
Save Our Scrum
• • • — — — • • •
Tools, Tips, and Techniques
For Teams in Trouble
Matt Heusser
Markus Gärtner

Save Our Scrum

Matt Heusser and Markus Gärtner

This book is for sale at http://leanpub.com/saveourscrum

This version was published on 2018-04-10

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Preface

> Scrum is ubiquitous in Software Development [1] - Jeff Sutherland, co-creator of Scrum

> I estimate that 75% of those organizations using Scrum will not succeed in getting the benefits they hope from it.[2] - Ken Schwaber, the other co-creator of Scrum

If both statements above are true, it would mean that everyone does Scrum and almost everyone fails at it.

We have a different interpretation: most software teams are doing something inspired by or related to Scrum, but do not see the improvement they hoped for.

That is a bittersweet truth, the kind of problem that might take an entire book to address. Before we go on, let us look back in time.

A Brief Timeline

1986 - The article "The New New Product Development Game" appears in the Harvard Business Review. The article describes how Japanese companies are doing product development in an all-activities-at-once model designed to move the ball forward. It resembles a Scrum in the game of Rugby.

[1] http://www.randomhouse.com/book/226850/scrum-by-jeff-sutherland
[2] http://mattcallanan.blogspot.com/2010/02/ken-schwaber-on-scrum.html

1986 - Ken Schwaber suggests using the Scrum metaphor at the software company he works at: Individual Inc.

1993 - Jeff Sutherland starts using similar methods at Easel Corporation.

1995 - Schwaber publishes "Scrum Development Process"[3] at the OOPSLA[4] conference.

2001 - The Agile Manifesto combines Scrum, XP, DSDM, Crystal, and other lightweight methods under the general label "Agile".

2001 - Ken Schwaber and Mike Beedle publish the book "Agile Software Development With Scrum".

2004 - Ken Schwaber forms the Scrum Alliance.

By 2004, modern Scrum was well defined with its rituals and roles: Scrum Masters, sprint planning, sprints and so on. Changes after 2004 are mostly tweaks and clarifications. Backlog grooming, for example, is changed into backlog refinement. And since 2004 the method's popularity has exploded, with over *three hundred and fifty thousand* Certified Scrum Masters.

Yet we still have that Sutherland quote to deal with.

Scrum In the Wild

Most of the teams that we have worked with lately are "doing Scrum" or some definition of Scrum. They are doing things inspired by Scrum. Or they use pieces of Scrum. Or … something like Scrum. Sort of. Some examples :

[3] http://navegapolis.net/files/Scrum_Development_Process.pdf

[4] OOPSLA is the "Object-Oriented Programming, Systems, Languages and Applications" conference

- Teams today are likely to do daily standup meetings. Or at least have a meeting once a day that is supposed to be quick.
- They often measure time in "iterations" or "sprints" of one to four weeks in length.
- Teams talk about the work in terms of "cards" or "stories".

These all have a Scrum appearance. They are Scrum-ish or Scrum-like. If we want to be polite, we would say they are "inspired by Scrum". If we want to be judgmental, we could call it Scrum-But, ScrummerFall or Fake-Scrum. It looks like Scrum but it somehow misses out on the power.[5] The Scrum Guide would even say these teams are not doing Scrum. Although implementing only parts of Scrum is possible, the result will not be Scrum.[6]

While that might be true, it is not helpful for teams struggling to get better. And make no mistake: changing the way software is delivered is a struggle. Even if the teams involved are doing Scrum, the model itself will point out the problems but it does not propose solutions. Finding the solution is left to good judgment and experience of the team.

Sadly enough, if there is no clear direction for improvement, teams find themselves taking too many "next steps" over a too long period of time. They are struggling to get from A to B at a point in time where they should already have reached Z. Today, we define the entire process in detailed steps to make training easier. Tomorrow, we will be lean and cut it down to a checklist. The next day the cycle is repeated.

We have witnessed this cycle over and over again. Tailoring and adapting the process - the issues Scrum does not address directly - is the hard thing. To be able to tailor, you need judgment and experience (which you do not have yet) along with a direction for improvement.

[5] Yes, We are aware of the parallel to the writing of the first century religious leader Paul of Tarsus in his second letter to Timothy, in the third chapter and 5th verse.

[6] http://www.scrumguides.org/docs/scrumguide/v1/scrum-guide-us.pdf

This book aims at providing examples for both. If those examples resonate with your experience, they can inspire you to take action. We will provide alternatives to help you decide if they fit your situation and what to do about it.

Getting to real improvement is hard which is why we decided to create some guideposts. That eventually led to the book you hold in your hands.

Above My Pay Grade

When we talk to people about improvement, we tend to see the same replies over and over again. Managers say "that's above my pay grade." Directs say "that sounds good, but we need to convince our peer directors and our VP of software." We went to the VPs, who gave us the same answer. At some level, usually at the CIO's office, someone says, "Look, my role is to provide a vision and interface with the business. I really don't have anything to do with how teams run projects on a day to day basis. You have to talk to the people one level down."

Wait, what? We talked to the people one level down. They said to talk to you!

Actually getting better at software development seems to be nobody's business.

We don't think so. We suspect, that it may fall to you.

After all, who else will do it?

So we decided to offer some advice. To *you*. The individual who wants to improve, and might not have the resources to fly us out for a consulting or training assignment. We turned down billable work and, we have to admit it, in some cases neglected our very families for weeks on end to create this book for you. It is the best

we have to offer - until now - based on our successes (and a few failures) in software development.

Changing things is going to be hard, but we will be right there with you. Let's get started.

How To Use This Book

If you have read this far, we hope you agree that:

(A) The ideas behind Scrum influence the way the work happens.

(B) Many organizations could use some help in tailoring or even implementing the ideas.

If your organization is in the second category, or if you work with one that is, then this book is for you.

Problem of Audience

It may sound like the second category mentioned above is specific, but it is broad enough to cover the following situations:

- You are excited about Scrum and just want to learn.
- Scrum was imposed and you want to comply.
- The person who imposed Scrum on you only has a buzzword-level of understanding and you want to respond.
- You want to use Scrum to get specific and tactical.
- You learned about Scrum on a definition-level and want to learn the theory.
- You are an expert in the Scrum theory and want practical advice.
- You are an expert already, and just need to figure out what small experiment to do next.
- You are great at Scrum on a team level and struggle to integrate it cross-team.
- Because of integration problems, people at team level are not willing to give Scrum a try.

Then here are the people related situations: you know what to do but can't seem to get real change to happen. This can be about the people in charge, or even more powerful, the ones doing the work who do not want to change. Maybe you are leery of a big bang change and want to change incrementally.

If we tried to write a book to be everything to all people, we would have more than one problem. The expert would be bored by some parts while the beginner would be confused by others. Contemporary readers used to blog-post-sized conversations on the internet will lose their attention. Extending scope beyond scrum - to chase value? That's a graduate textbook, at best.

A more focused work would leave some people disappointed. A book with a bit of each would disappoint everyone.

We have a solution - and we think you will like it.

Relax, We Have You Covered

We have structured *Save Our Scrum* in a way for you to quickly find the answers to the problems you want to solve. Think of it more as a magazine with a table of contents than of a book.

You can also use our tables of contents (we have three, pick your favorite) to find something interesting to read, and to expand your reading into circles that grow wider over time. You might possibly read the entire book, or not. If you are an expert, you might skip the introductory matter on what Scrum is entirely. Or, once you understand how we think, you might read our take on Scrum at the end, just to see the subtle nuance between our wording and that of others - especially to evaluate if this is the kind of material that you would like to pass on to a new learner.

Here are four places to start:

- The next-to-last section, "*Common Questions and Problems*",

lists specific issues for teams, along with a set of answers to browse through.
- The *table of contents* might be helpful to get an overview, to find something that strikes you as interesting or fascinating.
- The *index to the book* provides a specific list of keywords and the places where they are discussed in the book.
- If you have the *kindle edition*, you can search for common or popular words and go directly to the related pages.

Or you could keep on reading down here, where we tell you how we laid things out.

Part One - Simply Scrum

The first chapter explains Scrum as a bare-bones framework. If you do not really know what Scrum is or if you have only been given a "sketchy" introduction, you will want to read this chapter.

Do not worry, it will not take up a lot of your time. The essence of Scrum is incredibly simple. We leave the theory and philosophy to chapter two, the why of Scrum, that talks about the problems Scrum comes to solve and how it solves them.

We end the part with chapter three: common problems in Scrum adoption. These are problems we have seen so often that we think everyone learning scrum should read about them.

That is it for the introduction, which is more like twenty pages than two hundred. You might want to read this part even if you are a deep expert, simply so you understand the wording and terminology we use.

If you skip this section and get confused later, you can always come back here. We will leave the light on for you.

Part Two - Scrum Nuggets

Instead of a collection of essays about a topic (e.g. when to test in Scrum), we have a bunch of nuggets, each one taking a couple of paragraphs to a couple of pages. Instead of enforcing a set of predefined chapters, we decided to write something emergent, first creating the nuggets and then finding patterns to organize them.

The result is a set of structured chapters (we have one on when to test) but instead of being a big chunk of running text, chapters are a collection of nuggets. Find the chapter that interests you and start reading. If a nugget is not relevant, skip it. That is OK, you have our permission. You can come back later if you like.

Part Three - Case Studies

The right thing to do sounds easy. Getting a team, a department or an entire company to change direction is a lot harder. We therefor offer you a variety of case studies delivered by ourselves and by several other people who have worked in a company for a longer time. Every case study covers several themes - we suggest you start with the case study that sounds most like your company and the problems you are trying to solve.

They are not wrap-it-up-into-a-bow case studies. They are not written in the (1) "problem", (2) "introduce Scrum", (3) "everything is better" format. They are real stories that go into depth on the trade-offs that real people had to make on real projects. We particularly went for long-term case studies that talk about the consequences of decisions months and years down the road, not about the state the system appeared to be in three weeks later when the consultant left the building.

To collect the use cases, we invited people we trust and who have experience we think is relevant. We might not agree with their

interpretation of events, but we believe you should hear a variety of perspectives and make your own decisions.

Part Four - The Back Matter

It is tempting to skip the back matter of the book, just like you might skip the preface. Maybe this is preaching in front of the choir, we get it. Know that this last part is packed with value, including the list of problems, the index, a list of places to go for more in-depth information on a specific subject, and some last words. This section is not made for you to read every word, but to give you a final set of ideas to help you solve your problems.

One thing we wanted to do with the front matter was to keep the introduction short. I think we did well.

Now on with the show. We think you will like it.

Scrum as a Framework for improvement... and that's it

This chapter will cover one thing: What Scrum is as we see it. Further chapters go out in a spiral, explain more on the benefits, or how it can wrong — but before we can analyze things, we need to understand them. So find a quiet place and give yourself a little time to concentrate.

Oh, and buckle your seatbelt, because it may be a bit of a bumpy ride. We'll start with the what and move to the philosophy, the why, because that's where the power is.

Scrum Roles

When people talk about software teams, there are typically two models. The more popular model, which we'll call the Prussian model, includes only the people building the software, and divides people into specialities by role - Business Analyst, Programmer, Tester, Database Analyst, Operations, and so on. The "egalitarian" (treat all people equally), or "Xerox PARC" model, puts less emphasis on specialty and even pay grade, and tends to refer to everyone as a "member of the technical staff", or MTS.

Scrum doesn't tell you what job descriptions to give people. As far as Scrum is concerned, the technical staff consists of MTSs. If your company wants to have roles, titles, job descriptions and pay grades, Scrum is neither for nor against. Yet Scrum does have roles.

The Scrum concept of the group building software is wider than the delivery team. The delivery team, which consists of the programmers, graphic designers, and other contributors, is itself a

role. Scrum calls this the "development team", which "builds the product."

Then there is a role that has to make sure that the team works on the right things by making sure those things are defined, places them in a priority order, answers any questions that arises, and manages the stakeholders. That is the Product Owner. Finally there is the role Scrum Master, the servant leader to the other two roles, who makes sure that all people involved can work effectively with each other.

The Scrum Team, or "Development Team", is responsible for delivering a working product increment within a specific period of time. We'll call this period of time a "Sprint"; a typical Sprint has a length of two weeks. That means the idea is clear, programmed, tested, and unless there are some business reasons not to, deployed to production. The software build, which we'll call an "increment" of functionality, is verified by a forth, role, which is implicit: the Stakeholder. A Stakeholder is anyone that has a stake in the success of the product. That includes, but is not limited to, users, customers, sales representatives, customer support, senior managers and the Development Team itself. The Product Owner has to work with all of these Stakeholders to make the most valuable product. The entire Scrum Team receives the most feedback from the Stakeholders if an affective Sprint Review is in place.

Let us take a look at these individual roles and how they contribute to a successful product.

Development Team

The Development Team is responsible for producing a potentially shippable, high quality product increment every Sprint at a sustainable pace. That phrase Sustainable Pace is important. Constant pressure to deliver leads to "sprints" that are full-out runs, which leads to poor long-term performance. The team will need to be

able to manage the possible mess (e.g. poor design) it created up till today.

Sadly "marathon increment" doesn't have quite the same ring to it as "sprint."

Sacrifices to product quality are usually a bad start for a Scrum Team and in Markus's experience backfires within five Sprints at the latest. (Matt thinks five sprints is optimistic.) In order to prevent Development Team members from having to do overtime, the team should be able to continuously develop tiny useful and working pieces of product functionality. Any problem that remains unsolved will keep the Development Team from producing more product functionality in the next Sprint. And that is what we want to avoid.

Teams that follow this advice, that try not to over-sprint, and try to break the work down into small chunks, will have to learn entirely new sets of behaviors while developing the software. As they learn, things slow down - organizations new to Scrum will develop an impression that "not much is happening" at first. That's a system effect, and a common one. Become aware of this and do something about it by helping the members of the Development Team adopt new skills for producing software. In our experience, this is an unavoidable step in a successful Scrum implementation; you need to slow down, at least a little to go fast. If investing in professional skills makes you feel uncomfortable, consider the alternative: What would happen if you skipped training the team members?

The Development Team needs the skills that are necessary to create a potentially shippable product increment every single Sprint. Ideally, this means that the Development Team covers all roles found in a traditional organization: architects, designers, programmers, UX experts, testers and operations. Most companies we have worked with do not fulfill this ideal - especially when starting to introduce Scrum in an organization. In that case our choice is usually to staff the Development Team with at least programmers and testers depending on the state of the organization, the skills available and

the team discipline. This is not how it will remain however, it is more of a starting point for the whole transition.

If the ability to push to production, or modify a database, for example, lies outside the Scrum structure, then production pushes and database modifications create a dependency on another team. The time from "ask" to "get" these things injects a *wait state* into the work. For now, it's enough to know that wait states are bad.

A common misconception is the thought that everyone in the Development Team has to be able to take over anyone else's job. Scrum does not prescribe this as mandatory. However, some teams found out that by applying Scrum and by focusing on delivering a potentially shippable product increment every Sprint, a certain degree of cross-training has advantages. It is useful e.g. in case someone gets ill or goes on vacation. Most teams established that there should be at least two people available that can take on any particular task. There may be tasks where more than two are beneficial such as writing code or writing automated micro-level unit tests.

Other teams find that continuous pairing, across specialities (testers pairing with developers, developers with designers, anyone with anyone else) is a valuable way to move toward "anyone has to be able to take over anyone else's job" - or, at least "anyone has to be able to pair with someone else on the team to get work done." If pairing makes you really uncomfortable, don't worry, it's not required, but please note that "x makes me uncomfortable so let's not do it" is probably in the top three barriers we see to real long-term success with Scrum.

But remember, just because it proved to be beneficial to most Development Teams over time, a lack of cross-training and specialists are no reason not to start with Scrum. Otherwise, no company whatsoever would have been able to start using Scrum. In such companies we usually determine primary skills, discover which team members already have secondary and tertiary skills and define

how the team wants to work with them.

The common element here, though, is that the development team is an actual team, with real objectives. If the team needs something to get done, say a bit of testing, to move the sprint forward, and the only person available is a graphic designer capable of doing the work, then the graphic designer can do to the work. "Union"-Style organizations that insist that only a electrician can move a power extension from one corner of the room to another will be creating dependencies outside the team, which will force requests, which will inject wait states into the process.

Wait states are bad. A team that can do more of the process themselves with no waits will be more successful.

Product Owner

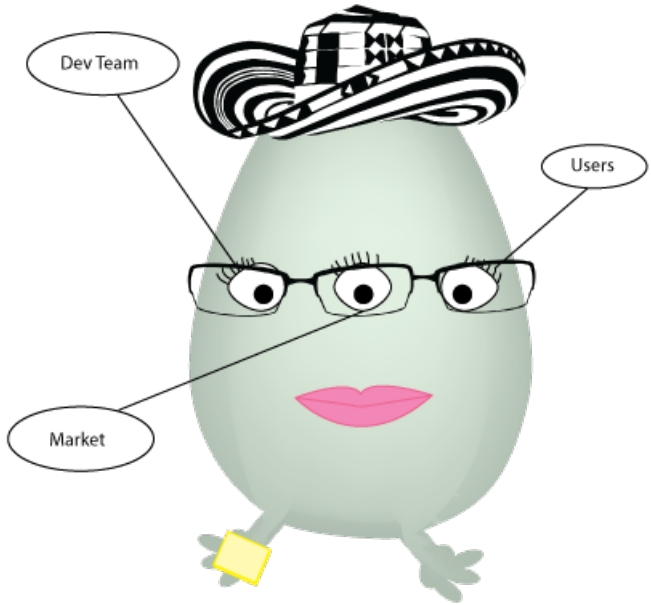
In order to build anything, the team "just" needs to hear what to build, in what order, from a single voice.

We put "just" in quotes because many of the teams we have worked with, over the years, do not have that single voice. Instead they have a steering committee, with multiple stakeholders, with different agendas, all of which are impossible to satisfy and some are in direct conflict.

Since we believe that voice is ideal most of the time (it is possible the team itself could figure out what to do and be judged by results), Scrum requires a central role to do it called the Product Owner. That means the team is never put in a position to be responsible for juggling different features from the various Stakeholders. It is the Product Owner's role to collaborate, negotiate, and ultimately, make the business priority decisions for the team themselves. We see the "Owner" part of Product Owner as being accurate — the Product Owner decides what to build and is judged by the outcome, preferably the success or failure of the product in the market

— instead of being judged by their ability to placate powerful Stakeholders with pet features.

That said, the Product Owner (PO) takes care that the features worth doing for a product are put in a list, called the *Backlog*, and then puts that list in a priority order which can change over time. The PO works closely with the Stakeholders to understand their needs and assesses the relative importance of each individual feature they bring compared to the others already in the Backlog. He also has to work with the Development Team, for example to get a guesstimate for each feature, to make better informed decisions based on the assumed Return-On-Investment.



The Product Owner as a three-eyed alien.

The Product Owner can be thought of as a three-eyed alien. One

of the eyes keeps track of the Stakeholder needs. He therefore has a closer look at particular demands from people he is directly connected with. He has another eye focusing on the market, the competition: existing features that might decrease the value of the work left in the Product Backlog. And the third eye watches the Development Team as it finishes more and more working features during the Sprint.

Some of the misconceptions about the Product Owner role that appear to occur every now and again has to do with the involvement of the Product Owner during the Sprint, especially during the Sprint events or rituals as we call them. While early Scrum evangelists had the understanding that the Development Team has to be protected from the Product Owner as he would interfere too often, that was due to the problems they were solving, when priorities were shifting too much, too often. Many organizations do not have this problem; in some organizations, the PO can add a great deal of value by clarifying questions and collaborating to change things, even during a Sprint. We tell teams new to Scrum to include the Product Owner in all of the rituals, including the Daily Scrum and the Sprint Retrospective.

Scrum Master

The Scrum Master is a servant leader. That means that the Scrum Master dedicates his talents to help the entire Scrum Team to improve. That usually means taking care of organizing the regular rituals (for now, think of them as required meetings), probably including facilitating most of these meetings. Also, the Scrum Master helps the Development Team to get better. In most companies that we have seen, there are many factors at play that may need improving. Yet, there is only one Scrum Master per team with limited time available. We advise a Scrum Master to select a few important improvement to focus on, instead of having a long list. This idea, of managing the work in progress, selecting a few things

to do well at a time, will be thematic in this book.

Again, Scrum Masters are servant leaders. That means that the Scrum Master will look for ways to improve the work of the whole team. In an extreme case, that may even mean that the Scrum Master will fetch fresh coffee for all the team members. Though we have not seen the lack of coffee being the most important impediment for any Scrum Team … yet.

The Scrum Master has to have knowledge in several areas to do the job well. To start with, the Scrum Master should know about Agile, Lean and Scrum in general. They also need certain skills to facilitate meetings as well as team building activities to make the Scrum Team better in self-organization over time. Since the job is also to coach the Product Owner, the Scrum Master should know (or learn) some basics of product development. That can include knowledge on how to build great products from the customer perspective. But that can also include keeping on top of technical practices or at least know when to call in an expert. Finally, the Scrum Master is a coach, mentor and mediator for the Scrum Team. In that role, the Scrum Master will try to extend the team member's individual performance to the benefit of the whole team.

If you look at the description above through command and control goggles, it looks like the Scrum Master "just" runs the daily standup and other forced meetings. At long as the meetings happen at different times, a Scrum Master could do the job for five or six teams once, right? Well, we've tried that. Markus started his very first engagement as a Scrum Master for seven teams at a time!

That missed the point entirely. Once teams understand the game, they can run the meetings themselves. A good Scrum Master seeks to know how to add value and does that - whether by understanding the business enough to advise the team on how to get things done, or consulting with the product owner on what to build, or even just reminded the team of the promises it has made and the experiments it is running. We truly believe that is why the saying goes, "A good

Scrum Master can take on two teams, a great one takes one." It is up to you to decide whether you want good or great Scrum Masters.

[more on the Scrum Master role in chapter xxx]

Stakeholder

While the The Stakeholder is not explicitly listed in the Scrum Guide, the team can't get along without. We decided to make it explicit in *our* guide. In general, the Stakeholder is defined as anyone that has a stake in the product that the Scrum Team delivers. That includes the Development Team and the Product Owner itself. Beyond that, there are usually general managers, support crews, marketing people, customers and end users that have a stake in the products that you build.

The Stakeholders work continuously with the Product Owner to improve the Product Backlog. At the end of each Sprint, they are invited regularly to verify the work the Scrum Team has delivered (the "Sprint Review"), and to provide feedback to make the product even better than it already is. This feedback is incorporated into the Product Backlog and ranked in comparison with existing entries. The Product Owner and Development Team may get in touch with particular Stakeholders during a Sprint to clarify open questions on the items they are working on or the ones that are upcoming in future Sprints.

Stakeholders are important in Scrum since their feedback will greatly improve the work of the whole Scrum Team. One of us once attended a Sprint Review meeting where the Scrum Team witnessed for the first time that a product manager explained important product decisions to the other Stakeholders attending that meeting. The Scrum Team later was positively taken by that moment as they realized that they had the product manager's full support, and that he was not working against them as it felt during previous Sprints.

Scrum Rituals

No, don't worry. When Scrum uses this term, it does not mean incense and prayers. Thy are more like required meetings; meetings that must happen for the team to claim that it is doing Scrum. Like rituals, it is possible for these to have great value and meaning, and it is also possible to go through the motions and miss the point. We'll explain the rituals, and also what they are for, the "point" we are trying to accomplish by having them.

First, "Sprint Planning": At the beginning of a Sprint, the Scrum Team plans the upcoming few weeks. On a daily basis, the team gets together to coordinate their work towards the goal of the Sprint. At the end of the Sprint, they invite the Stakeholders to provide feedback on the work they have finished and to improve the product in upcoming Sprints. After that the Scrum Team inspects the work and looks for opportunities to improve their process. Throughout the Sprint, the whole team has to work on the Product Backlog to prepare it for the next Sprint.

Sprint Planning

The purpose of the Sprint Planning is to agree no what is coming next, in enough detail to get started immediately after. Sprint Planning "kicks off" every new Sprint. The Scrum Team is present to get answers two main questions: what are we going to deliver during this Sprint, and how do we plan to implement those functionalities.

The answer to the first question lies in the Product Backlog, kept by the Product Owner. The Product Backlog is a sorted and ranked list of functionalities that are not yet included in the product. The Product Owner introduces or recaps the most important item in his Product Backlog to the Development Team. The Development Team can ask questions for the Product Owner to answer. When all is clear, the Development Team makes a forecast about whether they

will be able to deliver that item in view of the work that is already planned. If the Development Team pulls the item into the Sprint, the Product Owner removes it from the Product Backlog, introduces the next most important item and the process starts over.

For the what-question, the Scrum Team also tries to determine a Sprint Goal. The Sprint Goal surrounds the Product Backlog items that the Development Team has selected for the Sprint. It can happen that the Product Owner has an idea for a Sprint Goal before the Development Team picked items for the Sprint. Sometimes, the Scrum Team has to see the individual items for the upcoming Sprint first to find a suitable Sprint Goal. It does not matter that much whether you find a Sprint Goal up-front or after forecasting the Sprint. You should have a Sprint Goal at the end of Sprint Planning, and the Scrum Team has to be committed to it.

The second question on how to implement the selected items is usually answered within the Development Team. That is the time where the team members define high-level design and architecture. This gives the Development Team the information they need to derive tasks for their daily work. Since the Development Team might have questions for the Product Owner during this process, he or she should be on short call. The Development Team continues to break down their work into smaller pieces to ultimately deliver the answer to the how-question.

Originally, these two questions were answered in two different parts of the Sprint Planning meeting. Nowadays, many teams intertwine these two parts. For some teams it makes more sense to first find an answer to the what-question. They then excuse the Product Owner from the meeting and work on the answer to the how-question. This may mean that the Development Team finds out that it has to re-visit its forecast after breaking down all the items and has to get back to the Product Owner about that. That is probably why some teams started to pull the most important item into the Sprint and break it down immediately before pulling the

next item into the Sprint. In practice, this means that you now have a larger degree of freedom when planning your Sprint.

At the end of the Sprint Planning, the Scrum Team should have a Sprint Backlog. The Sprint Backlog consists of the items from the Product Backlog that it forecasted, the individual tasks for implementing those items and the Sprint Goal. The Sprint Backlog also includes any improvements that the Scrum Team committed to during the Sprint Retrospective of the previous Sprint. That may also take some time from the Scrum Team, time that will not be available for working on new functionalities in this Sprint.

As all meetings are time-boxed in Scrum, so is the Sprint Planning meeting. Sprint Planning should not take longer than 5% of the overall Sprint length.

Daily Scrum

Every Scrum Team we have ever worked with has some way to visualize the work in progress. They break the work down into chunks, called stories or requirements, then put the "state" of every story on the wall or web-based tool. To figure out the status of the sprint, you look at the board, figure out what day of the sprint you are in, and see if the stories are on track to all get done by the end of the sprint. Teams that want to figure out how long stories have spent in a given state ("Needs details" / "Development" / "Test") can mark them each day with a colored dot by state, or, more likely, just ask a web-based tool.

That means there is no need for a status meeting; status is visible.

So the fifteen minute or less daily Standup meeting, ("The Daily Scrum") is not about status. It solves a different problem - how the team can make best use of their time, to get as much as possible done today. The Daily Scrum *pulls the team forward.* For that to be effective, it helps to take a look at what has happened with yesterday's plan, observe possible obstacles and have the

Development Team assign itself the work for the next day. We limit, or "time box" this minute to fifteen minutes per day to keep it focused; a team of twenty people burn twenty person-hours, a half a week, with a single one-hour meeting!

The Daily Scrum is a stand-up meeting meaning that everyone is physically standing up during the full fifteen minutes. Most people tend to feel uncomfortable standing up for more than fifteen minutes, so requiring standup tends to keep people focused. That has the side-effect that participants make brief statements during the meeting, are are likely to take complex technical, process, or product discussions to a smaller group after. Note: If your team comes from a culture that associates standing up with long speeches, you might not physically stand up. That is okay. We won't take your Scrum Card away. This is true for all these practices. We personally find standing up to be worth following generally, but the definition of a generality is that it has at least one exception.

During the Daily Scrum, everyone should answer three questions:

1. What did I do yesterday that helped the Development Team get closer to the Sprint Goal?
2. What will I do today to help the Development Team get closer to the Sprint Goal?
3. Do I see any impediment that prevents me or the Development Team from getting closer to the Sprint Goal?

The first and the third question are about looking back; the second question is about looking forward. In a variation of the Daily Scrum, all participants answer the first and the third question individually. Based on these answers, the Sprint Backlog is updated. After that, the team answers the second question as a group. This has the benefit that the "what am I going to do today" question is answered knowing the progress made by all colleagues.

At least the Scrum Master and the Development Team attend the Daily Scrum meeting. The meeting is open to other parties such

as the Product Owner and Stakeholders to listen in. However, they cannot not participate. Sometimes we encourage the Product Owner to attend the meeting and share the latest insights (with input from the Stakeholders) regarding the upcoming Sprint. In its current edition (August 2013), the Scrum Guide discourages participation of the Product Owner in the Daily Scrum meeting. Though we feel that teams working closely together with the Product Owner deliver higher business value since the Product Owner has a view on the potentially finished functionality before the Sprint is over.

The "three questions" are just one way to hold the Daily Scrum. The Scrum guide says that during the meeting, the team members explain answers to the questions, but does not literally say how. There are other ways to answer these questions indirectly while meeting the goal ("The Point") of accelerating the sprint. For more on this, consider [TBD].

Sprint Review

The Sprint Review meeting is a larger meeting meeting with Stakeholders to demonstrate what the team built this sprint and get feedback on it, and what to build next. To decide who to invite, look at the Sprint Goal; invite the relevant people.

There are two main activities in this meeting to achieve it's purpose. First the product increment is demonstrated ("Sprint Demo") and after that the feedback is collected. During the first part, the Scrum Team has a bigger role to play. In the second part it is important for the team to stay out of the justification zone and to keep an open mind for feedback.

During the presentation of the new functionality, don't just walk through the new features; show what new capabilities you've created for the user. That is, what the new features integrated with the old features make possible - focus on the literal user story, not

the feature. That might take up some time, but it will keep your audience listening in this and future Sprint Review meetings. And that is the most important thing, to have your Stakeholders with you as they will be delivering crucial feedback.

The review meeting is not an acceptance meeting for the Product Owner and the Development Team. Consider the following situation: The Development member is proud of work it did I in the past Sprint. They demonstrate it to the Product Owner and the Stakeholders, including the company's CEO and CTO. The Product Owner says in front of everyone that he does not accept the functionality because something is missing or because it was intended differently. That can be very painful and demotivating, but just as importantly: So what? "Failing" a sprint means more work. "Succeeding" a sprint means: More work. New work. Different work. New features. Bug Fixes.

The software is what it is. The key questions for next Sprint are: Does the software need more development, and if yes, what should that work be? That's what the backlog and planning meetings are for. So avoid that shameful situation. It is not honoring people and may get in the way of working collaboratively together. The Product Owner should be engaged enough to know if he will "accept" the work as done. If the PO doesn't accept it, it still means the same thing: More work.

The Scrum Master as is facilitator of Sprint Review; the Product Owner has to define how to gather the feedback from the Stakeholders. It is usually verbal feedback but can also be written feedback on post-its or a flipchart. Do not use this meeting as way to justify the product decisions. The Development Team, the PO, and the Scrum Master need to keep their mouths shut and their minds open to receive crucial feedback. After the feedback, it might be appropriate to tell the Stakeholders that you will consider their additional features for the Backlog, or, if you are lucky, how the existing Backlog already addresses their concerns.

The Sprint Review meeting is time-boxed to four hours for a month-long Sprint. For shorter Sprints, the meeting usually is shorter.

Sprint Retrospective

The purpose of the Sprint Retrospective is to help the team move faster; think of it as "turning the crank" to accelerate the team. (Markus uses slightly different language; he focuses on improve the collaboration within the Scrum Team in order to become more effective). The retrospective should include the Scrum Master, the Development Team and the Product Owner; all those roles can improve the way they work together.

The Sprint Retrospective is the final activity of a Sprint. It follows the Sprint Review meeting and precedes the Sprint Planning meeting of the next Sprint. To find out what need improving, the Scrum Team looks back at the past Sprint and determines experiments for improvement in the next Sprint. Usually the Sprint Retrospective results in two to three experiments for the next Sprint.

There are many good books on retrospectives in an agile context. The general underlying approach is to gather data about the past Sprint, generate insights with that information and come up with improvements for the next Sprint. All of this is surrounded by some activities to get into a reflective mood at the start of the retrospective and to close the retrospective with feedback on the facilitation so that aspect can be improved as well. The simplest format may be a military After Action Review: What went right, what went wrong, what should we do differently next time.

The Sprint Retrospective is time-boxed to 45 minutes per Sprint week. Neither of us ever facilitated an effective Sprint Retrospective in less than 90 minutes though. Your mileage may vary.

Backlog Refinement

The purpose of Backlog Refinement is to prepare the Product Backlog for the next Sprint. Depending on the literature, as much as 5% to 10% of a Development Team's time may be used to prepare the next Sprint. Backlog Refinement is an activity carried out over the course of the whole Sprint.

There are three main activities during Backlog Refinement:

- Split larger items in the Product Backlog into smaller ones.
- Estimate new Product Backlog items.
- Identify acceptance criteria for items in the Product Backlog.

These activities can be intertwined. For example, while estimating an item, you find out that you missed an important acceptance criterion. Or you may find out while estimating that an item is too big for one Sprint, and that it you have to break it down into smaller items.

Some teams set aside a dedicated hour per week to work on the Product Backlog. This is usually not enough time and they can only estimate three or four items during this hour. To understand more items from the Product Backlog, your team may need more time. It is obviously allowed to spend all the time required.

The Secret Sauce of Scrum

Earlier we talked about Scrum's meetings as rituals, and the problem of missing the point. But *what is the point?*

Visualization of the flow of the work (A "Scrum Board") means we don't need status meetings. By breaking the work into small chunks, and letting people "sign up" for the work they want to do next, we *don't need to assign work to individuals*, so they can pull the next

assignment just in time. By letting people work on what they see needs to be done, we can swarm around the work based on ability - not defined roles and handoffs.

All of these things point to *self-organization* and *self-direction*. The product owner tells the team what they would like to see (more often the two groups collaborate to decide the best product), then management steps back and lets the technical staff "do their thing." The retrospective makes sure the team is learning and improving at that over time.

It's possible to "do" stuff a different way - by implementing "All the Scrum things." That means having a Product Owner, Scrum Master, Technical Staff, Sprint Increment, Planning, Review, Backlog, and Daily Scrum, but just dumping that extra work onto the existing team, without empowering them to change their work process. Matt calls this "Scrum Jemima", because it pours a sticky syrup of Scrum Ritual on top of what was done before without changing it.

Without the value of Scrum, the secret sauce, the team just has more work to do in ceremonies that don't add value. That's the kind of thing that gets people to say "we don't like Agile - too much process."

It makes us sick. Physical sick.

You may ask us "So what are you doing about it?"

We wrote a book.

Conclusions

This concludes our introduction to Scrum. By now you know the concept of a sprint, the roles, the rituals, and the point, which is so often missed. Next we'll talk about what Scrum does for you (the "why"), then move on to common failure points, then jump into nuggets.

Keep that seat belt on. You'll need it.