The Pragmatic Programmers

SAMPLE

Fixing Your Scrum

Practical Solutions to Common Scrum Problems

Ryan Ripley
Todd Miller
edited by Dawn Schanafelt

Praise for *Fixing Your Scrum*

"This book is a friendly and conversational partner for today's Scrum Master. It provides real-world solutions to real-world problems that many teams face in pursuit of being truly self-organized and cross-functional. Ryan and Todd have identified some of the most common anti-patterns that result in bad experiences and less-than-optimal outcomes with Scrum, and encourage teams to face them with practical tools and humor."
— **Melissa Boggs**, Chief ScrumMaster Scrum Alliance

"Ryan's and Todd's book is filled with practical solutions you can use to fix your Scrum. Every day, Scrum Masters come to me asking for practical ideas they can quickly put into practice, and that's exactly the kind of advice that Ryan and Todd have put together for this actionable, takeaway-oriented book. It's a must read for all Scrum Masters who want to continuously improve their practice. Read it and then put it into practice!"
— **Vasco Duarte**, Host, Scrum Master Toolbox Podcast

"This book is an essential read for any Scrum Team who wants to deliver value frequently and enjoy doing it. It illuminates the most common Scrum anti-patterns, their underlying causes, and practical ways to make them transparent and overcome them. You will appreciate both the directness and the humility in the authors' positive approach to fix your Scrum."
— **Stephanie Ockerman**, Professional Scrum Trainer and Author, Agile Socks LLC

"Scrum: So few instructions, yet so many misunderstandings. Ryan and Todd offer you a multitude of ways to get your Scrum back on track. Not for the sake of Scrum or 'theory,' but for the sake of addressing real-world problems. Use their combined experiences and expertise to remove long-lived obfuscation and fix your Scrum."
— **Gunther Verheyen**, Independent Scrum Caretaker

"Ken Schwaber, the co-creator of Scrum, often says that Scrum takes a moment to understand and a lifetime to master. Scrum isn't hard to understand, but applying it is difficult because the environments in which Scrum plays are complex. People, organizations, business problems, suppliers, predefined processes, and behavioral norms all seem to work against the simple framework. In this book, Todd and Ryan provide practical remedies to these problems. They focus on ways that you can make your use of Scrum better, not because Scrum is important, but because - when applied well - Scrum can help you change the world. Scrum on!"
— **Dave West**, CEO Scrum.org

"As an agile coach, I encounter Scrum done poorly all the time. It's not doing or not doing Scrum that bothers me most. It's that, by doing it poorly, the team (or organization) isn't achieving the great promises associated with doing it well. In other words, they're missing the results and the impact. Ryan and Todd's book has become my go-to initial action for those organizations - handing them the book and telling them to read and apply the simple, yet powerful advice within. It does a masterful job of helping teams to fix their Scrum. Now I only hope it doesn't impact my coaching practice too much."
— **Bob Galen**, Author and Agile Coach at Vaco

Fixing Your Scrum

Practical Solutions to Common Scrum Problems

Ryan Ripley
Todd Miller

The Pragmatic Bookshelf
Raleigh, North Carolina

Contents

CHAPTER 2

Why Scrum Goes Bad

Scrum itself doesn't go bad—it's the ways that organizations implement it that can be problematic. We frequently see people changing the Scrum framework to fit their organization rather than the organization itself changing. Changing an organization can be slow and frustrating but the whole point of adopting Scrum is to switch to a more efficient, empowering way of creating products, so change is a must.

We've experienced, and even promoted at times, many of the Scrum anti-patterns that we'll describe in this book. The root cause of these anti-patterns is typically policies that have been in place since long before the company adopted Scrum. Such longstanding policies can sometimes make what Scrum brings to light counterintuitive. For example, the idea of having a dedicated product owner who is fully empowered to make all decisions about their product can feel really foreign to many organizations, but having someone in that role is crucial to the success of any Scrum team.

As Scrum masters, part of our service is to the organization, and that means one aspect of our role is making sure the organization is fully embracing the Scrum way of doing things—which may require changes. Organizational change can come in many forms, such as:

- Working with HR to redefine job roles.
- Working with Finance to understand how budgeting processes impact the way teams work.
- Helping management adopt agile leadership principles.
- Removing the divide between IT and business partners.

In order to know which change(s) your organization needs to make, you first need to understand the underlying causes of bad Scrum.

Turning Scrum into Best Practices

Ryan once worked with an organization whose Project Management Office created a 500-page Scrum manual that detailed the processes and "best" practices that *all* their Scrum teams had to follow. Not surprisingly, these "Scrum" best practices looked a lot like the old waterfall processes the company used prior to adopting Scrum. We aren't against teams creating their own practices within the Scrum framework—that's actually a good sign of a team that's working well together. But we've seen many organizations take the idea of "best" practices too far (like with the 500-page manual). A best practice is a practice you adopt and use *everywhere, all the time* as the best way to do your work. But the type of work that Scrum teams do is typically too complex for any particular practice to *always* be the best approach, so there's no such thing as a best practice in Scrum.

At the other end of the "Scrum as a best practice" spectrum is the tendency to modify the Scrum framework itself. This can be very tempting, but don't do it. When a team only uses part of the framework, they lose most of the benefits of working with Scrum. What do these framework changes look like? Here are some examples:

- Treating every Scrum event as optional. We have enough meetings as it is!
- Skipping the sprint review when there isn't any work to show. I mean, it's just a demo, isn't it?
- Holding the daily scrum bi-weekly. Just because it's called the *daily* scrum doesn't mean we have to do it every day, right?
- Cancelling the sprint retrospective in favor of getting more work done. Because who has time to improve?
- Ignoring the Scrum event time-boxes. I mean, who doesn't love a 45-minute daily scrum? (Spoiler: no one loves it.)

Scrum is a collaborative framework that teams work within to help them frequently deliver working software. Your team needs to deliver an increment every sprint so that they can determine whether they've met stakeholder expectations, whether customers actually want the product that they've delivered, whether they've created value at a reasonable cost and timeframe, and whether they're using the appropriate technologies. This feedback loop (between the Scrum team, stakeholders, and customers) that the Scrum framework implements is an example of risk reduction.

Along the way, you and your teams will discover that it's really hard to deliver product increments. Changes will be required at all levels of your organization,

from the Scrum team all the way up and down the org chart. Changing the Scrum framework, ignoring the rules of Scrum, and simply swapping out old jargon for new-and-improved terms won't get you there.

When an organization is reluctant to fully adopt Scrum without making lots of customizations, we've found it useful to make a clear distinction between the Scrum framework and other complementary practices that the Scrum team is using. Often in our consulting engagements, we spend most of our time tearing out complementary practices in order to simplify things and firmly establish the core Scrum elements. Scrum teams should strive to get the basic elements of the framework solidly in place *before* they add any additional practices that may be needed.

Where does your team stand in terms of having the basic Scrum framework elements in place? To find out, try this simple exercise alone or with your Scrum team:

1. Write each element of the Scrum framework on a separte sticky note: Product Owner, Scrum Master, Development Team, Sprint, Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective, Product Backlog, Sprint Backlog, and Increment.
2. Next, grab a different color sticky note and write down each complementary practice your team uses (one per sticky note), such as story points, poker planning, or sprint burndown charts.
3. Now that you have a list of the Scrum elements and a list of the complementary practices, on every sticky note, write a score from 1 (we aren't doing this at all) to 5 (we've mastered this).
4. Finally, reflect on how you can get every Scrum element sticky to a 5. Are you spending too much time and effort on complementary practices? Are any complementary practices prohibiting a Scrum element from getting to a 5?

Best practices work well for work that is repeatable or standardizable, but Scrum is designed for solving complex problems. All too often, we hold on to best practices as a way to feel like we know exactly what's going to happen. Wouldn't it be great if we could follow a 100-step process (filled with best practices) and always get the results that we expect? But complex work doesn't play out that way. Instead, we need to be transparent both within the Scrum team and to the wider organization, have frequent opportunities to inspect and evaluate our work, and have the ability to make adaptations as needed. Scrum is a simple framework with eleven elements that give you a means to use empiricism (transparency, inspection, and adaptation) to your advantage.

Adding complementary practices to Scrum may enhance the framework, but you must be doing Scrum well *before* you adopt any additional practices.

Lacking Goals

Want to know the best way to demotivate a development team? Keep them from seeing how their work impacts your customers. Likewise, it's demotivating for a Scrum team when they don't know how their product relates to the overall mission of the company. This misalignment of value and purpose can make it difficult for a Scrum team to have their own clear goals that reflect larger organizational goals. This cascades all the way down to sprint goals. And without sprint goals, the development team lacks urgency and inspiration. In that situation, the developers become backlog lumberjacks: they chop through features and stories without really understanding why they're doing that work.

What does this look like in practice?:

- Carrying work over across multiple sprints becomes the norm. We'll finish that PBI in the *next* sprint, right?
- Your team actually starts using a sprint goal (hurray!) but it's basically just "finish the sprint backlog" (boo!).
- Quality suffers. We don't know why we're doing the work or who it helps, so who cares how good it is?
- Stakeholders get upset. A lack of a product vision is a vaccuum of product leadership that ultimately gets filled by the development team. And there's no way for developers to magically know what customers want, especially if they don't know how their work is impacting customers.

Scrum teams work best when they are aligned with the organization's goals and customers' needs. Leadership sets the high-level goals for a company, product visions serve these goals and the customers' needs, and sprint goals keep Scrum teams aligned with the customers. When this alignment is in place, the Scrum team has purpose. Purpose is a powerful tool that can bring a team together and keep them inspired and motivated to deliver great products.

Does your Scrum team lack goals? This happens when executives fail to advertise company goals clearly, or fail to communicate changes to company goals. Goals, from the organizational level cascading all the way down to a Scrum team's sprint goal, should be measurable and customer focused. Work with your product owner to clearly understand *why* your team is doing the work they're doing. Trace it all the way up to the company's goals and mission.

During your next retrospective, see if your team can make the connection between their sprint goal and a corporate goal. Ask the product owner and development team to reflect on the sprint goal from a recently completed sprint, and have them connect this goal to the product vision and ultimately to a corporate goal. Making these connections explicit can help keep the Scrum team focused on the impact of their work.

Taylorism Creeping Back in

We've worked in countless organizations where people hold strong beliefs about product development that aren't compatible with the proper use of the Scrum framework. These beliefs are often held both by people in various levels of management and among the people doing the work. These beliefs are often overlooked and rarely discussed because the trust needed to talk about them is lacking. To make matters more difficult, many of these beliefs are deeply rooted in corporate culture and difficult to work through.

What kinds of beliefs, you ask? Ones that people in management and large organizations have held for a *long* time. In fact, many of these beliefs are over 100 years old. We won't bore you with the details, but in 1911, Frederick Winslow Taylor published a study called The Scientific Principles of Management[1], which resulted in a methodology known as Taylorism. Taylorism is rooted in the idea that we can break tasks in to very small, simple steps that can be analyzed, taught, and repeated. The goal was to separate a workers' brains from their hands. In other words, remove the need to *think* about the work and simply give people small steps to follow to complete a task. With small steps in place, the result is predictability and compliance, not innovation.

Taylorism was designed to solve manufacturing problems that were prevalent in the industrial era (which emphasized repeatable work), and it worked quite well. But when people apply the principles of Taylorism to complex work in the innovation and development space, the results are typically disastrous.

1. https://en.wikipedia.org/wiki/The_Principles_of_Scientific_Management

Taylorism differs *dramatically* from the Scrum way of doing things. Here's a quick summary of the main beliefs in Scrum and Tayloism:

Taylorism	Scrum
Workers only know how to do the specific tasks they've been assigned; they don't have or need a big-picture view of what their organization is trying to accomplish and are not encouraged to broaden their skillsets.	Work is performed by cross-functional teams that have all the skills they need to get the job done. These teams are supported by leadership and high levels of trust are leveraged between leadership and Scrum teams to make decisions quickly and frequently deliver increments of working software to customers. Workers are encouraged to broaden their skillsets and collaborate.
Managers plan work without input from the people who perform the work.	Planning happens at varying levels across all of the Scrum roles. Management is invited in to inspect the team's work during sprint reviews and to collaborate with the product owner so that the team can take management and stakeholder opinions into consideration as they figure out what to do during the next sprint.
Management tries to make the work as predictable as possible by precisely managing resource utilization with exact estimates.	Scrum teams manage their time and focus as they plan their work. The development team is responsible for resource utilization. They use estimates to trigger conversations within the Scrum team and with management—not to make the work as predictable as possible.
Management optimizes workers' performance using metrics and measurements.	Scrum teams use metrics to optimize outcomes that benefit the customer.
Management uses money and rewards as primary motivators for performance. Workers are motivated to achieve rewards and avoid punishment (extrinsic motivation).	Scrum team members have a goal they are trying to achieve, the autonomy to achieve it, and a comprehensive set of skills to accomplish their goal. The also have opportunities to learn from each other. Team members are motivated to perform their work because they enjoy what they do and

Taylorism	Scrum
	they feel a sense of personal accomplishment (intrinsic motivation).

Table 1—Taylorism vs. Scrum

As you can see, Taylorism and Scrum are two *very* different mindsets. They're basically opposite ways of approaching work. So it's no surprise that adopting Scrum in an organization that's used to the Taylorism way of doing things can be an uphill battle. Keep this info in mind when you're trying to bring people in your organization around to the Scrum way of doing things. It'll help you understand where people's opposition to Scrum practices comes from.

When Taylorism and Scrum are at odds in your organization, you'll likely see some of these signs:

- A mechanical implementation of the Scrum framework without truly embracing its principles and values.
- Scrum team members view Scrum as just new way to get micro-managed by management.
- No meaningful signs of collaboration between Scrum team members.
- The Scrum team is producing poor-quality increments.
- People are still being measured by how busy they are, not by the outcomes of their work.
- The Scrum team is unable to deliver increments of product every sprint
- Reverting back to past practices—but calling them by new names

Keep an eye out for these signs that the old ways of working are still at play in your organization. This book describes a lot of anti-patterns that occur when Taylorism and Scrum compete with each other. Adopting Scrum will require conversations about your company's culture that will give you an opportunity to create change. Take advantage of those opportunities so that you can positively impact your organization and move it more towards the Scrum model, which represents the world we work in today far better than Taylorism does.

Trust is Missing

Trust is a weird thing: it's contextual, exists on a spectrum, and is very transactional. Think about your relationships. You trust some people with certain aspects of your life, but not everything. For instance, Todd and Ryan trust each other to work on this book and to co-teach a good class together—but they don't trust each other to do each other's laundry. And trust

changes over time. It can take a long time to build up trust with someone, and then one misstep can wipe it out.

What does trust look like on a Scrum team?

- The product owner trusts the development team to create a done product increment by the end of every sprint.
- The development team trusts the product owner to provide them with a clear product vision.
- Development team member trust one another to do their best and support each another.
- Management trusts the Scrum team and removes any impediments to delivery.

Without trust, you can't have transparency. If the members of a Scrum team don't trust each other or an organization doesn't trust a Scrum team, then it's impossible to make the team's work and progress clear to stakeholders. Instead, people play defense: they blame one another and fail to work as truly collaborative team.

Want to quickly make your organization trust your Scrum team? Deliver. We've found no better way to build trust than consistently delivering increments of products every sprint.

But how do you increase the likelihood that a Scrum team can work together well and deliver successfully? Well, other than solving the many anti-patterns in this book, here are a few quick ideas you can try:

- Shorten your sprint length. Instead of a four-week sprint, try a one-week sprint. The development team will have fewer product backlog items to focus on, the product owner will have stakeholders in the sprint review even more frequently, and collaboration will happen in shorter intervals.
- Focus sprint planning on setting a sprint goal that has a true impact on the customer. This helps inspire the team and gives them something real to work toward.
- Use the daily scrum as an opportunity for the development team to inspect their progress towards their sprint goal. Celebrate progress and promote opportunities for Scrum team members to support and help one another. This helps increase trust within the team.
- Create opportunities for the Scrum team to collaborate with real customers. The sprint review event is perfect for this. Who better to talk about the impact of the team's work than the people who are actually impacted by it?

- Introduce or re-emphasize the importance of the Scrum values. If you and your team keep the Scrum values in mind, then empiricism will shine.

If you see that trust is lacking—either between your Scrum team and the rest of the organization or within the team itself—do everything in your power to find a way to build trust. We offer tips and exercises for doing so throughout the rest of this book. For Scrum to work to its full potential, trust is mandatory.

Coach's Corner

Most big organizational structures are based on Taylorism, and something that has been in place for so long takes time to unwind. Scrum has been around for over 20 years, but a lot of large organizations have been around for far longer than that. Even newer organizations likely have employees who are used to the older, pre-Scrum ways of doing things.

But change *is* possible. As a Scrum master, you can unwind old policies and create new ones every day! A great way to figure out where to start (or continue) creating change in your organization is to perform this excercise, which is inspired by the liberating structure 15% Solutions [2]:

1. Gather all the Scrum masters in your organization.
2. Discuss with them how far your organization has come in terms of transitioning to the Scrum way of doing things and how much work there still is to do. If it's helpful, reference the Taylorism vs. Scrum info we presented in this chapter.
3. Have everyone in the group spend 5 minutes on their own answering the following question: What's within my control that I can change to get our organization 15% closer to where it needs to be?
4. Put people in groups of two to four, and have each person take 3 minutes to share their 15% Solution with their group. Make sure that the people who aren't sharing are actively listening and *not* giving feedback or advice.
5. Finally, have each team member spend 5 minutes sharing their idea(s) again, but this time encourage the other team members ask questions and give feedback to each 15% Solution.

Thinking in terms of 15% Solutions can keep people from feeling overwhelmed. Concentrating on what's within our control and finding a way to get 15% closer to where we want to be can be a powerful exercise.

2. http://www.liberatingstructures.com/7-15-solutions/

In this chapter we covered some of the old practices that Scrum often competes against in organizations. Next we'll delve into how we can break bad Scrum by concentrating on the values and principles that influence behaviors both within a Scrum team and in the wider organization.

About the Authors

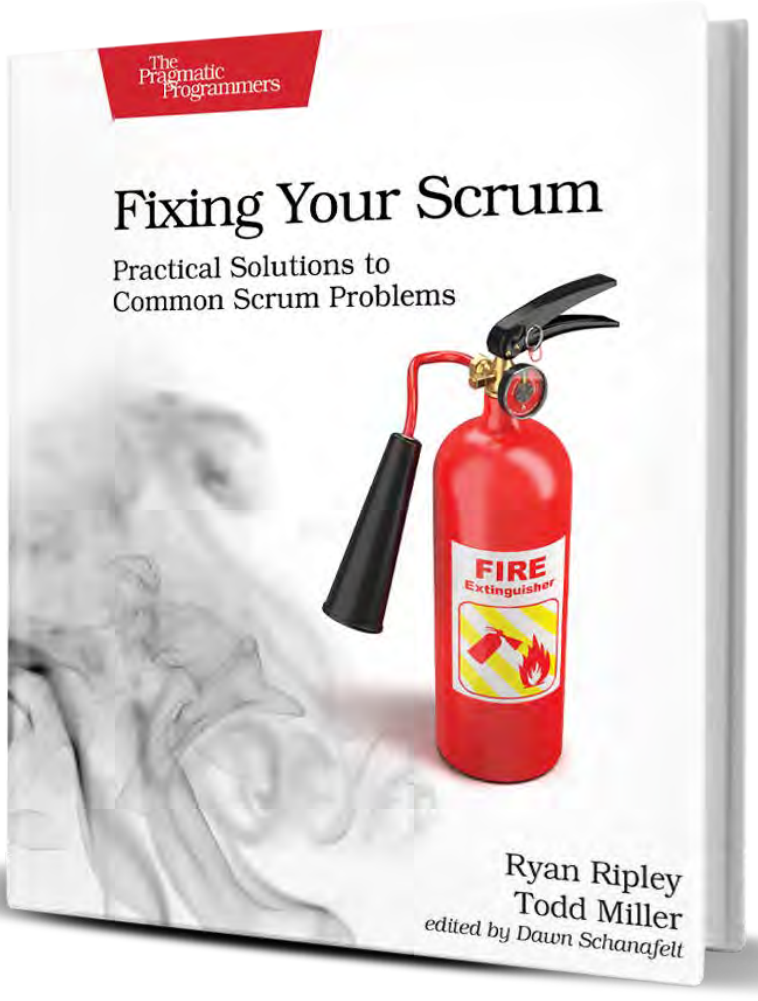
Ryan Ripley

A professional Scrum trainer with Scrum.org, Ryan Ripley has worked as a software developer, manager, director, and Scrum Master at various Fortune 500 companies in the medical device, wholesale, and financial services industries. He is host of "Agile for Humans," the top agile podcast on iTunes. Ryan lives in Indiana with his wife, Kristin, and three children. He blogs at ryanripley.com and is on Twitter @ryanripley.

Todd Miller

Todd Miller has practical experience as a Scrum Master, Product Owner, Software Developer, and Agile coach on a variety of technical and creative projects across a multitude of industries. He has been a professional Scrum trainer with Scrum.org 2016. Todd lives in Pennsylvania with his wife Jessica and two children. His blogs can be found on the Scrum.org website and his twitter handle is @todd_miller11.

Buy Your Copy Today



SHARE

BUY BN Hudson Booksellers Porchlight