Extracted from:

Help Your Boss Help You

Convert Conflict Into Opportunities

The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Pragmatic exPress

Help Your Boss Help You

Convert Conflict Into Opportunities

Ken Kousen

Foreword by Glenn Vanderburg

Edited by Michael Swaine

Help Your Boss Help You

Convert Conflict Into Opportunities

Ken Kousen

The Pragmatic Bookshelf

Raleigh, North Carolina

The team that produced this book includes:

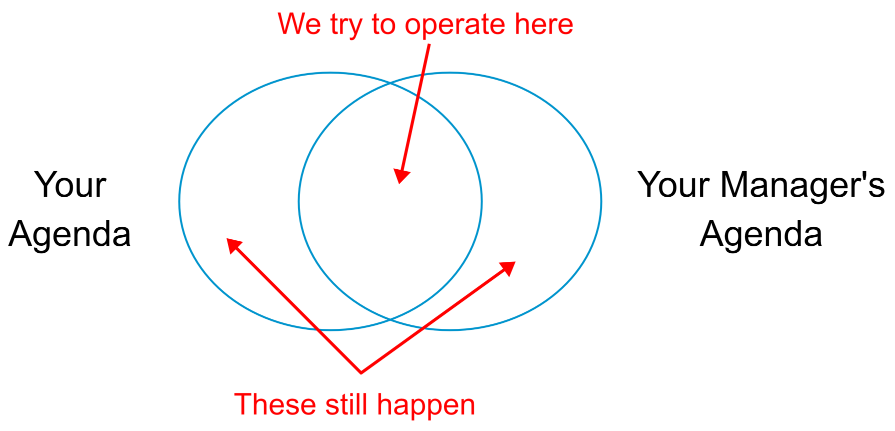
CEO: Dave Rankin
COO: Janet Furlow
Managing Editor: Tammy Coron
Development Editor: Michael Swaine
Copy Editor: Katharine Dvorak
Layout: Gilson Graphics
Founders: Andy Hunt and Dave Thomas

ISBN-13: 978-1-68050-822-2
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—July 2021

For Cheri, Mem, and Josh

Differing Agendas

Why are your goals different from those of your manager? The reason is that your job differs from your manager's job in fundamental ways, which means the rewards and incentives for each of you are different as well. They overlap, though; otherwise, why work together in the first place? It's just that the overlap isn't complete. Consider the following figure, which is a Venn diagram showing your agenda and your manager's agenda.



You have an agenda for your job, your career, and your life. It involves doing your job well, learning about new developments in your chosen field, and dealing with any work-life balance issues that evolve as you grow older. Your boss also has an agenda for their job, their career, and their life. While we mostly operate in the overlap, there will be times when what you want won't match what your manager wants, and vice versa.

It's not hard to imagine a situation that fits your agenda and not your manager's. To choose an example from software development, say your company has implemented its back-end systems in Java, but you attend a presentation on one of the alternate languages that run on the same platform, like Groovy, Clojure, or Kotlin. You're fascinated, and decide to read one of the top books on the subject (no doubt from The Pragmatic Bookshelf). Not only do you like the elegance and sophistication of the newer language, but also you know that learning it will help you become a better developer, and, as a side benefit, may open up new career opportunities for you in the future. You, therefore, want to try out the language on one of your current projects. Because you're deploying to the existing infrastructure, everything should work in the end. Eventually.

The problem is, you know convincing your manager to let you go in a completely new direction is a tough sell. You are proposing a change from what you already know into a new technology with which you have little or no experience for benefits that may never materialize—and without the expertise necessary to fix any problems that may come up. At best, your manager is going to be reluctant to go along with that.

Therefore, you decide on your own, without talking to your manager, to carve off a small piece of your current project and build it using the new language. You'll do it after hours, except that the tasks will inevitably creep into work time. After all, you're still making progress on a work project, right? Unfortunately, there's also the mental energy you are going to expend on the new approach that you might have applied to existing tasks, but hopefully the benefits will be worth it. If everything works, you'll later present the new implementation to the team as a *fait accompli,* minimizing the amount of effort that went into it. As a side benefit, you will also improve your resume, which may turn out to be necessary if your manager finds out what you've been doing and isn't happy about it.

On the other hand, if the new approach fails, nobody ever has to know about it, right?

What to Think About: Side Projects

Have you ever created a separate project at work just to learn a new tool or technology? If so, you're not alone. Did it work? Did you later share what you learned with others on your team, including your manager? More important, did your manager find out about it before you were ready to share? How did you handle that?

Consider the best possible outcome from that experiment. You implement a new subsystem in say, Kotlin, and everything works. You understand the underlying problems much better than before, and the new system is far more robust and powerful. When you demonstrate the new system to your group, everybody is impressed and agrees that your code is valuable.

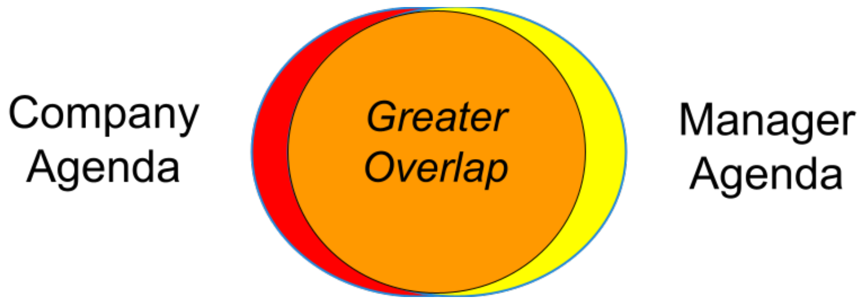
As a side result, you are now the company expert on Kotlin, which has both advantages and disadvantages. It may be good for your career, but from your manager's point of view, you've introduced a single point of failure into the system. Now you are the only person who can maintain that system. You say you're willing to help others in your group learn it, but inwardly you know how much time and effort it took for you to really understand the new language. When you talk to the rest of your group, you may minimize the time and effort coming up to speed required, but nothing worth doing is ever easy. Major transitions only appear straightforward after you've already made them.

The boss also may be glad that you—and, by extension, the group—have a new skill that they can potentially market to clients, but now you're the only resource who can handle that work. At some point in the future you may decide to leave, transfer to a new group, or simply get bored and want to do something else. What will the group do then?

As a natural reaction, the boss may decide to bring in a junior employee to work with you on a project. That way you can train this person on what you've done and how to go forward. That will give the team another resource and still get the job done. Unfortunately, from your perspective, that can be viewed as a problem. It may feel like the boss is trying to create a junior version of you. That person is likely to be less expensive and, perhaps more important, more willing to go along with what the manager wants. Apparently you have a problem doing that, as you just demonstrated that if you believe a new direction is worth following, you'll do so without telling anybody first.

In effect, from your perspective, the boss is asking you to train your own less expensive, more cooperative, replacement.

You can try to object, but in this case—as in the vast majority of cases—what your boss wants is likely to be much more aligned with what the company wants than what you want.



That's a common (though not inevitable) occurrence; your manager's priorities are usually going to be better aligned with those of the company. The higher level the manager, the more that will be true. In any conflict between you and the manager, you are automatically placed at a disadvantage. As far as the company is concerned, if there is a conflict between you and your manager, the company would prefer that you do what your manager tells you to do.

Even worse, what if your manager wasn't happy about your change in direction? The situation just described comes from what those in IT call the "happy path." Consider the alternative: what if the manager is angry at you? What if your boss tells you never to make a major change like that again without prior approval? What if they insist you redo the work on the system, but with the previous approach?

Now you are in a difficult situation. In any conflict between you and your boss, most employees believe they have only two possible courses of action:

1. You can go along with what the manager wants.
2. You can leave.

The problem is, neither of those approaches get you what you want. If a manager knows you'll just go along with whatever they ask, there's no incentive for them to modify their behavior in the future to accommodate your desires. Of course, you can't *not* do what they say, either. Passive aggression has its uses, but direct challenges to the boss's authority are also unlikely to get you closer to your goal, and may even get you fired.

Managing the Relationship

The difficulty comes from viewing the employee/manager relationship as a problem to be solved, rather than a long-term situation that must be, for want of a better word, *managed*. Working professionals are problem-solvers and tend to see the world in those terms. We want to fix problems and thereby make them go away. Your relationship with your boss is not like that. By viewing the relationship as a situation to be managed rather than to be solved, you can try to create a balance between you that evolves over time and ultimately gets both sides what they want.

The term that will be used in this book for such a relationship is *constructive loyalty*. Once you have a solid relationship with your manager based on constructive loyalty, you approach any potential conflict with your manager differently, with clear, honest communication that ultimately gets both sides what they need.

The following list summarizes some of the techniques that build a relationship like that, many of which will be discussed in more detail in future chapters:

- You and your manager *meet regularly*, so you can talk about what you'd like to do in the future (such as learn a new language) during one of your periodic meetings.

- You understand better *how to communicate* with your manager, so you can express your interests and concerns in a way your manager is most likely to hear and understand.

- You provide the boss with arguments they can use to justify the new approach to their bosses, and will *have their back* if problems arise.
- Your boss knows you will find a way to build and maintain the new system, regardless of approach. They know *you got this,* meaning you will take responsibility for the success of the project.
- You and your boss build up a history of successfully dealing with conflicts. Over time, you established that if the boss says no, you will *push back*, but in a way that doesn't threaten the constructive loyalty relationship.

Any disagreement, therefore, is not a crisis. It's an opportunity to express what you want and to encourage the manager to help find a way for you to get it—if not this time, maybe the next time, or the time after that. If you want to use a new language on the current system, you talk to your boss about it. Maybe you decide to just implement tests that way in order to stay off the critical path. Maybe you choose a less vital part of the system for your experiments. To satisfy the single-point-of-failure problem, maybe you arrange a book club or a "lunch and learn" group to share what you've learned. Perhaps your boss can arrange to bring in an outside resource to help, like a consultant, or provide access to an online library or arrange for a training class.

Instead of each party trying to help themselves exclusively, hoping for the best rather than discussing it, conflicts are recognized as a fundamental part of the relationship and both sides plan accordingly. You're playing the long game. You may not get exactly what you want in this particular conflict, but you are building a relationship that you can rely on in the future.

To accomplish this, it's important to understand what each of you want from your careers.