Practical Ways to MANAGE YOURSELF

MODERN MANAGEMENT MADE EASY: BOOK 1

Author of *Manage Your Project Portfolio: Increase Your Capacity and Finish More Projects*

JOHANNA ROTHMAN

Practical Ways to Manage Yourself

Modern Management Made Easy, Book 1

Johanna Rothman

ISBN 978-1-943487-12-7

Practical ink

5. Can You Be Effective as a Player-Coach?

Many senior managers believe a first-level manager can work as a player-coach. Too many first-level managers and senior technical people believe that, too.

If we continue the sports metaphor, how many coaches do you see who play as well as the people on the team? While many coaches are physically fit, they're not as fit as their players.

The "players"—here, the knowledge workers—excel at the tactical decisions. The developers in the code excel at design and architecture decisions, both small and large. The testers excel at seeing risks and providing information about the system under test.

The coaches—the managers—can excel at seeing the big picture, the system. The managers see the flow of information and where the bottlenecks are in the team or the project. The managers see the feedback loops and delays. The managers might even see the product strategy in ways the team members don't.

In the software industry, we've been burned by managers who didn't understand how to see the flow of information at all levels: in the team, between teams, between the parts of the organization. They especially didn't understand the relationship between the customers and the organization.

When managers don't understand the flow of information and the various relationships between the people involved, the managers cannot create a strategy for anything. Not for this team, project, program, or product. Certainly not for the organization itself.

Too many organizations have people who think, "A professional manager can manage anything." Nope. Not at all true. The domain matters. Managers who don't understand the domain can't see the system of work and the various risks.

While I am terrific at software and other technology management, I would need help to manage construction. I am a technical person, and I don't understand enough about the domain of construction to be an effective manager.

Effective managers understand the system of what they manage. They don't have to be super technical, but they do have to understand the system.

When we misunderstand the role of management, we believe the manager can be a player-coach and still do significant technical work. That's wishful thinking.

This myth is related to the various other delegation problems: not taking a vacation, feeling indispensable, and this problem of being a player-coach.

Here is a tip for managers:

Once you delegate something, do not take it back.

When we, as managers, see a technical problem, it's tempting to want to wade in and help—either by fixing the problem ourselves or by telling people how to fix it. But that destroys the trust we have built with our teams. Once we delegate the problems to the team, we need to leave the problem delegated.

If you're worried about being a hands-off manager, think about the times you've asked the team to deliver outcomes. I'll talk more about this in Book 2.

If you try to take technical problems back, you are not doing your team any favors. They will eventually stop trying to solve problems, anticipating your lack of delegation. Why should they solve problems if they know you are going to grab the problems back once things get sticky?

5.1 Myth: I Can Still Do Significant Technical Work

“You know, if you want something done, you just have to do it yourself,” Clive muttered as he strode down to his office.

Susan looked up from her desk and sighed. She stood, followed Clive down the hall, and knocked on his office door.

“What? I’m a little busy right now!” he replied and turned back to his computer. Then, he turned to Susan and said, “Explain this part of the framework to me.”

“No.”

He looked at her, raised what Susan thought of as his “Spock” eyebrow. “No?” he asked. “You know I’m your boss.”

Susan grinned. “You’re right,” she said. “You’re the manager, but you’re not asking me as my manager. You’re asking me as someone who’s going to go in and do some damage, as opposed to helping. How long have we known each other?” Clive sighed. “Ten years? Maybe more?”

“Yup,” Susan said. “I had my ten-year anniversary last month. Remember when you said I was the calming influence on you?”

“Yes.”

“Here’s why,” she said. “I’m the technical lead. If you have a problem with what’s going on technically, you’re supposed to talk to me. You’re supposed to talk to the team. Why are you not talking to me? Why are you not talking to the team? Why are you messing with the code?”

Clive snorted. “Did you see what Todd did?”

“Yes.”

“You can stand there calmly and say yes? You’re not freaking out?” he asked.

“No, I’m not freaking out, because the team and I solved this problem this morning, which is more than you have done. Whose problem is this to solve?”

Clive took his fingers off the keyboard. “Uh, yours and the team’s.”

“Thank you. And did you ask me or the team how we were solving the problem?”

“Uh, no.”

“So, you missed that Todd and Cindy are pairing on the fix for this problem, that we already have a patch on the production server, and that Dick and Samara are pairing on performance test development so we can catch this in the future. You missed all of that, right? Oh, and that we will be reviewing the code and the tests, right? You missed that?”

“Uh, yes.”

“You were going to put on your Superman coder cape, and do it all yourself?”

“Uh, yes.”

Susan sighed. Then she grinned. “Look, Clive, you were the Superman developer back in the day, but your day was more than five years ago. Even if it had been a year ago, you wouldn’t know what we’ve done with the code. You cannot possibly be current with what we are doing. Back when you were active in the code, you were part of one five-person team, right?”

Clive nodded. “Yup, just five of us.”

Susan nodded. “That’s what I thought. We now have five teams of six people each. And, we’ve changed all the automated tests. And, we use different frameworks than you had access to all those years ago. You don't know what we're doing and what we're not doing any longer. You cannot do significant technical work anymore without doing real damage. Stop thinking you can."

Clive sighed. "But, what can I do to help when I see a problem?" he asked.

"Ask me what we need. Ask the team what we need. Make sure we are fed and watered," Susan said and grinned again. "Mostly, just ask. We'll tell you. Give us moral support, but don't mess with the code."

Clive started to nod, slowly.

Susan stood up to leave and said, "Oh, I've revoked your check-in privileges. You can read anything you want. You can't check anything in. And, I've changed the root password. You can't mess with anything technical. You're a manager. Be one."

Clive nodded again. "Probably about time," he said. "Susan, I'm going to want details until this is really all fixed."

"No problem, boss!" she said.

5.2 Do You Still Understand the Details?

Now, be honest with yourself. Do you still understand all the details of your team's technical work?

As soon as we become managers, our *technical* problem-solving skills start to erode. At least they *should* if we work on our management skills. Why? Because managers spend more time with people on the team and with other managers and less time with the code, the tests, the requirements, or whatever functional part of the organization you came from.

That doesn't mean we shouldn't know whom to ask about a problem, but we shouldn't know about all the technical details.

Once, when I was a Director of Development, a developer rushed into my office with a highly technical problem that he thought I could solve. I listened to make sure I understood his concerns. I knew I couldn't pinpoint where the problems were in the code.

However, the performance team had changed several aspects of performance the previous week. I could steer him to the correct people because I remained current with discussions. But I couldn't solve the problem.

If you don't understand the details anymore, is it your job to wade into the details? No, it's not. You can facilitate problem-solving. Do not solve the problems yourself.

5.3 Know What You Can Do

The best thing we can do as managers is to create an environment in which people can do their best work. Sometimes that means facilitating a problem-solving meeting. Sometimes it means making sure they have enough servers or debuggers or whiteboards. Sometimes it means keeping other people such as *your* manager out of the way.

Congruent managers will consider the needs of everyone—not just themselves—when they think about their role in the products and services your team provides.

5.4 Consider the Role of a Technical Manager

To paraphrase Peter Drucker, managers exist to organize with purpose. They organize people or help people organize themselves into project teams or workgroups. They organize problem-solving or project-solving teams. They organize ideas. Sometimes, they organize coffee, bagels, or pizza.

Managers see the flow of information, the work, and the people. When managers have perspective on the situation, they can help everyone solve problems.

5.5 Create an Environment Where People Can Solve Problems

Managers rarely solve the team's problems for them. Managers create an environment where people can solve their problems. As a manager, consider these options:

- Create a community of practice so people can learn together informally. That might mean finding funds for a book club. It might mean arranging space for "Lunch-and-Learn" meetings. It might mean asking people to offer their services as coaches and rotating the coach service throughout the organization.
- Arrange specific training for teams or projects or groups.
- Arrange space so people have enough room to work. That work might be code review, test review, or a lab.

In addition, managers remove impediments or challenges the team can't solve on its own. Consider these scenarios:

- If the team uses ancient machines to build and smoke test, you might need to advocate for faster machines that mimic customer configurations.
- If the team doesn't have all the people they need to complete the work, you might need to advocate for more people.

- If the team can’t easily communicate because they don’t have the licenses or tools, such as cameras for a distributed team, you might need to help buy those things.

Your time is much better spent on arranging space for the team to solve its own problems and for removing impediments the team can’t remove. They can handle the code and tests. You handle the management work.

What if you were promoted from within the organization? Consider pairing or mobbing with the team so they learn what you used to do. As a manager, the last thing you need to do is offer technical assistance. Why? You’re dealing with management issues.

5.6 Can You Contribute Technically?

Every manager has a tipping point with respect to how many people the manager can manage and still do technical work. The less seasoned you are as a manager, the faster you reach the tipping point.

If you are new to management, you might be able to lead two other people and still contribute. However, as soon as you add a third person to lead and serve, you may find that you only have about half the working hours available to perform technical work. And the more integrated you are with the technical work, the more likely you are to have some sort of an emergency that requires you to do more of the management.

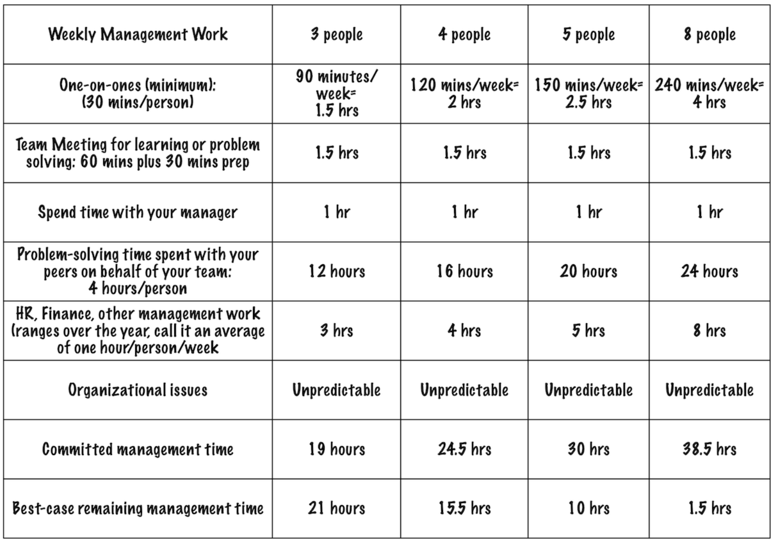
That’s because your management time is not totally predictable.

5.7 Where Does Management Time Go?

Your management time will depend on the kind of team you serve.

Long ago, I charted where my management time went. At the time, I managed a *function*, not a cross-functional, self-managing team.

I was busy all day, every day, running around. Figure 5.1 shows an approximation of where I spent my time:



Weekly Management Work	3 people	4 people	5 people	8 people
One-on-ones (minimum): (30 mins/person)	90 minutes/ week= 1.5 hrs	120 mins/week= 2 hrs	150 mins/week= 2.5 hrs	240 mins/week= 4 hrs
Team Meeting for learning or problem solving: 60 mins plus 30 mins prep	1.5 hrs	1.5 hrs	1.5 hrs	1.5 hrs
Spend time with your manager	1 hr	1 hr	1 hr	1 hr
Problem-solving time spent with your peers on behalf of your team: 4 hours/person	12 hours	16 hours	20 hours	24 hours
HR, Finance, other management work (ranges over the year, call it an average of one hour/person/week	3 hrs	4 hrs	5 hrs	8 hrs
Organizational issues	Unpredictable	Unpredictable	Unpredictable	Unpredictable
Committed management time	19 hours	24.5 hrs	30 hrs	38.5 hrs
Best-case remaining management time	21 hours	15.5 hrs	10 hrs	1.5 hrs

Figure 5.1: Management Time with Manager-Led Team

Because the team members were matrixed into projects, I still had one-on-ones with each person each week. I offered feedback and coaching on their work, their team relationships, and their relationships across the organization.

Notice that I could not easily manage as few as eight people when I had to work with them on their projects. I spent way too much time working with them or for them on their projects.

If I wanted to also contribute technically, my management limit was three people. I could contribute a little bit most days to one project when I only managed three people.

Once I started managing four people, however, the balance of my work swung from technical work to management work. You might be able to manage four people and still do considerable technical work. I had too many emergencies.

I was able to offer feedback. I could coach and teach. I was able to create an environment where everyone could solve problems. But I could no longer perform the technical work myself.

If you use an agile approach, you might need to spend a lot less time personally coaching, offering feedback, and serving the individual people. Successful agile teams integrate several key pieces of the work managers used to perform:

- They can both offer and receive feedback about the work and the team's environment.
- They can both offer and receive coaching about their technical and team concerns.
- And because when people coach and offer feedback to each other, they can create a psychologically safe team environment. That safety can help the team take responsibility for their technical excellence.

The manager is no longer the *sole* practitioner of feedback, coaching, and checking on the team. The team checks itself.

If your team is self-managing, you serve the entire team. Your time might look more like this: