The Scrum Guide™ Explained

A Comprehensive Analysis of the Scrum Guide

Moritz Knueppel

To those who have shaped my path in Scrum

Dr. Peter Störmer
my first boss after college,
who encouraged me to pursue the Scrum Master path

Sumeet Madan
the PST who taught me about Product Ownership and helped me in my pursuit of PSM-III

Sofia Katsaouni
my girlfriend and fellow Scrum Master, who has helped my writing of this books with countless discussions during walks by the lake

Self-published in September 2020
First edition

Moritz Knueppel
Hermannstal 89
22119 Hamburg, DE
scrumguide@moritz-knueppel.eu

Contents

Preface

Scrum is simple to understand but difficult to master.

This paraphrased statement from the Scrum Guide holds a special meaning for me. When first I came into contact with Scrum, I read the Scrum Guide a few times and thought I had understood all there was to understand, that I would now know all there was to know. Glossing over the Scrum Guide, I accepted it as ”one way to do things”, not questioning any meaning behind it. My modus operandi was hence all too often ”this is how it would be done by the books, but it doesn’t really work for us, so we will do it differently”.

As I moved around different companies and got to observe Scrum in practice in many Scrum Teams, from small teams of a handful of people to scaled environments with several dozen developers, deviation from the principles laid out by the Scrum Guide appeared to be the norm rather than the exception to it. Other Scrum Masters I would work with didn’t seem to see this as problematic either. But as I saw more and learned more, I began to question the validity of my existing knowledge, or - as I would come to realize - significant lack thereof. I decided for this to change and set out to understand Scrum more appropriately.

This book represents my attempt to explore the Scrum Guide in-depth to understand what this truly means. It is an attempt to inspect the delicate balance between rules, roles, events, values and artifacts the Scrum Guide describes. Most of all, it is my attempt to understand not only WHAT the Scrum Guide says, but WHY it says what it says.

To explain the Scrum Guide, I am taking apart the entire Scrum Guide. In what grew to be over 200 pages, I am examining the Scrum Guide sentence by sentence, sometimes even word by word. I provide background information about the history of Scrum and about underlying

concepts. Most of all, I am trying to explain the interconnectedness of the Scrum Guide, how the sections refer to each other, and everything in Scrum is dependent on each other.

Every Scrum Guide section and, for the most part, every sentence can be read individually in this book. Critical dependencies between them, such as an explanation of Scrum related terminology and essential concepts, are marked in footnotes. Therefore, the book can be used as a reference book to look up the meaning of and ideas behind individual aspects of Scrum.

For an in-depth study of the Scrum Guide using this book, I recommend reading the entire Scrum Guide and then working through this book back to back. While this book is not explicitly intended as study material for any Scrum examinations, I believe that it can certainly help you prepare for exams that require not only a superficial but a more thorough understanding of Scrum.

I sincerely hope this book can be of help to anybody who is seeking to understand Scrum more thoroughly. I hope it helps them gain a more in-depth insight into this genuinely ingenious framework. I wish for this book to be a valuable contribution to the ever-growing Scrum Community.

1 Purpose of the Scrum Guide

Scrum is a framework for developing, delivering, and sustaining complex products.

The first sentence of the Scrum Guide is only 11 words long; however, it contains an essence of what Scrum is and is not, what it is for, and even what understanding it has of product development. This sentence can be broken down into three key components:

1. The term ”*framework*” outlines that Scrum does not have a handbook for every given situation and does not claim to have the solution to every single challenge arising in product development. Rather, it provides a set of rules within which the people involved are free to operate with the tools, methods, and techniques they deem appropriate. Scrum provides a frame to work within, hence the term ”framework”.

2. The trifold of ”*developing, delivering, and sustaining*” products lays out the understanding of the product development lifecycle: Scrum acknowledges that software is not simply shipped when it is finished and left to the customer to use, but rather there ought to be an ongoing process of development in collaboration with the customer, continually adjusting the product to the needs of the market.

3. When speaking of ”*complex products*” , the Scrum guide makes two implicit claims:

    a) There are situations in which using Scrum is appropriate. When developing a product that is complex, especially in environments that are complex, Scrum is indeed capable of providing a

framework with which to address the challenge of complexity.[1]
Example: The development of an intricate mobile application with a team of six developers over the course of a year may warrant the use of Scrum.

b) Conversely, there are situations for which Scrum is not appropriate, namely when the product in development is not complex. While Scrum is a lightweight framework, it nonetheless creates overhead by defining roles, events, roles and what it calls artifacts. This overhead may not be justified if the product development at hand is simple.
Example: setting up a simple website with two people in a few days may not warrant the use of Scrum.

This Guide contains the definition of Scrum. This definition consists of Scrum's roles, events, artifacts, and the rules that bind them together.

Unlike some other project management methodologies or frameworks, Scrum has a clear definition of what it is and what it is not, leaving no room for subjective interpretations or variations while still operating under the title of ”Scrum”.

The Scrum Guide is not descriptive in nature, meaning it does not try to describe an already existing real-world practice by abstracting models from it, but rather it is prescriptive, meaning that by definition, what is written in the Scrum Guide and it alone serves as the definitive source.

The Guide defines the roles, events, artifacts and rules which constitute Scrum. In practice, there are often deviations on one or more of these – often called ”ScrumButs” deriving from the expression ”we do Scrum, but. . . ” followed by an outline of the deviation.

By definition, these approaches are not Scrum, as they violate the prescriptive definitions outlined in the Scrum Guide. Only what adheres to the definitions laid out in the Scrum Guide is truly Scrum.

[1] see explanation of complexity in the Cynefin Framework on pages 5ff.

Ken Schwaber and Jeff Sutherland developed Scrum; the Scrum Guide is written and provided by them. Together, they stand behind the Scrum Guide.

Some approaches to product development gradually emerged as a wide set of practices. Scrum, however, was explicitly defined by Ken Schwaber and Jeff Sutherland. Both developed approaches in their respective companies that would lay the foundation for the modern Scrum. In 1995 they joined forces and integrated their ideas into a framework that is known today as Scrum. Since then, it has been further improved multiple times, most recently in the 2017 update to the Scrum Guide.

Figure 1.1: Jeff Sutherland (left) and Ken Schwaber (right)

In 2002, Schwaber, together with others, founded the Scrum Alliance[2] and launched the Certified Scrum accreditations to spread Scrum.

[2] https://www.scrumalliance.org/

In 2009, he left over differences – mainly regarding the standardization of the curriculum – and created Scrum.org[3], which now oversees the Professional Scrum accreditations.
Parallel to this, Sutherland set up a company called Scrum Inc,[4] offering consulting and training on Scrum, setting up the Licensed Scrum accreditations.

These two sentences in the Scrum Guide can be seen as a commitment to further cooperation in developing Scrum. It further outlines that no organization owns the definition of what Scrum is, but that instead, that definition is up to the Scrum Guide, which is jointly maintained by Schwaber and Sutherland.

[3] https://www.scrum.org/
[4] https://www.scruminc.com/

2 Definition of Scrum

Scrum (n): A framework within which people can address complex adaptive problems, while productively and creatively delivering products of the highest possible value.

This one sentence definition contains a highly distilled essence of Scrum and can be inspected almost word by word:

The use of *"Scrum (n):"* [with *"(n)"* indicating that Scrum is a noun] mirrors a dictionary entry. This can be seen as having two purposes:

1. It reinforces the idea that the Scrum Guide is the definitive source, in which the one and only true Scrum is defined.
2. It clarifies the correct way of spelling Scrum, which is often written as "SCRUM", likely because in an early paper on the subject by Ken Schwaber, the document title contained the word Scrum in all capital letters. The term "Scrum" is based on the rugby formation known as "scrummage" or "scrum" for short. As a proper noun, it is capitalized.

"*A framework within which...*" once again indicates that Scrum does not provide a definitive methodology, but a set of definitions and rules, within which other tools, approaches and methods can be used.

The choice of "*address complex [...] problems*" can be explained by looking at the Cynefin-Framework, which categorizes problems into four primary clusters: simple, complicated, complex and chaotic.[1]

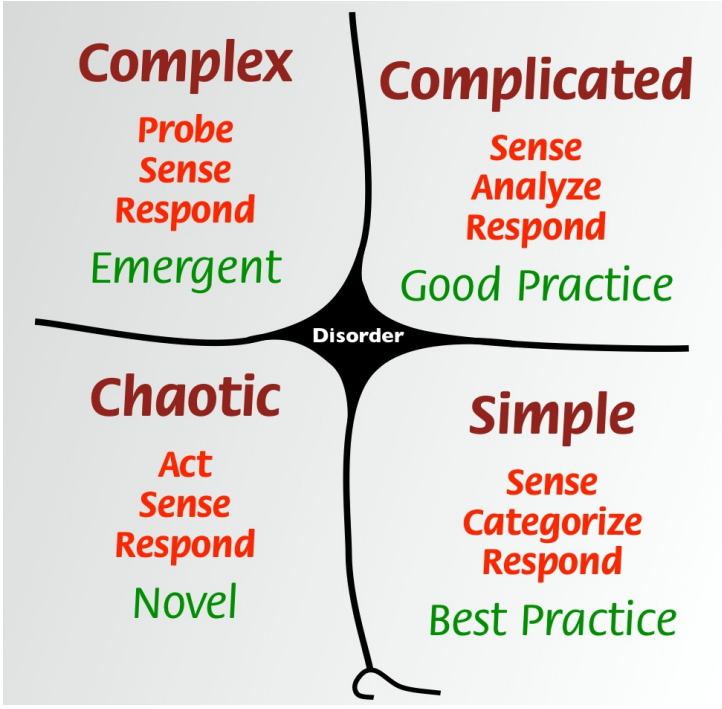

Figure 2.1: A graphic representation of the Cynefin quadrants.

1. Simple problems are ones with ”known knowns”, in which the rules are clear and known, and the situation is considered stable. In situations with these problems, a defined ruleset of best practices can be followed. The advice given by the framework is ”sense, categorize, respond”, meaning that one observes the problem, categorizes it into existing categories and applies the standard solution for that category.

2. Complicated problems are ones where ”known unknowns” exist, meaning there is awareness of what is not known. These are situations where experts can be consulted, and conducting deeper research can lead to solving the problems. Good practices may be identified based on the analysis conducted. The framework’s advice is thus similar to that for simple problems but replaces ”categorize” with ”analyze”, indicating a need for a more in-depth look.

3. Complex problems are ones in which the ”unknown unknowns”

dominate. The nature of cause and effect cannot be known in advance; thus, categorization or conventional analysis will not be useful. According to the Cynefin-Framework, the proper approach is thus to ”probe, sense, respond”, meaning that experiments should be undertaken, of which the results can be observed, and actions can then be defined accordingly. It is impossible to plan the actions far in advance as the nature of the metaphoric playing field has yet to be discovered.

4. Chaotic problems are ones where any knowledge-based response is not possible, and the rules of cause and effect cannot be deducted. The advice given by the framework is ”act, sense, respond”, which differs from complex problems insofar that the first step is to take immediate action rather than experiment first. The goal is to take effective action – as reasonable as can possibly be estimated with good judgment – and based on observations, determine a way to turn the problem from a chaotic one into a complex one.

In the context of product development, those categories imply the following respective approaches:

1. Simple: inspect the problem beforehand, choose the appropriate methods and tools, implement the solution
2. Complicated: analyze the problem beforehand, choose the methods and tools that appear most appropriate, implement the solution
3. Complex: run an experiment, evaluate the result and determine the best course of action based on the insights gained
4. Chaotic: do anything, evaluate the result, based on that determine the best course of action to turn the problem into a complex one

The Scrum Guide states that Scrum is appropriate for complex problems, which is consistent with the ideas of the Cynefin framework: Scrum is founded on empiricism,[1] with the three pillars of ”transparency”, ”inspection” and ”adaptation”, which are mirrored in the Cynefin framework’s recommended response for complex problems:

[1] see the section on empirical process control on pages 21ff.

1. ”probe”, which aims at generating data, or in other words, creating transparency
2. ”sense”, which describes the inspection of the generated data
3. ”respond”, which means to take appropriate action based on the newly won insights, or in other words to adapt actions to the situation as it is now understood

By ”*address [. . .] adaptive problems*” the Scrum Guide claims that Scrum is capable of addressing problems that are not static but can change over time. While many conventional approaches to product development assume the general environment of the product during its development to not change in fundamental ways, Scrum acknowledges that the conditions of the problem may change over time.

A problem that is complex but not adaptive may be solved by conducting an experiment (probing) once, evaluating it (sensing) and executing one derived set of actions (responding). If the problem itself changes though (adaptive problem), the derived set of actions may be outdated. This can be avoided by repeating the process of probing, sensing and responding often enough to adapt to the problem, while not too often to hinder actual progress in developing the solution to the problem. This is the basis for the iterative nature of Scrum.

Example: a company develops a mobile app. At the beginning of the development, it is estimated that building the entire app to fulfill the current expectations will take 24 months. Considering this as a static problem would mean to assume that in 24 months, the conditions around the app will still be the same, and development could be done with an approach like the Waterfall Model.
In contrast, considering it an adaptive problem recognizes that the conditions surrounding the app may change significantly within the development period. In this example, hardware specifications could change, such as higher-resolution screens demanding other designs or the newly widespread availability of NFC in phones creating new opportunities for the app that were not initially planned. Alternatively, potential customers’ expectations for the app may change, as competitors may set standards

with their applications.
Thus, it should be considered an adaptive problem and is hence a problem for which Scrum may be a valid framework to choose due to its iterative nature.

Using the adjective "*productively*" highlights Scrum's focus on efficiency. Scrum aims to utilize the available (human) resources efficiently. This focus can, for example, be seen by the use of time-boxes or the concept of the Daily Scrum, which aims to increase productivity by facilitating a high-density exchange aimed at synchronizing the entire team, rather than letting information spread decentralized and risking miscommunication, which would lower productivity.

The adjective "*creatively*" outlines the necessary approach to solving the problems, which earlier in the sentence were outlined to be complex and adaptive. Unlike simple or complicated problems, for which best practices may already exist, complex problems - and complex adaptive in particular – require trying new approaches often and thus require creativity.[2]

Finishing the sentence with "*delivering products of the highest value*" again underlines the focus on productivity, but also emphasizes that the key metric in product development with Scrum is the value delivered to the stakeholders and especially to the final users. This is a crucial factor in the Scrum mindset, especially for Product Owners: a key measure of whether Scrum is applied properly is whether every iteration aims to deliver the highest possible value product.

To illustrate: A Sprint spent on cleaning up technical debt (often called a "hardening Sprint") could at first glance be considered a good idea. However, it is not conforming with Scrum as it does not deliver the maximum value possible to the stakeholders.

When the Scrum Guide uses the term "product", this includes physical and non-physical products, as well as services of any kind.

[2] for a wider definition of creativity in the Scrum context and how Scrum optimizes for creativity, see pages 49ff.

Scrum is:

- **Lightweight**
- **Simple to understand**
- **Difficult to master**

Listing these three attributes of Scrum illustrates the sharp contrast between the ease with which Scrum can be grasped due to its light setup, on the one hand, yet difficult to understand in-depth on the other.

1. Characterizing Scrum as ”lightweight” refers to two aspects:

   a) The number of prescriptive rules and definitions Scrum forces on those who choose to apply it is relatively low. This can be powerfully illustrated by comparing the Scrum Guide, with its less than 20 pages, to the official guide for the V-Model XT, which contains more than 400 pages.[2]

   b) The amount of time the events Scrum demand creates relatively little overhead. It does not demand hour-long status meetings and time-intensive written updates within the team but instead has relatively short, time-boxed, face-to-face meetings. In terms of staffing and role division, Scrum can also be considered lightweight, as it recognizes merely three roles and centers the product creation within the Scrum Team itself, instead of demanding various new levels of hierarchies with titles and complex chains of command.

2. Scrum is ”simple to understand” in so far as reading the Scrum Guide a few times will get one to understand the overall concept, especially the more practical aspects, mainly the events and, to a lesser extent, the roles and the artifacts. With only a few hours of studying Scrum, it would be possible to observe a Scrum Team in action and identify many of the patterns described in the Scrum Guide.

3. While Scrum may be easy to understand, it is ”difficult to master”, particularly because it is lightweight. Scrum is based on a broad set of axioms about how people can work together to deliver value.
   It addresses these assumptions not only with explicit rules but with a

foundation of values.[3] Understanding how to use the ideas of Scrum and apply them in situations not explicitly covered by the Scrum Guide in the ”Spirit of Scrum” is what differentiates a novice from a master. Reaching this stage requires arguably much practical experience, theoretical knowledge and introspection, and thus mastery of Scrum can be considered truly difficult.

Scrum is a process framework that has been used to manage work on complex products since the early 1990s.

With this, Schwaber and Sutherland define the origins of Scrum. Based on the given time, it can be assumed that they understand their initial, separate approaches to be the roots of what we now call Scrum. Rather than defining their first joint paper outlining Scrum in 1995[3] as the birth of Scrum, they choose to include the early emergences. From this, it can be deduced that Scrum itself is not considered ”finished” by the creators, that it was not placed into existence at a specific time to remain static for all times, but instead is itself a continually improved increment through updates to the Scrum Guide.

Scrum is not a process, technique, or definitive method. Rather, it is a framework within which you can employ various processes and techniques.

Scrum can be thought of as a container, within which those driving the development are relatively free to choose their approaches. In his book ”Agile Project Management with Scrum” [4], Schwaber outlines in the foreword ”Why Scrum Works” that a crucial aspect for the success of product development with Scrum is that it acknowledges that in complex projects decisions should best be made is by the people actually conducting the work.

This sentence once again emphasizes that Scrum is not to be considered as anything beyond a framework, within which the people involved are free

[3] see chapter on Scrum Values on pages 33ff.

– and in a sense obligated – to choose other tools and paradigms according to their needs.

The only requirement for these additions is that they must comply with the rules outlined by Scrum, including the values and ownership regulations. Thus, it is acceptable, for example, to use Test Driven Development, run Product Discoveries or utilize Kanban metrics and WIP-limits as the Scrum Team sees fit. It is on the other hand not acceptable for the management to enforce upon a Development Team the way in which they work on their Sprint Backlog via prescribing Gantt charts, as this would violate the Development Team's ownership of the Sprint Backlog.[4]

Scrum makes clear the relative efficacy of your product management and work techniques so that you can continuously improve the product, the team, and the working environment.

The introduction of Scrum in projects is often considered painful because it reveals problems in the existing product management and makes issues that are hindering better performance transparent. Scrum, with its basis on empirical control theory[5] , is based on making things transparent in order to inspect them and adapt to the situation of newly gained information. The goal of Scrum, in a way, is perpetual improvement aimed towards ever-greater value delivery.

This is one reason why Scrum may only be adopted in its entirety, including the roles and terminology. Scrum is meant to break with conventional product and project management paradigms and allow the transparent inspection and consequent questioning of existing methods.

With this sentence, the Scrum Guide further stakes the claim, that Scrum is not only aimed at improving the product but also

- the team who develops the product, for example by fostering attributes such as trust by emphasizing the Scrum Values[6] , and

[4] for more on the Development Team's ownership of the Sprint Backlog see the section on the Sprint Backlog on pages 170ff.

[5] see the section on empirical process control on pages 21ff.

[6] see chapter on Scrum Values on pages 33ff.

- in a second step the working environment, later in the Guide referred to as ”organization”, by scaling the Scrum thinking beyond the Scrum Team into other parts of the organization; typical examples here are the sales department and the HR department

This last section of the sentence was only placed into the Guide with the latest update in 2017.

The Scrum framework consists of Scrum Teams and their associated roles, events, artifacts, and rules. Each component within the framework serves a specific purpose and is essential to Scrum’s success and usage.

This is often summarized by a quote from the section Endnotes[7], ”Scrum is . . . immutable”, meaning that there is a carefully crafted balance between the components, which will be inspected in this book. Using only parts of Scrum may be useful – e.g., using Sprints or having standup-meetings in the fashion of the Daily Scrum – but is not Scrum. Similarly, leaving out and ignoring components that appear not to fit the business situation reduces the potential positive effects that Scrum may bring.

A typical real-world example for this is adopting most ideas from the Scrum Guide but putting one Project Manager in charge of a Development Team, instead of providing it with a PO and Scrum Master. This may be done because higher management does not trust a self-organized, ”unmanaged” team to deliver relevant results, or even merely due to office politics.

The resulting construct may still be better than the previous arrangement, but it does not allow Scrum to unfold its full potential and hence is not Scrum. The roles, events, artifacts and rules are not created apart from each other, but as a system, which can only function if all components are in place.

[7] see chapter Endnotes on page 191

The rules of Scrum bind together the roles, events, and artifacts, governing the relationships and interaction between them. The rules of Scrum are described throughout the body of this document.

With this paragraph, the authors emphasize again

- the interconnectedness of Scrum's parts via its rules, which are laid out in the sections of the Guide
- the Guide's claim as the definitive source of Scrum in general and its rules in particular

It also outlines that while Scrum does not regulate every single aspect, it does prescribe the kind of interactions deemed most appropriate. An example for this is that not only are the positions of Product Owner and Development Team described, but also it is clarified that these two are not in a hierarchical relationship with each other, but both – together with the Scrum Master – make up a Scrum Team, in which all operate on eye-level.

Specific tactics for using the Scrum framework vary and are described elsewhere.

The authors contrast the use of Scrum as a general framework with specific tactics for using it, e.g., through the application of other techniques and methods, for example, different manners in which to conduct a Daily Scrum.[8] While these tactics may be useful depending on the situation, they should be tailored to the needs of the respective Scrum Team and are hence not part of the Scrum Guide. In turn, it clarifies again that the Scrum Guide is not a complete manual for all aspects of product development, but rather a basis on which teams can organize themselves and their processes.

[8] see explanation of the use of the three questions in the Daily Scrum on pages 129f.