on the office walls. Others might have more of a hand-wavy response. Whatever the case in yours, you should know and understand these details by heart. They should be the backbone of the organization and your North Star for decisions.

An exercise I sometimes do with clients when they detail their vision or values is to ask how these have manifested in the company's decisions. Are these merely a dead letter, read once during one's onboarding and forgotten? Or are they regularly referred to during strategy and roadmapping meetings, shaping the culture, and instilling focus (e.g., dropping a potential client because it does not match the company's values or long-term vision)?

Business Model and Health

Business health metrics change somewhat between different businesses and industries. Still, I would say that you need to have an understanding of the company's financial situation (e.g., what is your current runway and burn rate), earnings, and different client metrics such as acquisition, lifetime value, and churn. Business resilience is also essential to be aware of: does the company have a whale client bringing in 80% of its revenue?

Most companies have clear metrics and KPIs (Key Performance Indicators) that are mentioned during board meetings and quarterly roadmaps. Understand what these are for your company, why they are important, and how they get measured, and look at the past trend of these measurements.

The business model might not be crystal clear for many startups (frankly, we have seen some companies going public without having this part nailed down). Whether the business model is an actual one already deployed or a hypothetical one, understand the thinking behind it and the effects it has on the company as a whole. Some examples would be that user experience is often treated differently at B2C companies than it is at B2B companies, the added complexity that comes with a marketplace business model, or whether the company has a high-touch or low-touch sales approach.

History

Get a succinct summary of the company's history. You want to have a rough picture of

- **Timeline**: When was the company started, how many people it had every year, the timing of big deals or venture rounds, and so forth. This helps you gain another dimension in your discussions—the time axis—so you can have a general feeling of how old ago what you are talking about is and an assessment of the company's velocity and acceleration.
- **Business changes**: Has the company pivoted, changed markets, or gotten exclusive contracts? Every incarnation of previous business models and approaches likely lives on in the organization in several forms. You find it echoing in your structure, your code, and your budgets.

- **Key personnel changes**: Another kind of business ghost is the effect of prior promotions, demotions, and reorgs. The cofounder who was CEO and is now the CPO might still be treated so by employees. The team that already had three waves of layoffs in previous years will be wary of any change.
- **Defining moments**: An amalgamation of the previous points combined with other interesting twists and turns in the company's story. Was there a critical security breach that changed the way data is handled forever? A massive spike in usage happened because of some random event and changed the product's vision for the following six months? All are good to be aware of.

It is a bit dangerous to try and cover history in a crash course because we humans are very eager to discuss the past. Do not go on an archeological excavation. Also, while you may be curious and have a drive to "know it all," you are better off only getting the most important bits. That's because you want to retain your fresh perspective and not get bogged down by the past failures and internalized excuses the rest of the team has. However, you will end up hearing something that seems abnormal, harsh, or just surprising. In those situations, I have a strong recommendation: get another person's perspective on the matter. A bit like a journalist verifying a story, this provides more depth to your understanding.

These aspects should help you have a more holistic understanding of the business. Gaining most of this knowledge can be done with a few sessions with the relevant people and a specific agenda to direct the conversation efficiently. Next, you need to understand your users.

Why, Who, and How?

Peter Drucker, known as the "Father of Modern Management," said that the purpose of a business is to create and keep a customer. Such a customer-centered approach is healthy for you and your entire team to possess. To achieve that, you are required to do your homework and understand who your market is, why do they need your product, and how does your product effectively solve their problems. As opposed to the previous section, here you should not merely sit and listen, but also roll up your sleeves and experience things for yourself.

This part of your crash course is intended to arm you with intimate knowledge about your users, market, and competition. With this knowledge, you will make better, smarter decisions and be able to place your chips on the innovative ideas that are more likely to pay off.

The Problem

Every product exists to solve a problem or answer a need. Surely you already know in broad strokes what problem the company is solving and might even be intimate with the mechanisms of the technical aspects of that solution. That does not always promise that you have gained a real understanding of the situation from your users' viewpoint.

The following are a few ideas to learn this quickly:

Testimonials: Go over accounts of those affected by the issues and problems that your company is solving (or is intending to solve in the future). You might already have access to these in the form of collected testimonials from existing users and reviews of your competitors. In some cases, you can even find people talking about the problem on social media. You'd be surprised what a quick "X sucks" search on Twitter could turn up.

Reports: The problem might be well documented in different studies. While it might seem intimidating, I believe that combating these sorts of research documents by yourself will allow you to get a lot of background that just would be lost if you'd get the CliffsNotes from a colleague.

Competition and rival solutions: Most companies are not created to solve a new problem, but solve it in a new way. That means that other solutions are already out there, and reviewing them is a great learning experience. Going over their historical marketing material, earnings reports, and other collateral will shed light on the problem from another perspective.

Dogfooding: Whenever possible, I highly recommend "eating your own dogfood" to witness the problem firsthand. Sometimes, often for B2C companies, doing it is trivial and doesn't require any special efforts. In other situations, you might have to simulate it as, for example, most people don't have an old mainframe at home to see how backing it up to the cloud is a hassle. It is practically always worth it to invest in setting up such an environment to experience the issue, and we will return to this later when discussing product mastery in Chapter 10.

The Customers

This aspect of understanding the business has, naturally, some overlap with understanding the problem. However, the critical elements are to understand how overarching the problem you're solving is, the different types of users you have, and their personas.

The Market

This entails any information you can get access to that can shed light on how widespread the problem is and its history. How fast is the market growing, are there specific areas where it is growing faster or slower, is it a global or regional problem, and similar questions.

Whenever possible, I also recommend getting familiar with the evolution of the market. Does your company expect new markets to become available in the future due to advances in technology, regulation, or the development of countries? Why so?

Users vs. Buyers

A distinction that mainly applies to B2B products is that whoever that makes the purchase decision is not always the actual user of the product. For example, the company might be focusing on selling products to executives in enterprises who are responsible for regulation or compliance. However, the tool, once installed, will be in use by different departments. Such a pattern does not usually take place for B2C products where the end user is also the buyer.

When these two groups are not the same, it has effects on the product and different decisions. It might help you understand why the company is (or should stop) investing in some "demo features"—features that are only used when selling but not actually useful for the end clients. It might also surface areas where your user experience or solution is lacking because there is not enough visibility of who is using the product in the field.

Become accustomed with the titles, departments, and role definitions of your users and buyers. You should be using the right terms and jargon when describing what the company does. Such knowledge is important for all tech executives but vital for those holding outward-facing responsibilities.

The Solution

Lastly, your crash course comes to the understanding of your product and how it solves the problem you researched for the people you have just gotten to know. If you are a cofounder or have already been in the company for a while, you might be under the impression that you have this part already covered. Bear with me for a few paragraphs and see if that belief still holds. Product mastery is not merely about understanding the code, but understanding the intention behind it.

Product Walk-Through

You should be very familiar with the workings of your existing product and your approach. That is not to say that at any moment in the next four years, you should have the ability to recite by heart what every single knob and toggle in your product affects. It does mean that you should be proficient with the normal usage of your product and make sure that knowledge remains up to date as the product evolves.

Take the time to go through the onboarding process for your product and see how it solves the problem for your users. What parts of it are error-prone? Are there areas that you find particularly rough, smooth, or otherwise worth further research?

Without familiarity with the product and its inner workings, you won't be able to make fast decisions, spot errors, or even hire people who would be motivated to work on your team.

Competition

Whenever possible, I highly recommend taking the time to experience how your competition fares as well. It is good to be able to see what the incumbent is doing and how their process looks so that you can see where they are inferior compared to your approach. This knowledge also allows you as a tech executive to communicate fluently with customers and prospects in case your role includes outward-facing responsibilities.

Current Approach

What assumptions is your current approach reliant on? At earlier stages, it is typical to have solutions that are based on parts that "do not scale." If that is the case, you should understand how those existing solutions behave and what the plans are regarding their future replacement. If the evolution of the required technology is within your purview, ensure that you will have time soon to start planning the needed steps.

Research Your Pipeline

If your company has marketing and sales organizations, learn how they operate. What are the stages a prospective client goes through? Who are the touch points and how does the demo normally look like? What happens when someone decides to stop using your service?

I recommend joining a few sales calls, making sure that you are invited to the next Quarterly Business Review (QBR) sessions, and attending some churn calls.

Shadowing Customer Success

The last approach to understanding your solution—and, in fact, to understanding the problem and the users better as well—is to invest the time and experience firsthand how customer success or customer support (CS) for the product looks like. I recommend at least shadowing them, which means that you accompany someone on that team for several half-day sessions. During that time, you get to see what issues are being brought up repeatedly and what tools CS has to address these and gain insights about the day-to-day workings of your product and company.

Remember, this is not about you being able to replace your company's head of product. It is entirely acceptable if you do not remember all the numbers and different terms by heart within your first month on the job. However, don't allow that to water down your education. As you sit down with your peers to gather this knowledge, inquire, and ask questions. Just because there won't be a test doesn't mean you shouldn't give it your best.

Quick Self-Assessment

Congratulations on compiling a personal introductory course! Once you set it into motion, use these questions to assess how well you are doing. You should be able to answer the majority of these questions good enough to make sense in a media piece or if asked by a knowledgeable candidate in an interview:

- Who are the ideal users of your product?
- Why does the problem you're solving matter to them?
- How is your approach different than the others out there?
- What edge does your technology provide?
- Which features are currently implemented and working?

- Which features are most requested or slated for the next releases?
- What is the company's long-term vision?
- How have the company's values affected its approach so far?
- What are the most pressing issues for the business or assumptions that need to be validated?
- Which of your current value offerings have the most traction or impact on the company?

Maintaining Your Product Mastery

Companies move fast. Things change constantly. Whatever you knew might no longer be relevant in a couple of months. Right now, after following these steps, you have a solid mental model of the product. The second you move on to focus on other things, parts of that model will start getting stale and foggy. However, you do not need to repeat all of these steps regularly—you will have exposure to at least some of the relevant changes as part of your job.

Put this book aside and add a task to your personal task system (or an event to your calendar) to refresh your product mastery in three months. Make it recurring. When the time comes, review this section and pick some areas where a refresher would be worthwhile. Maybe you will feel the need to shadow customer success for half a day. Perhaps a new product walk-through is in order—after all, since the last one, your team has shipped a mobile app. Every quarterly refresher can and should be different and tailored to provide you anew with a crisp mental model.

Your Personal Upgrade

You should now be equipped with a few key factors to set yourself up for success: knowing how your boss will measure you and having a plan for acquiring mastery in your product. Just reading through this chapter and contemplating the ideas is not enough. Just as you should not tolerate your employees spending hours in planning meetings without making any decisions, you should not read this without doing the work.

The product mastery onboarding is relevant for all executives and usually requires three to four half days. It often goes hand in hand with getting allies, as mentioned earlier. Get it on the calendar as soon as possible.

The next chapter will help you plan and achieve success on your first 100 days on the job.

CHAPTER

3

First 100 Days

Booting up your operating system

The previous chapter detailed the steps required to create a quick crash course for your product mastery. Having a grasp of the business is a necessity, but it is not enough to guarantee success. This chapter is a direct continuation of the process of setting you up on the right path during your first 100 days in your new role.

I have seen the onboarding of new executives in charge of two developers or the entire engineering department of a unicorn. Either way, it is a rough ride. The sections detailed in the following will help you know what to anticipate, form your plan, and start addressing the most pressing issues.

What's Special About the First 100 Days?

I will admit that the number 100 is arbitrary. However, it is commonly referred to as the period of time new leaders are given to show their first results. Furthermore, experience has taught me that this is roughly enough time to learn the organization, put things into motion, and start seeing results. Specifically, 100 days should promise that you will participate in at least one quarterly planning process, which is paramount for leading an organization.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_3

Setting Expectations

Regardless of the state your organization was in when you took over, do not expect to see a complete 180-degree change within a few months. One of the reasons that the executive role is so vastly important is because executives lend a part of their character to be fused into the culture. Any complete transformation is going to take a while.

However, you should aim to show positive results within this time frame. Defining what those results would be is up to you and your boss, some of which might have come up when you were following the steps in the previous chapter. These goals are likely to be accompanied by emergencies—lots of them.

It is very typical to feel as if you are spending entire days putting out fires when you are new to your role. I will tell you a little secret: too many executives spend their careers doing just that. If you are taking over an organization—either replacing a predecessor or the first tech executive in charge of it—you will hear many stories and have issues brought up almost every single meeting. The trick is to learn to manage them.

You will not solve everything in a hundred days. You won't even know *how* to address all the issues. You will, however, triage the most critical aspects and start forming a vision and a roadmap.

The Window of Malleability

Another useful aspect of your first 100 days or so is that it is an instantiation of a unique period, which I call a *Window of Malleability*. These windows are time frames where certain people are more receptive to change or have a fresher perspective. During such periods, there's a sense of novelty that makes the relevant people more aware and more intentional in their behavior.

Imagine that you just bought a new car. In the first week, you still have to actively think for a few seconds when you use one of its features. In the first month, you are still finding more nylon wrappers on surfaces. For a couple of months, there is still a distinctly new smell to it. Eventually, it all starts to wear off. You no longer have a "new" car.

The same applies to new hires or changes in position. At the start of your role, you are more likely to pick up on issues and notice cognitive dissonances. A few months later, you will get used to them and no longer be able to detect them that easily. Furthermore, during this time, the people around you will be more receptive to the manner in which you manage your day and the model you give. Providing people feedback right now would be easier than it would be if you decided to "wait it out" until the first six months are over.

That is not to say that you have to get all of the changes you are thinking of within your first 100 days. That's not possible nor even advisable. But the standards you set and initial moves you make will determine how easy the next ones will be to initiate as well. Keep that in mind and take on every day with the explicit intention of noticing what doesn't make sense and modeling and demanding the values you expect to see in your team.

Your Goals

The layout of your first 100 days is detailed in Figure 3-1. As you read this chapter, and as you go through this period, keep in mind what you want to have by the end of it. Often, I advise clients to dedicate this time to achieve the following:

- Build relationships with their employees and colleagues.
- Unearth the most pressing issues.
- Form a roadmap for the next year or so.
- Assess the culture that's currently in place.
- Kick off critical hiring/firing decisions.

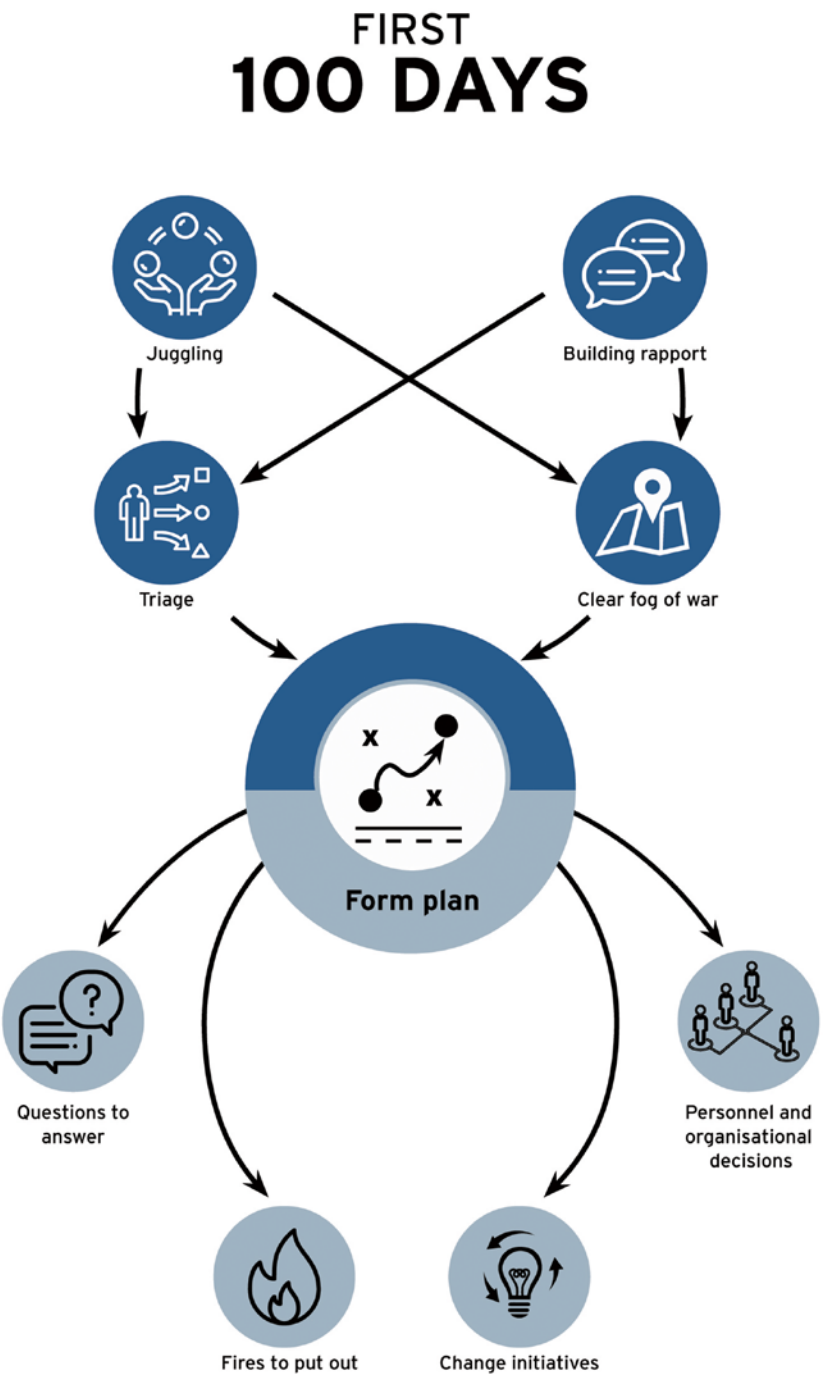

Figure 3-1. The Layout of Your First 100 Days

You, Juggler

All leaders have days where it seems like every chat they have and every email they receive are piling on more and more issues on their plates. Maybe someone grabbed you in the hall between meetings, and you get the feeling they are unhappy. You hear in passing that a project might not be progressing right. You are updated about an outage that happened overnight. Each issue is like another ball you have to keep in the air.

Initial Overwhelm

When all these issues start surfacing, it is natural to feel overwhelmed or sense stress stirring up. My clients often describe this as fearing they have been tricked—there are plenty of things that are harder than they seemed they would be just a couple of weeks ago. In my experience, though, just about every executive position looks like this. There's no trickery, only the human tendency to focus on the positive things and the good old "unknown unknowns" moving into the "known unknowns" category.

Do not succumb to this drowning feeling. Yes, there are challenges ahead. That is why you get paid the (figurative) big bucks. Where there is no risk and the path is clear, an executive is not called for. This is what you signed up for, and you can do it. You have already committed to becoming an exceptional executive and are investing in yourself to get better by reading this book. I will lay out a clear system for you to manage the juggling.

Don't Rush into Things

A lot of managers try to solve everything on the spot out of instinct. You hear about an issue, and you let it detract the entire meeting. You think you should have the right solution off the top of your head. While such an approach might work for a while when you're in charge of a small and new team, it is not the way to go forward as an executive. You should be prepared. Expect to get a constant stream of ideas, suggestions, and issues in the first days.

Always keep in mind what your current obligations are, what the purpose of the meeting you're in is, and your agenda. Do not let any stray comment hijack a conversation. I find this is similar to meditating. Just as you have to train your mind to become aware of when it wanders off in thought, you should train yourself to notice when you digress from your plans. In virtually all cases, you do not need to drop everything else you are doing to address the issue right this second.

Triage

You want to take charge of your day. That requires balancing not getting sucked into handling issues left and right while also not dropping the ball on anything important. One of the most important traits of leaders is that they are seen as accountable. When issues are brought to their attention, the team needs to trust that they will remember them. Being responsible and living up to your commitments is a required step to demand the same from your team. This is part of the modeling I described earlier.

You should have a triage process for handling these issues. Rather than rushing to put out fires, assess which fires genuinely require your involvement and the appropriate timing to manage them.

Write It Down

My suggestion for you might seem trivial, but every executive I coached who picked up this habit swears by it: write it down. Whenever someone grabs you or a thought strikes you, do not dive into it. Ensure that you will keep track of the issue and move along. The mechanism that you use to manage these is entirely up to you, and I advise you to try a few things until you find something that does not feel burdensome and that you are capable of maintaining long term.

Those who know me are used to seeing me with a notebook and a fancy fountain pen in hand. For me, the analog system works best to write things down and remember them. Others just use their phone's notes app or have a fancy getting-things-done sort of app. You can tinker later but start with choosing something and using it.

Decide on Next Steps

Routinely, preferably at least once a day, go over the issues you've collected and review them. Treat this like personal backlog trimming meetings, between you and yourself. For each problem, consider the following questions to decide on the way forward:

- Is this still an issue? Often things that seemed problematic at first don't seem that way after a couple of hours have passed.
- Are you the one that should be handling this issue, or should you delegate it to someone else and track its progress?

- Do you have enough information about the topic? Sometimes it is beneficial to get a "second opinion" and hear what others involved are thinking before committing to any specific solution.
- What is the severity of the issue?
- What is the urgency of the problem? Do you have to treat it right now, or can you postpone the decision to a later point where it makes more sense?
- Is this a matter where a decision can be easily reversed?

Armed with these, you make a decision regarding how to handle the issue. Maybe it should be added to the agenda of your next 1:1 with an employee. Perhaps it is a good idea to hold a brainstorming session about it with your senior staff. Know what you are doing next so you can commit to doing that and move along.

Again, make sure this session happens routinely. It might be daily, weekly, or twice a day—I've seen all of these work for different people. But you have to make a habit of doing it regularly. Again, you might benefit from a recurring event on your calendar or another solution. I live and die by an app called Due that keeps nagging me every few minutes until I tick off a task.

If this feels like Personal Time Management 101, that's because it is! Out of all the executives I've coached, I can say that no more than 10% had these habits nailed down when we started. You know the saying about the cobbler's children. Put aside your knee-jerk reaction and consider how well you perform these steps currently.

Follow Up

The triage cycle starts with writing down an issue and only ends when you finally can declare the matter handled. That is because following through on our promises is crucial for building trust, as described earlier. Most of us have worked with people who would regularly say they will do something and then forget about it. Eventually, their colleagues learn to work around them and not count on their word—not a fine example of executive leadership.

That is why you should track your accountabilities, follow up, and update whoever brought it to your attention. You might tell them you addressed it or that you decided it did not require any further investment at this point. No matter what the final status is, be sure to close the loop and let them know. Now you can cross it off your list.

I used to work with a leader who was frustrated with how he was regarded. The team did not have a lot of trust in his accountability, and that would result in rework as others also handled what the leader was responsible for, fearing he'll forget it.

He told me, "I never forget anything. Here, look at all the things I crossed off just in the last two weeks!" He really had this nailed down. Everything was orderly. He systematized going over his tasks perfectly. I asked, "If you never forget, what are they saying when you let them know you finished stuff?" The question was answered with a blank stare. It turns out he rarely let them know when things were done. That naturally leads people to assume he forgets things! Your personal Definition of Done should often include letting people know that you're done.

Putting Fires Out

Lastly, there's a caveat for this process. Hospital triage, as you likely have seen on TV, is used to act immediately where that is required. If you commit everything you hear to your notebook and review it weekly, you might only come back to an issue after it's too late.

Picking up the urgency of issues quickly is not always easy, and it is something that will become easier with time. Every person on your team will have a different threshold for escalating. Some will update you about every minor hiccup, while others only surface when things are in grave danger. As you get to know them better, you will see how each one's rheostat is configured and how to respond. With time, it might make sense to formalize these escalations. Create company-wide severity levels, and train the staff about triaging themselves.

Furthermore, some matters are too important to delay. Outages, major client blockers, and similar external problems are easy to spot. You should also listen to your "spidey sense" when someone lets you know they need to talk. It might seem casual when someone asks you as you walk the hall, "When will you have five minutes?" Most times, telling them to get back to you after lunch or whatever is fine. However, you would be surprised how many conversations start this way that end up being a big issue: a key employee getting job offers or an individual contributor who is displeased with their manager.

When you think you might be facing such an issue, it is acceptable to invest some time as soon as possible to assess the urgency of the problem quickly. Keep in mind that this should be the exception: the vast majority of cases should not require derailing your day.

Making Changes

As your first days in your new role progress and the juggling balls keep piling on with more and more issues, your triage queue will get longer. You already know that you should follow through on the issues, but some, especially the more pressing ones, will require putting changes in place. Maybe the team's on-call schedule is not done correctly. Or the feedback culture is out of whack, and you'd like to start changing that to align it with the company's values.

Rolling out successful change initiatives is an art. It requires deliberate planning, communication, and execution. That is why I dedicate Chapter 7 solely to doing just that. In this section, I will help you decide on which changes should be addressed during your first 100 days and how to communicate them. For the actual change steps, I commend you to that chapter.

Get to the Root Cause

We all know that you should handle the underlying causes of issues rather than the more readily identifiable symptoms. You can decide to make planning sessions longer because your managers complain the current 90-minute meetings are not enough to go over the issues. That might solve the symptom. But the root cause might be that people come to the meetings unprepared[1] or that the communication is not efficient.

Influential leaders draw out the real problems using Socratic questioning and collecting evidence.

Socratic Questioning

When investigating an issue and attempting to get to its root cause, we often rush into conclusions. You cannot be satisfied by hearing a single, shallow explanation. Also, you should not be pushing too much of your instincts and insights before you have let the other people express themselves. That is because you, as the leader, might sway their thinking.

Practice fundamental Socratic questioning to drive the conversation forward. This feels less robotic than the "five why's" method and leads to genuine discussion. Some common questions might be

- When you say "X," what exactly do you mean?
- Why do you think that is the case?

[1] See more in Chapter 5

- What other possible explanations are there?
- If I'd ask the other person, what would they say?
- What do you think would happen if we addressed this?
- How did this problem come to occur in the first place?

You are challenging beliefs, assumptions, and conclusions—both for the person you are talking to and for yourself. Words are tricky, and some extra digging is worth it.

Collecting Evidence

Socratic questioning is not always enough in order to initiate a change. That is because it is based solely on the thoughts and beliefs of those you talked to. The next step is to gain proof that things are as you have heard. We are not talking about evidence that could hold up in court, mind you. The purpose is to make sure that you are not rushing into action due to a misunderstanding or exaggeration.

When discussing matters between people, hear out all parties. If there are issues with processes and meetings, like the planning session example earlier, attend the meetings yourself to witness what is going on. Collect data and metrics like the number of defects software releases introduce, hiring pipeline conversions, and so forth.

Equipped with these numbers, you can make sure that whatever conclusion you reach and the steps you decide to take are guaranteed to address the root cause of the issue.

Changes as Tests

Let us get one thing clear here as you start making changes. Both you and your team must understand the nature of changes. Everyone involved should align their mindset around the concept of a Change Culture.

A Change Culture is a culture that understands that reassessing how things are being done regularly is worthwhile and a requirement for long-term success. You don't wait to try new things until your current process falls apart entirely and the whole team grinds to a halt. Instead, it is perfectly fine to try new approaches and ideas to keep the team agile. Once you stop evolving, you stop learning. That means that you might kick off some processes and the people involved should not see it as a sign that they were doing anything wrong necessarily.

Furthermore, the fact that you want to put in place a particular change does not mean that whatever you will try is going to work out. You might realize it has improvements, but it would still need more work. Or you could quickly realize it seemed like a good idea, but it doesn't work in practice. As with every challenge, if you always succeed in everything you take on, you are in your comfort zone.

That is because it is vital to introduce most changes as tests. Don't allow your plans to be interpreted as hailing "The king is dead. Long live the king." The reason is twofold. First, some people on your team surely liked the old ways of doing things and shouldn't feel ashamed for it. Second, if you eventually decide you want to make additional changes to the new approach or roll it back completely, you don't want to lose your team's trust and buy-in.

Treating changes, especially the initial ones, as tests in an outspoken manner reduces the chances of pushback, now or later. As time passes, keep in mind the need to update everyone involved about the progress. If you see the positive results you had wished for or if there is no noticeable improvement, they should know.

The process should always include these steps:

1. **Declare:** Whenever a new change is kick-started, let everyone involved know that this is a test. Communicate a clear goal or hypothesis. Inform them how success will be measured. When possible, set a deadline for the experiment in advance.
2. **Monitor:** As the experiment is ongoing, keep an eye on its progress and perform tweaks as necessary. Do note that if the adjustments you are making are radical, it might be better to declare the experiment as done and start a new one.
3. **Assess:** At the end of the declared period of time, evaluate how the test has performed. Are you better off than you were when it started? Were the objectives achieved?
4. **Apprise:** Let everyone involved know that the experiment has ended, what its result was, and what the next steps would be (e.g., more tests).

Pulling Up Your Sleeves

Another critical target for your first weeks should be to get to know a lot of the people, processes, and projects better. Your schedule will naturally start filling with meetings and syncs, and those will help. However, experience shows that swiftly getting "down to business" helps executives gain momentum and builds trust faster.

Building Rapport

Creating real relationships with your peers and direct reports is extremely important. Getting to know your entire organization is also essential. We humans have a hard time trusting people we haven't had real conversations with. Also, the amount of information that you pick up from having even a short discussion with someone is tremendous: people's assertiveness, language, fears, and more all bake into the company's culture. You have to experience it firsthand.

To do so, meet them! Start up your calendar and start firing away those invites.

One-on-Ones with Your Directs

Schedule get-to-know personal meetings with all of your direct reports as soon as the schedule allows. I recommend the initial meetings to be an hour long. Share backgrounds, mindsets, and thinking. Learn what is important to them and what is currently troubling them.

Often, these meetings will result in issues that will need to be triaged. That's fine. It means you are making progress.

One-on-Ones with Peers

Similarly, you want to create partnerships with your executive colleagues. Other than getting to know them, have them tell you about their teams and current challenges. First, it is part of your personal product mastery course (see Chapter 2). Second, hearing out their needs, even if there are no specific requests made, will help you shape your team and start the "coders without borders" mindset.

Shadow Meetings

To understand the processes and the operating mode of your team, you should witness them in their element. Ask to be invited to routine meetings. No background is necessary, and you shouldn't actively participate. See how the team's ceremonies take place, whether people arrive on time and prepared, and so on.

Learn the Projects

The product mastery crash course involved learning the current solution, but this step is more in depth. Become comfortable with the different projects that are currently ongoing, their milestones, and their objectives.

The Rest of Your Organization

Depending on how big your team is, you want to have face time with as many of them as possible. For smaller teams, that might mean quick 30-minute meetings with them over your first few months. With larger teams, it could make sense to meet them in small groups and extend an invitation to talk personally with whoever wants to. These group meetings tend to be a platform for them to ask you questions, but I recommend preparing some discussion points to see how the groups communicate. It's a consultant's trick to get more information quickly.

Fog of War

Along with these steps to handle the incoming issues and getting to know your team, you should also start forming a plan. To do that, you have to get the lay of the land. If you've ever played a strategy video game, you learned about the fog of war concept: some things will only be known by going toward them. This section covers areas that you will need to proactively ask about—you won't usually get a summary of all the issues with these areas. To quickly clear the fog, invest time in getting a relatively shallow understanding of these, and delve deeper where necessary. Think of this as priming your indexes: you might not have all the answers, but you'll know where to look for them.

Hiring and Firing Decisions

Especially for teams ongoing (or trying to achieve) rapid growth, personnel changes are frequently needed. You might understand that there's a vacuum that needs to be addressed in a specific technical or leadership role in your team. For example, the company's roadmap includes a mobile app out in a year, and so you should probably start hiring the relevant people. Additionally, others might not be working out well in the team currently.

I do not recommend letting go of people without due process. However, if you are handed over the baton and not starting your team from scratch, there may be people who are already on the figurative chopping block. For those situations, I often recommend making your decision within your first 100 days.

I've worked with several executives during their onboarding who, as they were taking over an existing team, were informed of low performers. Some were individual contributors, engineering managers, and even directors. During all the years I have seen this, I usually see the same pattern:

First, the new executives overwhelmingly default to giving those employees another chance. I don't know if it's some sort of hero complex, optimism, or simply all of them being good people in their core. Still, almost everyone accepts these cases as a personal challenge.

Second, the result of virtually all cases is that eventually, often within the first three months, the low performers end up leaving the company. Sometimes the new executives realize that they need to be let go, and sometimes the fresh feedback makes the employees realize it's time to move on.

The moral of this story isn't that you should fire everyone you hear bad things about on your first day nor that you should give up. I want to set your expectations to a reasonable level. Programmers often complain about whoever has written the system they inherited ("who gave that guy a keyboard?"). The same tendency applies as in your new role—you might think, "Just because Sarah couldn't make him productive doesn't mean I can't." The truth is, your predecessors are often very capable too. Keep that in mind, and don't attack every windmill.

I rarely allow my clients to draw out these situations for too long because it becomes unhealthy for everyone involved and is a substantial cognitive attention sink. Ripping off the bandage fast is often better.

If you spot a need for hiring, putting that plan into motion early is essential as most hiring efforts have a lead time of a couple of months in the least. If you are made aware of a few people who were poor performers before you took on your role, your first 100 days should be focused on providing them with candid feedback and extra attention to make a decision by that time.

Are Projects on Track?

Take a look at the projects that are already ongoing. Assessing the health of a project is tricky if you are in the thick of it and have been part of the process since its inception. It's even trickier if you are joining it somewhere along the road and trying to understand when it is going to reach the finish line (if at all).

Reassess the projects' objectives and definitions of done with those in charge of the projects and the product team. Work with them to evaluate the progress done and update the estimations. Take the pitfall discovery rate into account: a team that has gotten into 2× pitfalls in the last three months is likely to discover at least X more in the next three months. Pitfall discovery tends to go down as the project matures, but the show isn't over until it's over.

If you find that certain projects are in trouble, start revising your plans. Teams shouldn't be going on death marches, as it results in poor quality and long-term burnout. Take ownership of the projects, no matter what your predecessors have done wrong. Decide on a plan that works for the team and the business stakeholders.

Team's Health

Looking deeper than the surface, which consists of the ongoing projects and their progress, you should make use of your fresh perspective to decide what you need to focus on. The following different aspects will help you choose the most pressing issues to start with—I've used a similar checklist with my clients to hit the ground running, no matter their history or size. Depending on which of the LEAP responsibilities (see Chapter 1) you are in charge of, not all of these would be relevant, of course.

Velocity

Velocity is the speed at which the group is delivering results. It is different for every team and changes with time and from one project to the next. You can get a grasp of velocity by using objective measures, like tools that produce statistics on your past sprints and repository changes, as well as from your private meetings with the team members. Be attentive to mentions of things slowing down, especially when they come from individual contributors as they tend to be the canaries in the coal mine.

Architectural Soundness

You rarely have to (or should) do a whole rewrite to save the company, so don't expect to make the architecture look like you'd dream it. Nonetheless, it is crucial to evaluate the existing solution and its ability to keep expanding with the company's direction.

Is the team still doing "DevOps" in the form of bash scripts that are rarely maintained? Are they relying on an obsolete package that should be replaced as a preventative measure? Righting the course of the entire ship—the whole organization's architecture—takes a lot more effort than steering a specific project back on track.

Quality Issues

Often the other side of the coin of velocity is the team's quality bar. Some groups have terrific velocity on the surface, but that's because they push out low-quality solutions. Again, try and collect data that will be useful to assess the gravity of the issue (as we discussed earlier, you are gathering evidence for your decisions).

Are there teams that suffer from higher defect rates? How often does the system have an outage? Are the SLAs affected or close to that point? Has the team stopped making changes on Fridays because they know there are always repercussions? These are all indicators of important issues that might require your attention.

Maturity of Processes

Companies tend to outgrow their processes without noticing it for a while. That is how a sprint planning session slowly grows to take the better half of a day when it originally was a single hour. Another problem is lack of process where there should be one. It might not have mattered when the company was small and everyone sat in the same room, but two years later, it is likely to be a good idea to have a regular all-hands meeting.

Helping the team introduce or tweak processes so that they fit its current stage can be very helpful. Keep an eye open to understand what are the different techniques that already exist and where they are lacking. Here are some examples of processes that you should consider implementing depending on your organization's size and maturity:

- **Hiring pipeline**: Create a clearly defined pipeline that candidates go through. It should include training for those involved and guidelines for yes/no decisions. I recommend also collecting conversion metrics for the pipeline to see how the funnel works and debug any issues.

- **Onboarding**: You probably already have some process for setting up a new hire with a working computer to start developing on, but that's just the tip of the iceberg. A good onboarding process makes sure employees get exposed to the company, understand its values, and develop product mastery.
- **Offboarding**: When someone leaves the organization, certain measures should be taken, for example, an exit interview to monitor morale issues, clearing the plate to ensure there are no bugs assigned to them or tasks that will get lost, and clearing credentials.
- **Pre- and post-mortems**: Regular retrospectives are a constant wellspring of insights and improvements. Performing them regularly in a manner that involves the right people and concludes with action is vital.
- **Near-miss assessments**: Similar to post-mortems, organizations that hold a very high quality bar also assess situations where nothing actually went wrong but came very close to it. If you were within a hair's breadth this time, it might not be the case the next time.
- **Feedback**: How are one-on-ones handled across the team? Between different levels of management? How is negative feedback handled? Do you hold performance reviews?
- **Incident and on-call procedures**: How is an incident escalated? How do you make sure that there's always someone available who can tackle them? When and how do you notify the affected clients?
- **Talent growth**: Some companies create clear career ladders that are communicated openly. Others make do with simpler solutions. Every organization that has more than 20 engineers should have some process in place.
- **Infrastructure changes**: How are wide architectural decisions made and acted on? Do you have a defined architecture team? Can every team created run its own experiments?
- **Mentoring and knowledge sharing**: Another aspect of talent growth is the cultivation of expertise. Teams that improve rapidly often have processes (either explicit or implicit) for active mentoring and sharing knowledge across their different departments—holding internal tech talks, brain swaps (where people change teams for a sprint or two), code reviews across teams, and more.

Regulation and Compliance

It used to be the case that only companies in specific industries like medical or banking had to give thought to regulation and compliance. Today, regulations are expanding and are relevant to virtually all businesses. However, some companies tend to give lower priority to going over these or simply forget about it. Your team is likely to be responsible for deciding on some solutions, gathering data, and similar activities. Thus, work done by your organization will make the company liable for any gaps in compliance or regulation.

If there are no clear protocols for these issues and no one has the responsibility to review it regularly, you should take steps to decide on the most pressing issues. I recommend bringing in subject matter experts for a quick assessment instead of trying to do everything in-house.

Hiring and Turnover

Hiring is going to be a big part of your schedule. Even for teams that are not going through hyper-growth, hiring and merely maintaining your existing size requires effort. Go over the team's current hiring pipeline and strategy, open positions, and success rates. You can be a great leader, innovative, and experienced. However, if you cannot get enough people to join your team and follow your lead, you will never achieve your goals. Hiring is on your plate—you cannot get recruiters and expect them to do all the work. That simply not the way it works, at least not for teams that are aiming to be above average.

The other side of the coin is to take stock of your group's turnover or employee churn. Are people leaving faster than you're replacing them in specific teams? What is the trend of the team's churn, and are there any key employees who have already signaled that they are unhappy? This is critical information, as you might realize that some hiring efforts are a priority or that you should make changes in specific teams sooner.

Also, keep in mind that your team will probably already have an allocated budget for hiring and team growth. Get acquainted with it to learn your limitations and address if they will turn out to be showstoppers.[2]

Reputation

Every company has a reputation, and you need to be aware of yours. This is especially important for those executives who hold the *Evangelism* responsibility. When your role is to make others outside the company use its products or advocate for them inside their own companies, your reputation

[2]For a deeper look into managing your hiring strategy and headcount, see Chapter 9

is everything. It never fails to surprise me how many companies are not aware of "the word on the streets." I've caught my fair share of CEOs by surprise by merely showing them some quotes I found when searching online for the company's name on relevant sites, like Twitter, Reddit, or Hacker News.

Find out what is the sentiment that's associated with the company. Perform user (and past-user) interviews. If you are already seen as the slower company or the one that's only used by teams that can't "afford the good stuff," knowing that will shape your roadmap.

Similarly, assess the internal reputation—what are employees thinking of the company, its environment, and its values? A great source for relevant threads to pull here would be employee reviews on sites like Glassdoor.

Innovation

Measuring your team's innovation will be covered in depth in Chapter 8. As you read that part, evaluate your team's performance. Are people coming up with ideas and initiatives during your get-to-know one-on-one meetings? Has any of your colleagues mentioned something that your team did that surprised and delighted them? A team that doesn't have business-directed innovation happening regularly is a team that's slowly decaying—there's no way around it.

These areas should help you understand the team's health within a few weeks and adjust your plan accordingly. Don't try to fix everything at once, though. Pick those that are most pressing.

Forming a Plan

Armed with all this knowledge, having established the basis for your most important relationships and cleared some of the fog, now is the time to decide on your roadmap. Your first roadmap should usually be detailed only for the short term, about two to four months. Afterward, a lot of the uncertainty should be lowered, and you can form plans for 6 or 12 months. I'm not talking about the product roadmap—what features your team will be delivering—but about what you and your management team will be doing. Such a roadmap should include the following:

- **Questions to be answered**: The fog will never be lifted entirely. There will always be unanswered questions. Some questions are more important than others, and investing time into getting answers is essential, for example, why hiring is falling behind the stated goals or what should be used instead of the framework that is becoming obsolete.

- **Putting out fires**: Your triage process will uncover some fires and issues that need to be prioritized. Maybe the team is having a precipitous fall in quality that needs to be addressed or a significant regulation, compliance, or security gap that you should close.
- **Personnel and organizational shifts**: Some of these might be your own initiative (as with the firing decisions described earlier), and some might only be handling what others have put into motion. Within your first 100 days, you should have a clearer sense of where the organizational structure is lacking and needs fortification.
- **Change initiatives**: If nothing needed to be changed, you would not be needed. Clearly, you will see changes that need to be performed. Decide on the most crucial change initiatives to start with and plan the way to deploy them with your management team.

Your Personal Upgrade

Congratulations. Your first 100 days are always special. This chapter should have prepared you so that you won't become a full-time juggler and won't drop the balls. You are armed with a machine-like triage process and can execute on the steps to get to know your team and clear the fog.

Your first roadmap doesn't have to be perfect. However, what you choose as your priorities will model for the team what you value and hold dear. Keep that in mind when you are forming it. You've done your preparation. Now go forth to conquer those objectives.

In the next chapter, you will learn how to get a seat around the table where decisions are made. That is key to being able to achieve what you set out to do and to ensuring that you compose a roadmap that will benefit the business.

CHAPTER

4

Moving Upstream

Positioning yourself for optimal leverage

Possessing product mastery is a prerequisite for any executive intending to make exceptional progress. That is why you created a crash course in Chapter 2. However, knowledge is only power if you are able to put it to good use. Unlocking that power is the intent of this chapter.

Advising companies, I've found that a significant factor in whether the engineering team is an impact-driving enabling force is the organizational sway the tech executives have. Placing yourself in the right meetings and getting your voice heard is the process I call moving upstream.

If you do not move upstream, you will lack empowerment and ownership. That, in turn, leads to excuses. Eventually, something will go wrong, and if you do not feel that you have power and control, you will shrug and make an excuse. It is said that whenever Steve Jobs promoted someone to a VP position at Apple, he would explain there's a single difference between the janitor and executives: the janitor can make an excuse when they failed to

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_4

clean something. Executives, on the other hand, are no longer allowed to make excuses. Whatever happens in their organization is their fault. Such extreme ownership is the only way forward, but it requires that you wield the appropriate power.

With Great Responsibility Comes Great Power

Fostering autonomy in your teams means that they get to enjoy the rewards as well as be responsible for the problems. You cannot expect people to commit entirely to a plan where they have no input and are merely pawns following orders.

The same applies to you in your position as a tech executive. I know it might come as a surprise, as many people I talk to do not see themselves as disempowered in their jobs. However, if most of your roadmaps and agendas are handed over to you rather than involve you in their creation, then you lack empowerment.

THE ORDER-TAKING EXECUTIVES

I was advising a young startup, where every single employee was mind-bogglingly bright. You could feel it in the air when you stepped into the office.

However, two of the tech executives were not truly operating as executives. They deferred entirely to the CEO. Roadmap and strategy meetings rarely happened; most of the time, the CEO would simply come over and update them about his decisions. When these meetings eventually did take place, it didn't matter much what the tech executives said—the CEO was hearing, but he wasn't listening.

The result was that of a tech organization that was executing splendidly—but lacking any initiative. Would you spend months and hoards of money on hiring and putting in the same room a bunch of Mozarts and Beethovens only to have them take orders to create new ringtones?

Once I helped them become aware of this observation, we were able to rapidly change their leadership culture and instill an executive ***team***. Soon after, we saw results in the form of raised morales, initiatives started by the leadership team, and more time for the CEO to focus on what really required his attention.

If you are genuinely an executive, you should be able to make decisions, be treated as a peer in the executive team, and have autonomy over your team and roadmap. That autonomy should reach beyond the trivial issues like managing tech debt or choosing a cloud provider.

Your Location down the Decision Stream

Consider Figure 4-1, which pictures the decision stream. The main idea behind moving upstream is that the closer to the source that you are located, the easiest it is for you to change the way the water is flowing, and the better your outlook is over the horizon.

Figure 4-1. The Decision Stream

The diagram lists three interesting points:

> **Downstream—Order Taking:** At this point, managers do not have any input about the work they are doing, and they are not privy to the reasoning behind it. Lacking foresight about what is to come, they find it hard to plan ahead and form plans to improve their teams and be prepared for the destination they are moving toward. Often results in unsuitable investments in frameworks and tools, a mismatch between hiring done and the needs the team faces, and a lot of rework and failure work due to miscommunication and the inability to affect the tasks to tailor them to business value.

Midstream—Management: In better-run organizations, this is the kind of empowerment that I often see for directors and engineering managers. Leadership communicates the business objectives and the reasoning behind the roadmap and decisions. Some back-and-forth takes place to change the work and plans according to input from the teams.

Upstream—Leadership: This is the ultimate state where the executives maximize their ability to drive impact at all levels of their organization. Not only do they clearly understand the strategy and roadmap but they also take part in shaping it. Their location at the top of the hill means that they know where the business would like to be in the future and so they can start taking small tactical steps to align their group with the overall strategy. This freedom also enables them to bring forth innovation and novelty and help the creation of tech capital.

Manifestations of Leadership

Before we discuss the steps you need to take in order to ensure that you move yourself upstream, let us go through a few of the ways that you will exercise your leadership capabilities once you get there. This will help you realize the areas you should be focusing your efforts on and make it easier for you to make your case about your location on the decision stream.

Forming Strategy

Perhaps the most crucial aspect of real leadership. Taking part in the creation of the company's strategy as a tech executive enables you to ensure that your team's capabilities are used to their full potential. It does not matter if the strategy is updated once every five years or every other quarter. Being able to see where the company's vision is leading you allows you to put in place the right preparation to enable your group to be ready for the different challenges ahead.

Having your input incorporated into the strategy opens the doors for rapid innovation that might not be obvious to others in the executive team. This is precisely the kind of innovation that you and your team would not have been able to come up with without having a broader perspective.

Compiling Roadmaps

The next step after the creation of a strategy is to decide on the ways that it will be executed on. Coming up with objectives that drive the company forward the fastest becomes easier once you have the full context from taking part in the prior strategy discussions. Armed with this knowledge, tech executives should create balanced roadmaps with their teams. These are composed of a sensible mix of stretch goals and low-risk goals.

A litmus test for a good roadmap is that the team can clearly draw the lines between the strategy and vision and the objectives within the roadmap. Being able to make the connection makes the impact obvious, and work discussions can take shape on a different level. Rather than focusing on breaking down tasks to come up with estimates and commitments that won't get them into trouble, the team can effectively negotiate and bring their ideas to maximize the value that will be generated.

Cultivating Culture

It is hard to point your finger at what exactly you do during the day that consists of shaping the culture. Evidently, some parts of every company's culture develop without any guiding hand. The executive team should act as a culture lighthouse. Cultivating culture—as opposed to "creating" it—means that the executives routinely keep a watch about the environment, language, habits, and discussions going on. They should amplify the positive trends, guiding the team along. Aided with clear company and team values, executives must put a stop to any poisonous culture pockets that might be forming.

Virtually every single employee in the company affects its culture in some way or another. Still, managers, and even more so executives, act as multipliers. Whatever action they model will be picked up by many others.

Short-Circuiting Silo Thinking

As part of the executive team, your team members are the other seniors across the organization. They often come from entirely different backgrounds and have distinctly unique ideas and ways of operation. It is natural for organizations to cluster into silos, as per Conway's law.[1] Effective tech executives embrace their peers as partners. Enabling healthy and open communication lines between your teams starts at the top.

[1] "Any organization that designs a system (defined broadly) will produce a design whose structure is a copy of the organization's communication structure." —Melvin E. Conway

You wouldn't want to see your different teams working separately on a new product only to try and integrate it all two weeks before launch day. The same applies to bigger endeavors that involve people across multiple departments. It is your role to nip the us vs. them attitude in the bud.

Leading Risk Management

Every successful project requires active risk assessment and management to avoid getting blindsided. Executives should not be responsible for having the foresight to think and address every single thing that could go wrong—I believe that's called a scapegoat. Instead, it is up to them to guarantee that team members feel safe to speak up whenever they think that something is getting off track. Further, as those with the broadest perspective, they sometimes also have to make sure that the right hand is aware of what the left hand is doing. That's always easier if you've made it a point to break silos earlier; see the previous point.

Motivating

Lastly, executives ought to act as the team's energy reserve. This will be more thoroughly discussed when we introduce the concept of the Executive Mindset in Chapter 5. In the meantime, suffice it to say that in executive roles, one needs to put aside the cold cynicism and pessimism that comes with seniority and instead lead their teams with enthusiasm.

As an executive, you also get to set the tone about the drive and pace of the team. If you are seen relaxed and lackadaisical, so would everyone else be. If you act with a sense of positive urgency—not one that stems from the stress of hitting deadlines but from the anticipation of seeing fantastic results accomplished—then it becomes contagious and will become part of the team's culture. You are beating that drum, and it is up to you to produce the right tempo.

Claiming Your Seat Around the Table

The previous descriptions should help you realize where you might be lacking in your executive power. Notice that many of these leadership actions depend on you being upstream enough to have the clarity that is required to direct them. If, after reviewing these, you feel that you need to improve your position, I recommend that you take steps to move upstream actively.

Do not agree to be compartmentalized as the "tech person." Failing to place yourself as an equal partner in the executive team is doing a disservice to your team and your entire company. Do not make the common mistake to think that such a partnership is automatically granted based on your role in the company. I have seen hired executives who were more successful in this than technical cofounders.

Own It and Say So

The first thing to do is to take ownership of your need to move upstream. It is not likely that someone else will stop by your office and say, "You know, I think we should all be involving you more in our discussions." If you realize that you need to improve in this regard, accept the fact that the change starts with you taking the first step and candidly communicating the way you want to approach matters.

I recommend making your case in your next one-on-one with the CEO. Be clear about your current situation with examples from the manifestations of leadership earlier and the steps you intend to make. Be clear about the kind of help you need. You wouldn't listen to someone who sits begrudgingly in the corner, arms folded, saying, "I need to be involved!" Formulate a plan that is in everyone's best interest.

Get to the Meetings

For you to be able to affect the decisions, you have to be present when they are made. Ensure that you are routinely invited to the executive meetings. I've been known to opening the calendar with my clients and checking what appointments were scheduled on their peers' calendars (which were usually shared to a degree). If you spot a discussion you should be in, it's time to muster your *chutzpah* and invite yourself. I won't go into the appropriate way of going about this, as it changes from one company to the next, but I trust that you can effectively get this done.

Sometimes when doing this exercise, we might come up short. There do not seem to be any discussions going on. This is surprisingly the norm at younger companies. It doesn't mean that decisions aren't being made, of course. It usually means that the CEO is calling the shots on their own or that there are a lot of informal chats going on. In that case, you should work to establish regular leadership forums to discuss such matters. You would often be surprised to realize that your other peers would find such a meeting valuable as well. This might be easier to accomplish once you go through the advice to get allies, in the following.

Speak Up

You've made sure to be a regular party to the important meetings that are going on (or initiated them entirely). Great progress. The next step is to get involved in the discussion. I'm not referring to talking for the sake of talking—that's an ego play that I don't think benefits anyone. I do mean, though, that you should be wholly engaged in the conversation and voice your opinions, remarks, and questions.

Do not allow yourself to clam up because the discussion is not squarely in your comfort zone. The agenda doesn't list out tech issues, but your understanding and input can affect virtually every aspect of the company. That won't happen if you are content with passively listening. An advantage of moving upstream is that it's cheaper and easier to change the direction the water is going. Don't waste it.

THE CORPORAL WITH THE CHUTZPAH

I was nineteen, a corporal, and a few months into my service in the Israeli Intelligence Corps. Israel, at the time, was at war. I was summoned to a meeting to discuss a critical system that had to be altered swiftly to work reliably under the changing circumstances of battle.

In the room with me was a lieutenant colonel I had never met before. His title was, literally, "fount of knowledge." As the meeting was going on, I realized that he was describing a needlessly complex solution. No one else in the room understood the system enough to correct him. What would you have done as a fresh corporal in a room full of officers?

I spoke up and started describing the issues I was seeing. I stood my ground when objections were made. We agreed to run an experiment to validate my thinking. We were able to save a lot of precious time and put the system into working order in record time.

I wasn't special. That's due to the healthy, open, and ***chutzpah***-inducing culture that was cultivated in our unit. All it takes is speaking up and listening when someone does.

Having participated in hundreds of executive meetings, I attribute a big part of the tendency to remain silent to impostor syndrome. You might be a cofounder and still suffer from it. If that's you, you don't have to start with suggestions and calling the shots.

First, understand that most discourse among executives should not be argumentative, but based on Socratic questioning. If something does not make sense in your mind, vocalize your fears in the form of a question. You will likely help others in the room get a better understanding as well, and you just might uncover a genuine problem that went unnoticed so far.

Many coaching clients have made fast progress by focusing on two simple tactics. First, make it a habit to ask Socratic questions that allow you to understand what is going on. There are no stupid questions. Second, when appropriate, you can summarize what is being said. This has the value of ensuring that everyone has the same understanding of what is being discussed. After a while of doing so, raising some issues and your opinions becomes more straightforward.

Speak Business

Most of the professionals I work with have a tendency to speak their jargon, regardless of how likely I am to understand it. The auto mechanic always names parts that, as far as I know, could be made up. The insurance agent names policies and regulations and not why I should care. Don't get me started about lawyers. This issue is not unique to service providers, and I see it all the time in meetings.

Tech executives might be speaking in technical terms because that's what they're used to. Others do it as their way of asserting their smarts. I have also come across a couple of executives who knowingly used it because they knew they could use jargon speak to confuse others and get their way. The reason does not matter: it is wrong to do so.

Make it a personal rule to discuss your ideas, issues, and initiatives in terms that apply to the business and that your colleagues can understand. The impact of your endeavors, even if highly technical, should have an effect on the business—or they are not worth it. Sometimes, it's cutting costs or being able to deliver faster. Other times, it is merely making your people's lives easier or more interesting—which has clear value as well. Don't talk about "reducing tech debt." Turn the discussion to be about value.

We will see how this comes into play in more areas later. For now, I will say that committing to doing so yourself will also help you model the right attitude for your team to follow. This is an excellent enabler for breaking down silos—adopting a common language.

Get Allies

Don't attempt to do everything on your own. You shouldn't be the only one fighting against plans drifting away from your stated roadmap and vision. You might have a genius idea for changing some internal processes in a way that will save a lot of time or effort—but not enough sway to make the CEO want to commit to it at this point in time.

By following the previous advice, you will have fostered relationships with the executive team in ongoing meetings, learned to speak business, and displayed your commitment to propelling the company forward. That should make it easier for you to create honest partnerships and gain allies with your other peers. It is a lot easier to get buy-in for an idea if the majority of the executive team supports it from the get-go.

Don't Accept Tasks

All of these tactics eventually enable you to work at a higher altitude. Rather than receiving specific tasks, you should be forming your team's execution plan with them. You should agree to objectives that you commit to and that align with the strategy and roadmap you helped create. Deciding on the right way to make those objectives become a reality is up to you. When you agree to be handed over tasks that were already "chewed on," you lose clarity of the reasoning behind them and your autonomy to find the right way to do your job.

R&D and engineering teams that lose such autonomy are essentially functioning like low-importance outsourcing groups. They are not seen as partners, and they lose their ability to use their full potential. There's less room for insights and innovation. Failure work becomes par for the course. I call these in-sourcing departments, and I believe they're a colossal waste of talent and potential. Work with goals and objectives that you can get behind, not Jira tickets.

The tactics we just went through are both immediate and long term. You should start taking steps toward them right now, especially in those areas where you spot the most significant gaps. However, do not expect changes to happen overnight. In my experience, the transition upstream takes on average three to six months of work in most small to medium businesses.

Part of it is simply because of the cadence of executive decisions—quarterly planning happens to take place only once a quarter. The other part is because it takes time to establish the trust and relationships and adjust the already existing perception that the organization and your peers might hold of you. Imagine that one of your engineers who was focused entirely on tech for the last five years suddenly said they'd like to take on more responsibility and leadership roles. You would probably be happy about the change but will want

to see them progressing in the right way before letting them manage a team, for example. The same applies to you and your move upstream. This is the start of your own salmon run.

Get a Coach

While anyone will benefit from having a great coach, moving upstream is a particularly challenging activity for many. I've helped experienced leaders who didn't understand it was part of the (unwritten) job requirements. I also had to help some executive teams understand the vitality of having tech executives around the table.

Give the preceding tactics your best effort. In most cases, you will see a positive trend within a month of trying. In case you do not, I highly recommend inquiring about a great coach whom you and your boss trust. You'd be surprised what you can accomplish in a month with the right change catalyst!

With these tactics in hand, the move upstream is within your grasp. My experience is that about 10% of the executives already pretty much have it sorted out, 60% will see a formidable change within two months, and the remainder might need a coach to help them get over the initial hump.

Your Personal Upgrade

Moving upstream is not a single action but a journey. It requires a personal mindset shift from you. Furthermore, it takes time for the change to sink in for your colleagues and boss. You should now possess a vision of how things should look like when you establish it and grasp the almost boundless leverage it has. Thus, I trust that you will be motivated enough to pursue moving upstream as you go along. At this moment, you should start by choosing a couple of tactics to focus on in the next two weeks and gain momentum.

My coaching clients have gained great progress by revisiting this list of tactics every month and choosing two to three to use and improve at for a while.

Moving upstream means you will be able to wield power. The next chapter will provide you with tools to use that power for great effect.

CHAPTER

5

Your Executive Toolset

The secret sauce to make you the best leader they ever had

Up to this point, I took you through processes and plans to set you off in the right direction. This strategic focus, which in turn will define your vision and personal roadmap, is crucial to your success. Nonetheless, even a perfect strategy would not help if your tactics—the actions you take—are misguided.

In this chapter, you will gain capabilities to fill up the toolbox. The right tactics and approaches for issues you will have to address daily will easily make every day, meeting, and decision you make better—even if only by a fraction. Those improvements quickly add up and serve as the excellence foundation that will allow you to over-deliver compared to the average tech executive.

Think of it this way: do you remember the last time that you found out about a keyboard shortcut that you ended up repeatedly using every day? Or the first time that you used an IDE that allowed you to rename something in the code or extract a method with a single keyboard chord? From a time-consuming, error-prone, and thought-intensive process to a keystroke. When

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_5

something becomes a nonissue and requires no active thought, you gain a cognitive level of abstraction. A personal power-up. These tools are the executive's equivalent.

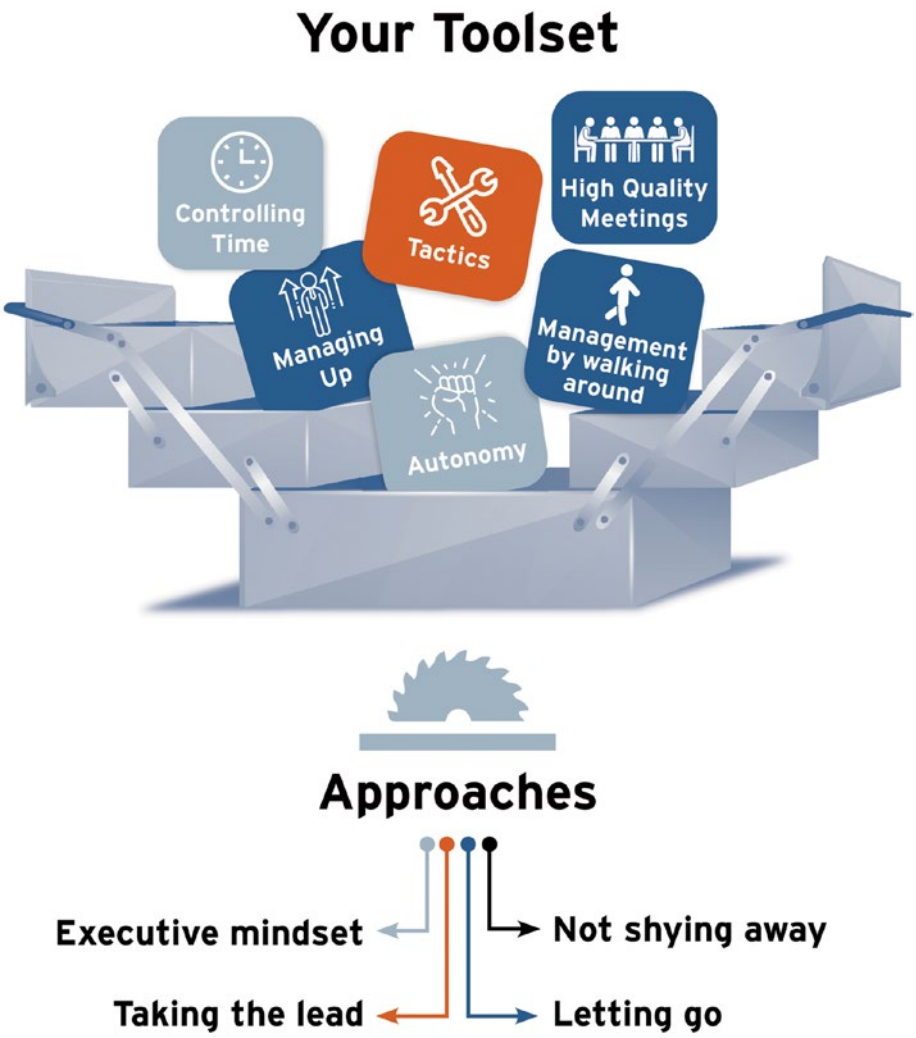

Figure 5-1. Your Executive Toolset

Figure 5-1 illustrates your toolset. The approaches—which comprise your mindsets and paradigms—are like your power tools. The different tactics we will cover are more specific and can be thought of as the different tools in your toolbox.

Tailoring Your Approach

First, we will start with an attitude adjustment. The manner in which you approach your daily challenges dramatically influences how they will come about. Here are a few mindset issues that you will frequently face.

Executive Mindset

You know the stereotypical senior tech person: cynical, mean, sarcastic, know-it-all. Often, when I first tell clients that they have to put a stop to such mannerisms in themselves, they think I'm joking. After all, being the pessimistic cynic is how they excelled in prior technical roles. And while that might be true, it simply doesn't work for leadership roles, and I advise you not to be so sanguine about it.

There's a reason that virtually every mention of Steve Jobs as a leader describes his "force field" that would make people believe whatever he was suggesting was doable: it has magical-like powers. Your role is to provide the belief that everything is possible or at least the temporary suspension of disbelief.

Executive Mindset is, essentially, radiating optimism. It makes your team act and try harder. Who would you follow to battle, the person chanting "We can do this" or the one who shrugs and quips, "Well, let's see if this works"?

The way humans are influenced is incredible. You will approach a chessboard entirely differently if you're simply shown a position and asked to make a move as opposed to being prompted with "white to move and win." All it takes is someone to suggest that there's a solution, and we focus on it—rather than on the notion that maybe there is no solution at all.

That is how, in your day-to-day meetings, you should work on increasing your awareness of the attitude you are showing. No, don't automatically say "We can do that" whenever you are presented with an impossible deadline. That's not Executive Mindset. It is about helping your team brainstorm a bit more, come up with a new idea, and find a bug.

One of the things I ask clients on their self-assessments is to rate themselves on Executive Mindset. There's a perfect correlation between high Executive Mindset and teams that are more innovative and hold a growth mindset.

Don't Shy Away

It can be hard to deliver someone harsh feedback, especially when that feedback isn't about something technical but about their personal skills or how their team performs. Nevertheless, that is your only real choice. The other option is letting things slowly become worse until the issue becomes an emergency—which is not really an option.

Ultimately this should be part of the core culture of the entire company: candor and open communication. However, even if the company is not there yet, you cannot use that as an excuse. Shying away from speaking what's on your mind is cowardly management.

The basic self-test I give executives to get a feeling for whether they are too shy is: how often do you feel, at the end of a workday, that unspoken issues are weighing down on you? It is okay if these things happen occasionally. Once this becomes a common occurrence and your default state whenever you call it a day, you know you have to take action.

When you finally do break the shyness and speak up, use these fundamental guidelines:

- *Don't deliver stale feedback*. You have to provide your feedback in a timely manner, or it doesn't matter. It's like hearing the continuous integration build broke because of something you did a week ago—you've missed the train.
- *Don't sugarcoat*. Your instinct might be to try and take the edge off of your feedback so as not to hurt the other person. The problem is that once you let yourself do that, you send the person mixed messages. Is it an issue, or isn't it? Be clear.
- *Commit to helping*. In most cases, the other person should see that you are on their side. One way to achieve that is to offer your help. You can mentor them personally, help them track their progress, or develop guidelines to help them move along. This kind of coaching builds trust and makes receiving most feedback an experience that people don't dread.

Every day ask yourself if you're leaving the office with too many things untold. If so, you're shying away from your role.

Letting Go

The last tweak to your approach should be to learn to reinvent yourself and reassess old habits and methodologies. This is especially relevant in cases where this is not your first leadership role and you have some experience. That experience can be priceless. It often provides you with better instincts and intuition for spotting issues earlier.

However, not everything you've learned in the past is still applicable. As part of the Change Culture mindset (as mentioned in Chapter 3), you should embrace your own changes regularly. The highest-performing companies in the industry are those where leaders regularly question processes, assumptions, and decisions. Your way of conducting all hands might not be working past a certain size of the team or simply doesn't fit the current company's culture. These things change from role to role and also in the same position as the company matures. If you do not let go of *some* of your old habits, **the company will outgrow you**.

Take the Lead

Every day, you have to remember that you are a leader. Even in an ideal situation, where your entire team is filled with all-stars, you still have to initiate plans and be proactive. Even more so as you are working on getting there.

These are parts of your attitude that will be relevant daily, often several times a day. By taking note of how you act and tuning your behavior according to the preceding principles, you will sow the seeds for the kind of organization you want to build. These will then make it easier to execute the right tactics day-to-day.

Tactics

Your toolset also expands to the way you manage your days and your reactions. To use the operating system analogy, you should ensure your core mechanisms operate smoothly, such as your scheduler and input-output processes. These will help you reduce the friction when you need to do the right thing. That, in turn, will save you many daily micro-decisions and help you maintain your personal velocity.

Controlling Your Time

Personal time management is perhaps the issue most executives, throughout the entire organization and in all sizes of companies, are struggling with the most. Somewhere along the line, being constantly busy and running from one meeting to the next became the norm.

Time to Lead

I'm sure you have certain feelings when it comes to your calendar. Do you dread the sound of its notifications? I once had a manager say their calendar was like Tetris: filling it out completely was the name of the game. However, we seem to forget the fact that the calendar should serve you, rather than you rushing from room to room as it instructs you.

It might come as a surprise, but the best leaders I've worked with, in all types of executive roles, are not always busy. They have time to chat in the office. Grabbing them for a few minutes isn't something that requires a week's notice. They are responsive. That is not to say that you should expect to be sitting by yourself most of the day, but if you are always trying to squeeze in another 15-minute chunk of time to talk to someone, you are doing something wrong.

Leadership Blocks

I often ask new clients when was the last time that they sat down and thought about their work in quiet for 30 minutes or more. Most don't have an answer. Some mention things like flights or vacations as those opportunities. Isn't it just a bit ridiculous that leaders need to step away from leadership to think about how they will do the leading?

Start with regularly blocking time on your calendar to think. Treat these time blocks like just any other commitment—don't allow them to be overridden because something comes up. I recommend having at least two different single-hour blocks every week.

You are likely scratching your head and wondering what one should do during such a block of time. The key here is that you get time to hear yourself think. Every session can be different.

Sometimes, during the days between these sessions, you will start mentally noting a personal agenda for the next block: thinking about the next quarter or freshening up your product mastery. That's the power of creating a dedicated space for these thoughts to take place in. Other times, you might start with nothing particular in mind—it's fine. Listen to the silence and think about whatever problems pop into your head. Go for a walk and replay the past couple of weeks in your mind. This introspection is how you induce more ideas to be created by supplying them with a platform. You know how you get the best ideas during the shower? These blocks are for the majority of leaders who cannot take an hour-long shower in their offices.

The Calendar Reset

More often than not, when I start working with new clients, I see they are afflicted with busywork: they answer emails around midnight, every text starts with "Sorry for the delay," and scheduling time with them is always an issue. These are the same people who, when I suggest they add leadership blocks to their calendar, say they can't even fit a bio-break between meetings.

That is when I ask them to open their calendar, and we go through a calendar reset. A calendar reset is a simple process to free up as much of your time as possible—without even doing any extraordinary work (later in this chapter, we'll cover more ways of taking things off your plate). Armed with a printout of your last month or so, perform these steps:

- **Review all recurring meetings**: These are usually the biggest offenders, as they take up chunks of your time regularly. Are they all still necessary? Are you required to attend them? Can you change their frequency to be more seldom?
- **Do not allow meetings that do not have a positive ROI**: Each session should have a purpose, be it making a decision, sharing information, gathering information, and so on. Do not tolerate meetings that just waste people's time without actual results. During the Covid-19 pandemic, many of the teams I work with realized a lot of meetings could be effectively replaced with a well-written email. Also, make sure people are prepared, as described later in this chapter.
- **Trim attendees**: The culture of leading by consensus is also resulting in ever-growing attendee lists. If every attendee has to be heard, that means your meetings are going to get longer. Invite only those necessary for meetings to keep them on point—meetings do not need an audience.
- **Make them shorter**: We rarely consider how much time a discussion should take. Most meetings could be done in 20 minutes, but because people use the calendar's default of one hour, that's how much they'll use. In my experience, you can shorten 80% of your meetings by at least 25% without harming their effectiveness.

- **No back-to-back meetings**: Change the company's defaults (or at least your organization's) to allow at least five minutes between meetings (e.g., the 10 AM meeting ends by 10:55 so you have some time before your 11 AM starts). Otherwise, you will have to get used to people being late to every single meeting, never even getting a second to take in what happened in the previous session before rushing to the next.
- **Defrag**: As you take the previous steps, you will start noticing "holes" in your schedule. Try and bunch meetings so that you gain bigger blocks of unscheduled time. The ideal case would be to have some days without any recurring appointments and not allowing days to be filled with meetings. If that's still hard, at least defrag your calendar, so, for example, you have your meetings taking place only after lunch.

Leaders who follow these steps often free up as much as 50% of their calendar. If you make these steps and the reasoning behind them known to your team, you will save months of work. If possible, I suggest doing the reset with another person who can challenge you on items you leave untouched. It makes a world of difference.

As with most such resets, your calendar's fragmentation will deteriorate over time. I recommend following this process quarterly—it's spring cleaning but before every season.

Getting Assistance

A substantial asset to controlling your time is having an assistant. It remains vastly unusual for tech executives to have executive assistants, but obtaining a skilled one can do wonders to your personal management. It is another instance of something that might feel awkward to consider at first—virtually every executive I suggested this to had the knee-jerk reaction of dismissing it. Nevertheless, as your organization gets bigger, it gets inevitable. Once your organization is reaching 60–80 people, you should weigh this option.

A Typical Week

Managing time well is vital for any professional, but it can make or break an executive. Because of this importance, I will finish this section by providing you a rough description of what a typical week should look like for you. You will make changes here to fit your style and situation. Nevertheless, this should serve as a yardstick to see how well you are allocating your time. I do

not want to get into how many hours you should work. Therefore, the time chunks here are in percentages out of your total working hours in a week. Figure 5-2 shows the time allocation of each activity in proportion to the others.

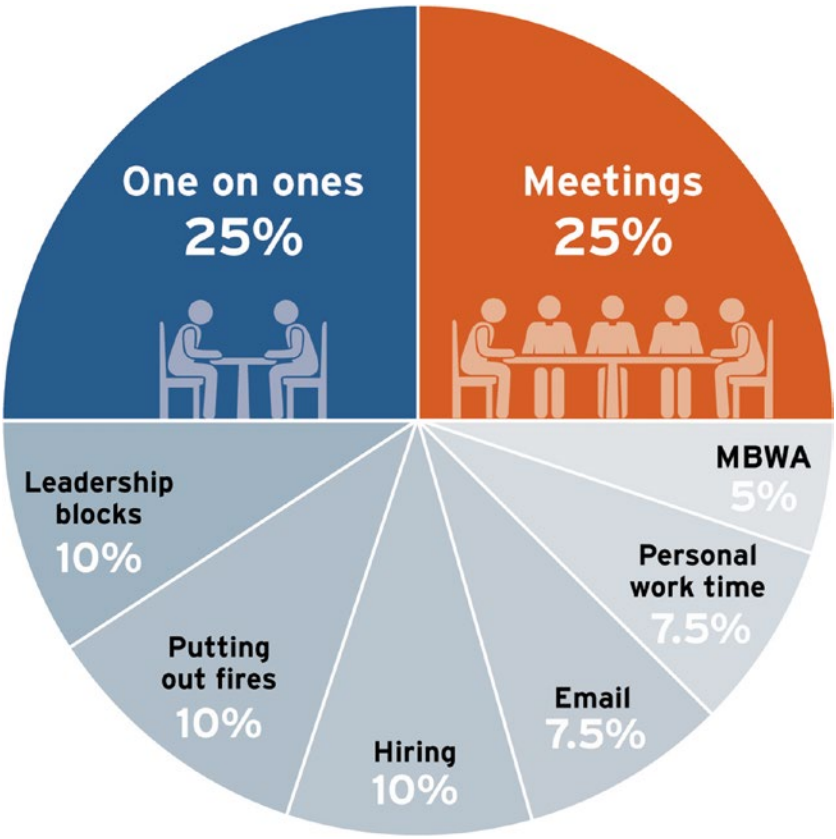

Figure 5-2. Typical Time Allocation

One-on-ones—25%: These meetings form the basis of your organization's feedback skeleton, which will be discussed in depth later. Since this platform is immensely valuable for creating a solid culture and helping the personal development of your team, it should take a big part of your time. These include meetings with your employees and peers.

Leadership blocks—10%: These are the "time to think" blocks mentioned earlier. You might spread them in small chunks every day or have one or two longer sessions a week. The key is to have them take place regularly, ideally at the same time every week.

Emails and communication—7.5%: Asynchronous communication can unlock a host of different working conditions (most notably, remote work). Furthermore, as mentioned already, a well-written email or document can often replace an hour-long meeting. Leaders should spend time articulating updates, approaches, and summaries. Every minute spent doing so is returned tenfold and likely even more as this habit spreads across your team.

Personal work time—7.5%: Naturally, not all of your work would be done by participating in meetings. For example, let us say that you are leading a change initiative to increase your team's product mastery. In that scenario, you might spend time planning a hackathon, collecting material about competitors, or preparing a talk. These efforts tend to oscillate as your different projects go on, but I find this amount of time to be a standard average.

Hiring—10%: For any R&D group that is growing, hiring is going to be a significant part of your time investment. This includes interviews, debriefs, and onboarding involvement. For companies undergoing "hyper-growth," this amount of time is often doubled or more, at the expense of other parts (e.g., if you have to triple your team, this is effectively the project you are responsible for and thus fuses with personal work time).

Management by wandering around—5%: I will explain what MBWA is later in this chapter. For now, suffice it to say that you should allocate some time every week to practice it. Think of this as turning an "open-door policy" to a "walking out the door policy."

Meetings—25%: There's no running away from meetings. The point of this section isn't to make you not have meetings at all. Rather, it is to make sure that you have *effective* meetings. Those are well worth their time investment and are your primary vehicle of influence in your organization.

> **Putting out fires—10%:** No matter what, you will have things happening that you had not intended. There might be a crisis, meetings that run over, or merely an off morning where you didn't get things done. Slack time is incredibly important to be able to handle changes as they come in—for your employees and yourself. Give extra attention to how much time you spend on these activities as they might be tricky to spot. Often, putting out fires isn't on your calendar.

As you follow the steps in this section, I recommend reviewing your calendar to see how close your time allocation is to this suggested schedule. It will help you spot the places where you can make the most significant gains. Remember that you should never be aiming for "perfection," but for effectiveness. By instilling the right habits, you will be able to weather the storms that will undoubtedly come across your path. A while after the Covid-19 pandemic started, I talked to hundreds of leaders globally. I repeatedly heard from CEOs that those tech executives I've helped with time management were the same ones who operated best as companies struggled to hit their stride again.

Managing Your Manager

Managing up is a skill that can help both in achieving better results and simply being more satisfied with your role. It is the act of knowing how to communicate effectively with your manager, reaching out to them at the right times, and forming a healthy relationship that provides you with the needed autonomy and allows you to continue moving upstream.

Results

First and foremost, start by focusing your meetings on sharing what results you've achieved and finding out which new objectives you should take on. If you want to be given autonomy, you have to take things off your boss's plate rather than being another item on it. That's only possible if you take full responsibility for your goals and then deliver.

It should come as no surprise that if you fail to do this, you will likely find yourself being micromanaged. That's why working in a results-driven manner is crucial for moving upstream. However, it is not always easy to get your manager to discuss objectives. This is where it is perfectly acceptable to push back and insist on not being handed tasks. Do not accept work without understanding the reasoning behind it and why it matters. Chapter 6 covers how this mindset is crucial for your team as a whole. Therefore, you have to walk the walk.

Business Talk

Many tech executives use too much tech jargon rather than phrasing their needs, issues, or ideas in a way that is accessible to a broader audience. It is possible to get away with this for a while as some people when bombarded with jargon just shrug and go with the other person. That is not a healthy way to hold discussions, though, and is especially important when working with your manager.

Think back to the last time you had to take your car to the mechanic or had the washing machine technician visit. Were you inundated with names of parts that you never heard of before and a bunch of things you "should do" without any explanation of why they were necessary? That's, in a way, what you are doing when you fail to "talk business."

Articulate your arguments in a way that is accessible and clear to others. It might require an extra effort, but it will pay off. Think about the bottom-line implications of different endeavors, the opportunity cost involved, and the long-term impact.

Don't Go Incommunicado

If you've been managing people in any capacity, you know that the most alarming thing is when someone seems to drop off the radar. Hearing that something is going wrong is frequently not as intimidating as *fearing* something is going wrong and you're not aware of it. This is another reason that people go into micromanagement mode: they have a need to stay apprised.

When matters are going well and when they aren't, inform your boss of progress regularly. If you committed to checking up on something, come back with an update instead of waiting to be asked again.

Speak Your Mind

When there's something that's bothering you or when you have an idea, vocalize it. Do not always defer to others in the executive team unless asked. If you fail to speak up, do not be surprised when your boss and colleagues stop involving you in discussions altogether. After all, if you do not actively participate, why waste your time?

If you repeatedly find yourself waiting for "the right time," stop it. Make the time. Bring up issues in your regular one-on-ones. (You do have these, right? If not, speak up and get them scheduled). Come prepared to leadership meetings to make your case. That's how you use your executive leverage to sway the company's direction.

CLAIMING A LEADERSHIP PLATFORM

Ron, a first-time VP I was working with, was having trouble speaking his mind. He felt that decisions were being made in the hall, between meetings—often not involving him. Ron didn't enjoy feeling left out and initially tried asking that whenever a decision is made, it will be communicated to all executives. That suggestion, while it might seem sensible, simply did not work in this young startup environment.

Together, we came up with a better approach. Rather than try and make decisions shared after the fact, the executive team needed a platform where they could act as a real ***team***. If they all met regularly and talked about current and upcoming issues, there would be less of a need to decide about things quickly as someone is walking the corridor from one meeting to the next.

He spoke up and pitched the concept of a regular leadership forum to his CEO. The CEO was skeptical, but Ron came prepared and talked to the other executives in advance. That way, he had buy-in to at least have these as an experiment (see "Changes as Tests" in Chapter 4).

A month later, the executive leadership forum was declared a success. Everyone found value in it, and Ron had become more involved in decisions that mattered to him. Moreover, the executives started working together more regularly, as this forum helped them gel as a team. All that accomplished with a weekly 30-minute-long meeting.

If you've been following along, you already took your first steps in managing up. That is what you did in "The CEO Meeting" in Chapter 2, where you had a discussion centered on objectives and goals. You already started gaining momentum and set yourself on the right path. Most of the work in managing up is similar: spotting important points where you should step up and take the lead. You have evidence that you can do this. Now all that is left is to keep at it.

Enabling Autonomy

Autonomous teams are somewhat of a management holy grail. I've yet to find managers who don't say they want such a team and that they provide autonomy gladly. However, often managers end up standing in the way of creating such a culture. Your toolkit should match this goal—to induce independence, your actions should not be taking ownership away from your team.

Default Responses

Every person tends to have a default knee-jerk reaction to specific events. Maybe you get offensive whenever someone asks a question that doubts a plan you're sharing. Or perhaps you use language that makes it so no one is interested in communicating bad news. Whatever it may be, the way you initially react will be seen by your team and peers, and they will learn to work around it.

To cultivate autonomy, you will need to ensure your default responses encourage it. Most people will not suggest ideas if they learn those suggestions result in an investigation. The best way to improve your default responses is with the help of a coach, but you can make progress on your own.

First, regularly evaluate your default responses after meetings. To do that, make a habit of taking five minutes after every meeting and call and take down notes. As part of that, assess how your default responses during those meetings affected whom you talked with. Did you rush to conclusions? Provide easy answers? Get sucked into debugging a tactical issue? I know it is easier said than done, but by becoming aware of your reflex actions, you can start short-circuiting the bad ones.

Second, invite feedback about your responses and their results. Had you had a coach, they would probably hold a few "360 interviews" to solicit this feedback from those you work with. Doing this alone is more challenging but still possible. In one-on-one meetings, make sure to routinely ask every few meetings if you did anything to prevent direct and healthy communication. The default responses concept applies to all the others in this section.

Guidelines

The other side of the coin for your default responses would be not defaulting to needlessly supplying answers. Consider the last few times when someone came up to you with an issue. Many leaders—especially those with a hands-on background—have a tendency to provide a solution when confronted with a problem. It might seem obvious. After all, if someone comes to you for help, you should help them, right? This is a classic trade-off of giving someone a fish or teaching them to fish.

In the majority of cases where you are approached with an issue, your response should not be a detailed answer. Whenever you provide answers and solutions, you teach your team to become dependent on you. They learn that you provide them with the right answer and so stop trying. You become their personal Stack Overflow replacement. This applies both for the managers reporting to you and individual contributors.

Instead, use these occurrences as learning opportunities: Can you teach your team to solve these by themselves? Can you provide a general answer rather than a specific one? The key is not to give readymade answers, but guidelines and principles that your team can follow by themselves. When they are required to draw their conclusions based on principles, it increases their autonomy and leaves space for their self-judgment. Further, it means that you will have to be involved in fewer issues as time goes on personally, and the team becomes accustomed to the guiding principles that underlie the right decisions.

THE PSEUDO-EFFECTIVE CTO

I had a call with a client of mine, a CTO of a growing startup. He described that he felt helpful and needed. Several times a day, his team would come up to him with issues, and he, in turn, fixed their problems. The crux of the matter was that he could not keep up with these issues as the team kept growing. There simply wasn't enough time in the day for him to help everyone and attend to the rest of his executive responsibilities.

I asked him how often he found himself answering the same question twice. "Not at all," he said. The team was outstanding in learning and didn't need to be told the same thing several times. This is good news, of course. I then asked him to write down all the issues he was handling for the next several days. When we reviewed them together, it was clear that there were a few underlying aspects that most issues revolved around: prioritization of company values, communication with the marketing team, and the need to change plans.

Together, we formed a few guiding principles that he would be able to refer to from that point forward. For example, instead of helping every technology decision, he explained the team's strategy for choosing tech. Rather than answering how the schedule should be changed for every delayed milestone, he made the quality bar explicit and then taught the team to make decisions based on that.

From that point on, he was able to get a lot more done, and the team got increasingly empowered by being able to call the shots on their own.

Create guidelines that will help your team to make decisions on their own without involving you. I recommend to put these in writing and in a place that is accessible to everyone regularly, especially as part of their onboarding. When these principles conform with the company's values and show how those values manifest in decision making, you inculcate a deeper understanding of the values and create an autonomous team.

Delegation

What about the cases where your default response isn't to provide an answer but to simply say "I got this"? Surely, there have to be issues where you are the right one to tackle the problem and take responsibility for it. However, if adding items on your personal to-do list is your default response, then you likely need to learn to control yourself.

Again, you might feel that taking care of matters is your role and is how you can be helpful. However, in many situations, that's merely a manifestation of micromanagement. I can almost hear your bewilderment, "What? Micromanagement? Me??" Let me assure you that even the best executives I've worked with had to struggle with micromanagement lifting its head every once in a while.

Similar to forming guiding principles earlier, consider situations that make you rush into action where you should let your team try for themselves. Don't instinctively add things to your plate, but help them take matters into their own hands. You might be required to advise them as they do it, but by delegating correctly, you help them become better and leave time for yourself to invest in more important issues.

"They Can't Do This Yet"

A common objection to delegation is that the executives do not trust their people are "ready" to take on these responsibilities. They keep waiting for some unknown event to happen, looking for a sign that indicates their team has evolved. The truth is that they will not get better without trying. I did not teach my kids to ride a bicycle by explaining it to them or have them watch me do it. Instead, my wife and I created a safe environment for them to practice independently and gain experience.

Think back to the earlier stages of your career. The chances are that you were performing tasks that you would not allow an employee with an equivalent experience to do now. That's a common bias that we hold and that the best teams actively seek to minimize.

I was 18 years old and only a few months into my first role in the IDF. Even though I was still quite "green," I was already in charge of several critical systems—as was the custom for everyone in my subunit. When the war started six months after I started my role, I had to make urgent changes. My commander, who happened to know the system I was working on, didn't barge in. He completely delegated the issue to me and made himself available. When I approached him with questions, he didn't provide readymade solutions. Instead, he shared guidelines and mentored me where I lacked in skills or knowledge. That's how you make 18-year-old juniors achieve the experience and responsibility others only get five or more years into their careers.

Coaching vs. Mentoring

Lastly, I'd like to make a distinction here between mentoring and coaching when it comes to how you increase the autonomy of your team. There are a lot of definitions and semantics when it comes to these two terms, but we will focus on one aspect of a difference that is key to our point: guidance. Mentoring, per my personal definition, involves active guidance. The mentor shows the mentee what should be done or suggests the steps that should be taken, even if not asked. It is incredibly valuable for all sorts of knowledge transfer, such as pair programming sessions, to help onboarding employees learn the ropes. Coaching, on the other hand, is about making your experience and knowledge available without taking charge.

As an executive, you should put on your coaching hat most often. It might not be an easy transition for your employees. They might be used to more guidance as well. When you have your coaching hat on and they don't seem to "get it," you should communicate about it. Let them know that you are trying to let them call the shots and will be there in case they need anything. Autonomy requires weaning both sides off of older habits.

These concepts, alongside the formation of clear goals as we will discuss in Chapter 6, will help you create an autonomous team.

Management by Wandering Around

The bigger your organization becomes, the farther removed that you get from the "actual" work. The different one-on-one meetings described earlier are one mechanism to retain your connection with all levels of your team. Another tool is management by wandering around (MBWA), which has been around since the 1970s.

In its origin, it was about people in management roles physically walking around the factory floor to witness with their own eyes how work progressed. In today's work environment, that's actually not all that different. If you recall, earlier under "A Typical Week," I recommended that you allocate 5% of your time to MBWA. It is as straightforward as taking your laptop to work from an open space, sitting in on some meetings that you see going on, or walking over to a "debugging huddle" (the grouping effect that an interesting issue has on engineers, during which they start assembling around a single display).

It might feel a bit unnatural the first couple of times, for you and the team. You have to endure the awkwardness for a few weeks, and it will wear off. After this, people will be accustomed to you popping in. As long as you don't use this to do any helicopter management, it will be fine. Let me be clear: *you are not supposed to get involved* when you are doing this—focus on observing. By seeing how things go down without intermediaries, you will get a better grasp of the engineering group's culture and situation.

Remote MBWA

Performing MBWA is not as straightforward if you are working in a remote (or semi-remote) environment. You have to perform some compensating actions to overcome remote inherent shortcomings. Start by making sure that the different remote meetings that take place are publicly available, so you can attend one without being invited. I don't mean any private one-on-ones, of course, but project syncs, retrospectives, and similar. I've seen people do this by watching recordings of meetings instead of joining them live. That's a possibility, but I find that it's not ideal: if you then decide to talk to someone after watching the recording, it elicits a "big brother" vibe. Further, it doesn't show people that you are involved and available, which is an advantage of MBWA.

Next, you will need to find a way to witness the way work is being done outside of meetings. Not being able to sit in the open space, the next best option for most companies is to lurk on Slack or your team's equivalent. There might not be a literal watercooler for you to hang around, and you will need to find its replacements.

If this sounds overwhelming, then let me clarify that I'm not suggesting that you read every single message. Join all the relevant channels your organization has, and mute the vast majority of them. When it's time for your MBWA session, go over some of those muted channels, and check the discussions. How did the team handle an outage? Was there an interesting discussion about a possible new hire?

This signal-to-noise ratio might be problematic, but if you do this systematically, you will be able to get a lot of valuable information faster than you would merely by sitting in an open space. I am routinely on half a dozen Slack organizations for clients, and doing this helps me grok their culture and deeper issues.

An Open-Door Policy That Works

Most leaders and executives have, at some point in their careers, told their teams that they have an "open-door policy." The problem is that this open door is seldom used. Holding regular meetings with all your team is one way of ensuring that your team will approach you when needed.

However, MBWA makes the door come to them. All of a sudden, you will be there present. That will make it easier for people to reach out to you, as things are happening or afterward. Even more so, by investing your time in the team so publicly, you are putting your money where your mouth is. This kind of role-modeling is vital, so you will not scupper your open-door policy.

High-Quality Meetings

The last item in this list of tactics is one that is not just relevant for you personally. In order to be an effective leader, all the meetings you attend have to be useful as well. The steps earlier under "Controlling Your Time" reduced the number of meetings you attend. That will only work if those meetings are worth their time investments. I've seen companies with hundreds of employees waste person-years on meetings that weren't well run. Here are my tips for unlocking a healthier meeting culture.

Have an Agenda

Most meetings should have a clear agenda declared in the invitation. I'm not talking about allocating time for every part of the discussion ahead of time—though that's undoubtedly alright if the meeting facilitator wishes to do so. However, meetings should state their purpose, the results you are looking to achieve, and whatever preparation is expected from the attendees. For example, you might ask all participants to review a document in advance or ask specific people to come equipped with data relevant to the question at hand.

There's no need to be dogmatic, but at least 75% of the meetings in your organization should have an agenda attached. Moreover, you should request the same for meetings with people outside of your organization. For events that are not a "happy hour" or equivalent, people should know what the purpose of the meeting is and why they were invited to it. If it's not completely obvious, a short agenda is needed.

Be Prepared

I'm willing to bet you've sat through your fair share of meetings where attendees were unprepared. Five minutes in, it is clear that Gwen hasn't had the time to collect all the data that's necessary to make the right decisions. Or half of the participants haven't done their homework and so are trying to be active in the meeting while also speed reading the documents.

What most leaders and nearly anyone running a meeting end up doing when this happens is *nothing*. They attempt to hold the meeting, even though it often results in lower quality of the decisions being made and wastes people's time, and at times eventually it means that the same meeting will have to take place again at a later time. Do not tolerate unpreparedness. Postpone the session immediately. I've seen managers ask those unprepared to use the original meeting time to do the work they failed to do before. Set an example and a standard where it is not acceptable to waste each other's time in this way.

Be On Time

I once consulted at a company where meeting start times had an alternate meaning. Naively, I thought those signified the time that everyone was supposed to be in the conference room, ready to start the discussion. There, people would wait for the notification telling them the meeting had begun. Then, they'd wait a few minutes. Only then someone, probably the person who cared most about the meeting, would start shepherding those invited. It was customary not to get up from your chair before you were individually approached. Then, those who were available would sit for a few minutes in the conference room, waiting for the managers to come from whatever meeting they were in previously, and that's running late.

This tardiness culture was ingrained in them completely. No one even batted an eye about the need to usher people personally. I later realized that this whole ordeal had originated because the managers never finished meetings on time. That made their calendars full of cascading delays and taught the team that whatever time the calendar said a session would start was only a suggestion. Just thinking about the number of person-hours lost to this is mind-boggling.

Start meetings on time. Don't tolerate pathological tardiness. Even more importantly, end meetings on time. If we're talking about meat-space meetings, on time means at least a couple of minutes before the actual end time so people will have the ability to get to the next meeting without being late.

Meetings Should Have Results

Some meetings have the purpose of making a decision. Others are about sharing or collecting information. And some are more of a basic heartbeat—they are intended to push things along. Knowing what the purpose of a meeting is also dictates the result it should achieve. If a decision-making meeting is finished without a decision, why was it held?

It is extremely frustrating for individual contributors: they go to meetings without any apparent effect. Just as agendas make clear the input needed for a meeting, you should make the *output* clear as well. At the end of the session, it should be clear what has been achieved, and the next steps should be laid out.

Sending out action items at the end of a meeting might seem tedious. However, failing to do so vastly increases the odds that things will be forgotten, and the meeting will not achieve its intended effect.

The Meeting Is the Work

To facilitate useful and effective meetings, the meetings have to be treated as part of the actual work for all participants. How many times have you seen meetings where a bunch of the engineers had their laptops in front of them, trying to finish the feature they were working on prior to the discussion? This often results in poorly written code and inattentive participants—grasp all, lose all.

It doesn't matter if people are typing away on their laptops or idly scrolling on their phones. The effort put into planning, facilitating, and timing meetings effectively means that there's no fluff. There is no need to multitask. Whenever it is allowed, it leads everyone else to think that they might as well be doing something else too.

Meeting Culture Is Your Culture

To summarize this section, meetings are a big part of how organizations operate and communicate. The meeting culture is inseparable from the rest of the culture. I've never seen a well-run organization with poor meetings culture and vice versa. Share these guidelines with all of your management team. Make it part of the management onboarding. The results will become evident faster than you'd realize.

Your Personal Upgrade

This chapter offers a panoply of tools to make you a better, more effective leader. No matter which of the tactics you start with, first, check your attitude. By embracing the importance of the Executive Mindset and taking the lead, you will significantly increase your impact on the company as an executive force multiplier.

Some of the tactics might feel awkward for the first few times. I can assure you that, like with working out, it becomes easier swiftly with some practice. Don't give up and keep at it. In the next chapter, we will walk through amplifying your personal impact by making your entire organization driven by impact and results.

CHAPTER

6

Impact

An R&D organization that doesn't move the needle is stuck in first gear

Why is your role needed? Why is the company investing in R&D? And why does the company exist in the first place? The answers to all of those should be tied to the results your products and services provide—the improvement for your customers. That improvement is what your customers hire your company to do for them. The bigger the improvement, the more *impact* that you have on them (and, as a result of that, on the business). That is our definition of success in our roles: *I have impact, therefore I am*. Placing impact as the foundation your organization is built on is essential.

What Does Impact Mean for R&D?

I was working with a VP Engineering in a promising startup who told me, "That group is always busy and delivering things on every development cycle, but nothing seems to be happening." That group comprised a cadre of competent, professional people, and the leader had prior experience in similar positions. They were doing excellent technical work. In many aspects, they were at the forefront with their tools and processes. At the end of every sprint, they'd clear most if not all of what they had committed to successfully. They had no quality issue.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_6

And yet, putting aside the pomp and circumstance, there was no output there for the clients and stakeholders. Due to various reasons, their prioritization was askew. They were working on the right objectives, but delivered what was easiest for them to define and create, as opposed to what was most valuable and impactful. Who cares that you have a beautifully designed screen for viewing and sorting alerts if they are not delivered in a reliable manner? When it came to value, a beeper was better for the users than the 21st-century high-tech solution this group created.

The team, misguided, spent enormous amounts of time and effort into creating things that weren't what the business needed. Was the group leader at fault here? No, at least not solely. That group never received the proper guidance until it got too late.

This is a common occurrence in tech: Engineers go through the motions of delivering things and might even be doing so in a fashion that seems plausible at first—all tasks they commit to are done on time in a reliable manner. However, that means nothing if those do not achieve any tangible business goals and improve upon the state of the users.

One should not default to classifying people automatically using stereotypes, but some generalizations have a grain of truth to them: Too many people in engineering leadership find it easier to work in a vacuum or do work for work's sake. When all is said and done, no one cares how pretty your team's code is, whether you're using bleeding-edge tech or ancient mainframes, and how many conference talks are based on your fantastic culture. If those don't translate into business success, you're rearranging chairs aboard the Titanic.

To provide the business with meaningful impact, your team's delivery must be easily translated into perceptible results or outputs. Those can usually be grouped into one of the following:

- **Moving the needle:** Every company has (or should have) Key Performance Indicators or metrics it is keeping track of to stay apprised of its state and health. These usually include measures such as customer lifetime value, retention, engagement, and similar metrics. If you can trace improvements in such KPIs to your team's delivery, that means you're making an impact.

- **Reducing waste:** R&D is commonly considered as a cost center, which we will tackle in another part of this book, but an easy step at demystifying that belief starts with reducing waste in other areas of the company. Can you provide tools that make the work of business analysts more timely or accurate? How about automatically segmenting existing customers, so the sales and customer happiness teams know whom to invest more time in? By improving the efficacy of other organizations, you are increasing your own team's leverage in the company.
- **Innovation:** An oft-neglected concept is that R&D should be providing the business with novelty. If your organization is focused on delivering precisely what you are being asked to do, then you will miss out on opportunities to come up with new ways to do things, using newly available technology to provide customers with more value, extending the distance between your company and its competitors.

To maximize your impact, you first have to realize who your customers are. For most engineering teams, it's not only the business's customers. You will often have a handful of internal clients, such as sales, product, customer success, and so on. These rely on your group's output to maximize their own impact and reach their goals.

By having a clear vision of who you are serving in the company, you can prioritize better and form a more coherent strategy to create an organization that best fits the company's needs. This clarity of priorities, often achieved by having clear objectives (be it in KPIs, OKRs [Objectives and Key Results], or other TLAs), will furnish you with a reliable yardstick to focus your efforts.

That's not to say that things such as quality, delivery practices, and urgency don't matter. All the focus in the world won't help if any new update you produce is full of bugs and is unusable. Similarly, focus doesn't matter if you're not delivering anything. These are tenets of execution, and they are needed to produce anything worthwhile. And yet, without the proper focus, you will likely find your team pulling and pushing in many directions, without gaining significant progress in any one truly valuable direction.

As you can see in Figure 6-1, your objectives act as a focusing lens, putting all the levers of execution to work together toward a common goal. Without it, your output would be a dimly scattered light. Apply it wisely, and you can spark a fire.

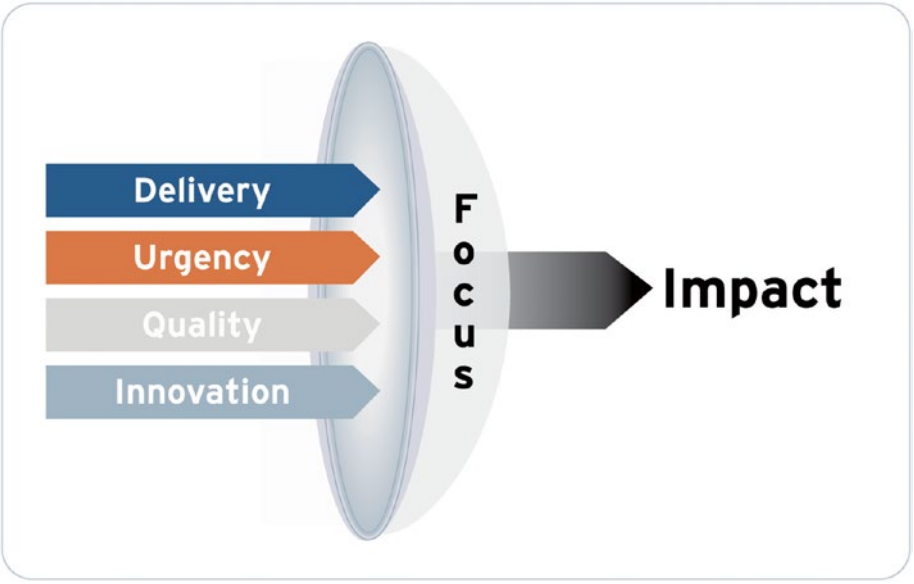

Figure 6-1. How Focus Leads to Impact

I was helping the CTO of a small startup that was a great exemplar of the impact-driven approach. Every single engineer had an exceptional understanding of the product, the users, and the company's strategy and goals. They were regularly refreshing their knowledge of those crucial things that may not be considered in most companies as their role to know. This, in turn, resulted in R&D regularly providing critical feedback to the product team about the scope of work, coming up with new ideas, and reducing engineering overhead to a minimum. Working tightly in lockstep with the product team and iterating on their ideas in small increments, they often achieved more impact with "squads" having only two engineers than most companies get out of teams of five. Such is the value of alignment and focus, and we will cover achieving this more in depth later when discussing product mastery in Chapter 10.

Another role of leadership in maximizing impact is in being involved with the creation of work upstream. It is easy to consider R&D as an outsourcing department that happens to be sitting in the same building. They get defined tasks from other parts of the business and are expected to turn out widgets according to spec. Yet by accepting this mode of operation, you are limiting your ability to deliver enormous impact to the hopes that your counterparts in the company can create the best requirements for you.

Instead, place yourself and your team higher upstream and shape the work coming down to you. One way of doing this is by being a catalyst for innovation: you deserve a seat at the table to think about the strategy of the organization, where you can shed light on what novel approaches can achieve tremendous new growth or increased speed of work. Once the strategy has been decided upon and your team is fed the chewed OKRs, your ability to think outside the box is limited to that box.

It is your job to ensure a great relationship is built between your team and the product team. Do not succumb to the standoffish default popular in our industry. Cooperation with the product team, a genuine partnership, will result in your ability to share guidance. When a new feature is being shaped to be put on the roadmap, your involvement can quickly provide more options and estimates for the effectiveness of different approaches, as well as their cost and risks.

The benefits here are twofold. First, there's the distinct advantage that your team will get work best suited to their skills and will be more likely to deliver on those goals rapidly. Second, and more important in the long term, is that your involvement means you will be apprised of the things coming your way as much in advance as possible. Having a clear vision of all the possibilities, risks, and goals that might unfold in the next year or later allows you to best prepare your team for the challenges ahead.

To drive this point through, understand that being present upstream can make all the difference in the world. You can be caught unaware, telling the CEO that you're sorry, but there's no way your team can provide what they are asking without first hiring a new team and having them come up with a solution, taking at least six months. Or you can avoid the surprises and be already there at the table saying what you believe would be the best course of action and that you already have two people with relevant backgrounds on your team who can start working on that next Monday. Being upstream can provide you with that foresight. Which leader do you want to be?

Creating Focus

Now that the crucial need for focus is clear and it is apparent that by applying focus to efforts we achieve impact, we need to address the matter of creating said focus. Referring to the preceding diagram again, what steps are needed to acquire a lens, and how do you place it correctly?

It's almost impossible to put in place the right lens without knowing where you want to be looking. Engineering organizations often accept a myopic mode of operation. They are given bite-sized tasks without understanding the overarching goals and so are not capable of making any informed decisions. The first time something goes even slightly awry, they either go on in the wrong direction or have to stop dead in their tracks waiting for someone higher up to make the decision for them.

In the Israeli Army, there's common terminology for soldiers acting in a "large head" or "little head" manner. Being a "little head" means you act like a rudimentary robot: you are given a task and perform that. You do not ask questions, and you do not take the initiative. "Large heads," on the other hand, understand the underlying reasoning behind matters and so can operate even without communicating with headquarters for a while. The same concepts, though not as life-threatening, apply to engineering. No one can be mindful of each employee 24/7 to ensure everyone is on the right track.

For people to be "large heads," they have to be given the commander's intent: values and goals. Only by making these clear and public can an organization operate in an empowered manner.

Company values are a big topic, and covering it all is outside the scope of this chapter. In relevance to the matter at hand, when it comes to teams operating autonomously, values define what the permissible ways to engage in order to achieve your stated goals are. These should span across different areas, such as ethics and finances. Each business should set its own values and ensure that they are adhered to by leadership before expecting everyone else to regard them.

IMPACT-DRIVEN INNOVATION

A client had mastered the tradeoffs between cutting-edge tech and business value. The CTO knew that his team's tech stack needed upgrades to support their expected growth, yet he feared to kick off a big rewrite when he had no idea how and when it would end. Being one of the founders, he could have used his power to make such a move anyway, but he wanted the group to learn that pushing for such changes has to provide the business with impact.

The team understood it would be a grave mistake to spend time doing "tech for tech's sake" in a business where every outage resulted in a fortune being deprived from its customers. Making a leap was necessary, but it also had to have a clear ROI to be worth it. Then they spotted the right opportunity.

Product approached them with an idea for a new offering. It was almost entirely disconnected from their existing product and so would require a lot of rapid development, but would also make the work less prone to harm the majority of the business. A few individual contributors made a case for everyone involved demonstrating that this was the perfect time to create a new microservice, the first that's not written in Ruby on Rails. They chose a tech stack that would shine in these conditions, as well as complement the issues they were having in the entire organization.

That new product offering was deployed in record speed, and the team learned a lot of their lessons in a safer environment. This change then blazed the way for them to start gradually transferring parts of their entire product line to this new stack. With the clear benefits laid out, I helped them put together a plan consisting of small and impactful stages. Even though we had gotten buy-in from stakeholders for this initiative, we did not go into a full-blown rewrite but focused on smaller iterations. These, in turn, left the company's leadership in control to stop the process whenever the need crept up, without the risk of losing all the effort as would have happened in an all-or-nothing approach.

Impact-driven goals can be your yardstick to assess the success and productivity of your organization. To compile them, you will have to work as a senior leader to collect them, verify that they are actionable, and set up your team to execute on them. The further upstream that you (and your managers) are involved in compiling these goals, the bigger the weight your input has on them and so are your chances of setting up your team to achieve remarkable work.

Remember that the intention of being involved in the goal-setting is not to "rig the system." You should not strive to set goals so that you have removed all risks and can be sure that everything you commit to will always be done on time. Such politics results in a culture of lowered expectations and a team that will constantly erode its own performance. On the contrary, some of your goals better be stretch goals: objectives that require significant effort, novelty, and sometimes luck to be achieved.

When you compile these bigger goals—we are not talking about a month's worth of tasks, but business-level objectives that will remain relevant for at least a quarter—use the following attributes to create goals you can hand off to your team and trust they can act on them:

Business focused: Impact is measured as having moved a needle on your business's dashboard. Each goal should be clearly tied to the benefits it is expected to provide. This both ensures the impact and provides your team with the *commander's intent* so they can understand the reasoning behind the tasks they are executing on.

Measurable: How will you know that the goal has been achieved and that the team can move on to the next task at hand? Your *Definition of Done* is crucial to ensure the right amount of investment will be performed. Otherwise, engineering might hand off deliverables that do not sufficiently answer the business need, or they will overengineer the solution to an unjustifiable extent.

Non-prescriptive: To create an autonomous and empowered team, they must be given the freedom to formulate their own plan of execution to achieve the goal best. If the objectives resemble a big to-do list, then you are treating your (most highly paid) staff as a bunch of high school students instead of the engineers they are. Insist that implementation details will not be part of the definition of the objectives so the team can focus on its *output*.

High level: Often going hand in hand with non-prescriptive, high-level goals are not merely glorified tasks. Allow your team to look into the horizon and plan in chunks bigger than the next week. A team reorienting every month because they were not given a bigger vision often feels confused and becomes myopic. This also means you should not have many goals at any one point. One cannot *focus* on too many things.

Achievable: No one enjoys participating in death march projects. If it is clear to everyone that the goals they are fighting for will never be achieved, the group is likely to give up right off the bat instead of pushing for it entirely. That is why the majority of the goals should be goals you are relatively sure can be achieved, at least to a certain extent. Your stretch goals should be explicitly defined, so, for example, you are aiming for a system that will support five million concurrent users by year's end, but you *have* to at least allow for a million users.

Novel: While we cannot expect all goals to have a novelty aspect to them (e.g., even Apple's industrial engineering team doesn't update their devices' exteriors every release), ensure that some goals require innovation. Organizations that never put out anything new atrophy. These are great contenders for your portion of stretch goals.

By applying the preceding descriptions to each goal or objective you commit to, you will already set yourself and your organization for success and will have placed yourself in a better position than most companies, based on my experience.

Measuring Impact

Now that you have the criteria for impact-driven goals, you need to work on another part (other than the actual implementation, obviously), and that's *closing the feedback loop*. Virtually every company creates roadmaps for the next year or quarter, yet those are neglected within a month (much like new year's resolutions), never to be considered again. Quarter after quarter, the preparation work is done, but the previous attempt and planning are not used as input to the process. Thus, no learning is done.

Adjusting your process to evaluate the impact of the different divisions, teams, and individuals has several immediate advantages:

Improvement: As will be covered later, feedback is one of the skeletons of a high-performing and fast-learning organization. That feedback has to rely, at least in part, on real-world performance. Assessing what you had thought was important enough to commit to and what has become of those commitments will help you make a better decision next time and reduce the risks of missing important deadlines or business opportunities.

Focus: Similarly, being confronted with previous optimistic plans and their results helps stakeholders focus on the most important goals when it is clear not everything can get done. Note this is not about the creation of a culture of lower expectations (the opposite of which we will cover in another chapter), but about the need to create the appropriate number of good goals as we described earlier. Aim to have around two to four such high-level goals.

Buy-in: The review of past results to estimate future performance, especially for technically oriented staff, increases the confidence and therefore the buy-in of those involved in the work.

Celebration: A recurring theme in high-impact cultures is the regular habit of learning from successes and celebrating them. Without the review, you are likely to miss such events due to the tendency of high performers to concentrate on the matters ahead and not look back.

Therefore, introducing the right processes and habits to your team to assess their impact is incredibly valuable and, also, an essential part of measuring their overall performance. After all, can a division be "world-class" if it has had no measurable business impact in the past year?

Instilling a culture of tracking impact into your organization can allow you as the leader to not just evaluate success in hindsight (a *lagging indicator*, letting you act only after the fact), but also as the work is being done. By introducing the right *leading indicators*, you will cut on failure work and be able to refocus the team faster once they have strayed off track.

Measuring Impact of Teams

First and most obvious is the need to evaluate the team's performance as defined by the goals/objectives it was given. The goal attribute *measurable*, mentioned earlier, is there to fulfill this specific need. Each objective should have an accompanying *Definition of Done* that is clear so there would be no ambiguity on whether it has been reached. Vague definitions commonly used include phrases such as "improved performance," "finish architecture change," and "higher quality." Just reading those, you can already hear in your head all the excuses and semantics that will be heard during the review meeting.

Instead, useful definitions will, whenever possible, look more like these: "improve performance, so we can support 4× customers with the same amount of cloud resources," "architecture change is deployed and used by 30% of customers and requires no more involvement of the architects team," and "reduce the number of client-escalated Priority 1 issues by 50%."

At least every time that new high-level goals are being decided and ideally more often, you and the relevant manager for each team should go over their goals and measure their progress against the yardstick that is the goal's metric. The examples provided earlier all have a number component, and so relative progress can be measured: if there's a week left and the architecture change is only used by 2% of the customer base, it is not likely to be done on time. Some goals, though, might have binary metrics: pass or fail, for example, "become HIPAA/SOC 2 compliant." Binary goals are harder to use as *leading indicators* and so should be avoided whenever possible.

Some goals are defined as stretch goals, and so it is expected that some goals would not be achieved, at least not entirely. When evaluating stretch goals, one should always be on guard not to allow them to be treated as nice-to-haves. A team that repeatedly fails to achieve its stretch goals might lack urgency and commitment to the goals.

Similarly, review the track record of the team for *too much* success. It might be a stellar team, but always be on the lookout for the possibility that the team is not taking on enough challenges so that even the stretch goals are within their comfort zones. Like the classic tale of the sales departments with a cascading system of quotas that every level is sure can be met, such "stretch" goals cause a culture of lowering standards—not even a standstill, as once you lose momentum for improvement, deterioration sets in.

FAUX SUCCESS

I was brought in to coach a leader in a startup whose team was not doing well. On paper, the team would usually achieve what it had committed to complete on every iteration. However, the rate was not fast enough, and the VP said improvement must be shown as the time estimates the team was supplying were ludicrous.

Working with the leader, we quickly identified several issues. First, that manager was not clear on her own way to impact and so did not invest enough time in developing her junior people. Further, they did not have a feeling of safety to take on even minor bets ("risks" would be an exaggeration here) and were lacking on the urgency to deliver—again partly due to their vague objectives.

Once we had these figured out, we worked together to distill the most critical goals to align the team on and on the creation of safety in their part of the organization. Aligning the goals provided the needed focus for the team to become impactful again.

Just as important as measuring the team's success in following the plan and sticking to its goals is evaluating where the team intentionally did something that wasn't planned in advance. This is your tool for assessing the team's innovation and novelty. Follow these steps:

1. Collect delivered work that was not planned and was initiated by the group (i.e., not urgent requests from the product group).
2. Review work that was scrapped. Not all novel ideas are destined to be worthwhile.
3. Together with the team, question whether any opportunities were missed.

Combining these will provide you with an estimate for the amount of innovation initiated by the team, the crucial matter being that it was *their* initiative. Teams that always stick to the plan and play it safe will miss out on chances to reap great rewards. During the 2020 coronavirus pandemic, many smart teams took the initiative of using the drop in the usage of their systems to initiate technological improvements and tests during times where failures were easier to mitigate. These sometimes translated to change initiatives being completed in weeks rather than months.

Measuring Impact of Individuals

As with teams, the main point when managing and measuring the impact of the different individuals that comprise teams is to look at their outputs. What matters are the outcomes and results they've achieved—KPIs showing better numbers, sales won, and improved quality—and not the inputs—hours in the office, butts-in-seats, and lines of code.

Other than their direct outputs, measured by the planned work they have accomplished, individuals can increase their team's impact and thus create greater leverage for moving the needle. Examples of such efforts to remember include

- **Initiatives and innovation**: As described earlier, the more they initiate rather than wait for instructions, the more autonomous the team can be.
- **Growing their team**: Even new team members can help others become better at their job by mentoring and coaching on specific skills.
- **Connecting to peers**: Individual contributors with a concrete grasp of the business and the product (*product mastery* described in Chapter 10), as well as their own specific domain in the company, can improve communication across teams and reduce useless rework and bugs.

Remember that once you discuss inputs with the team, it becomes what people optimize for. That is always a mistake unless your business model is based on billing by their own time (e.g., lawyers). For several years, I provided services as a high-end engineering consultant. The most common testimonial I got from clients went along the lines of "You do in two days what most of my engineers do in a week." That's mainly because I was laser-focused on achieving the results I was hired for, while many employees have learned that they will have to be in the office for a set amount of hours every day, so there's no rush.

Measuring Impact of Managers

Managers are first measured by a combination of the two previous sections: the completion of their team's goals and their effort in innovation and growth. A manager is responsible for their team's output as it is clear that a manager cannot be deemed successful when their team has failed miserably to achieve its goals.

Managers have more responsibilities, some of which are not always explicitly indicated by objectives from higher up. These are

- **Leverage:** Does the manager reliably act as a power multiplier for their team's efforts? Examples include removing roadblocks, improving processes, and streamlining work planning.
- **Hiring:** Every growing organization needs to invest time and effort into getting new talent to join. Managers must be able to "sell" the team's open positions and staff them in a timely manner.
- **Turnover:** Similarly, once employees are hired, they also need to be successfully retained. The majority of every team's turnover is due to the manager, and therefore outliers need to be measured and reviewed.
- **Staff growth:** For R&D organizations, talent is by far the most expensive investment. A great manager helps team members improve and grow instead of being stuck in the same place (see "Peter Pan Count" in Chapter 9).
- **Autonomy:** While harder to measure at times of peace, some managers prevent autonomy in their teams. This manifests in a lack of empowerment or delegation, micromanagement, and functioning as a hub for all communication. These teams are brittle and stop being productive (if they ever were) once the manager is not available.

Combining these, strive for your managers to create teams that deliver on their promises, continually improve professionally, and challenge the status quo (to introduce innovation). These teams then act as a flywheel where the manager's work is to provide even more momentum rather than working hard to kick-start things into motion anew every week.

Organizational Structure for Impact

Achieving your goals is, unsurprisingly, easier in some structures than in others. I always advise leaders of even the tiniest startups to give this thought early on. While structural decisions might change later, these sorts of issues do not tend to be overturned in time frames of less than a year in most companies. By considering which kind of organization you'd like to see grow, you can staff the team in a healthier way. Some applicants might be great in one kind of structure, but not in another.

There's a lot of talk lately about the ideal way to structure teams so they are productive and easier to manage. Research done recently surveying engineering managers reveals one of the most common issues they report is difficulty with handling cross-team dependencies. It is all too common for a person or an entire team to be blocking on another's work or to waste time due to miscommunications.

We have established the need to measure groups and teams based on business goals. Since a homogenous team, say comprising only iOS developers, data scientists, or DevOps engineers, cannot usually deliver on a business goal from end to end, they have to have all sorts of dependencies on other teams, making things harder to measure, communicate, and manage. Therefore, it naturally follows that the most straightforward structure for measuring business goals and impact is of cross-functional teams

What's a Cross-Functional Team?

There are many names for this structure going around, the most common I've witnessed being squads, task forces, product pods, and interdisciplinary teams. Plainly, a cross-functional team is one whose members perform different functions. Most often, these teams have a mix of backend, frontend, mobile, DevOps, and data engineers (of course, not all in every group, but those are the most common roles).

These teams can also become cross-functional in a broader sense and comprise QA and UI/UX engineers and even product managers. As a rule, the more end to end you can get your teams to be, the better. However, in most organizations, there's a limit due to politics or team sizes. To be pragmatic, I advise going with as much as you can get; this isn't usually the right hill to die on.

Having all these roles under the same entity can be a blessing for an assortment of reasons:

- **Improved communication**: Since getting the API defined between the server and the client is no longer a matter of cross-teamwork, there's no politics involved. Teammates, who are colleagues and already have an established rapport, discuss each side's needs and come up together with the ideal solution. Decisions are less likely to be made simply to conform with whoever has most political power in the organization, and that's because everyone is working in the same team, toward the same objectives, and is measured together—did we ship what we promised this sprint or not?
- **Better product mastery**: Working shoulder-to-shoulder with people performing different functions enforces each member of the team to have a broader viewpoint about the product and the various aspects of it. Breaking up the silos between different positions, even partially, can considerably help with product mastery.
- **Impact thinking**: Cross-functional teams, being measured by business goals and delivering whole working features and not just parts, intuitively make their members think in terms of impact and moving the needle. By simply making it easier to draw the line from the finished work to its real-life benefits, people make the connection more.

The main upside for sticking to homogenous teams is that everyone is quite used to it and so finds it easier to default to. There's no need to explain anything to other departments. As will be covered later, the key to effective change is to know the objections you will face upfront, so you can consider them ahead of time and push the change forward. Let's handle the ambiguity and discuss the most common objection I hear when advising my clients.

The most common issue, by far, is the need to handle a responsibility that doesn't feel "task force-ish." The usual suspects are a DevOps/SRE team or some sort of a platform team. You may be thinking, "There's no business goal in a team of architects pushing for reduced tech debt or improved CI." I'd disagree.

First, assess whether this functionality won't work dispersed among the task forces. Sometimes, having a DevOps engineer in every squad works great. However, if you have six squads and only two DevOps engineers (and don't feel like more are needed), then a different structure will have to be created.

In these cases, you have to dig a bit deeper to create a valuable and measurable goal. It might be an SRE team dedicated to reducing mean time to recovery, increasing mean time between failures, or enabling much faster deployments. All of these have a clear implication on business.

Similarly, some sort of a platform team dedicated to creating cross-organizational tools and infrastructure used by everyone might be something that you're contemplating. Again, it feels like working on frameworks "in the background" doesn't directly affect the business. However, if the platform team measures itself by developer happiness in the organization or developer empowerment, that's a whole different game. Imagine a team focused on solving the most repetitive issues so that, for example, testing a product hypothesis becomes a standard thing that doesn't require extra setup to deploy and roll back. How much faster would the product team be able to iterate?

Sometimes dispersing the responsibility across multiple teams simply doesn't make sense. For me, the point of this structure is less the cross-functioning part (meaning there doesn't have to be multiple functions), as long as it's an end-to-end team that can deliver on its projects from A to Z.

Manage your team in squads, and you will be eliminating all sorts of daily excuses and time-wasting activities. No longer will managers need to bridge between teams when two teammates sitting across the room from one another can simply talk about their task. There's no "throwing over the wall" of completed work, as now the team succeeds and fails together. And, very important in an economy where talent is extremely valuable, you will be providing your employees with plenty of opportunities to learn and grow without having to move between teams and projects.

THE PERFECTLY USELESS SOLUTION

I was consulting a startup during one of its most crucial periods, at a time the founders defined as "make or break." We were helping them migrate a text-chat app with tens of millions of users to a newer platform that would enable modern capabilities and improved quality of service. The CTO provided us with the objective of performing the switch seamlessly so the existing users would not be aware anything was changed under the hood and all their old messages would stay intact.

Accepting that as the objective, we had implemented state-of-the-art mechanisms to allow for the operation to happen live. Many tests were made until we were sure the process was iron-clad. It was performed in piecemeal over an extended period that had cost the company a lot of money and precious time to finish the move. Eventually, it was done, no major glitches were reported, and we celebrated the successful delivery.

Later, I was chatting to the founders and asked them why that chat app's history was so important to be worth months of work. "Months? That was most of the work? If we knew that, we would have scrapped the message history in a second!"

How Will You Know You Had Impact?

Earlier, we covered that it is vital to go over a team's planned objectives and the real-world result of their efforts to account for that as part of assessing the team's performance. Let us now expand on how to perform such a review to achieve overall improvement.

By making sure always to compare what has been planned and what was delivered, you will already do more than most other leaders to achieve an impact-driven culture. However, simply checking off which objectives have been accomplished is not enough, as it assumes that even in hindsight, those objectives were justified. This is rarely the case for all goals. By investing the time to evaluate the commitments made, you can save enormous amounts of failure work in the future.

For each objective delivered as planned, reach out to the relevant stakeholder to achieve closure: has this effort truly moved the needle as much as was thought? It is not always possible to tell immediately after handover, as other departments might then need to act or more time might be necessary to gather data. Still, track everything completed and return to it until you have an answer.

This knowledge is essential whether the decision turns out to have been justified or not. If it was justified, let your organization know. It's one thing to celebrate the successful delivery of a feature, but it's a whole different matter to celebrate the business success that is due to it. And for the other side of the coin, whenever you have committed to something that did not move the needle, you essentially invested time, effort, and money that could have been spent doing something else.

This is why your involvement "upstream" in the decision-making process is critical. When you learn to spot similar objectives in the future, you can help ensure the efficacy of your team's work. For example, drive for a minimal and straightforward proof of value before committing to a full-blown solution.

Remember that it is extremely rare for a company to dispose of work done even if it doesn't result in any material business impact. Your organization will be in charge of maintaining most failed experiments and features used by a handful of people for years before being allowed to sunset them. As every engineer learns quickly, the only perfect and bug-free piece of code is the code that was never written in the first place.

Your Personal Upgrade

Establishing a culture that's founded on the drive for impact is the biggest gift you can give your team and your company. It's the difference between merely having a job and feeling achievement. The result of enough impact just might be a dent in the universe.

Coupling focus with the right structure to deploy it, along with methodical measurement of the impact, will make wonders. And, with time, the best teams find their groove and maintain a certain impact velocity—which can feel magical. To get there, you might have to change some things in your organization along the way, which is exactly what the next chapter will assist you in doing.

CHAPTER

7

Driving Change

Accelerate the organization's Speed of Improvement and stop herding cats

Change is the lifeblood of organizations. Without change, teams calcify and degenerate. At the core of a healthy and sustainable team is its ability to adapt to new situations and opportunities. One of the hats you wear as an executive is that of the change catalyst: rather than asking, "Who moved my cheese?" you will be trailblazing and calling others to help you move the cheese to where it should be going.

Nonetheless, it is rare for people to embrace change happily. There are terabytes of forgotten "change initiative" documents piling up in companies' wikis. Successfully driving changes takes effort and planning. There are ways of going about a change that will increase your chances of success. That is what this chapter is all about.

Change Is a Must

For starters, let's define what a *change* is. I'm not referring solely to the obvious reorg, nor am I suggesting you have a reorg once a quarter. Companies are living organisms and so should adapt regularly. Those adaptations take many forms, including process changes, product shifts, culture adjustments, and many more. A change initiative is any intentional modification to the way things were previously.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_7

Here are some examples, big and small:

- Organizational restructuring (a "reorg")
- Changing the process of your iterations/sprints
- Introducing an on-call rotation
- Improving quality by introducing post-mortems
- Creating a feedback culture, built on regular one-on-ones and performance reviews
- Shortening cycle time
- Adding a protocol for handling user-impacting outages
- Introducing a new tool
- Moving up the value chain product-wise

As we gain more experience, we tend to become more rigid. We might think that we've learned enough so that we know how things should be done. If your team has been operating great for the past year, why should you change anything? The crux is that there are no constants in our line of work. Your team is growing, some people are leaving, and the market is evolving. Insisting on doing what "has been proved to work" will result in performance that slowly deteriorates. That is because merely maintaining your current level of operation for a sustained period requires active effort, not to mention seeing improvements.

There's no running away from change. Therefore, it is best to learn to use it to your advantage. Imagine that you were a hermit crab. Eventually, your shell would not suffice, and you'd have to gamble, leaving it aside to find a new one. Sticking to the one that has served you well will, quite literally, deter growth. Of course, just ditching the shell without any planning might result in a search for a new one that takes too much time. Hermit crabs have been watched to line up by size, and then each one takes the shell of the one next to it. If they can orchestrate such a life-threatening change, surely you can help your team improve regularly. To start, teach your team to value changes as well.

Change Culture: No Status Quo Allowed

The concept of Change Culture was introduced in Chapter 3, and it is a vital one for the longevity of any enterprise. The importance of changes for growth should be expressed in your hiring process, the company's values, and performance reviews. People at all levels of the organization should see that changes are treated as a must, not a necessary evil.

A part of creating such an attitude is making sure that the status quo is not sacrosanct. For this to work, you and the rest of your leadership team have to walk the walk. Treat new ideas and suggestions with an open mind, no matter what rank they come from. Invite your team to challenge the current way of doing things. Welcome thoughtful laid-out arguments and healthy debates (as opposed to having meetings turn into collaborative ranting).

The phrase "that's how we do things" should be verboten if used as a way to shut down ideas. If, on the other hand, it is followed with curiosity and attentiveness, new and exciting discussions can take place.

Consider this: you might already have some ceremony when one quarter ends and the next begins. As you discuss what work was delivered and what is next in store, make sure to highlight changes. What changes has your organization gone through? What have they taught you? What changes might be coming next? Treating it as part of the "real" work will help establish it as so.

Pitfall: Don't Create Change Fatigue

Armed with the apprehension of the importance of changes, one might get a bit trigger-happy. I've seen companies where change initiatives were introduced haphazardly. If the team eventually learns to treat the executive team as capricious, trying to change things on a whim, they will stop cooperating. As the confidence in your initiatives erodes, you will not be able to get people on board and establish the required buy-in to complete the changes successfully.

This is why you should only take on changes that are worthwhile enough to warrant your ongoing attention and follow-up. While we welcome changes as often as needed, you should remember that with changes, quality trumps quantity. Your attitude toward these changes will teach those around you how to treat changes as well. As with everything, you are continually modeling—even when you are not conscious of it.

Algorithm for Changes

Successful change initiatives tend to follow a very similar lifecycle, as you can see in Figure 7-1. Skip a step, and you risk failing the entire transformation, which is why I advise you to use this as a checklist until it becomes ingrained (and even then). Note that steering changes is a skill that will require deliberate practice to master. This approach and algorithm are based on hands-on experience as well as a lot of great books on the subjects of organizational change and forming habits. For further reading, I highly recommend checking out *Switch* by Dan and Chip Heath. Let us review the change algorithm.

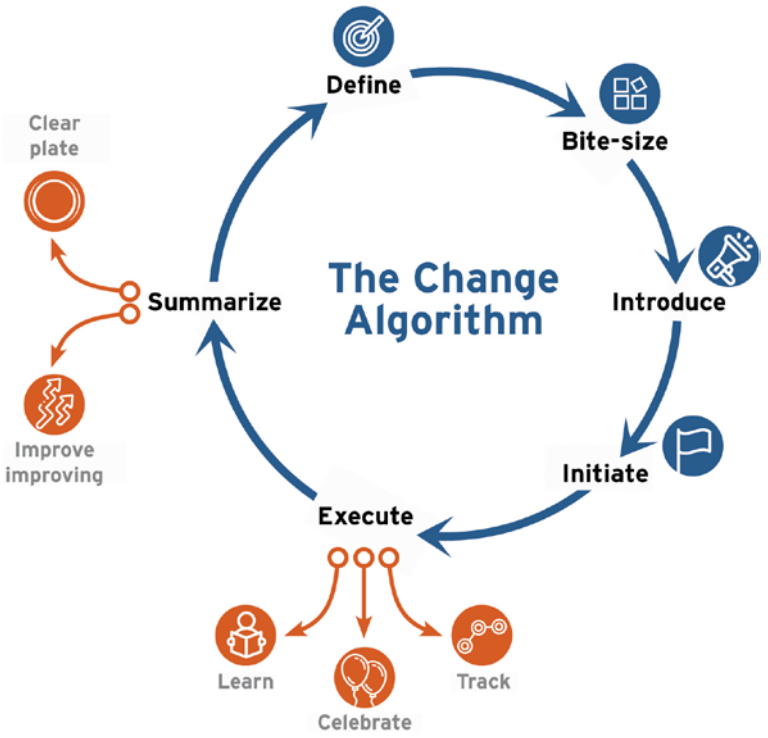

Figure 7-1. The Change Algorithm

Defining and Choosing the Initiatives

First, accept that you can only have so many things on your plate. If committing to another change would come at the expense of other ongoing projects you've already committed to or means that you will not be able to manage this new one properly, I'd reconsider waiting for a better opportunity. Otherwise, you run the risk of tripping on the pitfall mentioned earlier.

Once you know what initiative you want to commit to, it is time to define it. You know that you've defined the initiative enough when you can answer these questions:

- What is the purpose of the change? How does it improve on the current situation?
- Who needs to be involved?
- Who needs to be informed?
- What are the risks and objections that might arise?

- How would you know that the change is going well?
- How would you determine the change is a success?
- How much time are you willing to invest in this initiative?

This might seem intimidating at first, but it is a necessity. Without this prep work, you will be committing to something without due process. This is an excellent point to clarify that you are not required to do all of the steps here on your own. You can and should involve your lieutenants and others as needed. Doing so will also allow you to introduce the change in a tiered way, collect objections, and refine it by the time you reach the point of initiation.

Bite-Sizing for Momentum

The next step before you put the plan into motion is to chunk it up. A major part of the objections that come up when changes are introduced—the "who moved my cheese" resistance—is due to people's fear of big goals that feel unconquerable. To reduce that factor, the change should comprise several baby steps or milestones.

It is far easier to see the gradual process when there's a clear plan, which avoids unnecessary anxiety and stress. This chunking also has two added bonuses. The first is that by breaking the plan down, you think it through, increase the likelihood of spotting issues ahead of time, and remove uncertainty. The second advantage is that you create built-in opportunities to celebrate along the way—we will get to those again soon.

Introducing the Plan

Now comes the part where you have to get everyone else on board. It's a crucial maneuver, and this is precisely why you've spent that extra time during the previous stages. I recommend going over the change, the motivation behind it, and the information compiled in answering the questions in the first step. But before jumping to all the technical details, you should spend some effort getting your people genuinely involved and committed.

That starts by finding a WIIFM: *what's in it for me*. A successful initiative requires everyone involved to care about the outcome and to achieve the objectives. That's more likely to happen if you find the right appeal to engage not just their rational but also their emotional side. Do you have a great story behind this? Is this change connected to the core of the company's values and the kind of culture you seek to create? Will it merely allow them to sleep better at night because things will be done in an orderly fashion? Whatever it is, share it, so they care as well.

Next, treat them as partners and entrust them with all the reasoning and logic behind this effort. They need to understand the impetus, so they are able to make decisions, spot risks as they arise, and bring important tradeoffs to everyone's attention. Without this, you and your management team will have to exhort such information continuously and spend more time hand-holding and effectively micromanaging.

Then, share with the group the milestones you've created in the previous step along with a clear description of the destination—how will you all be better off after each milestone and specifically after all of them have been completed. That clarity helps disperse the "fog of war" about this change that's currently in their minds. Always remember that you might have been planning this for weeks and months, but they are only now starting to grasp it. It will take them some time to wrap their heads around it all.

Lastly, now comes the time for a trust-building session of questions and answers. Invite everyone to speak their minds or poke holes. Defending your plan in the open is a great way to build confidence, and you just might learn something new along the way.

Initiation

To get the ball rolling, you should continue with determination so as not to lose momentum. After introducing the change to everyone, it's time to start doing the actual work. Introduce in full the next milestone, assign objectives, and discuss timelines. Then, it's a great opportunity to tie a knot on some of the following steps and their intricacies.

Initiatives frequently go off track right at this stage because people kept nodding along when you discussed everything, but they have yet to actually "get" it. To make things stick, you should examine some of the critical steps ahead and how they will be addressed. For example, if you're changing your process to include regular one-on-ones and ongoing feedback, you can practice scenarios that are likely to happen: employees not saying anything or having to deliver negative feedback. Practice makes perfect, and even a tiny amount of such preparation can remarkably improve the chances that everyone involved will follow the plan correctly.

Then, as things get into motion, try to use the path of least resistance to your advantage. The way to do that is to carve the path to fit your needs. That is, rather than trying to put in place a bunch of new things the team needs to keep in mind, you can make things happen more naturally. I like moving people to sit closer to one another when they are working on something new or at least close enough that they are bound to bump into each other. You can put up a project dashboard or post-it wall located in a corner frequented by the relevant people. Have a Slack bot that gathers or asks for the needed

information automatically to remind those involved to prepare it. When possible, use existing ceremonies and tools rather than creating new ones—you can have the sync be on the agenda of the weekly management meeting, and a "dashboard" can consist of a simple spreadsheet.

These steps are intended to form habits quickly and naturally so that the meta-level work of managing the change initiative does not become too taxing. Designing these alterations in a way that's "just right" is not always trivial. I highly recommend consulting different people at different ranks of the organization as you do so.

Execute

Once that the initiative is in motion, the hard part becomes maintaining the momentum. In the preponderance of cases, the original drive is the highest it will be during the lifetime of the change initiative, after which it starts slowly declining. It is up to you and your change champions to preserve the higher motivation as much as possible.

Track

Effective execution starts with a mechanism for tracking the progress of the change throughout the organization. It should be trivial for everyone involved to be able to see what tasks are currently in progress, what the next milestone is, and the state of the different metrics you've established for measuring your success. It might be straightforward to collect this data. For example, if your change is about improving quality, you simply need to collect the average amount of production outages or bugs reported post-release. Other times it might require more legwork and compile into a unified metric, for example, when all managers ask their employees during one-on-ones to rate the new process once a month.

Nevertheless, tracking progress visibly helps those involved see that they are not in this alone and creates traction due to simple peer pressure: "If everyone else is doing their part, so should I."

Celebrate Often

An added bonus from tracking your progress is that it becomes visible when such progress is made. Make full use of these wins as opportunities to celebrate and help maintain your team's momentum. Celebrations recharge everyone and again help establish collective ownership where people become aware of the work others are putting in and so don't want to be those who stand in the way or hinder the team from making headway.

Taking note of milestones can be as simple as a "way to go" during your weekly all hands, but for bigger endeavors, I wholeheartedly urge you to go a step further. Have a toast with everyone involved, or create swag for the change initiative team that helps them feel like they're part of something together. I love visiting offices and seeing people have a bunch of trinkets from previous successes on their desks.

Learn from Setbacks

The other side of the coin is that as you track progress, you will undoubtedly come across issues from time to time. The wording is meaningful here, and that's why I'm not calling these failures or even obstacles—they are often merely setbacks. Things might not happen as quickly as you had hoped, but it doesn't mean the change you're trying to create is doomed.

When you notice these, start by addressing the root cause of the issue and redirecting efforts to overcome it. That might require changing some of the deadlines you've decided on—so be it. Having deadlines that no one believes in is toxic.

The mindset for handling these problems should be that they are learning opportunities. An unexpected issue often means that you now have information you didn't before. Make use of that information by seeing where it might apply elsewhere and improving your plan.

Furthermore, it is an opportunity to help people grow and improve. When there are no issues, we remain in our comfort zone and never stray off the "happy path." If you're not sweating from time to time, you're probably not gaining enough. Treating these with a growth mindset is the way to go. Don't let a setback knock all the wind out of your sails and instead find ways to grow your people to be capable of the task. This creates a win-win: not only do you get the improvement of the change itself but your people will also get a nice side benefit in the form of personal growth as they learn to do it.

And once your metrics indicate that you're done, it's time for the last part.

Summarize and Learn

Things need a proper end (I still want to get back all the hours I spent watching *Lost*). More often than not, change initiatives never really end. They slowly fizzle out without much attention, or they drag on ad infinitum, never seeming to reach an end. People need closure, and it is a veritable opportunity to create a lasting impression of success.

That is why we've started the whole discussion about change initiatives by defining metrics for success and thinking about a time frame. All good things should come to an end.

Celebrate

If we made sure to have a little celebration when milestones were completed, I'm sure you can imagine the kind of hoopla you should be doing when the operation has completed successfully. How you celebrate is entirely up to you, but my go-to has remained the same for over a decade—get a few nice bottles of wine and put labels on them commemorating the event. Everyone gets to share good wine, and then the empty bottles are collected with time on shelves in the offices. I have clients that have kept those I gave them for years, with newcomers asking, "How do I get one of those?" That's how you build up the momentum to have more people interested in helping out when the next change is ready to start.

It's rare for many of us to stop and smell the roses, especially while we're running toward a goal. Helping your team do so and showing you appreciate their efforts is one of the most satisfying things you can do as a leader.

Improve

There should be continuous learning going on to tweak and tune the processes as you're working. Still, it is a lot easier to gain insights when looking at the completed change in retrospect and assessing your performance as a whole. We will cover this in more detail later in this chapter, but for now, the point is that every completed change should be leveraged as a stepping stone to make the next change even more straightforward, better, or faster.

Clean Your Plate

We want to make sure that initiatives have a defined end. To achieve that, you define the next steps for your initiative. Whatever syncing meetings and open tasks should probably be canceled if no longer necessary. Otherwise, they should become habitual by this point and require no ongoing supervision from you.

Ensure that whatever processes, definitions, and watchdogs required are in place and move on. Everyone should know that along with the celebration comes a purging of the extra that's no longer needed.

Empirically, this step is skipped over in the vast majority of companies. People's calendars are brimming with zombie syncs and ghost projects. The higher-ranking people stop showing up or find a way out, but the rest have to live with this ever-growing burden.

Think about all the initiatives you've started like more and more concurrently executing processes. While they might not seem that costly when viewed separately, together, they form a substantial drain on cognitive resources. Cleaning your plate (and everyone else's) is the act of declaring things as done and moving on so that you get some of your cognitive capacity back. Never underestimate the value of less open loops running in your mind.

Following this algorithm, you have a reproducible way of making progress that you can rely on whenever the need arises. What each stage looks like can and should be adjusted for your company. Removing steps, though, should be done very carefully. Your next goal should be to create an organization that is able to adapt to changes fast enough.

OKISH OKRS

You are likely familiar with OKRs (Objectives and Key Results): a goal-setting framework that originated at Intel and brought into prominence by Google. I'm sure it won't come as a surprise to hear that I hear about companies using this framework very often, as someone who advises tech executives.

One startup I was helping had started using OKRs, and the founders spent significant time forming OKRs with their board and the entire team. The CTO shared those with me a couple of months later, and I noticed a clear (for me) dissonance.

The OKRs looked compelling, but they had almost nothing whatsoever to do with the team's work in the past month! The founders kicked off this change initiative but neglected two crucial aspects. First, they didn't invest enough effort into making the team understand the reasoning and motivation. That resulted in cargo-culting: they went through the motions without understanding what it meant for them in practice. Second, they didn't define a process for tracking the initiative's progress. It's always compelling to imagine you can "set it and forget it." Alas, that's rarely the case.

Once this dissonance was spelled out for them and the mistakes made, the founders immediately knew what they should do to fix it. Two months later, the team was chugging along with improved OKRs that were clearly connected to everything the team had in progress. It became a crucial tool for their success and planning.

Speed of Change

Note that the preceding algorithm doesn't address the timeliness of the actual change taking place. That's because every change is different. Even more so, undergoing the same change initiative will not take the same time at different companies.

The *Speed of Change* is a bit like measuring delivery velocity, but for these meta-level changes. And increasing it means that your team can adapt faster when the need arises. With a low Speed of Change, it is impossible to put in place a Change Culture—changes will be too rare by definition. A big part of your actual Speed of Change is up to you, though.

Stale Velocity

As teams are established, many go through a period of slowly finding out what their delivery velocity is. They might use something like story points for estimating work and then track that over a few sprints to establish a baseline. The issue I see is that these baselines become a self-fulfilling prophecy in some cases, where we get used to things being done at a certain pace and so stop trying to get any better.

The same applies to your change initiatives and the organization's Speed of Change. Your expectations greatly influence it. If you expect something to require four months, I can assure you it will not take any less. This goes to say that we have to be mindful of the cadence that we put in place for tracking the change initiatives, as well as the expectation setting that's done at kickoff.

These expectations should be revised on a case-by-case basis. Don't assume that everything requires at least two months just to get started. Don't allow bureaucracy to rear its head. Use the learning you've made at the conclusion of previous changes to set a new standard every time.

If something took six weeks, it doesn't mean a similar change will require the same time. Perhaps it will take less, as you can do things after, or it should even take longer to perform better. Don't automatically copy-and-paste the stale velocity you have in mind.

Arbitrary Deadlines

Some change initiatives come with an inherent deadline, for example, when you decide to change the way performance reviews are done and you need to provide the updated reviews by the end of the year. More often than not, there's no obvious time by which these changes have to get done. In those scenarios, setting deadlines might not come naturally to you, which directly affects your Speed of Change.

Many people feel that setting arbitrary deadlines is not moral. By *arbitrary deadlines* I mean deadlines where nothing bad would actually happen were they to whoosh by. This comes from negative past experience where we were subjected to managers who set arbitrary deadlines that were too taxing and caused burnout. You should not be setting aggressive deadlines that no one believes are within reason merely to drive people forward. That's an excellent way to teach everyone to disregard deadlines completely.

The problems of aggressive deadlines notwithstanding, not having deadlines at all can be just as bad. Without a set time, the work has no particular priority over the rest of the tasks that your people have to get done. What's stopping it from dropping to the bottom of the pile?

Deadlines have many positives. First, there's Parkinson's law[1] that states that "work expands so as to fill the time available for its completion." If there's no deadline, the work will just draw on and on. Second, deadlines impose a finish. That finish is crucial for clearing your and everyone else's plates and freeing their cognitive cycles to focus on the next challenges. Lastly, deadlines—any deadlines—enable leadership the ability to plan ahead. Without knowing roughly when something will be done, your options are limited when it comes to vision and creating roadmaps. So even if the date is "arbitrary," it doesn't mean that's a bad thing.

It's unlikely that you would allow teams to schedule work without a deadline or even an estimate. Likewise, you shouldn't be investing in change initiatives without making sure they will execute promptly. If it means you have to prioritize it at the expense of other work, so be it.

One last point about deadlines is that it is perfectly acceptable to communicate their nature. You shouldn't proclaim missing a deadline would be a disaster when it wouldn't. Your team should understand the importance of sticking to its word. If something ends up being postponed, chances are it will be alright. We want people to give their best efforts, nothing more.

ARRANGE TO CHANGE

I was advising an executive, and my spidey sense was tingling. Meeting after meeting, the list of changes that were supposedly inflight was growing ever longer. Literally, nothing was moved to the "done" column. We reached a point where I halted discussions of anything further to understand what was going on.

It's one thing that org-wide changes, like feedback mechanisms and growth coaching, were taking a long time. Even the simplest of changes that consisted of a champion talking to half a dozen people and making a decision were going on for months on end. It turned out that the executive and his peers were wary of using deadlines, scared that it would have adverse effects. I hate useless stress as much as the next guy, but this was something special.

We realized we had to put a stop to this immediately. A clarifying conversation with all involved about the concept of Speed of Change, as well as the importance of clearing your plate, helped them realize something had to give. Once the importance of setting deadlines was conveyed to the team, matters started changing rapidly.

[1] https://en.wikipedia.org/wiki/Parkinson's_law

Improving How You Are Improving

Accepting that changes will always be present, we want to set the team up to get the most out of them. The purpose of change initiatives is to make our condition better—they create a "new and improved" version of your organization. Your team's job is to go through the steps of these processes accurately and as fast as possible, to receive the biggest benefit out of their efforts. Your job includes another layer, and that is to ensure that the next initiatives will be even better.

By "improving improving," we are focusing on increasing your change acceleration—how fast your Speed of Change accelerates. To accomplish that, you should make time for regular introspection and reviewing efforts that are being made. You might have noticed a recurring theme: introspection, retrospection, and reviews make an appearance repeatedly. That's because you must understand where you are coming from to control where you are heading. Nevertheless, don't let these turn into navel-gazing activities. To help you avoid that, here are some prompts to instruct you and your team of champions as you are trying to make sure the next change will be even better:

- What are the common impressions and remarks team members brought up about the initiative during their one-on-ones?
- What were the unexpected issues, and how can you avoid similar ones in the future?
- Think back to the risk assessment you compiled before kickoff: how did those hold up, and were some of the severities skewed too much one way or another?
- Were there people who you should have involved earlier but failed to? Others whose involvement did not contribute enough?
- What caused things to go slower, and what helped you pick up momentum again? Different celebrations work better for different teams, and sometimes a different combination of bigger and smaller milestones might create a better rhythm.
- What case studies can you make public? Merely sharing your victories and progress is a good start, but to help bolster changes as part of your culture, it is great to have actual stories that people will remember. For example, if Gill created a Slack bot that kept everyone on track, that's worth noting and encouraging similar innovations.

- Were you successful in externalizing ownership, or did the onus on moving this thing forward stay with you and your team of champions? If initiatives never become something that the rest of the group cares about, you will not be able to get them done much faster.

Treat these as opportunities to have post-mortems without any deaths—or critical errors—occurring. Regularly debriefing and investigating your way of work improves everyone's awareness and will make it easier to spot issues the next time. Getting even a slight improvement every time adds up quickly. Improving how you're improving has the same effects as compounding interest, and it's in your favor.

Pushing for Change

Up to this point, we've focused on changes that are local to your organization and are usually wholly within your purview. The majority of change initiatives you are involved with should fall into this category. However, there will be scenarios where you will want to drive a change that has broader effects. After all, that's part of moving upstream (see Chapter 4)—not having all your thinking, innovation, and influence be local and contained.

Working on this plane, though, requires some different approaches. While I'm sure you are not getting people in your team to work on initiatives by being authoritarian, being the boss still makes things easier. Furthermore, you likely already have a lot of trust and healthy relationships that are working in your favor. That's not always the case when you want to start an initiative that requires, for example, your colleagues in sales and marketing to devote time and energy to kick off and support the change as well. Other times, you might want to pass an even more significant reform, which will have to be backed by the CEO and require consensus. It's time to put on your lobbying hat.

No One Wants to Be a Pushover

When we discuss pushing for changes, it might sound as if it is acceptable to try and force people to go along with what you want to do. Things would be a lot easier if you could just voice your great idea at the next executive meeting and have everyone happily nod in agreement, but that's bordering on fiction. In the real world, everyone has priorities and deadlines, so that even good ideas will face objections frequently.

Just as you would not like it if you felt that new processes, initiatives, and tasks were forced upon you haphazardly, so would your peers. To start pushing for changes, accept that it doesn't begin with pushing per se, but with a well-thought-out series of nudges.

Preliminary Legwork

If ramming your ideas isn't going to work, we have to come up with a different game plan. Before any change that's significant enough, you will have to do the legwork. Now is the time to approach each relevant person and confab about your plan. You should be having regular one-on-ones with most of your peers (see Chapter 2), but if not, make sure to grab them privately before bringing this up in any bigger forum. Those discussions should include these three parts in order to form a coalition.

Appealing to Their Self-Interest

Similar to the algorithm displayed earlier, you have to find a WIIFM (what's in it for me). The difference is that in this case, you should carefully articulate something for every executive who's involved. They are all as busy as you are, and they probably weren't planning on adding anything extra.

How will implementing this change improve their condition? What makes this worth their while? When I help clients prepare for this step, many feel uneasy at first or find it unnecessary. The truth is that without it, you are considerably less likely to maximize your leverage.

Furthermore, keep in mind that your case might be better off appealing to their emotions than to their brains. Some people make decisions coolly, rationally weighing every option. Others require an emotional connection. For example, sometimes making them see that peers in well-known companies are using the same approach will make your suggestion appeal to them more.

Hearing Objections Beforehand

Every bright person can quickly come up with a stream of objections for whatever idea you present to them. Just imagine how you might've dismissed a pitch for the iPhone in 2005. If you're not cautious, the same can happen to you.

Executives who come from a technical background are sometimes dumbfounded when they first pitch an idea in an executive forum. Technical discussions might be won over in a debate-like manner, where everyone considers best practices. That's often different in executive forums, where people do not appreciate being surprised and guard their and their team's time like a hawk.

You don't want to be the one catching your peers unaware. They are not likely to appreciate it, and doing so is essentially setting yourself up to fail. Instead, make sure to collect all the objections during your preliminary one-on-ones. Hear whatever might be bothering them and take it into account. Ultimately, you should be surprising virtually no one when it comes time to suggest the initiative formally, and you should not hear any objection that wasn't already voiced privately. That way, everyone will have had the opportunity to prepare and plan for this.

Creating a Shared Vision

The last part of the conversations is to attempt to get at least a couple of allies. While it might feel better to claim full ownership over a novel idea, you should be optimizing for success, not credits. That optimization requires trying to win others over to become enthusiastic about your vision. That starts by ongoing a transformation so that it is no longer solely *yours*.

As you share your vision, invite them to voice their concerns, spot risks, and improve on it. When you frame the idea as a suggestion that still requires work, they are more likely to try and cooperate with you, as opposed to when they are faced with a readymade approach that requires a yes/no decision.

Then, you can improve upon your suggestion together. This might require making some concessions, but that's the political game we all have to play. Ideally, working on shaping this idea together will result in them feeling ownership of it as well—it becomes their idea as well as yours. Once you reach a state that resembles a consensus (or at least have removed all prominent objections), you will be able to move on to actually doing the work.

Pitfall: Avoid the Consensus Trap

When you are working to win over your peers and colleagues for your initiative, working toward a consensus or something close to it makes sense. Nevertheless, you shouldn't assume that the same approach is required for change initiatives local to your organization.

We've discussed the importance of finding a WIIFM to make your team want to be part of the work. Still, do not expect or even try to get a unanimous vote of confidence every step of the way. Sometimes, leadership is about taking the lead. Some holdouts should not deter you from moving forward, as long as they disagree and commit.

Your Personal Upgrade

Kicking off changes is always accompanied by some uncertainty and sometimes even fear. However, it is an engine of growth and one of the main ways you have to help your organization become better with time. Embrace the art of change initiatives. It will take a few tries, but eventually, it will become easier—or at least less scary.

Start by picking a first change initiative and start following the algorithm to kick-start your training. There's no reason to wait for the "perfect" first initiative. Choose something that will deliver value and isn't too big and then get your hands dirty. Then, improve at it!

The next chapter introduces a paradigm shift for engineering organizations that not many have fully grasped in our industry. It might just provide you with ideas for your first project.

CHAPTER

8

R&D as the Innovation Center

An R&D team that doesn't provide business-related innovation might as well be outsourced

In plain financial terms, departments are viewed as either profit centers or cost centers. The former generate revenues, while the latter only do so indirectly. Sales and marketing are typical examples of profit centers. R&D and engineering are often counted as cost centers. My premise is that by accepting this viewpoint, you will be losing out on substantial gains. In this chapter, you will learn how to make innovation an integral part of your organization. Doing so will unlock business opportunities, market-leading edges, and, ultimately, profits.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_8

Cost Center?

Maybe you never gave this distinction any thought, or perhaps you have taken it at face value. If you're wondering whether it is that bad, let's start by understanding how this can handicap your team.

Cost centers are expected to operate at, or below, budget. The budget is often set by taking into account the work that the team is expected to perform, and that's it. Moreover, in some kinds of businesses, budgets are first allocated to the profit centers. Whatever is left then determines what will be expected of the cost centers to perform with the remaining budget.

When the R&D budget is deduced from a given feature roadmap, it is cut precisely for that. Rarely do you retain enough wiggle room to allow for surprises, experimentation, novelty, or innovation in general. This leads to a vicious cycle. The team does not have enough resources to do anything except for the tasks it was given, no extra value is generated, and it performs as expected: as a cost center. When it comes time to allocate budgets again and there is no past experience suggesting otherwise, the same process will be done.

Operating in a way that is this tightly coupled with the product roadmap doesn't leave a lot of room for anything else. Whether you like it or not, you will align your team to achieve those goals and those alone. The expectations will be of delivery, and a good team will live up to those expectations. This betrays an essential chunk of the impact that could be achieved.

Frankly, this way of thinking is one of the biggest wastes I see in our industry. I can't help but cringe when I think of all the wasted potential. We are taking a whole slew of extremely bright and talented people and pay them a fortune accordingly. All that to pigeonhole them to comply with a given roadmap and to focus on rote delivery. That's a costly mistake to make.

The Innovation Vision

There are kinds of R&D investments that you can relate to surely. Apple shows advancements in sensors, processing, materials, and algorithms every year. These don't just spontaneously happen. Behind the scenes, extensive efforts are being made. A swath of experiments take place, with many failing, before progress is achieved. That's the price of innovation. To take the lead and bring something novel to the world, you will need to put in the effort, the investment, and the time. But it is worth it.

First, you get to enjoy the fruits of your novel ideas. Apple is leading the market in many aspects, precisely due to those. When creativity complements your existing offerings and product direction, it acts as a catalyst for sales, adoption, and more.

More than providing you an edge in the market *right now*, regular innovation is also the only way to create a long-lasting business. The reason "enterprise software" is whispered with slight terror in most startups is because they fear becoming a slow-moving giant. That inherent tardiness includes a lack of innovation. When you stop reinventing, challenging the status quo, and moving forward, you will eventually atrophy. Lacking the ability to innovate regularly will result in an utter failure when a market-changing moment arrives—you cannot grow an innovative culture on demand after letting all your creative muscles shrivel away.

Successful innovation will allow your company to "skate to where the puck is going to be" rather than where it currently is. This ingenuity works wonders for everyone involved in your organization. Evidently, people enjoy being part of an innovative and creative team. That is how a culture of innovation adds many intangible benefits, such as better employee retention, improved morale, and increased motivation.

Lastly, to be able to innovate, the team has to obtain remarkable *product mastery*. In Chapter 2, we covered your executive product mastery, and Chapter 10 focuses on product mastery for your employees. Since product mastery is a prerequisite for valuable innovation, you get two for the price of one.

Types of Innovation

No one will say Apple should be stopping creative endeavors and experiments to focus solely on delivery and marketing. It is a vital part of their edge in the market. Not all companies require this kind of trailblazing to succeed. But every business can benefit from the right amount and the right *types* of innovation. Let us consider the major types. At least one will likely apply to your company.

Product Novelty

This type of innovation consists of new ways of doing things or new kinds of offerings. Often, it does not require inventing anything groundbreaking. Instead, it is about connecting existing solutions together or enabling things that were not possible before.

For example, think of something as simple as a meditation app, a piece of software that mostly consists of playing back recorded sound clips and counting time. Yet there was enough novelty in enabling this for consumers that there are several unicorns in this space.

Another well-known example would be Intuit. They provide a whole slew of products for financial needs, like payroll and taxes. Again, at the core, these products consist of straightforward operations that are codified in a way that nonprofessionals can use.

A standard class of these kinds of apps falls under the name "CRUD applications." Those are mainly database applications that create value by digitizing processes.

Technological Edge

This type consists of profound technical advantages that frequently stem from significant investments in innovation. The earlier examples of Apple's innovations in screens, materials, sensors, and so on are great examples. Nevertheless, not all technological edge is about hardware.

Think about the team that has created a world-leading machine learning model that is more accurate than anything else on the market. Another example would be a cyber startup with renowned security researchers who repeatedly find zero days and capabilities to retain their edge. One last example of this innovation type would be the life-saving startup that can analyze medical test results to find issues before human doctors can. All of these are based on innovative approaches that are not easily copied elsewhere.

Tech Capital

Tech debt is something that engineering teams tend to discuss regularly. Picture what you would do if you had a friend who repeatedly talked about debt, loans, and maxing credit cards. Most code and features end up becoming a liability—another thing to keep refreshed and maintained as time goes on.

Tech capital is about those endeavors that form assets that keep accruing in value with time. Think about the creation of tools that empower and enable new capabilities internally. This isn't any single feature, rather the ability to create a series of related features or allowing counterparts in the company to do things that were impossible prior.

The regular creation of tech capital is magical, where every new piece of it acts like another force multiplier working in your favor. The most basic example would be internal tooling that saves engineers' time and reduces mistakes in error-prone processes. Another everyday use case is the creation of tools for other departments, like a data collection tool that helps salespeople reach deeper insights and target the right prospects or an internal CMS that allows people in marketing to change emails without having to wait for engineering's time.

To use a well-known example, think of Basecamp's Ruby on Rails framework. It started off as an internal tool that allowed the company to deliver features at breakneck speed, iterate rapidly, and grow quickly. It is an asset that enables incredible, superpower-like progress to be made at times—and it is so valuable it is used by other developers worldwide.

Process Ingenuity

One last type of innovation is about finding better ways of doing things. Whenever you release a new feature, your competitors can check it out and often copy it relatively fast. If, however, you change the process by which you create value, then it will be much harder to copy.

Processes are tightly coupled to culture, which is why changing them is hard (and why discussing change management is so important; see Chapter 7). Copying them from one company to the other, especially without guidance, can sometimes be impossible.

The company that works with agile methodologies will always beat a big-design-upfront competitor. Organizations that are working in a cross-functional way and have autonomous teams are likely to drive more impact than others. Even the way that you handle production outages or the personal growth of your people can be seen as processes that provide you with an edge. Process ingenuity helps you create and maintain a better product, but it won't always affect the product itself directly.

Which Fits Your Needs?

Your company does need to be world-class at all of these types of innovation. For your organization and the market that you operate in, consider which of these is best to invest in. You should not try to spin up multiple of these unless you already have a substantial lead in one, as it will make your team feel like a glob of butter scraped over too much bread.

Assess and choose which is likely to provide you with a better ROI right off the bat:

- Do you already have an advantage in one area?
- Is there a type that none of your competitors excel at?
- In large enough R&D organizations, it can make sense to have different groups focus on different innovation types.
- Does any of these kinds of innovation compliment the grander parts of your company's vision or uncharted areas of the product roadmap?

After you decide which is the most important to you right now, start innovating.

Implementing Innovation

Unless we are talking about research teams, it is not like anyone on your team will have a Jira task that says "Innovate" with a deadline for the end of the sprint. The onus is on you and your management team to establish a culture where innovation is a regular occurrence. You need to provide the appropriate conditions for it to take place.

While the right set of steps you should take will need to be tailored for your team, there are some that frequently help. These are the ones I've found to show the best results empirically.

Get Some Wiggle Room

Thinking back to the problem of sizing your group precisely to the measures of the declared roadmap, you want to add some buffers. Many tech executives treat the budget numbers that are initially discussed as set in stone. It is up to you to speak up and make a case for hiring extra people to account for experimentation.

If your teams know they cannot tolerate any failure because you've allocated every single minute of every single sprint this year, then they won't innovate at all. Once you have in mind the types of innovation that you'd like to see more of, help your colleagues see the benefits and the potential upside from such an investment. That is the only way to secure a bigger budget that will allow it.

A concept that I find helpful is to discuss those investments that fit Nassim Taleb's barbell strategy. While about 90% of your efforts should be dependable and relatively low risk, you should invest in some efforts where the risk is capped (i.e., you waste a specified amount of time on development), but the reward is virtually unlimited. Then, staff up to allow for this to happen without risking your product roadmap.

Make It a Part of the Job

For your team to believe that innovating is valued and vital for the company, you have to put your money where your mouth is. You should make the time for them to try things out, tinker, and work on *non-feature work*.

Non-feature Work

Non-feature work comprises tasks that are not defined by the product team. It includes tech debt, bug fixes, maintenance work, and, most importantly, tech capital. Things like bug fixes should be prioritized by product, and it can be hard for them to determine the value of tech debt and tech capital.

What many organizations try to do to strike a balance is to decide on a fixed portion of their time to be allocated to tech debt. It's prevalent to have a proportion of 20–30% of an engineering organization's time to be set for internal needs. I have qualms with this pattern. First, it is often an organizational "smell"—it is created to avoid friction between engineering and product. As mentioned in previous chapters, the engineering-product partnership is the most important one for you and therefore should not be worked around. Second, this fixed allocation often results in wasted efforts. Sometimes tech debt tasks of very low impact are done to fill up the time. It also prevents any big leaps in technical investment as the team is forced to work in these 20–30% increments.

Intermissions

Innovation is not likely to prosper in such a constrained way of work, which is why I recommend a wholly different approach: **intermissions**. Intermissions are regularly scheduled blocks of time where the team has full autonomy to decide what to spend them on. Many successful companies use these or similar concepts, such as Basecamp's sabbaticals. It boils down to creating a schedule so that teams have 3–5 days of intermissions between every one or two sprints.

There are a few significant advantages to investing time in intermissions regularly. First, just the act of having some extra time between the end of one sprint and the start of the next can result in a more impactful sprint. When sprints are always back-to-back, the product team has to "feed the beast" and provide the developers with tasks without having time to see the results of the last sprint in action. A sabbatical allows for new features to be used in production and responding to what the team learns faster.

A second advantage is that it helps focus the team during sprints. When there's a sprint, everyone is involved in achieving its objective. When in an intermission, everyone is trying out things together. I prefer this to the more common pattern where a couple of engineers tend to get the majority of tech debt and platform tasks while the rest focus on feature work.

Yet another upside is that it helps with motivation and makes your people be more proactive and take initiatives. That's because they see that they have the ability to determine what they do, at least in part of their time, and the best engineers I work with are always eager to maximize these periods.

A common objection for intermissions is the fear that the entire engineering organization will be unavailable for a week, making it more likely that issues will slowly become critical before being noticed, that momentum will be lost, or that there will be nothing tangible and user-facing to show for the work, inviting criticism from others. I believe the best way to do intermissions with multiple teams is to have the intermissions on a cascading schedule so that they do not overlap across the entire department, as you can see in Figure 8-1. It avoids these issues while also allowing for a nice passing of the innovation torch between teams—there's always someone up to interesting stuff in the company!

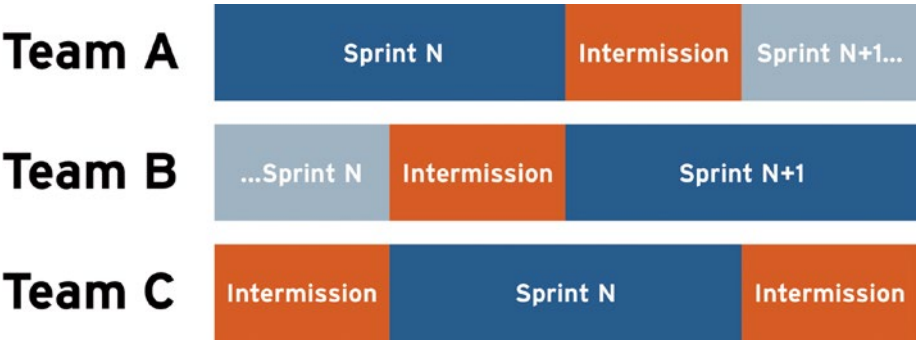

Figure 8-1. Cascading Intermissions Between Sprints

What Do You Do on Intermissions?

One thing that is essential to realize about intermissions is that they are not "free time." The concept might remind you of Google's 20% time, where people are allowed to work on whatever project they want. Ideally, your teams should have internal backlogs and decide on how to use the intermissions best. I know that these have been wholly ingrained in a company's culture where every engineer maintains a private backlog in a file and comes equipped with it to the intermission kickoff meeting. That's a great example of autonomy and initiative by developers.

Some of the work can be tackling tech debt, like upgrading an old package you depend on. Some of it can be used for learning and trying things out. For example, one team might decide to create a prototype of a new service using new technology they're not currently using to assess better whether it will be a worthwhile addition to their arsenal.

A considerable portion of intermissions, though, should be invested in innovation, trying out things that have the potential to pay off well. Think of it as managing an angel investing portfolio. Most of the things you do will not end up being tremendous game-changers. Nevertheless, having even 10% of those become wildly successful will help your team amass a vast armament of force multipliers with time.

Google's 20% time has brought the company many wildly successful achievements, like Gmail or Google News, and other companies, like Atlassian, agree. The significant difference between that approach and intermissions is that we aim to make this a team effort that gets progress in bursts. That is as opposed to 20% time that every individual allocates differently. When progress is only gained in single-day steps and people can rarely work together, you lose out on a lot of the benefits of brainstorming, working collaboratively, and fruitful debate.

THE BONANZA

One of the brightest VPs I was working with knew that his group was not spending enough time on big-bet innovations. Their heads were filled to the brim with new ideas and approaches, but they were entirely focused on routine work, never making the time to get it done. After all, no one wanted a team to come up after a sprint or two saying that they had nothing to show for it. That was a waste, as a bunch of incredibly talented engineers was not fulfilling their potential.

Together, we implemented some of the tactics described in this chapter, like getting wiggle room, setting innovation as a value, and making it part of the work. He articulated the barbell strategy that he was taking and got approval from the rest of the executive team to change the roadmap accordingly.

Two intermissions later, one of his teams unlocked what they called a "bonanza"—a capability that will allow them to create a wide moat that will differentiate them from the competition, likely for several years. The intermission showed a successful proof of concept, and the product roadmap was altered again to account for the new possibilities this introduced. To this day, they are still unchallenged in this space.

Kicking Off Intermissions

Introducing intermissions to your time is a change initiative, as covered in Chapter 7. I recommend choosing a few champions who are likely to have a personal backlog of ideas (whether written down or solely in their heads). Having a few key motivated people who are already chomping at the bit and eager to start can be contagious.

Make it clear to everyone that the time is theirs to decide how to manage it. Your champions are likely to get some people involved in their ideas to work on them together. Others can try out small changes and improvements.

I highly recommend leaving product out of these blocks of time—your people should find something worthwhile to do on their own or, if necessary, with some guidance from their managers. Only if someone has wholly exhausted other options should you let them, for example, pick up a bug from product's list. Note that bugs the team cares about are completely fine to address. A common example of bugs that product wouldn't care about for low user impact but that your group would like to see resolved are bugs that make data "dirty" or spam their logs. Those are valuable to solve in an intermission if the team thinks so.

Account for Innovation

Another aspect of making innovation part of the work is to make it everyone's job. It should be something that is brought up in one-on-ones and performance reviews. If you create a career ladder for both individual contributors and managers, make sure that it contains an assessment for innovation, creativity, novelty, and other such initiatives. Otherwise, speaking about innovation will merely be a dead letter.

Measuring innovation as part of an organization is detailed at more length a bit later. However, for individuals, it has to be done more loosely. Start by counting how many ideas they've brought up when kicking off intermissions. If you're wondering whether you should measure their successful ideas, not just all ideas, then move along to the next session.

Failing Spectacularly

You've heard a hundred stories where a field was disrupted not by a subject matter expert but by someone who was a relative newcomer. Part of it is due to the fresh perspective that newcomers have. The other part is that the more entrenched we are in a specific field, the more likely we are to grow an ego that we coddle. One becomes afraid of making mistakes and, in turn, less likely to try something new that might not work.

Add to that the apparent refrain from missing deadlines or being wrong in a work environment. It can take some effort to cure what hundreds of sprints have ingrained in people. The first step toward that is to embrace failure. You and your lieutenants should be extremely clear about the "angel investing" mindset: most of what you try will not become bonanzas. As long as enough do to get a positive ROI, that's alright.

For this to stick, there should be psychological safety when it comes to failing or trying out things that don't work. As with many other cases of cultural changes, for many people, seeing is believing. As you kick off the first few intermissions, many might still be sticking to relatively low-risk (and likely low-reward) endeavors. Your champions, though, should be starting with more challenging stuff. As we know, a lot of those will not have an amazing result. That is the point where you and your management team will be required to walk the walk.

We've already discussed the importance of celebrating wins in previous chapters, but now comes the time where you should celebrate taking the right kinds of risks. I wouldn't celebrate someone failing to use a bleeding-edge tool with 1.5 maintainers just for the coolness factor. On the other hand, trying to use a new library that could've saved costs but turned out to be still immature? Sounds well worth the shot.

Celebrating failures might seem weird, but that is how you encourage taking the right kinds of risks and moonshots. Some companies are known for handing out prizes not just to the best innovations that worked but also for the craziest idea that failed. Give you and your team a license to make a leap of faith and try something that might not work. Anything else, and you'll be capping your likely innovation.

Hackathons are a Hack

One last point about implementing innovation in your organization is about avoiding hackathons for this. Hackathons provide a welcome breath of fresh air for organizations that rarely spend time doing anything new or innovative, and therein lies the rub. Hackathons are a good idea for teams that are starved: it's better than nothing at all.

The problem with the once-every-quarter (or longer) hackathon is that it teaches people to quench all their ideas and wait for the "allowed innovation time." You end up creating creativity black holes, where people learn that they're not supposed to be letting their ideas come out till permission is granted. Putting innovation silos in place is counterproductive. Don't get me wrong; you can have regular hackathons happening to get the benefit of the entire team (or even company) working together as opposed to the smaller scale of intermissions. But these should not replace ongoing, more frequent intermissions to allow for constant tinkering and improvement.

Making Smart People *Think*

Implementing innovation is by itself a good start, but might not be enough for great results. If your team merely goes through the motions without the ability to be creative and try new things out, they are not likely to accomplish significant breakthroughs, no matter how much they try.

This section covers the different ways of encouraging a culture with enough trust to unlock bold ideas and prosper.

Invest in Learning

The first part of unlocking creativity is that the team has to possess mastery of its tools, environments, languages, and frameworks. You know the saying that to a person that only has a hammer, all problems look like nails. Engineers rarely have only a single tool. Nowadays, they have a truckload of tools at their fingertips, but they are likely to have only mastered a handful. That's why sometimes they might go off reinventing the wheel—they don't know they have the perfect one already lying at the bottom of their toolbox.

Most learning inevitably happens as part of the job, which is a good thing. You can still induce accelerated learning throughout your team with a few adjustments.

Learning Budgets

Every person's optimal learning is achieved differently. Some people get a lot from conferences and hands-on workshops. Others like online courses. I mostly prefer a great book, along with a small project. To each their own. That goes to say that rather than trying to control all the learning that takes place in the organization, you should allow for autonomy here whenever possible.

One typical example is to allow for personal learning budgets. Each person gets a yearly budget to spend how they see fit. Encourage your team to invest in their own learning to continue broadening their horizons and get better acquainted with their tools.

Be Old-School: Dig One Level Deeper

I wrote a wildly popular blog post in 2011 lamenting the fact that many engineers have stopped understanding how their tools worked. A decade later, it is more relevant than ever. I call it "being old-school," as a couple of decades ago, engineers couldn't rely on Stack Overflow and the likes to solve everything for them.

Don't read this as someone yelling "Get off my lawn!" at youngsters. This is about ensuring that with time the team as a whole understands the underlying mechanisms of its tech stack to be able to solve issues faster or avoid them altogether.

I cannot tell you how many times I've helped debug issues others took hours to solve by understanding how TCP/IP works and being able to pinpoint the cause faster. How many web developers don't know how HTTP cookies or CORS work, so once they have bugs, they lose days on end trying to find the right blog post to save them? My favorite of all is whipping out `git bisect` to find the cause of a bug in minutes—it always looks like sorcery to people who have never heard of it. We have a saying: no matter the time of day, if you've used `git bisect` to find a bug, you can call it a day.

Whenever your post-mortems spot these kinds of blind spots in your organizational knowledge, make it a point to address it. I love having someone research the topic and then share it internally in a tech talk. A team that learns together grows together.

Tinker Away

All work and no play makes your people miss out on loads of learning opportunities. It is incredible the amount of creativity and learning we can accomplish just because we want to make something seemingly stupid happen. However, tinkering and playing are how we learn a lot and practice our creativity muscles.

In *The Creativity Leap*, creativity strategist and my friend Natalie Nixon describes the importance of integrating regular play into the day-to-day work to accelerate learning and help creativity prosper. After all, a lot of innovation is connecting old ideas in a way that was not done before. That's harder to come by if you don't know a lot of ideas to start with.

I know that I dismissed hackathons a bit earlier, but they are great precisely for these kinds of purposes: trying out new things in a fun and stress-free environment. It might seem odd to let people learn how to create Slack bots so that they create one that helps arrange basketball meetups after work. Nevertheless, you'd be surprised how many of these end up evolving into practical tools when suddenly you have members who can think of such integrations as straightforward.

WORK-FRIENDLY SHENANIGANS

In my army service, we had a habit of finding geeky ways to prank one another (to my defense, we were 18–20-year-olds). In hindsight, those ended up contributing to our daily work. There was this one time where my team visited a server room in a base to do some maintenance but realized that the servers had no identification stickers on them and the paper mapping out their layout was lost. We thought we were going to spend hours trying to untangle wires to find out which is which. Then I had an idea: I connected remotely to servers and ejected the CD tray to see where it was physically. We were done in minutes. Where did I get the idea from? I learned how to mess with CD trays just so I could randomly open them on our workstations, poking my teammates' knees.

Another prank involved learning how to transplant an icon on someone's screen remotely. Keep in mind this was before screen sharing and remote control were easily doable. The knowledge of doing this was later used to support users of our systems remotely: we could draw arrows pointing at what they needed to click as we were helping them on the phone.

Tinkering helps you dig deeper and better understand our tools. That way, when the need arises, you can think of ideas much faster and are already aware of various possibilities. You are practicing to make it easier when it's game time.

Cross-Fertilization

When it comes to creativity, every single person added to the conversation brings another fresh set of eyes and a different perspective for looking at the problem. That addition might just be what was missing to make a leap forward. You can design your work processes to allow for more cross-fertilization to take place and break silos. This is especially important as teams tend to think more and more alike with time. By regularly working with people from across the organization, you prevent idea degeneration.

First, start by ensuring that technical mentoring and consulting are not compartmentalized between teams. The most obvious example would be regular code reviews and pull requests. In a cross-functional organization, there are engineers in other teams with the technical capability to assess the work—even if they are not masters of the specific system that is being changed. They can weigh in about clarity, conventions, ideas for libraries that can save work, and so forth. Having several people review work and not all of them from the same team brings in more ideas and helps with collaboration and spreading work habits and norms across the organization.

This can also be applied to other ongoing work, like design reviews, brainstorming, and more. There is no reason to be limited to a specific team or group. The biggest obstacle to this healthy collaboration is often that engineers don't want to "waste" time on tasks that belong to other teams (or their managers don't want them to). One of the guiding principles in every R&D group should be mutual assistance. If you notice that you are experiencing an issue here, bring up its importance to your managers and find a balance that works.

Another way to achieve cross-fertilization is to make sharing knowledge routine. I love seeing teams do regular tech talks internally (e.g., about what they learned in an intermission). Not only does this help spread knowledge throughout the team, but it is also conducive to the growth of your individual contributors and improves communication. Similarly, you can make a habit of putting people "on loan": every engineer can pick a sprint (say every 6 or 12 months) to join a different team. While they might not be able to be as productive or independent there, everyone gains a better understanding of what is happening around them.

Avoid Groupthink

A great way to hinder innovation is to go into a brainstorming session with many people unprepared. What ends up happening most of the time is that one or two relatively senior or valued engineers speak up and the rest tend to align with them—no one wants to risk sounding stupid or going against others. Think about the different practices of "planning poker" from agile methodologies: each person involved is asked to write an estimate, and only after doing that everyone shares. Otherwise, once Emma has said she thinks something will require two days, no one will want to voice concerns and say a week is what they had in mind.

Similarly, when you really want to collect different ideas and perspectives about an important problem, start by collecting ideas individually or in small groups. Once you have a bunch of different approaches, making them part of the agenda of a brainstorming session is a lot easier. That's a surefire way to create healthier discussions and collect more ideas.

Hire Heterogeneously

Isn't it great to hire someone whom others in the team have worked with in the past so that you have a recommendation that you can trust? It seems to reduce friction, you have better "culture fit," and things are just easier (not to mention the benefit of referrals from employees: you don't need to pay external recruiting fees).

I often come across companies that hire most of their team from a specific "pocket." Everyone is alumni of the same university, used to work together at a tech giant, and so on. This speeds up hiring and makes the team gel faster, but at the cost of losing out on a larger variety of backgrounds, perspectives, and approaches.

The most creative groups I've worked with were those that had minimal use of such pockets. Every person was a multiplier of creativity and innovation. Make sure not to get addicted to easy hiring at the cost of creating a too homogeneous team.

And as you do your best to kick off innovation and make everyone think, it is good to create a yardstick to help you tell whether or not your efforts are paying off.

Measuring Innovation

Going through these lengths to encourage and induce innovation is a significant investment. As with every such step, you want to make sure that you get a positive return. Innovation, though, is not straightforward to account for, especially when you are not managing a group that is entirely dedicated to innovation and research.

The perfect innovation metric is elusive and for all I know doesn't exist. You will have to find one or several indicators that fit your team. The following are approaches that other consultants from around the globe and I have found valuable empirically:

> **Ideas deployed:** The most straightforward indicator would be to keep count of the number of initiatives and ideas that originated from your team and ended up being deployed and positively impacting your users or the business. Did the push for a mobile app increase engagement? Had you improved the loading times of your website and saw an uptick in sales? Depending on the kind of business you are in, assessing the ROI of those improvements over 6–24 months can be a good indicator.

Internal acceleration: Innovation does not always directly affect the bottom line. For example, tech capital efforts can result in new internal tools. Similarly, process novelty might translate into faster cycle times or improved quality. These are so invaluable that I look for them when starting with a new client to get a quick grasp of their innovation. Measuring the leaps forward these internal tools achieve is another great indicator, by looking at the time, quality, and similar.

Product regeneration: Company-wide innovation can sometimes be measured by tracking the percentage of revenue that is being driven by new products or features in a 12–24-month period. This is a great way to ensure that you are not letting the product become stale and allow the competition to leapfrog your offering.

Capability: Lastly, if none of the previous work for you, you can get a feel for how good you are doing by tracking your group's ability to accommodate the company's roadmaps, goals, and new initiatives on time and budget. Without innovation, these abilities don't tend to hold up over time.

Using any one of these or even an adjusted metric that you come up with is a great way to keep an eye on your innovation efforts.

Your Personal Upgrade

Leading a team of motivated professionals is incredible. Helping such a group be a driver of growth and creativity is likely our profession's pinnacle. Shifting your organization from a cost center to an innovation center takes time, money, and effort. However, without doing it, you will be wasting potential, missing out on business opportunities, and, frankly, losing out on a whole bunch of fun.

Making the switch starts with your own perception of your team. Once you believe the leaps you all can provide, you will be able to convince everyone else. You might need to adjust your organization, such as hiring more or mixing things up. The next chapter is all about that.

CHAPTER

9

Organizing the Organization

Even though it's your hardware, it doesn't mean you can't upgrade it

R&D organizations can accumulate a lot of tech debt and thus suffer from low engineering velocity and quality issues. Software, though, is extremely malleable. It doesn't mind that you refactor it or throw it away entirely. Your employees—your *hardware*, if you will—are not as flexible. You can A/B test a feature, but it is a lot harder to A/B test a reorg or a promotion to management.

Mistakes in your organization's formation are expensive: they are not easily undone and sometimes can take months or even years to fix. In this chapter, we will list principles and guidelines for shaping your organization to help you avoid pitfalls and create a robust team.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_9

Structure Principles

As time goes on, the organizational structure will have to change. New teams will need to be created, merged, or shifted. More layers of management might be necessary. Some people might ask for specific titles or responsibilities. The principles in this section should be considered whenever you are faced with these sorts of decisions.

Teams

A team is the smallest organizational unit. As the basic building block of your org chart, it is a foundation that should be solid enough for the rest of the business to rely on. Creating the right teams will help with productivity, autonomy, communication, and ownership.

Team Role

First and foremost, each team should know what it is responsible for. In Chapter 6, we discussed organizational structures that help with achieving more impact, mainly cross-functional teams. When you just start out, you have one team, and that team is responsible for everything. As you grow, you default to splitting into groups by profession (e.g., backend and frontend).

That's not ideal because those teams are incapable of delivering tasks end to end, almost by definition. It also encourages a mentality of "doing my part" as opposed to complete ownership over a feature or product. How many times have you heard someone say "The backend is done" when you inquired about a feature not being done on time? If the backend is done, but there's no UI to make the feature available for users, does it really matter that it was done? Cross-functional teams have less synchronization overhead and so can finish things faster and own them wholly.

Team Size

People with an engineering background sometimes treat organizational structure like they would treat software: they look for concise definitions and borders. The problem is that you cannot easily undo a "refactoring" when it comes to teams, and so the *Extract Team* refactoring should not be executed carelessly.

Ideally, teams should have about six members. It can be a bit smaller when the team is being formed (more on that in a bit) or a bit bigger if needed. An experienced manager with a relatively independent team can manage up to ten or so people, in my experience. More than that, even if it seems fine, often results in lower performance over the long term. I am not a proponent

of "flat teams" for anything but relatively small organizations comprised of senior people.

The same number of direct reports is also a good rule of thumb on all levels of the organization. One level up, when it comes to how many teams leaders (or engineering managers) should report to the same directors, how many directors report to the VP (or to a senior director), and so forth. This rule has an added bonus in that it saves on the overall number of managers you need to recruit. Thirty engineers, in six-person teams, require five managers. The same in four-person teams means eight managers: almost double.

Figure 9-1 depicts a few examples of organizational trees. Relying on concepts you might be familiar with from computer science courses, you should aim for balanced trees without each person managing too many or too few. Tree *A* shows a healthy organization. Tree *B* shows an organization that has remained flat for too long. Tree *C* shows an imbalance. Tree *D* is perfectly balanced but indicates a premature organization with many micro-teams.

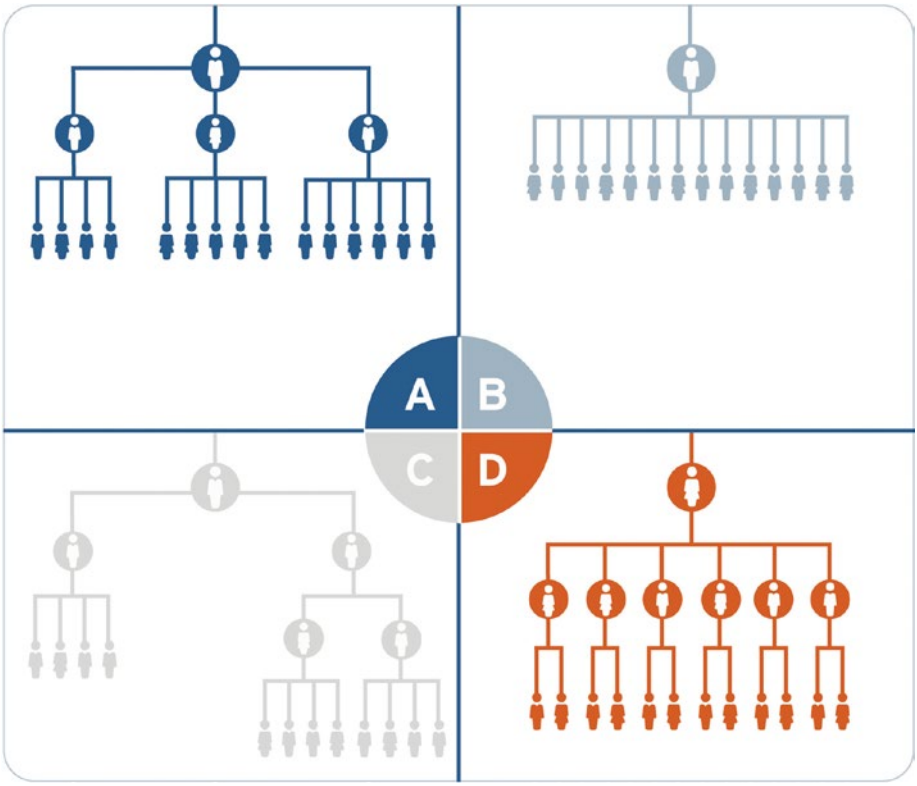

Figure 9-1. Examples of Organizational Charts

Forming Teams

Given our rule of thumb for team sizes, the best-case scenario for forming a new team is like natural cell division: the team grows to about ten people and can then be split into two separate teams. If you are able to create a clear division between those different teams so that each has its own role and autonomy to accomplish things end to end, then you are on the "happy path" here.

A lot of times, though, the split will not be straightforward. Many executives reach for the refactoring prematurely, spinning out of the bigger team a one- to three-people team. As long as possible, I would refrain from doing so. These smaller teams, primarily when bigger teams already exist, tend to introduce more overhead than is worth it. If the existing manager can manage the team a bit longer, until it grows a bit more, that would be better.

Another thing to keep in mind is that sometimes when teams are split prematurely, you might be pigeonholing them to a role that's too small. When that happens, the team's scope will never justify growing, and with time you end up with a lot of tiny teams, requiring more managers, syncs, goal-setting, and so on.

The matter of who manages these teams will be covered next, but it is first essential to take care of the teammates themselves. Be clear about the changed scope of each group. Talk about any transition period where they will still have to do some cross-team work. Share the revised objectives for the team. As we covered in Chapter 6, the goals should be tailored to each group for maximum impact and so might need to be updated and not merely split between the new teams. As these teams are created, ensure that the process will be adjusted fast: meetings updated, Slack channels created, and so on. The team should feel like a real team and fast.

Lastly, these principles apply when you consider splitting a group into two groups or merging existing teams. It's important to communicate your reasoning and, in general, treat such changes as the change initiatives covered in Chapter 7.

Managers

One might assume that any change that applies to teams also applies to managers, but reality tends to be a bit fuzzier. When you create a new team, likely, you will also appoint a new manager for it. However, that's not always possible or even the right thing to do, and in those situations, you should be aware and make conscious decisions about the path you take.

Ideally, you will have a great candidate for promoting or stepping up when a new team is created or when someone switches positions. When that is not the case, you should avoid doing what many think is the next best thing: prematurely putting someone in charge of the team.

By "premature," I do not mean that it is not acceptable to promote someone who does not have prior experience in a managerial position. I mean that the transition is done without taking into account several aspects:

- Make sure that the person is likely to be a good fit for the role, including expectation setting and painting a picture for them to envision how their day-to-day will look like.
- Take timing into account. I do not believe that a new manager should only be appointed after whatever current rushes are settled and the team is in a lull. Timing is about the availability of the support system the new manager will need to succeed (covered in the next section).
- Consider other likely changes that are imminent. No organizational change is going to be your last. Nevertheless, if you know of some that are overdue and will be decided on soon, it might be best to wait with introducing a new manager (e.g., if you are going to change a director for that group or move the team to be under a different director, you might decide to wait to get their input on the appointment as well).

When you take these aspects into consideration and realize that appointing someone right now would be a mistake, you have to buy time. That often comes down to having an existing manager in charge of that team. It is not uncommon to have a good team leader also be the interim manager of another team. Similarly, it is possible to have the group's director be the interim manager. Either scenario is not ideal, but it is often better than the other options you have. Looking at the bright side, this temporary solution allows members of your management team to get acquainted with groups they wouldn't have otherwise. That often has a positive long-term impact.

Another possibility, more common in larger organizations, is having an interim manager who's not currently a manager. If you have someone who's considering becoming a manager, you can make them an interim, with responsibility for some, but not all, the aspects of the team. The rest of the responsibilities, as well as oversight of the interim manager, are kept at a different manager (similar to what was described earlier). After a predefined period, you and the interim manager can decide whether this was a success and formally promote them or roll this back for various reasons. For example, you might realize they are not ready yet, or they decide management isn't what they thought it would be.

KNEE-JERK PROMOTIONS

I once started working with a VP of R&D about a week after he had decided to create the first two teams in his organization and promote two engineers inexperienced with management. Unfortunately, he instinctively skipped all of the steps I recommend: he didn't wait for the right timing, didn't correctly help them assess their fit for this role, and went along with it right before a big reorg. While one team lead was doing great with some coaching and guidance, the other struggled from the get-go.

We worked together to create a mentoring and coaching framework for the struggling lead and do the expectation setting (better late than never). The VP himself needed some coaching on coaching, and we made sure becoming a manager in that team was not merely "flipping a bit."

These steps made everyone understand quite fast that the excellent engineer was not a good fit for a manager in this organization (at that point in time). The VP was able to reverse the promotion while retaining her and dodging a near-miss that could have significantly harmed team morale. Knowing to deal with your mistakes is important, but some mistakes you can avoid.

The Management Bit

How do you create a new manager in your company? Is there any specific process? When I worked at IBM, we used to joke that being in management was merely a matter of having someone "flipping a bit" in an internal system. You kneel, someone clicks a button, and you rise again, dubbed "team leader."

Joking aside, that's not far off from the way many companies end up handling these promotions. You have a few discussions with someone who wants to become a manager (most often, along with discussions with their direct managers and HR). The day arrives where you have a single meeting with them about "management practices," and off they go. Management, though, is not like Neo plugging into the Matrix for training. You cannot go through a single meeting and emerge from it a manager (not a good one, at least).

Teams sometimes go to great lengths to create smooth and elaborate onboarding courses internally for their technical people. Rarely do the managers get a similar treatment. You can do better.

Manager Onboarding

You wouldn't give someone who just finished reading their first programming book commit access to your repositories and wish them good luck. The same goes for managers. Manager onboarding can be as elaborate as you'd like it to be, but it boils down to two key elements: knowledge transfer and an ongoing support system.

Knowledge Transfer

You obtain a lot of information about the way the organization works, how managers are expected to behave, what the managers will be measured by, and so on. This might all seem self-evident and obvious to you, but the truth is that it is not straightforward to others. Even if they are not new to the organization, some of these aspects of their roles can be ambiguous in their minds. For this reason you should make knowledge transfer explicit when promoting or hiring managers. Plan to have a series of discussions about the topics in the following checklist over a period of a month or so. Remember that it takes time to think and practice what you are discussing.

As your team gets bigger, it might make sense to have these "manager orientations" happen in batches. However, if your team is still small and promotions are not common enough, invest the time in having these done "just" for the one new manager. It might seem like extra overhead, but it is worth it. Keep in mind that you do not have to do all of these yourself. Some of the items in this checklist can be delivered by your directors, for example:

- **One-on-ones and feedback:** Talent growth is the most crucial aspect of a manager's role in the long term, other than generating impact. Being explicit about the company's standards for conducting these, the needed preparation, and even simulations are immensely helpful. This often ties to the way that performance management is done.
- **Metrics for success:** Communicate the way managers are evaluated in the company. Without aligning goals and objectives, they are likely to focus on different things.
- **Boundaries:** New managers might need to be told what exactly is within their purview. Are they allowed to talk about salaries and raises? Are there issues that have to be escalated immediately?

- **Situations and responses:** A standard part of all worthwhile training programs. There are typical scenarios that managers face. Providing your new managers with guidelines can significantly help to set them up to succeed. What do they do when things go off schedule, an employee is not delivering up to standards, or hearing someone on their team is looking for a new job?
- **Partnerships:** As covered in previous chapters, managers can greatly increase their leverage on the organization by having trusted relationships with colleagues and counterparts across the company. Introduce them to the critical people they should be working with. If needed, provide them with basic training about those worlds and help them set up working relationships (e.g., regular sync meetings).
- **Transitioning to management:** The move from being an engineer to managing a team is not always straightforward. Discuss whether they are expected to be involved in hands-on delivery, how their schedule should look like (similar to the time allocation covered in Chapter 3), and how to be informed of what the team is doing with micromanaging it.
- **Coaching:** Another vital part of the manager's toolbox is the ability to coach one's team effectively. Good engineers often make for lousy managers at first. That is because they are so acquainted with the intricacies of the job that they become too involved, never really letting go. Learning to delegate and provide the team with autonomy to make their own calls and mistakes is vital. Coaching new managers in coaching is important and will likely be an ongoing part of their support system.

Support System

As time progresses, managers will benefit greatly from having others whom they can share and consult with. A reliable support system is a must, not just for first-timers. Putting these in place at all levels of management is a catalyst for managers' growth.

First, there's coaching, either from their direct manager or an external coach. This is especially useful for matters that cannot be easily shared with others in the organization, such as their teammates' personal issues.

Similarly, there's mentoring by a peer, for example, assigning a more experienced "buddy" to new managers, so their questions have an address. Depending on the seniority and availability of your managers, this can be a great tool to speed up onboarding for new managers.

Another tool is management forums: regular meetings of managers at the same level and the same part of the organization (e.g., all engineering managers who report to the same director). Just as engineers can sit down together to tackle a technical issue without involving their managers, managers should have a *team* that they can rely on—without supervision. This approach has an added bonus where the managers learn to work together better, making their manager less of a hub.

You can pick and choose from these options if you are just kicking them off. However, all of them are immensely useful, and as a rule of thumb, all should be implemented by the time you have two directors or more.

Premature Organization

You've likely heard the old adage (attributed to Donald Knuth): "premature optimization is the root of all evil." Indeed, when it comes to creating software, nailing down details and decisions prematurely is error-prone and often a costly mistake.

The equivalent for management and leadership would be rushing to create clearly defined lines in a forming organization. A widespread occurrence in startups, other than the straightforward premature promotions covered earlier, is the tendency to provide titles, allocate responsibilities, and create a zero-sum game. I refer to this as *premature organization*, and it frequently creates a structure that's too rigid to change as the company grows.

As an example, I once helped an R&D organization of a young startup. Even before they had ten people, three were promised managerial roles, and technical responsibilities (e.g., backend architecture) were assigned to three others. Those early commitments ended up tying the executives' hands as the team tripled in the next 18 months. No one wanted to break a promise, but as they grew, they realized that some of those early decisions didn't make much sense. Furthermore, having already "assigned" responsibilities to some individual contributors was harming their efforts to hire more star engineers and capped the ownership of others in the team.

You may feel pressure to make promises to some of your team as well. Currently, the industry is in a state where any half-decent engineer can find a new job quickly. Still, if you give in to this instinctive urge, you will be slicing your team into fiefdoms much too soon. I've heard that "titles don't cost anything," but they do. They have the cost of organizational debt that is harder to overturn. If you think that technical debt is hard to keep under control, wait until organizational debt prevents you from hiring the people you need.

Organizational Growth

Many executives set up a hiring strategy that boils down to "get the best and most experienced people we can afford." Others, especially as organizations become bigger, have to accommodate the vast demand for senior engineers in the industry and hire more junior people with a lot of potential. In both cases, helping your team members' self-growth is paramount.

Some growth comes naturally, with the challenges of the job. Highly motivated individuals notwithstanding, you will have to take active measures to induce ideal growth.

Peter Pan Count

Every year, you can purchase Bordeaux wines at a very early stage—often before they are even bottled. Bordeaux futures are cheaper than buying the same wines five years later. That's because you make the buying decision based on some inputs that suggest the wines' potential, and that's it. The wine then needs some investment—time and storage in a proper cellar for a few years. If you can afford that investment and you know how to spot the high-potential vintages, you get a bargain.

Similarly, I recommend that most of my non-tiny clients start recruiting more medium and junior engineers. That is because there are more of them; they are cheaper and are more malleable. Yes, they require an investment in getting them to grow, but that investment has a tremendous return if done right. Further, your senior people should constantly be growing as well. They might be so good that they only improve by 5% a year as opposed to a junior who often advances 20–30% in that same time period. Still, 5% of improvement in your best people's output is worth a lot.

How do you keep track of growth across your entire organization? For that, I suggest that you keep track of Peter Pan employees. Peter Pan was a boy who never grew up—he remained a junior indefinitely. Most companies have Peter Pan employees—those who, even though they've been at the company for quite a while, have yet to improve their performance substantially.

My rule of thumb is that individual contributors, except for the most senior ones, should make a discernible "level-up" once every 1.5–2.5 years. For example, most junior engineers should not still be seen as juniors after two years on the job. On the other hand, if an employee makes a leap within a year or less, they are ahead of the curve.

Along with your performance reviews, keep track of the "leveling up" your employees do over time. At some companies, this is entirely subjective and at others is a matter of moving up the engineering ladder (see next). If you have too many Peter Pan employees—employees who are "stuck" in the same place for too long—it is likely that you have a training or a hiring issue.

Keeping track of your Peter Pan count (the number of stuck people) and of those who are on a trajectory to become so will be of help to address these issues on time. The best way to do so involves asking direct managers for their assessment, as well as tracking the progress individual contributors make along the different traits covered in your organization's engineering levels. It's a lot harder to justify demanding behavior change if you've only realized there's an issue after two years.

An organization with a healthy Peter Pan count will see the vast majority of people move up the different engineering levels (explained in the following) routinely. There's no specific number to hit, but if you were to plot how much time it takes employees with the company for more than a year to "level up," you should see a bell curve—the vast majority advance regularly, and on the edges you have a few exceptional people who do so fast and a few lagging behind. If the lagging becomes too substantial, you have a Peter Pan count that's going out of control.

THE PETER PAN–INDUCING CTO

A CEO approached me to help his company's struggling engineering team. Business was going great—prospects were urging the company to let them use the product. Unfortunately, things weren't that easy to get done technically. Schedules were continually being postponed, the quality was suffering, and the general feeling for the other executives in the company was that of working with a slow and antiquated team. Bear in mind this was a startup that was only a few years old.

As we started looking into it, everyone agreed that the Peter Pan count was not looking good—other than the CTO and maybe one lieutenant, no one was capable of taking on responsibility. The CTO had to spend the vast majority of his time helping the team, making decisions, and acting as a hub and puppeteer (things were not moving unless he was involved). Some more digging revealed that it was due to the CTO having trouble letting go of things and genuinely delegating them to others.

Realizing the company could not survive even six more months in this mode, we acted quickly: first, personnel changes and restructuring the organization to provide the team with more autonomy; then, tight coaching to key people along with incorporating the principles of impact-driven engineering and timely feedback. These allowed the team to make a sharp turn and start delivering.

Career Ladders

Career ladders or engineering levels are names for a framework that defines the growth path individual contributors have in your company—the "professional" track. They initially gained dominance at larger companies but are becoming more and more common across the industry. The main benefit is that by having a clear description of the different levels an engineer can go through, you create a path forward for the vast majority of your talented people who cannot or will not go into management. This relieves a lot of pressure to perform premature organization: not every good engineer will demand to manage a team as a way to progress.

Most organizations already have the beginning of a definition of engineering levels, though often very weak and ambiguous. It is likely that you already have "regular" engineers, along with junior or senior positions. A healthy career ladder will require clearly defining the distinctions between each rung on the ladder, along with more resolution: you will want to add more levels.

Crafting the right engineering ladder for your organization is serious business and will require effort as well as regular adjustment: as your team grows, you will realize that some previous definitions are not good enough or that an extra rung should be inserted somewhere along the way. Nevertheless, a good ladder makes hiring easier, creates a path for motivated engineers to follow, and removes a lot of cognitive load from your engineering managers.

While the definitions of different levels and naming them are outside the scope of this book,[1] here are the guiding principles you should keep in mind:

- **Cover multiple aspects:** Good engineering ladders should not be based solely on the technical proficiency of your people. For me, a senior engineer is about more than cranking out code. Include requirements for topics such as mentoring, evangelism, supervision, innovation, and communication.
- **Have enough resolution:** A ladder where everyone essentially gets stuck in the same level quickly or that has levels that require only a small effort to cross is not defined correctly. It might require some trial and error; however, this aspect of gamifying people's growth can genuinely help them achieve goals faster. That is only possible, though, if they feel like the challenge is fair and makes sense.

[1] Some good resources to look into are www.engineeringladders.com, www.progression.fyi, and https://ladder.glossier.io

- **Ladders should be transparent:** The definitions should be widely available, as opposed to the tendency of some companies to try and keep them obscure. Similarly, consider making the information about who has what title/level available as well.

Feedback Skeleton

A vital part of growth for everyone involved is to have quality feedback given continuously and in a timely manner. The feedback skeleton is an invisible structure that holds together the entire effort of advancing your team individually. It comprises the regular one-on-ones held across all levels of the organization, performance reviews, and peer feedback.

It is very rare that I talk about regular feedback and one-on-ones and hear a clear objection, such as "We don't do those." Just like testing, everyone "knows" these are good ideas, and so everyone claims to have them. Digging deeper, though, frequently uncovers that is a void statement: one-on-ones take place very rarely and without any depth or underlying structure.

As part of the manager onboarding described earlier in this chapter, you should make your expectations regarding feedback clear and share the way feedback is conducted in your organization. For example, one-on-ones with direct reports are expected to take place every one or two weeks instead of merely being a vehicle to perform work syncs.

Encourage candidness and open feedback from peers across the organization. The best teams I've seen are those where teammates could put egos aside and speak their minds frankly.

Providing Feedback Regularly

A major problem with some cultures is the radio silence: you toil away, hearing nothing bad about your work, only to find out 6 or 12 months later during the arduous yearly review processes that you are underperforming.

Getting feedback in such long intervals is akin to getting your code reviewed 3 months after it has been deployed to production: you no longer recall what exactly was done, it doesn't really matter anymore because things have changed since then, and you are biased to think that as it's been a while it's no longer valid.

A healthy organization should make it a habit to share feedback. I believe it is best to start, as a tech executive, by putting in place a laid-out process for management to follow about regular feedback and performance evaluations. As with code and product development, where we strive to get feedback and iterate fast to not waste time, the same can be reached with attention to your organization's talent. Helping them see where they are having issues in timely fashion means they are less likely to burn out or simply give up when they realize there's some hard work ahead.

One-on-ones are the regular meetings a manager has with their employee in private. They should be performed regularly. The cadence can vary, but for direct reports I recommend starting with weekly and maybe stretching it to biweekly. I'd be very wary of a longer interval. Assuming that your organization already has some loose structure of 1:1s in place, you should start with setting expectations with all managers to perform them in a timely manner from this point forward.

In case parts of your organization don't already do these or they were performed poorly, you will need to walk through your managers in starting to do these right. In one-on-ones, we want to focus on both positive and negative feedback. Note this isn't all what the meeting is about. It's a place for the employee to speak up and share, but we'll discuss that later.

This does not mean that after 9 months of radio silence your managers should merely send the invite for a one-on-one, get into the room, and start blasting away with all that needs improving right now. Anyone would feel blindsided by such a tactic and will not be likely to change or even take it well.

The longer this has been postponed, the harder it will be. The protocol I recommend is as follows:

- Managers should tell their employees that one-on-ones will now happen regularly and why, before sending the invitation. Otherwise, people tend to get needlessly worried, thinking that this new meeting is a sign of trouble.
- At the first meeting, the manager should come ready with an overview of the employee's accomplishments so far (meaning since the previous proper feedback was given), as well as a list of issues they will work together on improving.
- As this is likely the first time the employee hears the feedback, I recommend giving them a day to digest it. If needed, devote the next meeting to further discuss these.

- After this "sync" has been performed, the one-on-one handshake should be complete, and the process can now continue happening regularly.

Soliciting Feedback

As mentioned, one-on-ones are not just for providing the employee with feedback, but also about being a platform for the employee to speak up. Managers should actively solicit feedback from their employees in these meetings. Doing so is not always easy and requires effort, as some employees are not used to voicing issues or do not know how to do it in a constructive manner.

First, ensure that your managers make it clear that they expect employees to come to one-on-ones with their own agenda items (e.g., by starting the meetings by saying "Here are the two things I want to discuss. What are yours?"). If employees still come to the meeting unprepared, teach your managers to ask for feedback and sit there, not rushing to fill the silence.

Having some prepared prompting questions can also be useful. Some examples are

- What can I do to help you work better?
- What do you wish that I did less of?
- What has frustrated you in the last sprint?

Responding to Feedback

Once feedback has been successfully solicited, the worst thing possible would be to disregard it or shoot it down. Do it once and you are guaranteed to no longer hear candid feedback from that person again. You and the rest of the leaders in your organization have to treat incoming feedback with care.

First, managers should repeat what they have heard in their own words. This "mirroring" has two purposes. First, it helps the side giving the feedback to feel heard. Second, it helps handle the fact that words are tricky. We are trying to validate that both parties are talking about the same thing.

Next, accept the feedback and thank the employee for speaking up. I mean it: literally say the words "thank you." Do not rush into a "but…" or a "no…" response automatically. Thank them and think about it.

Only then, after you've stopped whatever knee-jerk reaction you might have had, should you act. Either discuss the issue with them further, or tell them that you will get back to them. Follow-up is important: if the manager decides that the feedback is invalid, they should not merely ignore it. They should get back to the employee and tell them why they have decided against the suggestion they got. Otherwise, again, you run the risk of team members learning that these are black holes: you provide feedback and nothing comes back.

Organizational Knowledge and Support

The bane of organizations, as they grow and bring on more and more people, is the formation of silos. It is tough enough to fight against silos forming between your department and others, such as cooperating effectively with the marketing or sales department. Allowing such thinking to occur within your organization can be a death blow to any team trying to achieve nontrivial goals.

I have a litmus test for spotting whether you already have silos forming without your knowledge. Consider the different teams in your entire engineering group. Are there any that behave differently than the rest of the organization? A team that has a slightly unique culture? It can start with small differences in ceremonies, using their own tools, or avoiding some overarching changes in the group. Once those start creeping up, silos follow swiftly.

To fortify your culture and reduce the chances of siloed thinking manifesting, you should take steps that ensure knowledge is shared across the entire organization.

Peer Forums

Everyone in your organization, no matter whether they are individual contributors or managers, should have at least one forum of colleagues that they can rely on. For managers, we've already covered management forums. These, when run across the organization, can help in increasing cross-group communications and creating healthier relationships.

For engineers, it is natural to assume that their teams are their peer forums. However, encouraging discussions across teams can result in ideas that have a bigger impact and better cross-pollination. Holding engineering forums (or sub-forums for different specializations, like mobile development) that involve all relevant engineers in your organization is a great way to cultivate openness and collaboration.

Professional Leadership

The other side of the coin for peer forums is direction. Committees rarely create innovative ideas, and so peer forums are best suited for tackling specific problems and knowledge sharing. However, some technical leadership is essential for calling the shots when it comes to leaping forward, setting standards across the organization, and similar.

Peer sub-forums are often referred to as guilds. Professional leadership would be the guild masters. As you grow, assigning this responsibility (even on a rotating basis) can be a catalyst for improvements and challenging the status quo. It also reduces the chances that a team will drift apart from organization norms and form a silo.

Horizontal Development

Another option for increasing cooperation is to leverage your organization's size for personal development. Often, talented employees are limited to growth within whatever team or group they originally joined. Keeping them in the same group has advantages, such as increased specialization. Additionally, managers hate "investing" in a teammate only to see them eventually move on.

Nevertheless, I strongly believe in natural movement between teams and groups as a developmental opportunity. For example, after 18–24 months in one team (and likely an engineering-level bump), encourage members to switch to a different team in the engineering organization. Constant swapping of experienced people between groups has positive long-term effects.

First, it becomes harder for culture pockets to take shape. The swaps "average out" a lot of the differences. Second, a fresh perspective is incredibly valuable, and a fresh perspective from someone who already understands a different aspect of the company is even more profound. A third advantage is that it creates better relationships between those teams and improves communication, thus resulting in easier collaboration when necessary.

One last advantage is that it is plainly good for your employees. Having to move out of comfort zones and readjust does wonders for motivated people and also supplies more interest to keep attrition low.

Your Personal Upgrade

Shaping your organization is hard work—some say it's the most challenging part of the job. The principles and tools described in this chapter should save you precious time, efforts, and relationship capital. You can leverage your new knowledge to create teams that will perform better in the long term and help your engineers be the best they can.

After all is said and done, your team is your greatest asset. Helping it operate smoothly is a necessity, and organizing it efficiently is granted to give you an advantage. Another surefire way to increase your chances of success is to foster product mastery across your team, which the next chapter will cover.

CHAPTER

10

Product Mastery

Cultivating an environment with a shared purpose and understanding of the company's vision

In technical organizations, it is common to hold in high regard those who know every nook and cranny of legacy systems, third-party APIs, and the tools you're using. They can be invaluable as they might short-circuit what would otherwise have been a couple of days of debugging by simply listening for a minute and saying, "I've got a hunch it's related to the caching mechanism in the gizmo system. I'd start with that."

As much as this technical mastery is essential, it is not enough in order to bring in real innovation and novelty, which are the staples of a leading R&D organization.

You can have the best senior engineers and architects tweaking and customizing different parts of your system, but it would have no real impact. For example, you could gain nothing more important than saving a bit on cloud costs. On the other hand, sometimes all it takes is exposing a single checkbox to marketing, and they can do things they never even dreamed of. How do you get more of the latter?

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_10

In Chapter 2, you learned how to hone your executive product mastery. Chapters 6 and 8 clarified how product mastery at all levels is necessary to unlock impact and innovation. In this chapter, you will learn how to bootstrap product mastery throughout your ranks.

How Product Mastery Looks

Before going into creating product mastery, you should learn how to spot it and assess your current situation. If you need to imagine what this product mastery is, it helps if you've ever worked at an early startup. At the stages where every single person is sitting in the same room and aware of the vast majority of micro-decisions being made, it is likely for the entire team to have high levels of product mastery. That's because they understand exactly why things are done the way they are, they know what their tangible goals are, and they have a real understanding of the users and their needs, either because they hear it first-hand from users or from the person who talked to them directly.

My first job out of the Israeli Defense Forces was in IBM, working on a storage product of a recently acquired company. In my whole time there, it was apparent that the old-schoolers understood precisely what to do and how they'd like to push the product. However, for us newcomers, it was a whole different world. We had no previous knowledge of enterprise storage. We were four times removed from the nearest customer *email*, let alone meeting or talking to them directly. It felt like operating one of those glove boxes for manipulating hazardous materials—we were doing things without any feeling, trusting the instructions we got.

In such a situation, how can you consider tradeoffs and trust your team to effectively make the dozens of micro-decisions that are required from software engineers? Contrast this with my very next job, the first employee at a fintech startup.

Even though the market was American credit card holders and we were all Israeli, we were fully invested in understanding the product. In that room, product mastery meant that we learned the names of all major US retailers to spot anomalies in credit card statements and to know how credit scores work. We read everything we could get our hands on and got statements from every American acquaintance who agreed to share them with us.

A bit later, I realized the stark contrast between the two positions and how that product mastery helped our team's efficacy, innovation, and motivation. Product mastery can sometimes provide you with a single giant leap forward. Other times, it manifests as dozens of tiny decisions taken every week that slowly but surely make the product better as a whole.

Hinderers of Mastery

There are a few reasons I've come to see as repeating factors of lack of mastery. They usually take place as the company grows. Here are the top five reasons for product illiteracy:

Organizational pigeonholing: As the company grew, it brought on more complex structures, and thus communication got scarcer across departments. If product, marketing, sales, and R&D never cooperate but pass down decisions and memos from one department to the other, you get silos. Too many companies are trying to "fix" communication by avoiding it, demanding everything be managed in Jira-like systems. When you're not conversing and cooperating, each side is forced to work at a lower altitude. We addressed some parts of this in the previous chapter, specifically when covering the right time to create new teams and their role definitions.

Relying on the old guard as a crutch: You might have senior people who have mastery and can answer the important questions. But, when those seniors do not work to teach the rest of the organization actively, they become oracles. You come to their room with a question, get an answer, and leave. The new people do not understand the "why." Thus, no development happens when it comes to their mastery.

Stressing technical mastery: Organizations often go through a phase where they feel the need to tighten quality and focus on "doing it right" vs. the haphazard manner with which things were done earlier. Of course, that's mostly for the better, yet focusing on this for too long or too much usually results in a hiring process that is heavily unbalanced toward technical mastery. It also creates a mindset that values learning the bits and bytes over learning the business.

Lacking transparency and impact as values: If the top leadership of your organization is not transparent, how many people do you imagine would work even harder to learn what's happening? If they are not rewarded for impact, why would they focus on that instead of things that are right in their comfort zone?

Isolation from the clients and solution: When people rarely answer support emails, never see a client face to face, and don't dogfood your own products, how can you expect them to understand what they are doing truly? Without a real connection to what you're working on, the product and its concepts remain intangibles in your team's mind.

Bootstrapping Product Mastery

Once you realize that a gap has been formed and too many on your team need to gain product mastery, it is time to kick off a process to ensure it happens and accelerate it as much as possible. This process should then be packed up and become a part of your onboarding training for new employees. Thus, you guarantee a shared base knowledge about the product that everyone in your group will share. Going through this just once, without the extra effort of systematizing it, is an awful waste.

Product 101

Product mastery must start with the product itself. A walk-through of how the product operates and how it should be used is an excellent first step. I find that the knowledge of teaching this often already exists in companies. The product managers, QA personnel, or customer success personnel are likely to be providing similar training to their new employees, and your people can tag along. I've seen successes in sessions that include hands-on parts where participants have to solve real-world problems with the product, just as the users would.

An important note here is to make this walk-through encompass all aspects of the product, including those that are not trivially part of it. For example, most B2B products nowadays consist of what's clearly "the product," which is often a web application that users click around. Still, for them to work, they have to be integrated with all sorts of other systems that the customers are using. Those integrations, by API, file transfers, or any other means, should also be covered. Ideally, people should go through the steps of setting such things up, even (or *especially*) if it is arduous to do so. Another example of hidden product realms is internal tools that are developed or maintained by R&D. These are rarely grasped as part of the product. However, they have the potential to be real force multipliers or productivity sinks. Learning how internal users use these is important as well.

Getting Down to Business

When discussing your executive product mastery crash course in Chapter 2, you learned about a whole slew of different business aspects that you should be aware of. Naturally, such a level of detail and transparency isn't relevant for the entire organization. Nevertheless, without enough understanding of how the business model is going to work, the team will not be able to focus on the right things.

Consider the following example of a common misalignment that can be prevented. Many enterprise products have different buyers and actual users, for example, compliance products will be purchased by the compliance officer but eventually used by others throughout the organization. When this is not clear to everyone involved, product teams are prone to focus too much effort on the regular users' needs. In fact, the actual selling points will be the control and reporting capabilities that are used by the compliance officer once a year.

Educating the team about the different aspects of the business and revenue engines will help avoid similar mistakes. I sometimes hear objections to doing this because it comes across as "unsexy"—engineers are naturally more interested in working on other parts of the product than features that are mainly there for aiding in sales. I get that, and I believe that a healthy balance must be achieved. Even so, such transparency is the only way to avoid frustration. Being told to work on the unsexy bits might not be fun, but doing so without understanding the business impact is even worse.

Roadmap and Vision

Just as you have to be involved in the formation of the roadmap to prepare your team best, your team members should be aware of the roadmap to make all kinds of better decisions along the way. For example, knowing that the company will be expanding to new markets in eight months can enable the team to make little investments to prevent an overreliance on only currently operating in a single country.

Alongside the clear roadmap that you have for the next few quarters, I highly recommend discussing the bigger vision for the company. Is the current product just the "first play" in greater world domination? Once again, awareness of the grand scheme of things, even if it might change later, can reduce rework later on. The vision, accompanied by the company's values, should be a major part in forming the team's mode of operation.

The Market

Finally, the team should be acquainted with the "problem space" as well as the "solution space." First, that means that they should be familiar with the kinds of users that the company might have, why those users need the problem solved, and so on. This includes familiarity with prominent figures in this space (like thought leaders or relevant regulators) and basics such as the jargon for your world.

Second, help the team learn about the other players in your space and how you differentiate from them. No one should aim to merely copycat your competitors or even one-up them slightly. The purpose of this is to make the team aware of how your market is operating and see how other smart companies are approaching the same problems.

When Apple first introduced its Apple Silicon Macs, all the reviews were that these first-generation, low-end machines were running circles around much more expensive models running Intel's offerings. For months, many bright Intel engineers and managers still ignored this. I consider this kind of disconnect from the industry a great example of the importance of product mastery. Had they learned a bit more, they might have realized how big of a deal this disruption of their industry was likely to be.

Maintaining Mastery

The effort you make at the beginning should provide your team members with a solid foundation of understanding of your product. Unattended, that foundation tends to dim in people's memory and get out of date. That is why you should have routines in place to keep your team's knowledge current.

Eating Your Own Dogfood

By far, the most effective way to gain product mastery is to have your team be actual users of the product. When people use the product, they gain first-hand knowledge and understanding of how it operates, what it does well, and where it needs improving. The rest of the methods described later in this chapter lack this advantage: they all consist of second-hand knowledge and experience.

I would urge you to go to great lengths to make it possible for your team to use the product. Sometimes, it is trivial, for example, for B2C applications that anyone can install from the App Store themselves. Other times, it might require more effort, like customized development to allow your team to use the product (a common occurrence here in Israel, where many companies only officially support international markets like the United States but have

hidden support for Israelis, so the R&D team will be able to try out the product). This special development might add complexity to the product and will require some extra work to make sure you do not taint your own metrics and data. Nevertheless, it is so valuable that it is worth it—don't hesitate to do it when at all possible.

Another critical aspect for enabling this dogfooding to happen is to have measures in place to promise the privacy of your team. I've seen teams where engineers were reluctant to use the product because they knew anyone else in the group could query the database and see their private information. Whether it is by obscuring, audits, or any other measure, sometimes you have to ensure privacy for your people to trust the product and their teammates.

Rubbing Elbows with Users

The next most helpful way of maintaining product mastery is to make sure there is constant "friction" between your team and users. The less filtered a view they can get of users, the better. Witnessing with their own eyes how the features they've worked on are used (or misused) in production can be a fantastic eye-opener.

Here are some approaches I've seen deployed successfully:

- **Regular user interviews**: Invite users for an interview and have it be recorded. Then share that recording with the entire team, so they can see what language users are using to describe the product and how they operate it. Some companies have turned these into podcasts that are distributed internally.
- **View usage recordings and analytics**: All sorts of analytics tools allow you to visualize how users go through a session using the product. These can be very helpful to expose dissonances where users end up using the product differently from how the company intended them to.
- **Lurk around user watering holes**: The team should regularly check out places where your users naturally congregate and discuss. Typical examples are organic blog posts, reviews, and mentions on social media. Other less common watering holes include the company's help forums or public issue reporting (e.g., GitHub issues on public repositories).
- **Work with users**: See in the following for more detail about leveraging customer support.

These are intended to make the users real and help the team have a vivid understanding of the customers' needs, preferences, and behavior. Otherwise, it will be hard for engineers to have a mental model of the usage that's true to the actual use.

THE QA-ING EXECUTIVE

I was attending daily syncs of a startup and noticed a pattern: the tech executive would routinely be the one spotting issues in the demos the team was doing or would be the first to report errors in production. After seeing him report a bug that caused the web application to hog 100% of the CPU, I realized the team was just not familiar with the product and not using it at all.

In discussions with the leadership and product teams, we created a boot camp similar to the one described earlier. To it, we added specific efforts like development during intermissions (see Chapter 8) that enabled most of the engineering team to use the product regularly—bugs in it would now prevent them from getting their own work done. Along with that, I helped the executive put on a coaching hat—rather than be a crutch that the team could rely on to spot issues, he learned how to help them spot those themselves and investigate them to prevent them from reoccurring.

A couple of months later, the team saw a clear improvement in its quality, engagement of R&D, and more product mastery–driven innovation and ideas in intermissions. Having a stake in the product (e.g., as they were using it, false-positive alerts would mean waking up at night for no reason) made all the difference.

Sharing Roadmaps and Vision

We covered the importance of sharing the high-level roadmap and goals with the team in Chapter 6. Comprehension of where the company is going helps individuals realize how their efforts help and hones their attention. A big part of product mastery is understanding the grander vision of the product and how it is evolving or expected to develop.

I've seen a lot of value coming not just from merely sharing these but doing so in a format that solicits feedback, for example, a quarterly all hands where everyone can voice concerns about the roadmap and prioritization. Being able to say something doesn't make sense to your direct manager is good. The ability to take it a step further and do that with the VP Product and VP Engineering is empowering.

Teams that are driven by impact are given business objectives or goals and then have to come up with a plan. Objectives that are easy to directly map to effect on the customers help keep teams in constant connection with the product and maintain mastery.

Watching the Market

The team cannot work in a vacuum. Even if they regularly see what your customers are doing, they might only be glimpsing a part of the bigger market that you are operating in. Therefore, you should find ways to stay up to date habitually.

Ideally, the team should hear about these things relatively close to when they happen:

- Competitors release a new feature that you do not have or a feature that's intended to compete with your existing offering.
- Competitors reveal interesting information in an earnings call, interview, or blog post.
- Academic research that's relevant to your product's future innovation is released.
- Discussions about possible regulatory or similar changes in your market.
- News items about the problem you are solving or the industry your customers are mainly from.

The problem is that these are a lot of things. Only the most motivated individuals will regularly follow up all the relevant news sources, blogs, and influencers to stay on top of along with the rest of their interests (e.g., technical aspects they are trying to remain on top of).

Some teams have success with having this be a rotation: every few weeks, someone else is tasked with sharing their findings in a summary email with many links or a quick 15-minute meeting. Others leverage those who naturally do this and give them a stage to share their learnings, like an "official" internal newsletter that people come to expect.

Customer Success

Lastly, a magic-like way to ensure product mastery is maintained is by having the team regularly shadow, or even practice, customer success. If every engineer takes part in answering emails and seeing what issues the users report most often, even merely one day every quarter, that will be enough. This also has many positive side effects, like better communication between customer success and your department.

A CTO I talked to about this told me how things changed after his company went through a hectic period and everyone *had* to help customer success. Innovative ideas about features and improved processes were coming from all directions. Engineers became more aware of the micro-decisions they were making in the product and how those affected users. New internal tools were created that turned into force multipliers for everyone involved in customer support.

Meta-reviews

A useful tool for sharpening the team's ears to notice impact better, and hone their mastery, is to perform regular meta-reviews. As opposed to regular retrospectives or reviews, which review the manner in which work was performed, meta-reviews shed a light on the results of the work after some time has passed and help everyone involved learn of its significance. Meta-reviews are done by the decision-makers and business owners to regularly review work that has been out and in the hands of users for a while. These reviews are the platform to evaluate the decisions made by the team when they chose what to focus on. If something is a success, how can you get more out of it and more like it? Otherwise, what can you do so you won't do this again, and should you plan work to remedy the situation?

Once you have established the meta-reviews, you get a tool that regularly helps you take stock of recent wins and gain vivid examples of how the work has affected your customers. That makes mastery tangible and ties the previous examples to the end result of product mastery, which is *impact*.

Your Personal Upgrade

Product mastery is a prerequisite for a team that will sustainably perform needle-moving work. Maintaining it as the company grows and people tend to pigeonhole is nontrivial, but the steps laid out in this chapter should provide you and your management team with an approach to bootstrap and maintain it across the team.

This mastery is so effective and conducive to amazing work, yet often left behind. Longing for "the good old days" where everyone around possessed such mastery is why many talented people end up leaving bigger companies and routinely go back to smaller teams. That is why this concludes our journey in forming your personal operating system. This last tool is not purely yours as an executive. Rather, it is a skill that you have to teach your organization to perform and keep current. Armed with it, you will all be doing amazing things together.

CHAPTER

11

Conclusion

Before you put down this book, make sure you are set up to get the most out of your investment in reading it. If you have not done so yet, choose the first matter or two where changing your approach in line with the principles outlined here will benefit you the most. Schedule the next leadership blocks on your calendar. Set a recurring task to review yourself and retrospect.

Every day, I see how easy it is for my clients in roles like yours to get sucked into the day-to-day and forget to take the time to lift their heads from the upcoming meetings and consider the bigger picture. I would recommend that you revisit this book quarterly to see where you can improve some more, but I know that is asking too much. I have compiled a digital cheat sheet to remind you of the main principles and guidelines in this book to help you with that. You can download it for free at https://avivbenyosef.com/teos-cheat-sheet/.

World-class teams are not created in an instant. Whether you are forming a team from scratch or managing an existing organization, you have a long way ahead of you. Lucky for you, that's the fun part of the job. The quest is about continually improving yourself and your team. By toiling daily to increase your abilities, and thus your leverage and impact, you add significance and meaning to your role. Thank you for letting me help you along your journey.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7_11

APPENDIX

A

Common Tech Executive Titles and Roles

This appendix lists titles you might come across in your career and how they differ. Keep in mind these are the most common cases, but your mileage may vary:

> **CTO:** Chief Technology Officer is by far the most common title and also the most ambiguous. In smaller companies, the CTO is likely to hold all of the LEAP responsibilities described in Chapter 1. If, along with the CTO, the company also has a VP managing the execution, the CTO tends to focus on *Architecture* and *Evangelism* responsibilities in most cases. In these cases, CTOs and their teams serve the role of bringing in innovation, researching new approaches to be taken, and communicating with the outside world. In the vast majority of companies, CTOs report to the CEO directly or to the COO. In some companies, these responsibilities are shared across a few VPs of Technology as opposed to a single CTO.

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7

CIO: Less typical at startups, though very prominent at later-stage companies. The Chief Information Officer is often in charge of internal processes, regulations, security, and infrastructure. Some companies have both a CTO and a CIO, others only one. The most distinctive classifier I know for determining which is more fitting for your company is whether the company is developing a product for external use (should have a CTO) or whether the technology is being used solely for internal purposes (should have a CIO).

VP Engineering: An executive who is responsible for and manages the different engineering teams in the organization. Though the title says "Engineering," these vice presidents might manage not solely engineers, but also product managers, analysts, UX engineers, and so forth. It is effectively always true that a VP of Engineering is in charge of the *People* responsibility. VPs of Engineering are executives and should be treated as such. In the majority of companies, they report directly to the CEO, though instances where they report to the CTO are not uncommon. In larger organizations, there might be several people with this title, each in charge of different parts of the entire tech team. These scenarios sometimes also result in a hierarchy where several VPs report to someone with the title of SVP (Senior Vice President) or EVP (Executive Vice President).

VP R&D: The Vice President of Research and Development is, de facto, almost a synonym for VP Engineering. In some companies, though, the distinction it is intended to create is that the person holding this title is in charge not just of normal execution, but also of research efforts. In other companies, there would be a VP Engineering and a VP Research. The research team often consists of data engineers, data scientists, researchers, and analysts.

VP Product Development: A newer appearance, this title is usually part of a move to focus on impact alongside a tech organization comprising cross-disciplinary teams (as discussed in Chapter 6). All in all, similar to the previous titles, but often used to signal that they are managing the product people of the company as well.

Chief Architect: Virtually always, this role is about the *Architecture* responsibility, though sometimes it might also include some of the *Evangelism* responsibility as well. The Chief Architect position often involves the management, either solid or dotted, of a team of architects or senior engineers.

Head of Engineering: Used less often and not always an indication that the holder is an executive. In some smaller companies, such a title is used as a placeholder where the person in charge of the engineering team might eventually be "bumped up" to one of the other titles or to allow the board to put in place someone more experienced in the executive position later on.

Director of Engineering: Normally, the title of a director means that the person is a manager of managers, usually managing team leaders or engineering managers. However, sometimes it is used in a similar fashion to the title of Head of Engineering, to indicate the most senior person in charge of the tech team, at least temporarily. In this latter usage, the Director of Engineering often holds the responsibilities and purview of an executive in practice, though not always by definition.

Field CTO: The newest and rarest of all of these titles as of the time of writing. Field CTOs are in charge of *Evangelism* and spend most of their time communicating with clients, prospects, and partners. They are the face of the company when it comes to technical matters and might assist with creating partnerships, sales conversations, and communicating client needs and feedback from the "field" back to the engineering team.

I

Index

A, B

C

D

A. Ben-Yosef, *The Tech Executive Operating System*,
https://doi.org/10.1007/978-1-4842-6895-7

E

F, G, H

I, J

R

S

T, U, V, W, X, Y, Z