# How the Fuck to do Scrum



How the fuck to do Scrum?
ISBN: 9789465388427
Version 1.0

www.agileviking.nl

# How the Fuck to do Scrum

By
Erwin Verweij

*“Scrum is simple. Try it as is and determine if its philosophy, theory, and structure help to achieve goals and create value.”*
**Ken Schwaber & Jeff Sutherland**

# fuck

/fʌk/

**VULGAR SLANG**

*verb*

1. have sex with (someone).
2. damage or ruin (something).

*noun*

1. an act or instance of having sex.

*exclamation*

1. used alone or as a noun or verb in various phrases to express annoyance, contempt, or impatience.

According to the **Oxford English Dictionary**.

Alright, that's dealt with (again). Let's move on.

## Table of contents

*“Your job isn’t to build more software faster, it’s to maximize the outcome and impact you get from what you choose to build.”*
**Jeff Patton**

## Foreword

Yes, this is another book about Scrum. Because clearly, the world did not have enough already. You might think I am trying to squeeze the last bit of juice out of a framework that is long past its expiration date. After all, scroll through LinkedIn for five minutes and you will see the same chorus: *Agile is dead. Scrum does not work. The big companies have moved on.*

Right. And yet those same companies are still running “Agile transformations” that involve three consultants, two steering groups, and at least one new set of buzzwords. So no, Scrum is not dead. Agile is not irrelevant. The framework is fine. The problem is us.

The truth is, using Scrum goes wrong more often than it goes right. Which is fascinating, because it is so simple it could almost fit on a coaster (it actually does). But simplicity does not equal easy. Humans have an incredible talent for taking something straightforward and turning it into a bureaucratic circus. That is why something designed around people so often ends up fighting against them.

This book does not promise to reinvent Scrum, nor to repeat every rule like a schoolteacher with a ruler in hand. What I want is to take you beyond the shiny posters and sticky notes. We are going to talk about what actually works, where people consistently screw it up, and why. Expect plain talk, a bit of grit, and the occasional poke in the ribs.

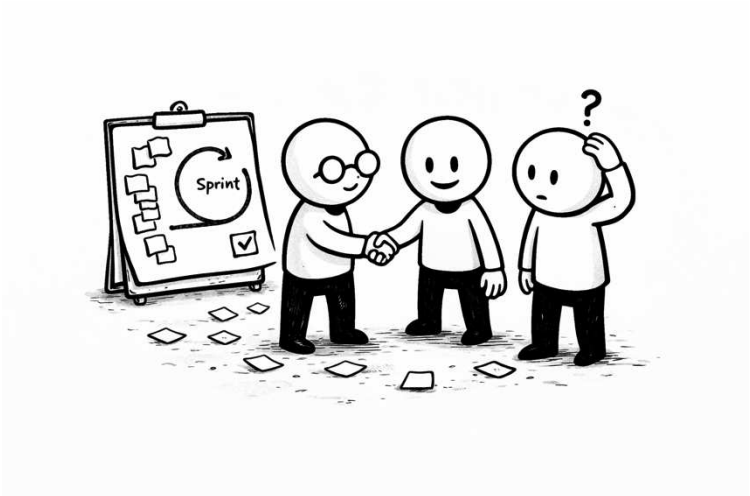
That is why I use the word fuck so much. Not in my everyday work because in know that being nice is often the better choice But because deep inside me there is always this tension. When I see organizations try but not understand. When I talk to collogues, teams, management and owners.

Most of the time they try so hard and do their best. And in most case, I just want to shout “Go and fucking do it the right way”, or “Then just stop fucking doing what does not work.” And simply “What the Fuck!”.

So forgive me if this disturbs you. I am not here to insult anyone and you might not even agree with me. But don’t forget, just because you feel insulted does not mean you are right. And yes I might be wrong in my knowledge and understandings, but even I am still learning.

## Who is this book for?

I wrote this book for people who are just stepping into Scrum, or who have stepped in and are already wondering what the hell they signed up for.




This book helps beginning ScrumMasters, Product Owners, and Scrum Teams understand how to work with the Scrum framework, not by repeating theory, but by grounding it in real-world experience and applied practice.

It's for beginning ScrumMasters who are trying hard to make the framework work inside an organization that nods politely at Agile but quietly keeps doing what it always did. The ones who feel the friction between theory and reality. The ones who sense that something is off but cannot yet fully articulate why.

It's for Product Owners who suddenly carry the title but are still figuring out what that actually means. Who are struggling to make their role valuable instead of becoming backlog administrators with a fancy name. Who want to understand how

to connect vision, stakeholders, and real value without drowning in tickets.

It's for Scrum Teams looking for solid ground. Teams that want more than rituals and buzzwords. Teams that want to understand why they are doing these events, what the purpose is, and how to make them actually useful instead of just another block in the calendar.

It's also for organizations trying to squeeze Scrum into their existing structures, hoping it will magically improve things without changing anything fundamental. If you are serious about understanding what Scrum really asks from you, this book is for you too.

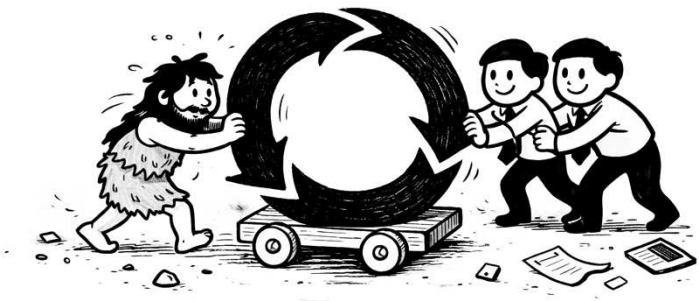
And finally, it's for anyone who is tired of fluffy theory and wants a clear, practical explanation in real-world language. No corporate perfume. No abstract diagrams without context. Just a grounded, honest look at what it takes to work with Scrum in practice.

## What this book is, and what it is not

I wrote this book with the Scrum Guide open on my desk. Because, let's be honest, a shocking number of people using Scrum have never actually read it. Or maybe you skimmed it once, nodded wisely, and then invented your own version. Fair enough. But it never hurts to go back to the source, with a practical lens this time. That is what this book is about: a grounded explanation that links theory to the mess of real life.

What this book is not, is a holy manual that tells you exactly how to "do Scrum properly." That is not the point. You still have to figure it out yourself, in your own context. My hope is that you do it with a bit more perspective, and a bit less bullshit.

# Part 1 What the fuck is it?



A bit of context
(before we dive deeper)

## The Misunderstandings about Scrum, and Agile

Let me start by clearing up a few misunderstandings. The biggest one of them all has a name: Waterfall.

There was a time when building software was treated like building a bridge. You plan everything in detail, pour it into concrete, and only start building once everyone has signed off. Linear. Predictable. Controlled. Welcome to the world of the Waterfall method. Neat, structured, and painfully slow.

What most people do not realize is that this model was described in 1970 by Winston W. Royce. And he did not present it as the solution. He presented it as a warning. In his own words: “The implementation described above is risky and invites failure.” He argued for iteration. For feedback. For collaboration. The very things that later became central to what we now call Agile and Scrum. He never even used the term “waterfall” himself.

But his paper included a tidy little diagram. Clean steps flowing downward. It looked so wonderfully organized that we copied the picture and ignored the text. His caution disappeared. The drawing remained. Applause.
On paper, the logic behind Waterfall is seductive. If you think hard enough upfront, define every requirement, design everything properly, build exactly what was specified, test it thoroughly, and then release it, what could possibly go wrong? If everything goes according to plan, it works beautifully.

**The problem is that it never goes according to plan.**
To be fair, in some domains this approach makes sense. If you are building a concrete bridge or installing a gas pipeline, or even a simple software solution you do not want to change your mind halfway through. Stability, safety, and predictability matter deeply there. But once you deal with evolving customer needs,

fast-moving technology, and people who do not yet fully understand what they want, the model starts to creak. It becomes rigid. Slow. Resistant to learning. What once looked structured turns into a swamp. Change is suffocated. Insights that arise along the way are ignored. Teams lose energy because adaptation is treated as failure instead of progress.

**And yet, it is still everywhere.**
Complicated projects still begin with thick requirement documents. Everything is locked down upfront. Nothing left to chance. And months later, a product is delivered that does exactly what was written down at the start. The only issue? Nobody wants it anymore. The user has moved on. The market has shifted. The original problem has evolved. And then comes the most painful sentence in the entire project: “But we followed the plan.”

**Yes. And the plan was outdated before the ink dried.**
It is not hard to understand why this approach refuses to die. Many organizations are built around the illusion of control. Paper feels safe. Budgets are approved based on predictable timelines. Contracts are written as if we are building factories, not software or services. Waterfall offers a comforting sense of stability in an uncertain world.

**But it is false stability.**
Value is not created by sticking to a plan. Value is created by the ability to adjust when reality changes. And that is precisely what Waterfall struggles with. It was never designed for continuous learning in complex environments.

That is where the misunderstanding begins. Agile and Scrum were not created to rebel against structure. They were created to handle uncertainty honestly. Not to throw away planning, but to accept that plans need to evolve.
And that makes all the difference.

## The New New Product Development Game

Long before the Manifesto for Agile Software Development was written.
Long before anyone stuck post-its to a wall and called it transformation.
In 1986, an article was published that quietly flipped the table.

It appeared in the Harvard Business Review and carried the ambitious title *The New New Product Development Game*. A mouthful. But what it described was revolutionary.

Two Japanese thinkers, Hirotaka Takeuchi and Ikujiro Nonaka, had studied successful product teams at companies like Honda, Canon, and Fuji Xerox. What they saw did not resemble the dominant way of working at the time. There was no linear waterfall. No neatly defined phases. No handovers from one department to the next like a relay race where nobody looks back.

Instead, they saw something closer to a rugby match.
A group of people moving down the field together. As one unit. With one shared goal. Not in strict sequence, but simultaneously. Not passing documents across organizational borders, but passing the ball. Adjusting in real time. Reacting to what was happening on the field.

That metaphor would later travel the world.

The word itself did not come from software. It came from sport. From that tight formation in rugby where players lock arms and push forward together. It symbolized collective effort. Shared direction. Momentum built through collaboration, not hierarchy.

What Takeuchi and Nonaka observed was simple and radical at the same time: the most successful product development did not thrive on control. It thrived on autonomy. Not on rigid stage gates, but on overlapping work. Design and development happening in parallel. Thinking and doing intertwined. Feedback embedded in the process, not postponed until the end.

They argued this in a time when command-and-control management was still the default. When predictability was worshipped. When detailed plans were considered proof of competence. And yet they said, out loud, that this approach no longer fit the complexity of modern product development.
They recognized something fundamental: product development is complex. It is dynamic. It is deeply human. And complexity does not respond well to rigid sequencing. It responds to learning. To interaction. To adaptation.

Speed, they showed, does not come from planning harder. It comes from shortening the distance between thinking and doing. Quality does not emerge from compliance alone. It emerges when capable people are trusted to solve problems together.

What is tragic is that many people working with Scrum today have no idea where it came from. They know the Daily Scrum. The Sprint. The Backlog. They can recite the roles and events. But they have never read that article. They do not realize that Scrum did not begin as a framework with rules. It began as an observation. As insight. As critique of a broken system.
It began as a question.

What if we let teams move together instead of handing work off?
What if we removed layers instead of adding them?
What if we allowed people to learn instead of forcing them to pretend the future is predictable?

The creators of Scrum later shaped this insight into a lightweight framework. But the spirit was already there in 1986. Freedom. Responsibility. Focus. Collaboration without bureaucratic friction.

Somewhere along the way, we turned Scrum into a recipe. A checklist. A set of ceremonies to perform correctly. And in doing so, we risked losing the essence that made it powerful in the first place.

If you truly want to understand Scrum, go back to that rugby field. Picture the team. Not waiting for instructions from the sidelines. Not stuck in departmental silos. But moving together. Adjusting. Learning. Pushing toward a shared goal.

That is where it started.
And that is the part we should never forget.