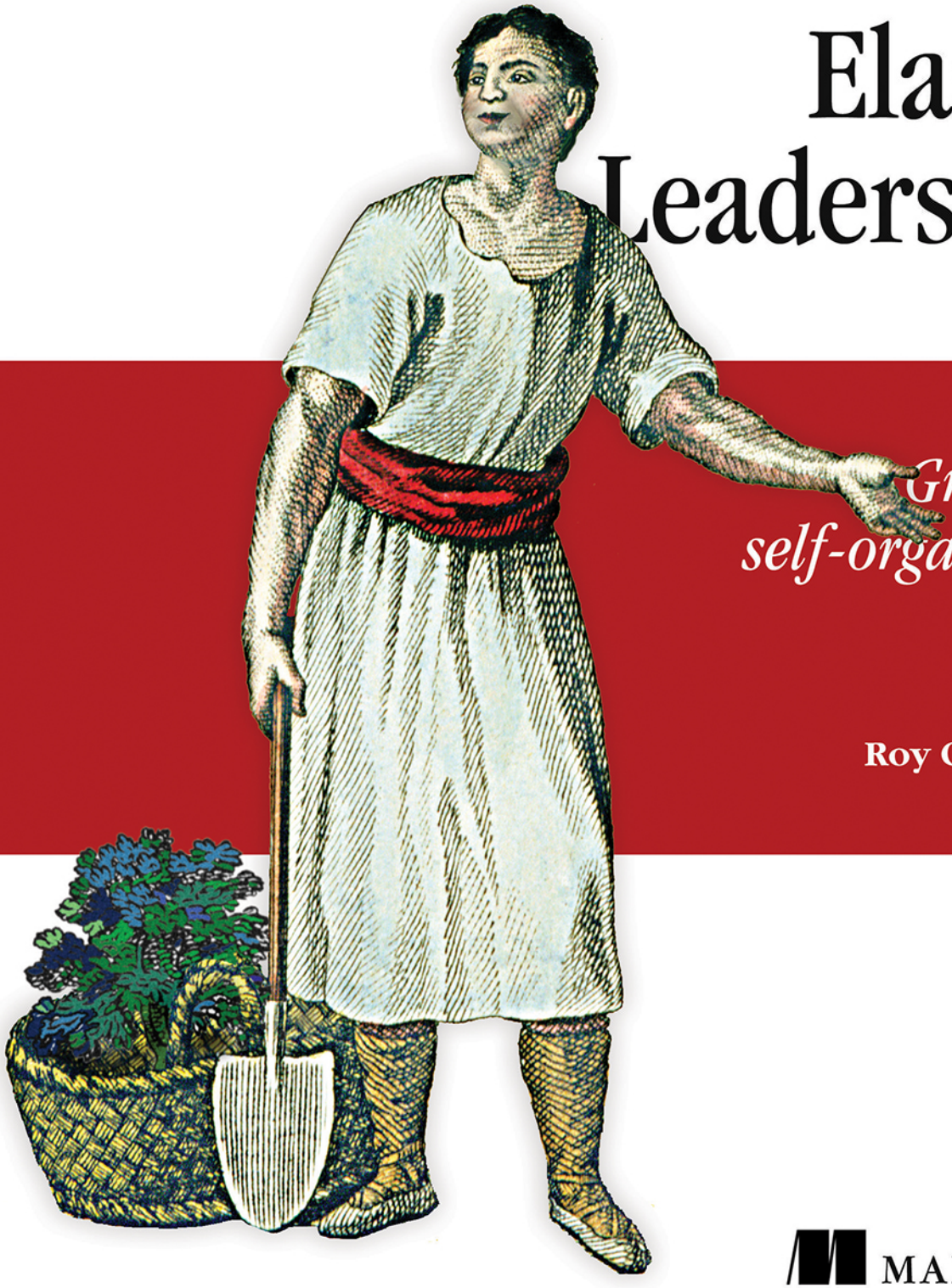
Elastic Leadership

Growing self-organizing teams



Roy Osherove

MANNING

Elastic Leadership

GROWING SELF-ORGANIZING TEAMS

ROY OSHEROVE

MANNING
SHELTER ISLAND

brief contents

preface *iv*

1 Striving toward a Team Leader Manifesto 1

1.1 Why should you care? 3
1.2 Don't be afraid to become management 3
1.3 The Team Leader Manifesto 5
1.4 Next up 6
Summary 6

2 Matching leadership styles to team phases 7

2.1 The role of the team leader 8
2.2 Growth through challenge 8
2.3 Crunch time and leadership styles 9
2.4 Which leadership style should you choose? 9
2.5 Leadership styles and team phases 11
2.6 The three team phases 11
2.7 When does a team move between phases? 13
2.8 Next up 14
Summary 14

preface

"There are no experts. There is only us."

These two simple sentences always make me feel lonely. They were uttered by Jeremy D. Miller, a software developer and architect I've come to appreciate over the years. These sentences give me the feeling that there's nobody else to turn to—that I have to start trusting my own instincts, and that whoever tells you they *are* an expert is either lying or wrong.

During my career, which consists of almost two decades in the IT business as of the time of this writing, I've come to realize that "There are no experts. There is only us" is very true. During one of my first jobs as a programmer, I joined a team working on a government project (the project was all in Visual Basic 6.0). The team, including my team leader, had no idea what they were doing, but because I *also* didn't know what I was doing, I *assumed* that whatever people were doing was the right way to do it.

As time went by, I began to read books about how software development *could* work, including *The Mythical Man-Month* by Fred Brooks (University of North Carolina at Chapel Hill, 1974) and *Peopleware* by Tom DeMarco and Timothy Lister (Dorset House Publishing, 1987). I looked around and recognized all the problems those books were talking about right there in front of me.

But nobody around me said anything about the crap we were building or the crap we were taking from our managers and customers, and *certainly* nobody said *anything* about the crap we were giving to our customers and managers—there was only silence. Nobody was talking about careless, helpless programmers. Everything was *fine*. To paraphrase (and counter) a famous saying by comedian Louis C.K., "Everything was crappy, and nobody cared."

These were *good people*. Some were my friends, and they didn't intend to do any harm. We were doing our best, in the same way ants do their best to vanquish raindrops along their path to the anthill. But we weren't looking *up*. We weren't trying to understand and predict *why* the rain fell or where it falls from. We didn't plan better

ways to get to the anthill, and we didn't get better raincoats to protect ourselves from the rain (OK, ant raincoats are a sign this analogy is breaking down, so I'll stop).

We were all just *there*, doing our ant-like jobs. Project late? Sure; that's life. Quality is lousy? Sure; that's life. Debugging until 3 a.m.? Sure; that's normal.

Was I the only one reading books? No. There weren't many trade books, but there were *some*. But the books my coworkers were reading weren't getting them anywhere; or if the books had the potential to help them, maybe they couldn't find the time to finish the books and get there.

There was no sense of craftsmanship. But there was also no sense of professionalism. There were just big downward spirals for every project, as far as the eye could see.

This workplace wasn't unique, by any means. I encountered many companies like this, with variations, over the years. I hate working at places like that; I always want to make a difference.

This book is about making a difference and getting other people to make a difference as well. It's the book for those who feel hopelessly trapped in their jobs, even though they're architects, scrum masters, team leaders, or senior developers. It's the book I wish I'd had when I first became a team leader.

The book started out as a passionate blog that I kept over several years at 5whys.com (now ElasticLeadership.com). At some point, I collected all the blog posts and published them in a self-published book at leanpub.com titled *Notes to a Software Team Leader*. When Manning offered to publish it a couple of years later, I jumped at the chance to revise and add content to the book and publish it with the same company that helped publish my first book, *The Art of Unit Testing*. I'd like to thank Manning for helping me publish this new edition of *Notes to a Software Team Leader*, now titled *Elastic Leadership* and with a new format and new and updated content.

1 Striving toward a Team Leader Manifesto

This chapter covers

- Principles of leadership
- How to become and remain a good leader
- Developing other leaders from your peers—growing people

> *The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore all progress depends on the unreasonable man.*
>
> —George Bernard Shaw,
> *Man and Superman*

This chapter outlines the principles for becoming, and remaining, a good leader, and also developing other leaders from among your peers.

Leadership in general—and team leadership in the software business in particular, where little training, if any, is provided—isn't easy to accomplish, or indeed to

measure. We can begin with the assumption that most people in software—like you—have no idea what they're doing or what they should be looking at when leading their software teams. Yes, that was me too.

This book springs from my personal experience of what *worked* for me and what *didn't* work for me when I was leading software teams.

One thing is certain: the way we work with our teams must improve. We must become better adjusted to the current reality and needs of our team, our business, and ourselves.

A few years ago, I was speaking at a programming conference. The person who introduced me also mentioned that I was looking for a job. At the end of the talk, a woman came up to me and said, "We want to hire you as a developer." Boy, was I proud of myself that day. I began working there, and the woman became my team leader.

My first job was to write code that uses the local network and searches for specific data on remote hard drives (no, it was not a virus). I worked on it alone. As the day went on, I struggled and struggled. It was harder than I thought. Another day came and went, and I was still at it. My ego wouldn't let me ask for help. They had hired me off a conference stage because I was supposed to be an expert. How bad would I look if I said I didn't know how to solve this problem—the *first* problem I had been given at that job? No. I was determined to work on it until I figured it out.

To add to my grief, there were no daily stand-ups, and every few days my team leader would pass by me and ask lightly how things were going. I would deliberately say it was "in progress" and move on with a solemn face. A week went by and I began avoiding people's eyes in the hallways. I pretended to be busy and pensive, but I was drowning inside. Every day that went by, it became harder to admit I was stuck. Every day that went by made it harder to ask for help; I would look more stupid with the little progress I had made during all that time.

At some point, I took sick leave for a few days. I couldn't handle the silent pressure. When I came back from the leave, my team leader approached me and said, "Hey, by the way, that thing you were working on? One of the devs and I sat on it and made it work in a few hours." They had found a simple way out of my mess, and I felt both betrayed and foolish. I was fired from that company shortly after with a budget-cut excuse, but I knew better.

There were many things that could have prevented this Greek tragedy:

- I could have been a bit more courageous and said, "I don't know how to fix this," early on.
- We could have had daily stand-ups where my predicament would have been discovered early.
- We could have had a rule whereby no one was allowed to work on something on their own for more than a day.
- My team leader could have approached me and done one-on-one meetings weekly or biweekly to discover what was up.

But nothing happened, until the worst happened.

The values I introduce here, and throughout this book, can help prevent such tragedies from transpiring, or at least make them very unlikely. I hope they can help spark a desire in you, the reader, to become better at what you do.

1.1 Why should you care?

You might feel helpless in leading your team to do the things you believe to be "right." This book can help.

You might feel like you want to keep your head above water. This book will help you accomplish much more. You might feel clueless as to what it is you're supposed to be doing with your (future) team. We'll tackle that too. You might be broken and scared because you have no idea how you're going to get out of a bad situation at work. Welcome to the club. I hope I can help. I hope I can because I've been there—clueless and scared. I was lucky enough to have some good mentors along the way who challenged me to do the things that I was afraid to do, to get out of my comfort zone, and to learn things I didn't even know I didn't know.

I think team leaders around the world all suffer from some of the same basic bad experiences. Most of us weren't taught how to do this type of work, and most of us are never going to get mentors to help us through it. Maybe what we're all missing are some good, old-fashioned people skills. Maybe we're missing some direction, some overall purpose for our leadership role. I feel that if we go uninformed into a leadership role in software, without a sense of purpose and strategy, we've already lost the battle to create real teams. Yes, our head might be above water, but are we there to slog through another day? Are we supposed to be this helpless? We want to be *leaders* who create not only real value but also happy teams, both fulfilled and loyal.

1.2 Don't be afraid to become management

A lot of developers who are promoted to leaders, or who are offered the opportunity to become team leaders, seem to be resistant to becoming management. I can understand some of the reasoning, but I don't accept it. You might be afraid that your time will be sucked up by meetings, that you won't have time to do the things you love the most (like coding), and that you might lose friendships with people you currently work with. I agree that there's a basis for those fears. We've all seen (or been) that person who doesn't have time to do what they love, or fumbles a friendship because they've turned into a boss from hell, and so on.

Paraphrasing Jerry Weinberg in his book *Managing Teams Congruently* (Weinberg & Weinberg, 2011): Management, done wrong, can make these fears manifest into reality. But management done right negates them. Management, done right, is a very tough job. That's why you get paid more.

I'll return to this concept later in this book.

1.2.1 You can make time for the things you care about

A good leader will challenge the team and the people around them to solve their own problems, instead of solving everyone's problems for them. As people gradually learn to solve their own problems, your time frees up more and more to do the things you care more about, and the things that matter more (sitting down with people, coding to keep in sync with what's going on in the team and the code). Doing your job as a leader and challenging or asking people to accomplish tasks may indeed feel weird, but in my experience, doing it will garner *more* respect for you, not less. Yes, some things will change, but change is inevitable. You might as well own and control how things change.

It also takes time to challenge people, time that most teams don't have in abundance. Making slack time to grow the team's skills will be necessary.

1.2.2 Take the opportunity to learn new, exciting things every day

Nothing beats gaining new skills. You and your team should always be getting better and going out of your comfort zone to learn new things. This is essential to what a team leader does. Becoming a team leader requires personal growth, rising to the challenge of knowing your team and what you can expect from them.

1.2.3 Experiment with human beings

Yes. I said it. You have a team, and you can experiment with goals, constraints, and the different leadership styles described in this book. Experimenting is one of the most enjoyable and interesting things I love about being a leader.

1.2.4 Be more than one thing

You're not only a developer; you're also a leader. You can change things that bother you and do things that you think are right. How many times have you said to yourself, "I wish I could change X?" As a leader, you can do something about it. If you choose not to become a manager, you'll have far less influence. As my friend Jeremy Miller said,

There are no experts. There is only us.

As you may have noticed in the preface to this book, I think that statement is spot on. Sometimes you must be the person who gets up and does something. You'll be surprised by how many people will follow you. Remember that 90% of success is stepping up to the plate.

1.2.5 Challenge yourself and your team

I've heard this basic idea expressed in a couple of different ways:

- Do one thing every day that scares you. (Eleanor Roosevelt)
- Get rejected at least once a day (also known as "Rejection Therapy").

These ideas are powerful and good ways to make sure you're learning something. I believe that learning something truly new is neither easy nor simple. In fact, many times it's scary, annoying, or discouraging enough to make you want to give up halfway through the challenge. Leadership, done right, can be difficult to learn. But as you make progress, you'll be amazed by your growth. You won't be the same person you were before embarking on this journey.

When you read the words *great challenge*, what comes to mind?

Depending on your state of mind, *great challenge* can be taken either gravely ("I'm facing an awful great challenge") or enthusiastically ("Wow, great challenge!"). Choose the second option.

The following section may present a *great challenge* to you.

1.3 The Team Leader Manifesto

This manifesto is continuously being developed and refined. You can find the latest version of it at http://5whys.com/manifesto. The high aspirations of the manifesto can be seen in figure 1.1.

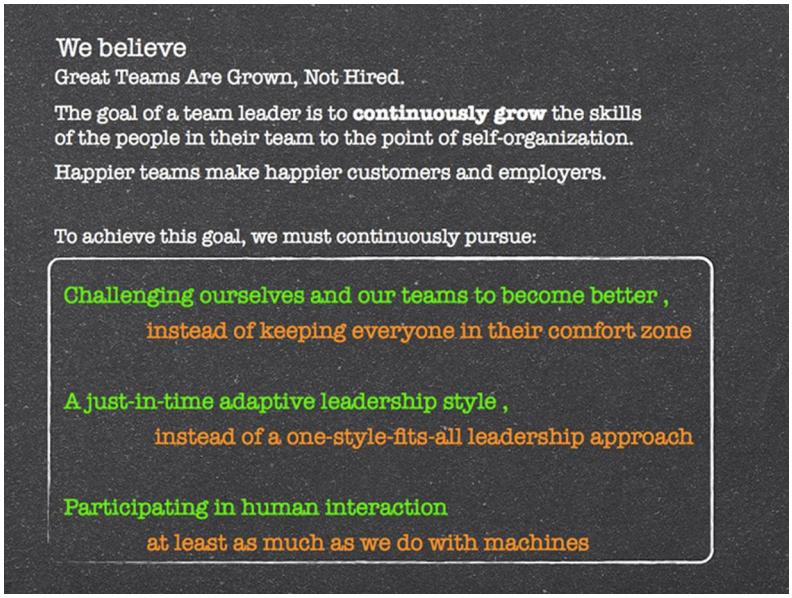


Figure 1.1 The Team Leader Manifesto

Here's a breakdown of the manifesto, line by line.

For us as team leaders, the goal and the way we measure our success is through the overall growth in skills of self-organization and self-maintenance in each member of our team, and in the team as a whole.

To that end:

- We accept that what the team needs from us changes continuously based on their skills for handling the reality of work, and we embrace a continuously changing leadership style over a one-style-fits-all approach.
- We believe in challenging ourselves and our teams to always improve; therefore:
  - We create slack time for the team to learn and be challenged.
  - We embrace taking risks for our team over staying safe.
  - We embrace fear and discomfort while learning new skills over keeping people within their comfort zone.
  - We embrace experimentation as a constant practice over maintaining the status quo
    - With people
    - With tools
    - With processes
    - With the work environment
- We believe our core practice is leading people, not wielding machines; therefore:
  - We embrace spending more time with our team than in meetings.
  - We embrace treating software problems as people problems.
  - We learn people skills and communication techniques.

1.4 Next up

The most important thing you'll need to know before you plan how to lead is the three team phases: survival mode, learning mode, and self-organization, which the next chapter discusses.

Summary

Leadership is a challenging thing to accept. If we don't accept it, there are others who will, who might not do justice to the power they wield.

There are many benefits to becoming a leader, including learning a new set of skills, becoming more valuable to the organization, and experiencing a greater feeling of accomplishment.

Sadly, many people take on a leadership role and don't know what to do with it, and they feel overwhelmed. This book is my way of contributing to help prevent that from happening.

Team leaders are the first line of defense (or attack) in making things work from the ground up. Without their help, an organization will be stuck spinning its wheels in neutral forever, no matter how hard upper management presses on the gas.

Without the leadership skills and practices outlined in the chapters that follow, it's hard to make positive changes in the work environment.

2 Matching leadership styles to team phases

This chapter covers

- Three team phases
- Leadership types
- Goals for team leaders

This chapter is a quick guide to recognizing the three team phases and the leadership types that make the most sense for each phase.

First, let's clarify *why* we need to define these phases. The reasons have to do with our overall goal as team leaders. What do *you* think your role is as a team leader? For a few years, I had to guess. Nobody told me and I had no one to learn from.

Before we go on, I want to clarify the terms used in this book. The words *phase* and *mode* are used almost interchangeably throughout the book. For example, in the *learning phase* you go into *learning mode*. *Phase* is where you think your team is; *mode* is how you react to which phase you're in. Think of it like the fight-or-flight instincts that we all have. When these instincts take over, we act a certain way. So

you might say that when we recognize we are in the survival phase, we initiate our survival instinct or survival mode. Or if we recognize that we're operating in learning mode while our organization is operating in survival mode, we can say we're in a survival phase and we should initiate survival mode behavior and instinct.

2.1 The role of the team leader

In the past, one of my biggest mistakes as a team leader was that I didn't recognize that my style of leadership was oblivious to the needs of my team.

Initially, my idea of what a team leader should do went something like this: a team leader should provide their team everything the team needs and then get out of their way.

Boy, did I think I was stellar! When people needed something—working code, infrastructure, a faster machine, or an answer to something—I was their guy. By my own definition back then, I was doing a great job. But that meant that I had little time for myself and was mostly in meetings or coding all day. I didn't allow myself to take even a couple of days to go on a vacation. I was always at work because people needed me. And it felt great to be needed.

Looking back, I can see lots of room for improvement, because the role of a team leader is vastly different from solving problems and getting out of the way.

2.2 Growth through challenge

Here's what I believe my role is today: a team leader helps develop quality people on the team.

I believe this should be your first "compass" in determining your behavior as a lead. You may ask, "What about delivering value to the company?" I believe that delivering value flows naturally from developing your team's skills. When people grow, value delivery also grows because skills grow. More importantly, the attachment and commitment of people to do the right thing also grows. Loyalty grows.

One of my mentors, Eli Lopian, told me this once:

People don't quit their jobs. They quit their managers.

I think that's a true statement (at least for the several companies I've quit). By coaching the people on your team to develop as team players or valuable working individuals, you generate internal value for them personally, not only for the company. True loyalty comes when you have everything to gain by sticking around and you realize it.

2.2.1 Challenge

More things logically follow if your guiding rule is to help people grow. To grow people at work means to help them acquire new skills. For them to acquire skills, you must challenge them. Therefore, you have to stop solving all their problems for them and coach them to solve problems on their own (with your guidance). If you solve all problems for your team, the only person learning how to do new things is you.

2.2.2 You're the bottleneck

When you solve all your team's problems, you're the bottleneck, and they'll find themselves unable to manage without you. If you're sick for three days, can you leave your phone turned off? Or are you constantly worried and logging in to the company's VPN to check and fix things that nobody else on the team can do? If the team has to wait for you to be available to solve problems, you're the bottleneck, and you'll never have time to do the things that matter most.

2.3 Crunch time and leadership styles

You might think, "Well, that makes no sense. We're in crunch time! The release is late, and now I'm supposed to take what little time we have left to teach people new things? I have enough on my plate as it is!" And you'd be right. It's not always a good idea to start challenging people. Sometimes, challenges don't make sense.

Challenging people is *one* style of leadership. Let's talk about two more:

- *Command-and-control* leadership
- *Facilitating* leadership

Why do we need to talk about the other styles? Because *challenging* to encourage growth isn't always a good idea. I know it sounds crazy and is contrary to the advice of agilists, but hear me out. Sometimes, a command-and-control style of leadership is required, especially in survival mode, as we'll discuss in the next few chapters.

Command and control is sometimes a good idea because there are times when a team leader must be able to direct their team down a path where the team has no time to learn the skills needed to deal with the current circumstances (such as when fighting many fires).

The third leadership style, *facilitation*, is described by many agile consultants this way: "Lock the team in a room, give them a goal, and get the hell out of their way." Agile methodologies sometimes call this a "self-organizing" team.

Facilitation is a good idea sometimes, if the team already knows how to do the work and solve their own problems. A command-and-control leader would get in the way of getting the job done.

2.4 Which leadership style should you choose?

It seems like the previously discussed approaches—challenge, command-and-control, and facilitation—are good styles at different points in time. Team leaders have succeeded by doing each, but many have failed with each as well. When *does* it make sense to use each of these different leadership ideas? When are the times that, as a leader, you need to take charge and start making hard decisions? When will using command-and-control leadership hurt more than it helps? When should you lock your team in a room and get out of their way because they know what they're doing?

I'll recap the three different leadership types that I've seen in the wild:

- Challenging/coaching leader
- Command-and-control leader
- Facilitating leader (self-organizing teams)

It's easier for me to start with an answer to an opposing question: "When should I *not* use each leadership style?"

Let's examine each one in turn and see when each can result in negative consequences.

2.4.1 Command and control

We have all seen or been this type of leader at some point. You tell people what to do. You are the "decider." You take one for the team, but you also have the team in your pocket in terms of hierarchy, decision-making, and control over everyone's actions.

The command-and-control leader might also try to solve everyone's problems. I once had a team leader who, my first day on the team, set up my laptop while typing blazingly fast on the keyboard and not sharing with me anything he was doing. When I asked questions, he muttered something along the lines of "Don't concern yourself with this now. You have more important things to do." (Read that sentence with a heavy Russian accent for better effect.)

With a controlling leader, there's little room for people to learn, take sole ownership, or take initiative that might go against the rules. The consequences are too undesirable.

The command-and-control approach won't work if your team already knows what they're doing or if they expect to learn new things and be challenged to become better.

2.4.2 Coach

The coach is also known as "the teacher" and is great at teaching new things to others. The opposite of the controlling leader, the coach is great at teaching others to make decisions while letting them make the wrong decisions as long as there's an important lesson to be learned.

Time is not an issue for a coach, because learning requires time. It's like teaching your kid to put on their shoes and tie their shoelaces—it takes time, but it's an important skill, and you'd be making a mistake not taking the time to let your kid go through this exercise on their own, cheering them from the sidelines.

The coaching approach won't work if you and your team don't have enough free time to practice and do any learning. If you're busy putting out fires all day, and you're already behind schedule anyway, you won't have time to also learn or try new things like refactoring or test-driven development.

2.4.3 Facilitator

The facilitator stays out of everyone's way. Whereas the coach challenges people to stop and learn something, the facilitator makes sure that the current environment, conditions, goals, and constraints are such that they will drive the team to get things done. The facilitator doesn't solve the team's problems but instead relies on the team's existing skills to solve their own problems.

The facilitator approach won't work if the team doesn't have sufficient skills to solve their own problems (such as slow machines, customer demands, and so on).

Now that we've discussed circumstances that are unfavorable for each leadership style, let's talk about when they're most effective.

2.5 Leadership styles and team phases

Each of these leadership types belongs in a different phase of the team's needs. There are times when a team needs a commander, times when it needs a coach, and times when it needs a facilitator. I call them the *three team phases*.

A beta tester for the book commented that the word *phase* gives him a bad vibe "due to bad memories on some poorly managed, waterfall-style projects." I'm still not sure what to call these things myself. *States* might be better suited, but I'm still debating this. If you have a better name for what I call team phases, email me at roy@osherove.com (or through contact.osherove.com) with the subject "Naming Phases."

2.6 The three team phases

These phases are how I decide which leadership type is required for the current team (see figure 2.1). The question "Which leadership type is right?" should be asked on a daily basis because teams can flow in and out of these phases based on many factors.

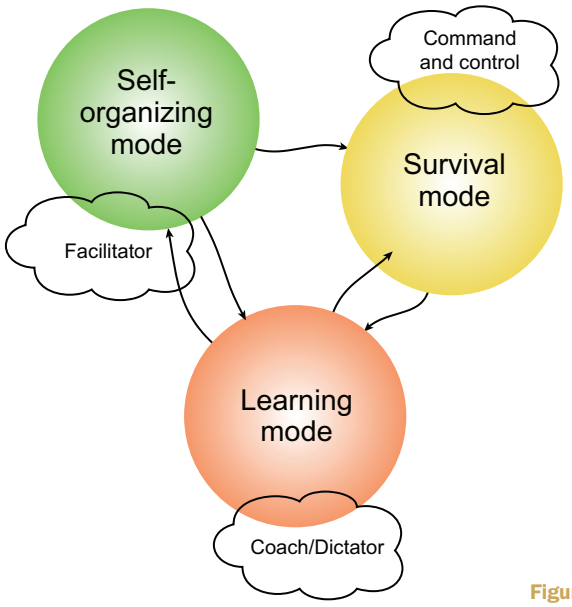


Figure 2.1 The three team phases

2.6.1 Survival phase (no time to learn)

Survival sounds dramatic and is as alarming as it sounds. It doesn't necessarily mean coffee-stained carpets and a sleepless staff. I define survival as *your team not having enough time to learn.*

In order to accomplish your goal as a leader (coaching people to grow), you need to make time to learn, and your main strategy, or instinct during this phase, is to *get the team out of the survival phase by creating slack time.* In order to get slack time, you'll most likely need to use a command-and-control style of leadership.

2.6.2 Learning phase (learning to solve your own problems)

You can tell you're in the learning phase when *your team has enough slack time to learn and experiment and you're using that slack time.*

Slack time can be used for learning new skills, removing technical debt, or, better yet, doing both at the same time:

- Learning and gradually implementing test-driven development, with people who have no experience
- Enhancing or building a continuous integration cycle, with people who have no experience
- Enhancing test coverage, with people who have no experience
- Learning about and refactoring code, with people who have no experience

In short, use slack time to do anything constructive, and tack on the phrase "with people who have no experience" at the end of the sentence.

Your main goal as a leader (in order to achieve your overall role of growing people) is to *grow the team to be self-organizing by teaching and challenging them to solve their own problems.*

In order to achieve that, you need to become more of a coaching leader, with the occasional intervention of the controlling leader for those cases when you don't have enough slack time to learn from a specific mistake.

2.6.3 Self-organizing phase (facilitate, experiment)

You can tell you're in the self-organizing phase if *you can leave work for a few days without being afraid to turn off your cell phone and laptop.* If you can do that, come back, and find that things are going well, your team is in the unique position of solving their own problems without your help.

Your goal in the self-organizing phase is to keep things as they are by being a facilitator, and keep a close eye on the team's ability to handle the current reality. When the team's dynamics change, you can determine which leadership style you need to use next.

The self-organizing phase is also a lot of fun because this is the phase where you have the most time to experiment and try different approaches, constraints, and team goals that will develop your team.

This is the point where you have time to do the things that matter most. As a leader, you have a vision. If you're always keeping your head down, you can't look up and see if your team is going in the right direction.

From my personal experience, most of the teams I've seen are far from self-organizing. My belief (though I have little more than gut feeling and anecdotal experience) is that maybe 5% of software teams *in the world* are truly self-organizing and are capable of solving their own problems. Some 80% of the software teams out there are probably in survival mode. How often have you been part of a team that kept putting out fires and never had time to do "the right thing"?

How do you switch to a different phase?

2.7 When does a team move between phases?

It's important that you recognize when your team needs a new type of leadership, and you'll have to keep a close eye on the team's main assets; see figure 2.2. Any event that can shift the balance of the following team assets can cause the team to have different needs from you as a leader:

- Asset #1: The team's knowledge and skill to solve their own problems
- Asset #2: The team's amount of slack time

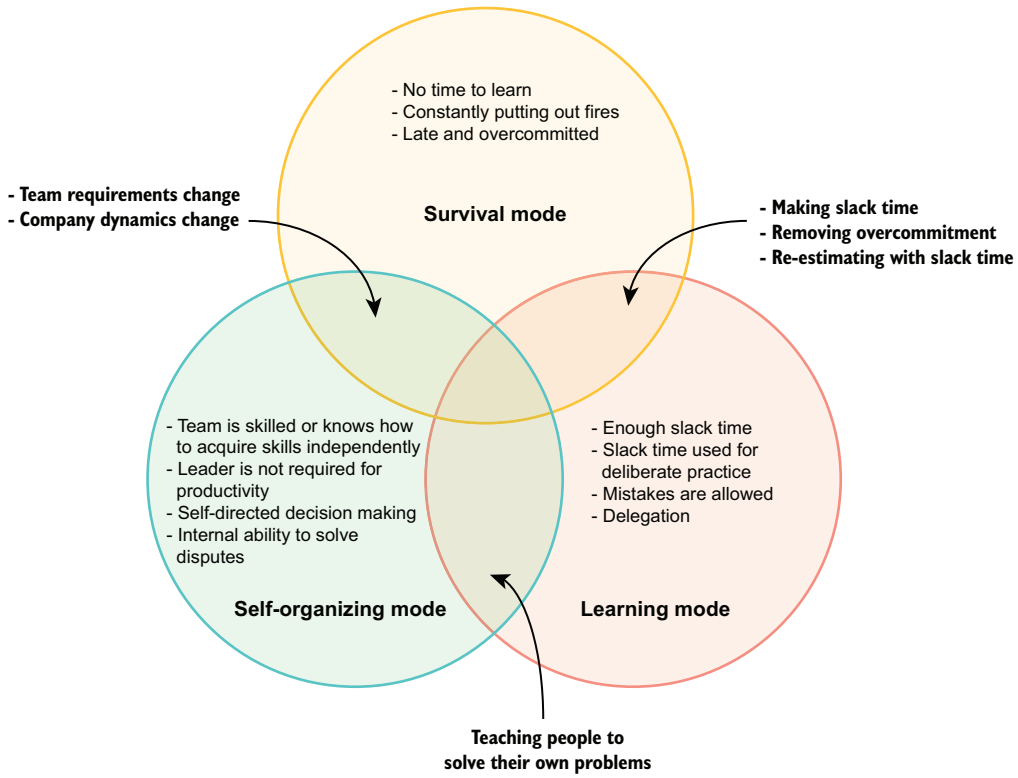


Figure 2.2 Detecting your team's mode

Here are examples of events that could trigger a team phase shift:

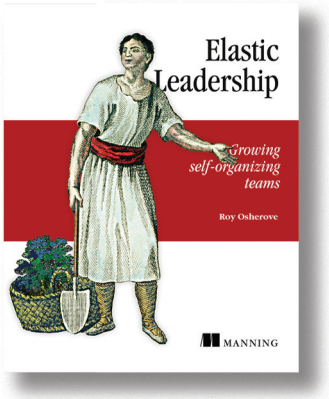
- *You bring into the team new people who lack the skills to solve their own problems.* You might be going into the learning phase. If they still have *time* to learn, then you are indeed in the learning mode. You can use this time to teach those problem-solving skills to the new team members. Better yet, you might take the opportunity to teach some of the more experienced folks on the team how to mentor the new team members to solve their own problems. That way everyone is challenged and growing, not only the new members.
- *You or someone else is changing deadlines on known goals.* This could possibly remove any slack time the team is using to learn. You might be in the survival phase. Time to get out of there fast by removing some commitments and making more slack time available for learning!

2.8 Next up

This chapter was a quick walk-through of recognizing the three team phases and leadership types that make the most sense for each phase. In the next few chapters, I'll go through each of the team phases, diving deeply into the specific leadership styles and techniques that have proved effective for me.

2.9 Summary

- The role of the team leader is to grow the people on their team. This growth is the overall compass through which the leader should navigate important decisions. To grow, the team must first have time to practice new skills and make mistakes; slack time is necessary.
- In survival mode, there's no time to learn, and the leader helps make time for the team. In the learning phase, the leader coaches the team and helps them grow; and in the self-organization phase, the leader acts more as a facilitator, letting the team move on without much interference.
- A team can move between the states rapidly based on the current reality and makeup of the team.



Elastic Leadership

Growing self-organizing teams

by Roy Osherove

ISBN 9781617293085
240 pages
October 2016