SCRUM HANDBOOK

SINGLE-TEAM SCRUM (STS)

What is Scrum?

Single-Team Scrum

Scrum Glossary

Dan Rawsthorne and Doug Shimp

> *"We make sense of the world through mental models, not just accumulating facts. Mental models help us recognize patterns. A good model allows us to see different things every time we look at it! The best mental models are simple, flexible, and open enough to capture complex situations and encourage us to see more, and ask better questions."*

- Marc Applebaum, PhD

Book Design by Tuna Traffic, LLC

First Edition Publication Date: December 25, 2017
12th version Publication Date: August 1, 2024

ISBN-13: 9781719047265

SCRUM
HANDBOOK

SINGLE-TEAM SCRUM (MTS)

Dan Rawsthorne, PhD, CST
Chief Scientist, 3Back LLC
ScrumZone.org

Doug Shimp, CST
President, 3Back LLC
ScrumZone.org

What is Scrum?

Single-Team Scrum

Scrum Glossary

Table of Contents

What is Scrum?

Scrum is Scrum, and it's a beautiful thing.
Unfortunately, Scrum has been described in many different, inconsistent, and incoherent ways. By focusing solely on Product Ownership we arrive at three common, existing, and different, descriptions of Scrum – which we refer to as $Scrum_P$, $Scrum_G$, and $Scrum_H$.

We both fell in love with Scrum in the late 1990's. It was clean, elegant, agile, and echoed all the good Teams we'd seen in our lives: in sports, in the military, and on the construction site. At its core, Scrum simply says:

- Have a Team that gets its work accomplished all by itself, and is not at the mercy of others;
- Have a Product Owner who is the single-source of requirements for the Team, so there is no bickering or arguing about what to do;
- Have a Scrum Master to help improve process and remove Impediments; and
- Have multiple, frequent, feedback loops in order to be appropriately agile.

What a beautiful concept!

Since then, we have seen, and helped, many teams implement Scrum. We have seen that Scrum can be successfully implemented by a Team if it has the freedom and authority to do so. Of course, each implementation is different in the details, as it adapts to its own environment and situation – but the core of Scrum will be there, and the Team will succeed.

Enlightened Organizations see that Scrum works, and they want some. These Organizations can see how powerful good Teams led by a single Product Owner can be. The problem is, they want to know how to make Scrum work for them — they want to know how to build their own Scrum Teams — so at their core they want definitions, recipes, or instructions...

Because of this need, we (the Scrum Community) had to develop Frameworks, Guidance, and Rules about Scrum. We had to be able to describe how to *tailor* Scrum to the Organizations where it lives. We had to be able to tell stories about how Scrum can be used to help Organizations provide value.

Originally, these stories were based on the experiences of us storytellers, and described how we had seen Scrum work. These stories had to be targeted at the Organizations we were trying to help, as they were useful only to the extent that they helped the Organizations provide value.

And we found that Product Ownership, in particular, was very difficult to describe in a consistent way. The concept of Product Ownership was clear — be a single-source of requirements for the Team — but the implementation was not. Organizations have different needs, and we saw three different versions of the Product Ownership story emerge.

These versions of Product Ownership lead to different variants (or strains) of Scrum, which we refer to as $Scrum_P$, $Scrum_G$, and $Scrum_H$, where the subscripts 'refer to' a primary source for the variant:

1. The *Scrum Primer* is authored by Pete Deemer, Gabrielle Benefield, Craig Larmon, and Bas Vodde, found at https://scrumprimer.org/
2. The *Scrum Guide* is authored by Ken Schwaber and Jeff Sutherland, found at scrumguide.org, and
3. The *Scrum Handbook* is authored by Doug Shimp and Dan Rawsthorne, found at: scrumguide.org.

Each of these descriptions of Scrum was created because of different 'forces' in the Organizations their authors were familiar with. We will give a brief overview of each description, but first let me tell you a *simple* story about Scrum… what most people would accept as a solid, high-level, description of Scrum.

Basic Scrum Description

Scrum is an agile, team-based, model of production that describes how a single Team produces valuable Results (often called Product) for Stakeholders in a complex development environment. While the behaviors described in the Scrum model have been around for decades – if not centuries – the first documented description of Scrum for software was presented at OOPSLA 1995, by Ken Schwaber.

At its core, Scrum is very simple, with few rules and definitions. Here is a typical description.

First,

Scrum has only four Roles:

- **Stakeholders** are people who want the Results;
- The **Team** is a self-organizing, cross-functional, group that produces the Results for the Stakeholders;
- The **Product Owner** does Product Ownership; working with both the Stakeholders and the Team to assure that the 'right' Results are being produced in the 'right' order; and
- The **Scrum Master** does Scrum Mastering; improving the ability for the Team to do its work.

Second,

There are two cadences in Scrum, punctuated by 'Inspect and Adapt' events that enable Scrum to be agile:

- The Team has a **Daily Scrum** (typically 15 minutes long or less) to discuss what was worked on since the last Daily Scrum, and create a 'plan' for what will be worked on until the next Daily Scrum *(the Daily Scrum works well with our natural, circadian, rhythm and is typically used to start the work day)*; and
- The Team has a **Sprint** (typically 1-4 weeks long) to accomplish progress towards work objectives; at the end of which the Team has 1) a **Review**, with the Product Owner and Stakeholders, of the Results it produced; 2) a **Retrospective**, which is an internal conversation about improving *how* the Team does its work; and 3) a **Planning** session with the Product Owner to determine what Results should be produced during the next Sprint. *(The Sprint cadence works well with the Stakeholder and Organization rhythms, interrupting them often enough to get meaningful feedback for the Team, but not so often that they feel 'put upon'.)*

Third,

There are three artifacts in Scrum (note: remember that the Team's Results have historically been called Product):

- The **Product Increment**, which is the result of the work the Team has completed, and is the current version of the Team's Results;
- The **Product Backlog**, which is a list of new, or additional, Results that the Stakeholders want, and the order in which they will be worked on; and
- The **Sprint Backlog**, which contains a list of Tasks that the Team is working on in the current Sprint.

Most people would agree that this is a good description of Scrum. Within this description, however, there is a lot of ambiguity, and this ambiguity has caused many strains of Scrum to emerge – often because of differences in **Product Ownership**, which we define as *"the collection of Responsibilities and Accountabilities that sit between the Stakeholders and the Developers."*

Basically, Product Ownership is what converts the 'wants and needs' of Stakeholders into work for Teams to do. In practice, we have seen three basic kinds of Product Ownership, which we refer to as $Scrum_P$, $Scrum_G$, and $Scrum_H$, as you see in the following figure.

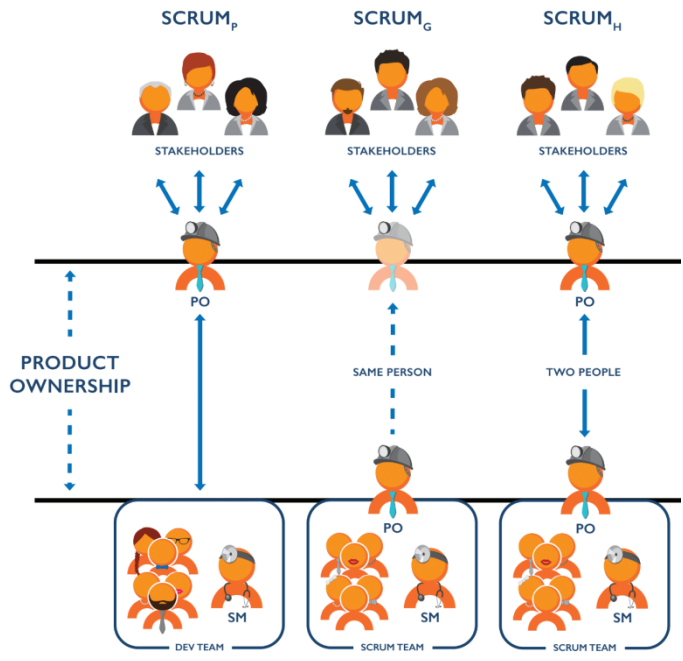


Figure 1: Three Strains of Product Ownership

In the next few sections, we will give overviews of these three strains of Scrum.

Scrum$_P$ Overview

Early implementations of Scrum enabled a single Team to develop a single Product for a single set of (relatively cohesive) Stakeholders. To do this, there was a Product Owner *outside* the Team with the Stakeholders. To distinguish this kind of Team from those that contain their own Product Owners, we will call it a DevTeam – a Team that does Development, without any Product Ownership.

This 'external' Product Owner prioritizes requirements, at the beginning of a Sprint, at a fairly high level, and works with the DevTeam to develop a Sprint Backlog that describes how they will implement these requirements.

The DevTeam commits to the Sprint Backlog, the Sprint Backlog is 'locked down', and the DevTeam is left alone to develop. Even though the Product Owner can't 'mess with' the Team during the Sprint, the Team can reach out and work with the Product Owner during the Sprint in order to get a better understanding of the requirements.

Figure 2 gives the basic structure of a Scrum$_P$ team and its surroundings.

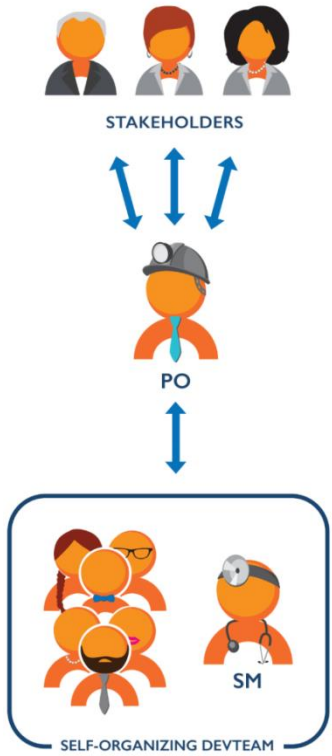


Figure 2: Scrum$_P$ Structure

As intended, Scrum$_P$ works well for developing a single Product for a single set of Stakeholders, especially if the Stakeholders' requirements are relatively well-known and not overly volatile. This type of Scrum is still around, still being useful.

We call this Scrum$_P$ because one of its early descriptions was the *Scrum Primer*.

$Scrum_G$ Overview

However, there is often the need for this single Team to not only *develop* the Product, but *maintain* the Product once it has been deployed. This requires the Team to be interrupted frequently in order to resolve time-sensitive issues related to bugs, building, testing or deployment that come up in Operations.

The Team needs to know: *"Which is more important, the new functionality we're already working on, or the new work that just came up?"* and this is a decision the Product Owner needs to make *right now*; the Product Owner needs to be agile *all the time*.

A 'Business Side' Product Owner (as we see in $Scrum_P$) is often incapable of making these decisions in a timely manner, and the response to this problem was to move the Product Owner onto the Team in order to be available to the Team to prioritize these interrupts real-time. This 'internal' Product Owner still works with the Stakeholders to gather and prioritize the functional requirements, but *lives on* the Team in order to prioritize new work as needed.

Figure 3 shows the overall structure of a $Scrum_G$ team and its surroundings.

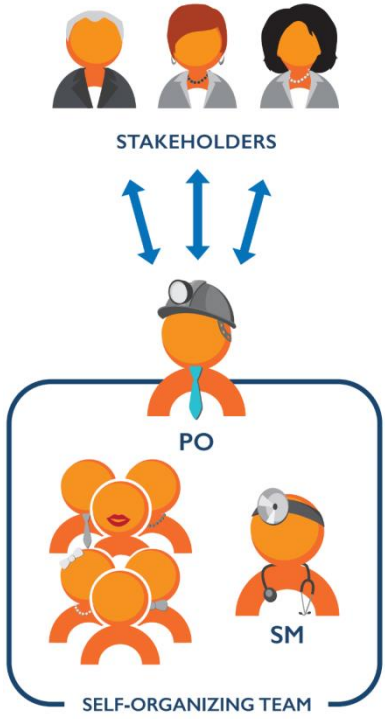


Figure 3: Scrum$_G$ Structure

The Sprint Backlog is malleable, not committed-to and locked-down (changes can be negotiated at any time). This does not mean the Team makes no commitments; there are two commitments the Scrum Team makes:

- There is a Strategic commitment; The Product Goal. This is a future state of the Product that has no explicit timeline. It is normally determined through consultation with the Stakeholders, is owned by the Product Owner, and is a commitment from the

Team to the Stakeholders. The Team commits to one Product Goal at a time, and they get a new one once the current Product Goal is either abandoned or accomplished.

- There is a Tactical commitment; the Sprint Goal. The Sprint Goal is created by the team during the sprint planning, and it is the single objective for the Sprint. The Sprint Goal is a commitment by the team, and provides some 'wiggle room' concerning the work they actually have to do.

As intended, $Scrum_G$ works well for developing and maintaining a single Product in production, even if the Stakeholders' are volatile.

We refer to this type of Scrum as $Scrum_G$ ('G' for 'Guide') because its most popular variant is described in the *Scrum Guide*.

$Scrum_H$ Overview

As you can see, these first two types of Scrum ('P' and 'G') are mutually incompatible (PO inside vs. outside the Team, Sprint Backlog locked-down vs. malleable, Commitment to Sprint Backlog vs. Sprint Goal), but each is useful when used appropriately.

They are both simple and easy to understand, and this very simplicity causes problems when people force-fit them into their Organizations. In many cases, neither of these strains of Scrum will work without modifications. Therefore, in order to make Scrum work, many Organizations made modifications, and new variants of Scrum arose.

Many of these variants arose in order to 'fix' problems the Organizations found with Product Ownership. For example, one of the major problems with both of these versions of Scrum is that each of them requires the Product Owner to be an effective link between the major Stakeholders and the Team.

The Product Owner must be able to spend a significant amount of time with each group. *(For example, Jeff Sutherland – one of the original authors of Scrum – recommends that the Product Owner spend 'half' his/her time with the Team, and 'half' his/her time with the Stakeholders.)*

So, these Organizations found themselves asking questions like: *"What happens when the Team is developing and maintaining multiple Products, each with its own set of Stakeholders?"* and *"What happens when the Scrum Team is in India, while the Stakeholders are in New York?"*

In these cases, and many others, the Team still needs the on-the-team Product Owner for the tactical prioritization of work, but this team-based Product Owner often has trouble managing the necessary Stakeholder interactions and still 'do right' by the Team.

The result was that another strain of Scrum arose that realizes that there often is a need for *two* Product Owner-types, one to work with the Stakeholders and prioritize what *Products/Features to work on* (strategic prioritization), and one to work with the Scrum Team to prioritize the *work being accomplished* (tactical prioritization).

Since this type of Scrum often occurs when there is more than one Product being produced (from the Organization's perspective), it often doesn't make much sense to call *anybody* a 'Product Owner' – there isn't an appropriate, single, Product to own. Therefore, we sometimes see names for these roles that more accurately describe their responsibilities (Business Owner, Team Captain, PO Proxy, Chief Product Owner, Squad Leader, Crew Chief, Helper PO, and so on).

Figure 4 shows a generic structure of a $Scrum_H$ team and its surroundings.

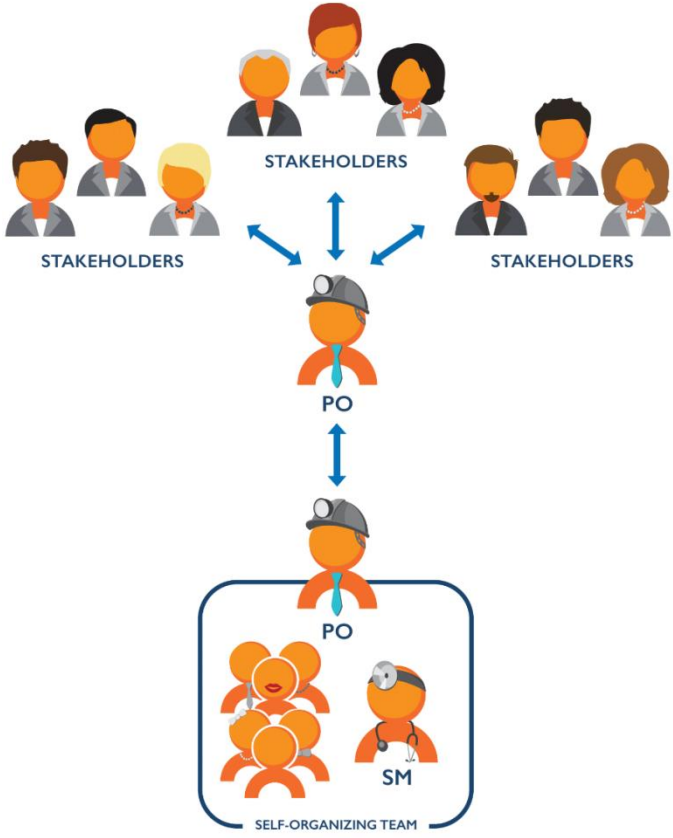


Figure 4: $Scrum_H$ Structure.

This is a different kind of Product Ownership… and this kind of Product Ownership is distributed and agile *all the time*. This is a common strain of Scrum in the real-world, and is

what many Organizations who try to do $Scrum_G$ end up *actually* doing because a single Product Owner can't do everything that is needed.

We refer to this new version of Scrum as $Scrum_H$ ('H' for 'Handbook'), and it is the subject of this Handbook.

However, $Scrum_H$ goes well beyond simply having two different Product Owner-types. This is the *start* of things – the reason we *need* a new description – but it is not all. There are many other adaptations that are commonly made in order to have a Scrum that is adequate for today's world, and we have combined many of them into $Scrum_H$.

We like $Scrum_G$, as far as it goes, so we have tried to make $Scrum_H$ as consistent with $Scrum_G$ as we could. We hope that $Scrum_H$ helps people understand how to extend and use $Scrum_G$ to be useful for *today's* Organizations.

$Scrum_H$ does not change the Scrum we find in $Scrum_G$, it just describes it differently. Because it extends $Scrum_G$, we consider $Scrum_H$ to be a new strain of Scrum, not a new Scrum. The end result is that $Scrum_H$ does several things:

1. $Scrum_H$ defines multiple Stakeholder, Product Ownership, and Scrum Mastering sub-roles (or facets), providing a more realistic model adequate for today's world;
2. $Scrum_H$ extends and unifies both $Scrum_P$ and $Scrum_G$ (though it leans heavily in the direction of $Scrum_G$); and

3. $Scrum_H$ is scale-ready; it can respond to growth or be applied to large Organizations (through fractalization) without any modifications or additions to the model.

So, let's move on to a more-detailed description/model of the $Scrum_H$ version of Single-Team Scrum…

THIS PAGE INTENTIONALLY LEFT BLANK

Single-Team Scrum

(with Commentary)

At its core, Scrum is a model showing how a single Team produces Results for Stakeholders*. The Scrum$_H$ model of single-Team Scrum is a clean, flexible, description of Scrum that is appropriate for most Organizations...*

Commentary*:*

Throughout this description of Single-Team Scrum, you will see boxes like this one. They are not part of the 'official' description, but contain advice, explanations, comments, and the like.

These commentaries are important; they will help you understand the description, and the ramifications of the description – but they are not part of the description…

Teams and Roles

Let's start with a picture. On the left we see the simple view of the Scrum Roles that we find in $Scrum_P$ and $Scrum_G$. On the right we see how the roles split into the entities we have in $Scrum_H$. These entities can be thought of as facets of the original roles, or as sub-roles…

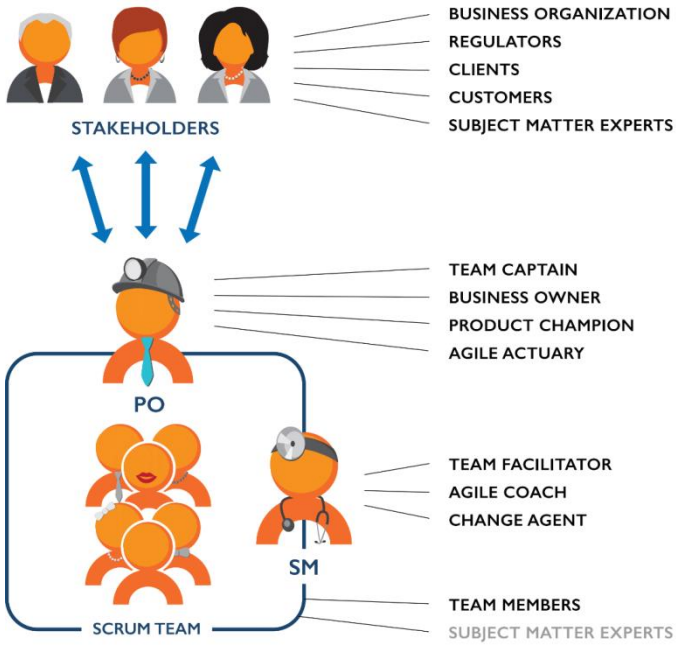


Figure 5: Expanding the Basic Scrum Roles

Now, let's define what we see here…

Stakeholders

A Team does not exist in a vacuum; it is surrounded by **Stakeholders**, who are people who are involved with, affected by, or have an effect on, the Team. There are many kinds of Stakeholders, and some of them *must* be included in our description in order to fully understand Scrum; the Stakeholder Groups we must have names for are:

- The **Business Organization**: The organization the Team belongs to, or works for. The Business Organization provides services the Team needs in order to exist, such as Human Resources, Logistics, Facilities, Contracting, Training, Legal, and so on. The Business Organization is often simply referred to as 'the Business' or 'the Organization'.
- **Regulators**: Entities, either inside or outside your Business Organization, who can regulate, or constrain, the Product/Results the Team is producing and/or the process that the Team uses. Examples include things like Cyber Security, CISA, ITAR Compliance, SOX, Medical, etc.
- **Clients**: The people who will be using, wanting, consuming, or working with the Product/Results the Team is developing. Clients are interested in the capabilities (features, usability, speed, etc.) that are, or will be, delivered.
- **Customers**: The people who are, or will be, paying for the Product/Results, the Team is producing.

Customers are interested in schedules, budgets, costs, delivery dates, ROI, NPV, Investment and Sales Opportunities, and so on.

- **Subject Matter Experts (SMEs)**: People, external to the Team, who will act as 'honorary' Team Members when needed. These are people with competencies (knowledge and/or skills) that the Team needs, but do not exist on the Team itself. These people could be from any other Stakeholder group, and their competencies could assist Production, Scrum Mastering, or Product Ownership.

Commentary:

These Stakeholder groups are neither independent nor mutually exclusive; the Stakeholder community is often a mess. We name these groups in order to help you make sense of this mess. For example:

- The Team must be able to clarify wishes, needs, and requirements with many different Stakeholders, but especially with the Regulators and Clients – or their surrogates. These clarifications need to be accomplished in near-real-time, so that Product Ownership does not have to be the go-between.
- One particular thing we like to do in order to get help when we need it is to co-opt Clients, who want a particular feature, to become SMEs, who will then work with the Team to develop that feature;

- Some People are both Customers and Clients, but they should play one role at a time – we don't want to see 'money people' in the Team room with the developers; and
- Many Organizations operate under complex Regulatory Regimens; Regulators produce standards, rules, and regulations… that can affect both what is built (e.g., it must be bi-lingual) and how it is built (e.g., the Code must be completely protected by Tests).

We talk a lot about self-organization in Scrum, but that doesn't mean that Teams can ignore their Stakeholders… My 'off the record' definition of a Stakeholder is *"somebody you ignore at your peril."*

The (Scrum) Team

There is only one Team in Scrum, and that is the Scrum Team. A Scrum Team (commonly called the 'Team') is a small, co-located, self-organized, self-contained, value-driven, group of full-time **Team Members** (including their SMEs on an as-needed basis) who are organized around a Mission. Let me define these terms:

- **Mission**: A short statement of why the Team exists – what it does, for whom, and why – the Mission is often obvious from looking at the Team's name – examples include "*Develop and Maintain the XYZ*

System", "*Project ABC*", or "*Acme Construction Plumbing Team*";

- **Small**: The Team must be small enough to remain nimble and large enough to complete significant work during a Sprint; typically, this means that a Team has between three and nine Team Members – the 'sweet spot' for software development seems to be around five;
- **Co-Located**: Team Members must be co-located (at least virtually) so that they can talk to other Team Members within seconds/minutes, rather than hours/days.
- **Self-Organized**: A self-organized team is one that chooses how best to accomplish its work, rather than being directed (micro-managed) by others outside the team.
- **Self-Contained**: A self-contained (also called cross-functional) team is one that contains all the knowledge and skills necessary to accomplish its objectives. In real life, this may require Subject Matter Experts (SMEs) who are *not*, actually, full-time Team Members.
- **Full-Time**: Each Team Member belongs to a single Team. However, the Team, because it is self-organized, may 'loan out' one or more of its Team Members to work elsewhere as a Subject Matter Expert. In general, when there are multiple Teams involved, an individual Team Member may work on

two Teams – a 'home' Team (the one he/she belongs to) and an 'away' Team (the one he/she visits)…

- **Value-Driven**: The Team Members value working together; they are constantly improving themselves, their Team, their environment, and their tools; and they strive to live up to **Values** that they – or their Organization – believe are important.

Commentary:

1) A Self-Organized Team usually develops a set of **Team Norms** that define how they are supposed to interact with each other and others outside the Team.
2) Along with its Team Norms, a Scrum Team often has a **Conflict Resolution Process**, which resolves issues resulting from disputes, disagreements, violations of Team Norms or Values, and so on. A Conflict Resolution Process should:
   - be fair and legal;
   - have an escalating series of steps (negotiations, agreements, discussions, sanctions, punishments, etc.), that
   - could ultimately lead to removal from the Team, expulsion from the Organization, or worse.
3) There are many ways to express values, and there are many different sets of values a Team could have. The

values the Team Members strive to meet could be things like:

- Let's all be REAL (Responsible, Ethical, Accountable, Loyal) to each other;
- We believe in Honor, Courage, and Commitment (US Navy, Marines);
- Team Members should embody and live by the values of commitment, courage, focus, openness, and respect (Scrum Guide);
- Or any other set of values the Team is willing to believe in…
- One collection of Values that we like is called the **Team Values**, which is an extension of the Scrum Values:

Openness	There should be no secrets between/ amongst Team Members about things relevant to the work and their ability to do the work.
Focus	Everything the Team does must have a focus, and the Team Members must focus on what is important in everything they do.
Commitment	Team Members commit to each other, to the Team's goals, to be ethical, and to the Product itself.

Respect	Team Members believe that people are always doing the best they can do at any given moment.
Courage	Team Members must have the courage to make reality visible, do the right thing, and work on tough problems.
Visibility	Team Members make the current state of the Team's Product/Results visible to each other, Stakeholders and the Business.
Humor	Everyone needs a sense of humor; if we can't laugh at some of the things we do, we'd have to cry.
Accountability	Everyone is ultimately answerable for their work or decisions; anyone can be 'held to account' for what he or she does.

Work Items the Scrum Team works on are called **Stories**, and as the Team completes the Stories, they produce Results in an iterative and incremental manner, thus providing opportunities for meaningful feedback from Stakeholders.

While they are doing their work – when they have **Work In Progress (WIP)** – they should *"do their due diligence and use an appropriate Standard of Care to do their work, without lollygagging or gold-plating,"* which will (as Project Management training will teach you) maximize the Team's sustainable throughput.

There are no internal, named, 'technical' roles on a Team — they are all called, simply, Team Members, Developers, Plumbers, Painters, or whatever is appropriate. Two of these Team Members have additional responsibilities/ accountabilities – the Team Facilitator (TF) and the Team Captain (TC), which we briefly describe here (more detailed descriptions to follow):

- **Team Facilitator (TF)**: The Team Member who facilitates the Team in order to help it do its work and improve the way it does its work; and
- **Team Captain (TC)**: The Team Member who is accountable to the Business for maximizing the value of the Team's Work.

Commentary:

1) Many people like to refer to the Team Facilitator and the Team Captain as the 'Scrum Master' and 'Product Owner', respectively. However, as we'll see in the next few sections, both Scrum Mastering and Product Ownership have multiple facets (contain multiple roles) that belong to the Business Organization as a whole, and the Team's Captain and Facilitator are the only two roles that are **required** to actually be on the Scrum Team.

   That being said, we often refer to these two roles as the **Team's Scrum Master (TSM)** and the **Team's Product Owner (TPO)**.

2) Both the Team Facilitator and Team Captain are roles that belong to the self-organized Scrum Team; and this is 'process-speak' that has particular meaning that you need to understand. A role is a 'named collection of behaviors and responsibilities', so having a role on a *self-organized Team* means that the Team will have the role's behaviors and responsibilities, but nobody outside the Team will tell the Team how to do it.

   So, even though we typically think of the Team's Captain and Facilitator as being embodied by individuals on the Team, all that is actually required is for the Scrum Team to have Team Facilitation and Team Captaincy.

   We know the Scrum Team Produces Results, and that, as a result of its self-organization, all the Team Members are *allowed to* help each other out with any work they do. The Scrum Team is a 'closed' unit that does Production, Scrum Mastering, and Product Ownership – but nobody can tell them (or even needs to know) how they do it.

3) Let's focus on Team Captaincy. There are variants of Team Captaincy; in fact, the term 'Team Captain' is not always appropriate. So, let's discuss some other terms we might use to describe the Team's Product Owner (TPO) and when we'd use them:

- Team Spokesperson (TS): In the real world, a Team Spokesperson is a Team Member who explains what the Team is doing, and why. Every TPO plays this role, but being a Spokesperson does *not* make a person a TPO; being a TPO also requires taking accountability for the Team's Results.

There are various ways a Scrum Team self-organizes to provide its TPO, and these define what kind of TPO we have. Here are a few we have found useful…

- Team Leader (TL): In the real world, a TL is someone who is the Team's Spokesperson *and* has ownership of the Team's 'play calling', even if the actual 'plays' are being determined by the Team. This ownership is what gives the Team Leader the authority they need in order to be accountable for the value of the Results produced by the Scrum Team.

  We like to use this term when the TPO manages the Team's Backlog but is not seen as a 'real' member of the Team by the other Team Members – TLs are seldom involved in either Development Work or Scrum Mastering. In other words,

  Team Leader = Team Spokesperson + 'owns the play calling'

- Squad Leader (SL): In the real world, a SL is a soldier who is in charge of a squad of lower-ranking soldiers, and is accountable for the work

they do. What is important about SLs is that they are soldiers themselves. We like to use this term when the TPO is involved with Development Work. In other words,

Squad Leader = Team Leader + Developer

- Crew Chief (CC): In the real world, a CC is a Journeyman who works, and directs work, at a work site – the CC is accountable for the work being done. As a Journeyman, the CC also mentors, trains, and coaches apprentices. We like to use this term when the TPO is involved with Development Work and Scrum Masters the Team, even if they're not the Team's 'formal' Facilitator. In other words,

Crew Chief = Squad Leader + Scrum Mastering

- Team Representative: In the case that the Team's accountability is owned by the self-organized Team (the Team Captaincy is done by the Team, not an individual), we say that the Team has a virtual TPO. For Teams like this, when a TPO is needed outside the Team for some reason, the Team selects a Team Representative (TR) to wear the *'PO hat'* for the required purpose. In this case, the Team Rep is both the Team's Spokesperson and can 'make deals' on the Team's behalf. In other words,

Team Representative = Spokesperson on demand
+ team accountability

Scrum Mastering

There are three roles involved in Scrum Mastering, including the Team Facilitator mentioned above. Each of these roles is a servant-leadership role, enabling and empowering people to do their jobs. Each of them is responsible for ensuring Scrum is understood and properly applied. Here are their definitions, their 'One Big Thing'. *(Note: some of the terms used here will be defined later in this handbook.)*

- **Team Facilitator (TF)**: A Team Member who is accountable for, and facilitates (makes easier), the Team's self-organization, growth, maturation, and improvement (on a daily basis) as the Team 1) does its work, 2) removes **Impediments** to progress, and 3) achieves its improvement objectives, or **Kaizens**. Every Team must have a Team Facilitator, who is often a technical contributor, as well.
- **Agile Coach (AC)**: A person who helps people improve their 'agile' skills. The skills might be technical or non-technical, and improvements are typically achieved through training and/or mentoring. The Agile Coach often works with the Team Facilitator to: 1) help the Team Members understand, implement, and improve their use of

Scrum and agility; and 2) help the Team identify its Kaizen every Sprint. Every Team must have access to an Agile Coach as needed; this usually requires one Agile Coach for every 2-10 Teams, with five being the 'sweet spot'.

- **Change Agent (CA)**: A person who helps the Organization adopt, implement, and sustain Scrum, and understands how best to support and work with Scrum Teams – this includes organizational design. There needs to be at least one Change Agent per Organization. The Change Agent is usually also an Agile Coach (with expertise in organizational behavior), and is considered the 'senior Scrum Master' in the Organization.

Scrum Mastering provides many other services to the Team and the Organization. Here is a list of additional responsibilities with indications of which Scrum Mastering roles would/could/should be involved *(These responsibilities are 'lifted' directly from the 2017 Scrum Guide).*

Scrum Mastering responsibility	**TF**	**AC**	**CA**
Helping people understand which of their interactions with the Scrum Team are helpful and which aren't	x	x	x
Creating interactions that maximize the value created by the Scrum Team	x	x	x
Finding techniques for managing the Team's work effectively	x	x	

Scrum Mastering responsibility	TF	AC	CA
Helping the Organization understand the need for clear and concise statements of work		x	x
Helping the Scrum Team understand the need for clear and concise statements of work	x	x	
Helping the Organization understand planning in an agile environment			x
Coaching the Team in self-organization and teamwork, including swarming	x	x	
Helping the Team to create high-quality Results	x		
Facilitating Scrum events as requested or needed	x	x	x
Coaching the Team in organizational environments in which Scrum is not yet fully adopted and understood	x	x	x
Leading and coaching the Organization in its Scrum adoption			x
Planning Scrum implementations within the Organization		x	x
Helping Stakeholders understand and enact Scrum and agility	x	x	x
Causing change that increases the productivity of the Scrum Team	x		
Working with other Scrum Masters to increase the effectiveness of the application of Scrum in the organization	x	x	x

Scrum Mastering responsibility	TF	AC	CA
Helping the Team acquire or deepen specific skills		x	
Ensuring the Organization is taking steps to develop and sustain necessary expert skills			x

This is a good list of responsibilities for the Scrum Mastering roles, but it is incomplete. You will find other responsibilities for these roles as you continue reading this Handbook.

Commentary:

1) These three Scrum Mastering roles can be covered in different ways:
   - On mature Teams it is common for 'everybody' on the Team to do Team Facilitation, as a self-organization issue. When the Team Facilitator is needed outside the Team, the Team selects and sends a Representative.
   - It is common for some (or all) Team Facilitators to also be Agile Coaches (especially about process issues). In this case, we refer to this combined role as a Facilitator/Coach (FC). These people are their own Team's Facilitator and Coach, and may also coach other Teams or their Facilitators. One or more of these FCs may also be Organizational

Change Agents. Being one of these 'do everything' Scrum Masters could cause 'overloading' problems – their own Team's Facilitation could suffer – and they may need 'deputy' Team Facilitators (one of the technical members of the Team) to pick up the load.

- It is common for the Change Agent to also be the lone Agile Coach, if the Organization is relatively small (fewer than ten Teams). In this case, it is probable that each Team's Facilitator is also a technical contributor.

In any case, the people who play these roles must collaborate closely in order to be aligned on, and perform, their functions.

The Change Agent helps identify needed changes in process, practices, and organizational design – which include changes in both Structure and Governance. The reason the Change Agent does this is to change the Organization's Culture to be more accepting and supporting of Scrum and agility. The improvement objectives for the Organization should be: 1) to increase its ability to change, and 2) to help it find high-value Items that need to be worked on. Often, the Change Agent will work with the Agile Actuary to evaluate proposed Organizational Design changes in an analytical way.

Product Ownership

There are four Product Ownership roles, including the Team Captain mentioned above. Each of these roles is involved in producing Results that make Stakeholders 'happy' and, collectively, they span the range from creating visions and objectives, to delivery management, to deciding what the Scrum Teams should do next. Here are their definitions, their 'One Big Thing'. *(Note: some of the terms used here will be defined later in this handbook.)*

- **Team Captain (TC)**: The Team Member who is the Team's Representative to the Business and is accountable for maximizing the value of the **Team's Work**. Each Team must have its own Team Captain to own the 'play calling' about what work the Team should do next.
- **Product Champion (PC)**: A person who is accountable to Stakeholders to deliver a **Product**. The Product Champion manages and prioritizes the **Product Backlog**, owns the Product's **Vision** and Development **Strategy**, represents the Product's **Clients** and **Customers** to the Organization, and provides/ identifies subject matter expertise to the Organization to aid in development and reviews.
- **Business Owner (BO)**: A person who is accountable to Stakeholders for ordering a **Results Backlog** in order to maximize the value of delivered **Results**; these Results can span several Products. We expect Business Owners to work with their Stakeholders

(including **Product Champions**) to determine which Results should be delivered, when, and why.

- **Agile Actuary (AA)**: A professional who does qualitative and quantitative analysis to help Business Owners and Product Champions mitigate risk, support decision-making, evaluate options, and produce **Delivery Forecasts** when necessary.

The first three roles (Team Captain, Product Champion, and Business Owner) are 'ownership' roles (their job is to make prioritization decisions), while the Agile Actuary is a more technical role that offers support to the others. Their responsibilities are tightly linked, and all impact the Scrum Teams. In particular, these roles interact to:

- Assure that the Work the Teams do is necessary to develop the Results that will be delivered to Stakeholders, and that the Teams understand both the Work and the Results to an appropriate level of detail – this includes knowing and meeting the appropriate **Definition of Done**;
- Assure that the expected scope of the Deliverable Results is consistent with appropriate Delivery Forecasts (usually one per Product), that the Teams understand the expected scope of each Deliverable Result, and that the Teams are not overly influenced by the predictions of the Delivery Forecasts;
- Assure that the predictions of the Delivery Forecasts adapt to stay in alignment with the realities of the

Work as they become visible, and that the Delivery Forecasts are used to establish realistic delivery expectations with the Stakeholders; and

- Assure that Product Ownership uses *informed* decision-making; in particular, that the Agile Actuaries 1) identify indicators and metrics to aid decision-making, 2) work with Teams to collect observations and data, and 3) analyze the data and prepare advice for the Business Owners, Product Champions, Team Captains, and Change Agents.

Commentary:

1) The relationships between the people playing these roles are very important, as they, the Teams, and the Stakeholders, must be in alignment about what Results are to be delivered (including the appropriate Definitions of Done), and what work is needed to produce those Results. There are a number of ways to do this:

   - The best way to stay aligned is if all three 'ownership' roles (Team Captain, Business Owner, and Product Champion) are played by the same person – as is required by both $Scrum_P$ and $Scrum_G$. This can, and has, happened. It's rare, though, and usually requires a one-team Organization with its Team Captain playing all the roles.

   - Some Organizations have spread a single Business Owner across several Teams and made this work;

we think this is because each of the individual Teams actually has a Team Representative or its equivalent. And, of course, since this situation usually happens when there is a single product, this Business Owner is actually the Product's Champion, as well.

- More often than not, however, the roles are dispersed throughout the Organization in some way. In a large Organization there could be multiple development Scrum Teams doing the work, multiple Product Champions representing Products for different sets of Stakeholders, and Business Owners holding it all together, making sure value (to the Business) is being maximized.

2) Everybody in 'ownership' roles must be in alignment about 1) what is being built, and 2) the flow of work within the system. This alignment is achieved through collaboration and accountability, and puts a huge burden on the Organization's Scrum Masters, who must work with – and advise – the Product Ownership about how to do this.

   So, here is some guidance about these issues…

   - Each Team Captain is the **only** person in the Organization who decides what work his or her Team should do next (and this decision should be the result of conversations with appropriate Team Members, SMEs, Business Owners, Product

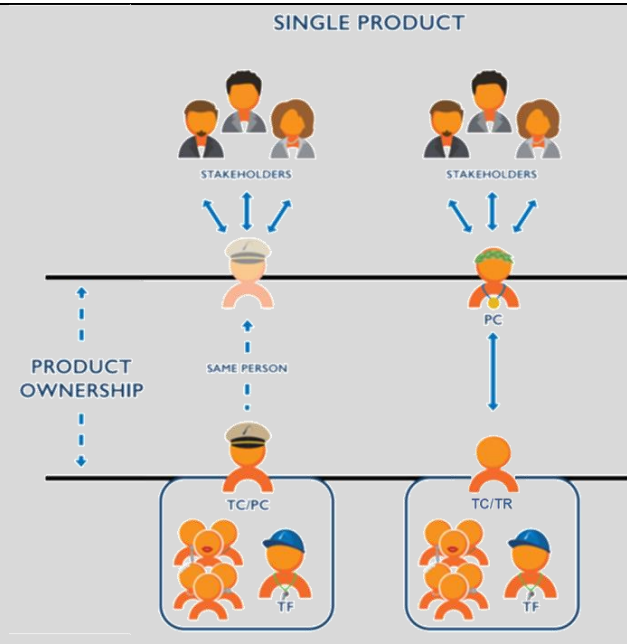
Champions, and others). Many people may be involved in figuring out what **should** be worked on next, but the Team Captain is accountable for the actual decision of what **will** be worked on next. The Team Captain **owns** the Team's **Work Backlog**, and the Team isn't allowed to act on what anyone else says.

- Each Business Owner **owns** his or her **Results Backlog** and is the **only** person in the Organization who decides the prioritization of the Results in the Results Backlog. When there is only one Product, the Business Owner is referred to as the Product Champion, and the Results Backlog is often referred to as the **Product Backlog**.
- Many people may have opinions about what the prioritization **should** be, but the Business Owner is accountable for the actual decision. The Business Owner is accountable for prioritizing the Results Backlog to maximize value to the Business.
- Each Product should have a Product Champion (PC) who embraces multiple accountabilities, in addition to those of being a Business Owner:
    - The PC is accountable to the Product's Clients for the capabilities and features that are (or will be) delivered;
    - The PC is accountable to the Product's Customers for the Delivery Forecast (cost and

schedule) for producing the Product (this usually involves working with the Agile Actuary);

- The PC (as a Business Owner) is accountable to the Business Organization to 'make money' on the project;
- The PC is accountable to the Regulators for 'following the rules'; and
- The PC is accountable to the development Scrum Teams for providing (either personally or through SMEs) the business expertise that is needed.

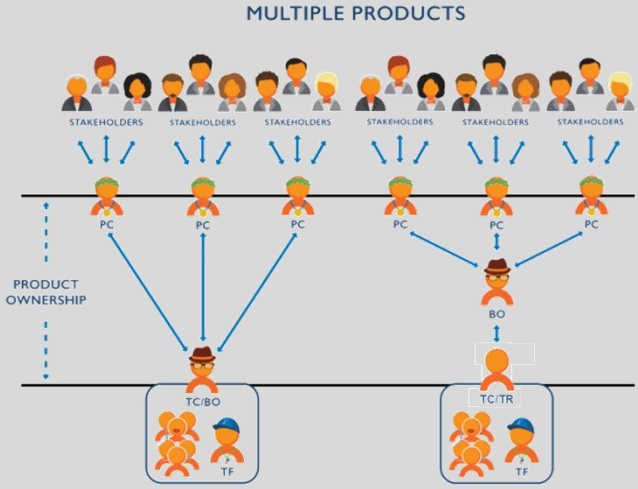
3) When there is a single Product, its PC could also be the Team's Team Captain. In the case the PC *is* outside the Team, the TC could be either physical or virtual. The following figure some variations when there is only a single Product.



4) Since having multiple Product Backlogs (and associated Product Champions) is the most common scaling problem we have (especially when we consider a 'Bug List' as its own Backlog with its own Champion); there usually needs to be a Business Owner who 'sorts out' the conflicting prioritizations.

   When we have only a single Scrum Team, this BO could be the Team's Captain, or external to the Team. If the BO is external to the Team, the BO interfaces with the Team's Team Captain (either physical or virtual). In either case, it makes no difference from the Team's perspective; they are completely focused

on what their Team Captain (physical or virtual) wants them to do Now and Next.



5) The Agile Actuary is an advisor who qualitatively and quantitatively evaluates options and risks, and produces the Delivery Forecast (predicted Dates and Dollars) based on observable Reality. The Agile Actuary knows what data to collect and understands, trusts, and believes the data. The AA's mantra is *"don't tell me a story, show me some data!"* We have found that many Organizations, Business Owners, and Product Champions like to tell stories rather than use data, and that's why they need an Agile Actuary along for the ride. Here is some more advice about the Agile Actuary:

- The Agile Actuary helps Business Owners and Product Champions use data to make 'tough decisions' about options and risks, and works with them to produce Delivery Forecasts and establish realistic delivery expectations with the Stakeholders. The Agile Actuary may **advise** Business Owners and Product Champions about 'what the numbers mean', but they (the 'Owners') make any decisions that need to be made.
- The Agile Actuary usually acts as an appendage of a Product Champion; by combining a Product Champion with an Agile Actuary we get a Delivery Owner (DO) – who 'owns' a delivery of a particular Product to its Stakeholders. Sometimes, when a Business Owner works with an Agile Actuary, we call the combination a Business Manager (BM), who is driving prioritizations with data.
- When multiple Scrum teams are collectively driven by a single Business Owner (acting as the Team Captain of a Leadership Team, for example), the Agile Actuary is usually a member of that Leadership Team who can be used as a Subject Matter Expert to help evaluate risks within the organization.
- The Agile Actuary is an expert in qualitative and quantitative analysis to support decision-making. The Change Agent suggests options for improvement of processes, practices, and

organizational design. We expect the two of them to work together to analyze these options in order to provide good advice for the people playing Product Ownership roles.

6) Since the Scrum Team's basic job is to *"do its due diligence and use an appropriate Standard of Care to do its work, without lollygagging or gold-plating,"* it is inappropriate (in fact, it is probably harmful) for the expectations created by the Delivery Forecast to be available to Team Members (with the possible exception of the Team Captain, but even that could be risky).

   Measuring the Team to help it learn more about how it could improve its work could be a good thing; however, if the Team knows that it is considered slow, expensive, late, or otherwise feels pressure to 'improve', this will **always** create a tension to compromise the appropriate Standard of Care. This **cannot** be allowed…

Ok, this is enough about Product Ownership for now. You will continue to find other responsibilities for these Product Ownership roles as you continue reading this Handbook.

Artifacts

Backlog

The work of a Team is driven by a Backlog, which is “a list of Work Items that represents everything that anyone interested in the Team‘s Results or Process has thought is needed or would be a good idea.” The Backlog is always a living document, and contains Work Items the Team might do, or Results (Deliverables) Stakeholders might want. There are several different parts of the Backlog, as we can see in Figure 6.

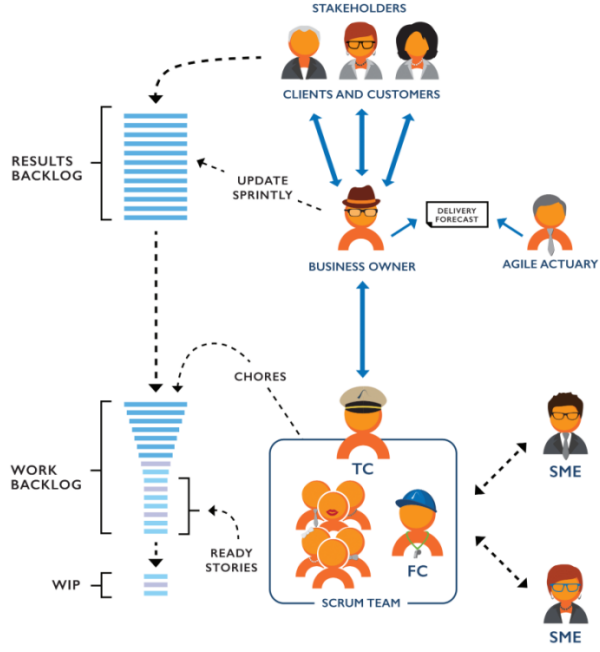


Figure 6: Parts of the Backlog

Results Backlog

The Results Backlog belongs to the Business Owner *(or a Product Champion, in the case that the Results Backlog represents a single Product and, in that case, the Results Backlog is often called a Product Backlog)*, and is a prioritized list of Results (Deliverables) that the Business Owner hopes to deliver to Clients. It also contains a **Business Goal** that the Business Owner has committed to. The Results Backlog is often accompanied by a Delivery Forecast that predicts when the Results (or Business Goal) will be delivered, and how much they will cost.

- The Business Owner works with the Clients to assure that the Results are what the Clients need. The Results (Deliverables) are often **Epics** — big, chunky, Work Items with ill-defined Acceptance Criteria.
- The Business Owner and Team Captain collaborate to determine which **Stories** (small, well-defined, Work Items) should be extracted from these Epics and 'passed down' to the Team's Work Backlog so that the Team can work on them.
- (if necessary) the Agile Actuary helps the Business Owner produce the Delivery Forecast for the Customers. The Delivery Forecast will normally be updated every Sprint based on data from the **Progress Review** to assure that the Results Backlog and the Delivery Forecast are 'in sync' and represent the realities of development.

Work Backlog

The Work Backlog belongs to the Team Captain, and consists of the Stories that the Team expects to be asked to work on 'soon'. These Stories are of two types:

1) **Capabilities** that the Business Owner wants the Team to produce in order to deliver Results (Deliverables) to external Stakeholders; these are the stories that were 'passed down' from the Results Backlog.
2) **Chores** added by the Team; work the Team must do that does not directly provide deliverable Results (e.g., 'do the dishes', 'clean up the code', or 'train the team'); the Chores typically include the Team's Kaizen.

The Team Backlog is constantly being Refined by the Team *(Backlog Refinement is described later in this Handbook)* so that there is always a subset of the Team Backlog that is immediately actionable, or **Ready** to be worked on. Typically, a Story is Ready when the Team knows the Story's Acceptance Criteria, what Standard Of Care to use for the Story, and which Subject Matter Experts they will be working with.

The Work In Progress (WIP)

The Work In Progress belongs to the Team and consists of the Stories the Team is actively working on, along with their associated **Tasks**. The Team often uses a **WIP Limit** to help

manage bottlenecks in development as the Team does its work.

Commentary:

1) You may be asking: *"Where are the Product Backlog and Sprint Backlog? That's what I'm used to seeing!"* Well, they are there too. In $Scrum_H$ we emphasize the Continuous Flow of work involved in Development, and think of the Sprint as being an interrupt of that Continuous Flow. In particular, we don't think of the Sprint as a Container of Stories. Unfortunately, our tools often treat the Sprint as if it *were* a Container of Stories… If you have to deal with such a tool, you can do that, with the following definitions:

   - **Sprint Backlog**: The Sprint Backlog consists of the Work In Progress along with the Ready Stories that have been brought into the Sprint, whether they are committed to, forecasted to be worked on, or whatever.
   - **Product Backlog**: The Product Backlog consists of the Results Backlog along with the portion of the Work Backlog that has not been moved to the Sprint Backlog.

2) Also, you may have noticed that there are three types of Work Items that are mentioned: Epics, Stories, and Tasks, with the following basic descriptions:

- **Epics** are big things that Business Owners (including Product Champions) talk to Stakeholders about. Basically, an Epic is 1) something that the Stakeholders think of as a single unit of work, or deliverable, and 2) is too big for the Team to get to Done all at once;
- **Stories** are small things that are either 1) passed down to Teams (Capabilities) or 2) created by the Teams themselves (Chores). Basically, Stories are things that the Team thinks of as a single unit of work that can be Done by working with SMEs; and
- **Tasks** are things that the Team creates in order to organize and do the work.

We have found that people like to argue about what these words mean and, in general, worry about them **way** too much. The main importance of these words is that they give the Team a vocabulary to use when discussing the size of the work – these sizes become the focus of conversations.

For example, there may be a big mismatch between the 'size' of the Results the Stakeholders want, and the 'size' of the Capabilities the Team can consume. For example, a Result the Stakeholders want may be like *"I want a new House",* and the Capabilities the Team is capable of dealing with may be like *"Paint this wall"*; it can easily require thousands of Stories to implement a single Deliverable Result.

3) The Business Owner (or Product Champion) and Team Captain are required to collaborate in order to extract and 'pass down' Stories to the Team, and they may not be capable of this without additional analytic help – which must come from somewhere. This is actually a scaling problem because the system, as it stands, does not have the capacity to consume the work they are being asked to consume. Anyway, you need to talk about it…

4) The Backlogs define a flow of work from the Stakeholders to the Team; this flow should be viewed from a lean perspective. From that Point of View, the Work Backlog is the Inventory that is feeding the WIP. Since Inventory should be minimized, the Work Backlog should be replenished slowly, and only because there is a 'pull' caused by work in the WIP getting to Done. In other words, Work flows because getting work to Done 'pulls' work down to the Team, not because the Stakeholders want to 'push' work down to the Team. Pull not Push…

Increment

As the Team completes each Capability Story it produces Results (proposed and partial Deliverables) that require meaningful feedback from appropriate Stakeholders. The collection of accumulated Capability Stories is called the **Increment**; the increment should always be Done and in reviewable condition.

Commentary:

The fact that the Increment is composed of *"proposed and partial Deliverables"* does not necessarily mean that we are building our Deliverable a piece at a time – that would be incremental, not agile, development. It does mean that the Increment is capable of eliciting meaningful feedback from Stakeholders, and that this feedback helps the Team, Business Owner, and Product Champions determine what to do next.

One of the things they could do next is decide to deliver the Increment. Another could be to add something to the Increment. Another could be to modify the Increment. Another could be to completely change directions and build something else. You never know…

There is an excellent video on this topic from Henrik Kniberg, *"Making sense of MVP (Minimum Viable Product)"*, which can be found on YouTube. We recommend that every Product Owner, of any type, watch it.

Definition of Done

For the Increment to be Done, *each* Story that was involved in producing the Results included in the Increment needs to be Done. For any particular Story, its definition of Done is the *"shared understanding, between the Team and the Story's Stakeholders, of what it means for the Story to be complete."* This is not as simple as it seems, as there are

many potential Stakeholders for a Story: there are Clients who are interested in Capabilities, Team Members who are interested in Chores, Customers who are interested in Costs, and you can't forget the Regulators, whose needs must be satisfied, as well.

Typically, this understanding has two parts:

- **Acceptance Criteria (AC)**: the objective criteria the Team will use to determine whether or not the Story achieves the Value the Stakeholders expect or need – this is what will (or could be) reviewed by the Stakeholders. In software development, for functional Stories, this is often a description of an Acceptance Test. If there is no objective test to use, the Story is usually time-boxed – the Acceptance Criteria is met when the time-box expires.
- **Standard of Care (SoC)**: the description of the objective criteria the Team will use to determine whether or not work on the Story meets quality, process, or regulatory standards and constraints – as opposed to providing Value to Clients. The Standard of Care is used to 'guarantee' that the Results meet Regulatory Constraints and have sufficient technical quality to be releasable.

A Story's Acceptance Criteria are usually negotiated and discussed with its Clients or surrogates, while the Standard of Care is a bit more complicated. Some parts of the SoC could come from outside the Organization, through

contractual obligations, a Regulatory body, or some such. Other parts of the SoC could be part of the conventions, standards, or guidelines of the Organization, and some could be developed by the Scrum Team itself. In any case, each Story the Team works on *must have* a Standard of Care and, when working with software, all Scrum Teams working in the same codebase, doing the same or similar Stories, must use the same Standard of Care for them.

Commentary:

Many Stories can have the same Standard of Care; these Stories are said to be of the same **Storyotype**. Storyotypes can be very useful because they capture common Standards of Care that can be reused as necessary. Some Organizations or Scrum Teams use the same Storyotype for all stories; for example, when all Stories produce Code for the same Product. In general, though, there need to be many Storyotypes to cover all the different types of work (Stories) the Team does. When using Storyotypes, we expect that they will be improved over time in order to provide higher quality Results more efficiently and effectively.

In any case, no matter what the Story, the Team makes the determination whether or not a Story is Done; this is not something that occurs outside the Team. In fact, the determination that the story is Done is itself part of the Story. This is important, and an extension of the requirement that the Team is self-contained – that the

Team has all the skills and knowledge to get its work completed.

In a Regulated environment that requires formal Inspections, there could be a 'final Inspection' that happens outside the Team (and is thus not part of the Story's Done), but this final Inspection does not relieve the Team from doing an internal inspection to verify that it is Done with the Story.

Commentary:

Since the Team Captain is accountable for the Team's Results, it is quite common for the Team to have a 'ceremony' that indicates that the Team Captain is taking ownership of a Story's Doneness. For example, when using a Storyboard, the Team Captain may be the Team Member who physically moves the Story to the Done column…

Delivery Forecast

The Agile Actuary helps the Business Owner and Product Champions develop Delivery Forecasts, which are artifacts that predict Delivery Dates and Costs for the Results being produced. Often, the Forecasts are predicting dates and costs for the completion of **Product Goal**, which are determined by Product Champions, and the **Business Goal** determined by the Business Owner.

Delivery Forecasts are optional artifacts, and prove to be necessary in many contexts. There is no established content or form for the Delivery Forecasts; each Organization must determine its own needs and development practices.

Commentary:

The Delivery Forecast can be something as simple as a predicted Delivery Date based on Velocity, to something as complex as a Project Plan comparing Baselines and Actuals to calculate Earned Value Metrics that support predictions of Cost, Scope, and Schedule. In any case, there are two things to keep in mind:

1) The Delivery Forecast should be based on Data, not hope. The Scrum Team telling you they will deliver on time is not Data. Telling the Team what it will deliver, and when, is not Data.
2) You should be very wary of making the Delivery Forecast visible to the Scrum Team (except, perhaps, the Team Captain), as it can be harmful. If the Team knows what the delivery expectations are, there are two 'bad' things that could happen:
    - If Team Members feel that they are 'beating expectations', they may slow down (as a consequence of Parkinson's Law), causing problems later on, or

- If Team Members feel that they are ‘too slow’, they will often try to speed up – and they will likely compromise the Standard of Care to do so.

Process

The Scrum Process has a fairly small collection of ceremonies, discussions, activities, or meetings. The purpose of these ceremonies is to have collaborative conversations, do work, sync-up, and make decisions. The overarching purpose of the Scrum Process is to help the Team surface useful information so that they can be agile by adapting to, and exploiting, opportunities to achieve Results. See the following diagram:

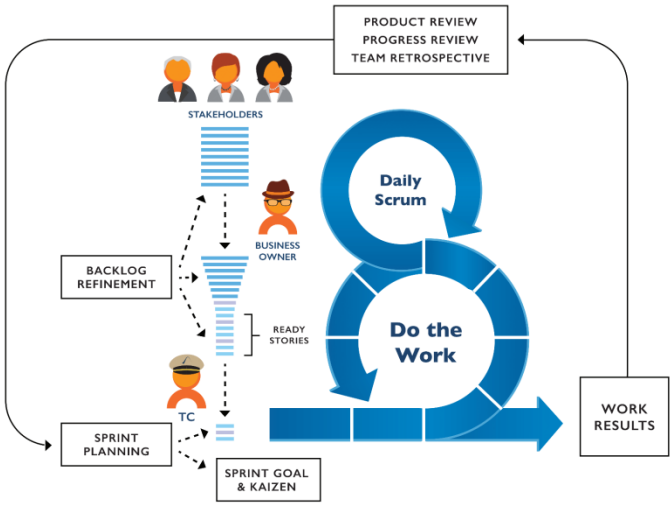


Figure 7: The Scrum Process

At its most basic, the Scrum Process can be thought of as a **Continuous Flow** of production work, with an interrupt every Sprint to allow for Feedback, Improvement, and Re-Planning.

Continuous Flow

Let me first describe the Continuous Flow, what the Team actually does day-by-day. Every day, the Scrum Team does the following things.

Daily Scrum

The Team has a Daily Scrum, when it self-organizes and plans its day. This discussion should take no more than 15 minutes. Typically, the Team discusses 1) its current state; 2) its progress towards its Sprint Goal, its Increment, and its Kaizen; and 3) the Impediments it's running into. This allows the Team to figure out what it hopes to do today. The result of the Daily Scrum is that the Team Members leave the Daily Scrum 'walking with a purpose' — they know what they're doing next — even though it could change in just a few minutes.

Commentary:

Many people think that the Daily Scrum is all about the three questions: 1) What did you do? 2) What are you planning to do? and 3) What's standing in your way? Actually, these questions are only one of many ways to help determine the Team's current state, and many Teams do it in different ways. For example, some Teams talk about the state of the work, rather than what people are working on…

What is important is that the Team Members use the Daily Scrum to understand 1) where they are, 2) where they want to be, and 3) what they are going to do today to help them get there…

Do Work

The Team Does Work, which means:

- The Team swarms with its Subject Matter Experts in order to get Stories that are in the WIP to Done.
- As Stories get to Done, the Team Captain determines which Ready Stories should be 'pulled' from the Work Backlog to the WIP,
- The Team makes advances on its Kaizen, and
- The Team discusses and removes Impediments.

Maintain the Results Backlog

The Stakeholders, Business Owner, and appropriate Product Champions (if necessary) maintain the Results Backlog, by

- Adding/Removing/Changing the Deliverables in the Results Backlog, and
- Ordering/Re-Ordering the Results Backlog.

Backlog Refinement

The Team does Backlog Refinement (as necessary) with its Subject Matter Experts and/or Product Champions, in order to assure that there are Ready Stories in the Work Backlog. These discussions are more-or-less continuous, with the total amount of Refinement usually taking no more than 10% of the Sprint. Refinement includes:

- 'Pulling' Capabilities from the Results Backlog to the Work Backlog, usually when the Work Backlog gets a 'little low';
- Adding necessary Chores to support those Capabilities;
- Ordering/Re-Ordering the Stories (both Capabilities and Chores) in the Work Backlog; and
- Refining the Stories near the *top/front* of the Work Backlog to make them Ready to be worked on. A Ready Story is one that is immediately actionable by the Team, and making a Story Ready typically includes:
    - Determining what Done means for the Story; defining and finalizing its
        - Acceptance Criteria, such as a functional test or time-box, and
        - Standard of Care, defining Quality (and other) Constraints,

- Splitting ‘large’ Stories into ‘small’ Stories as necessary,
- Sizing the Stories, if necessary, to support gathering data to update the Delivery Forecast, and
- Determining which Subject Matter Experts need to be available to work on the Story.

Commentary:

1) Since we are thinking of a Scrum Team’s work as a continuous flow with a Sprintly interrupt, we are not overly upset if a Story’s work overflows to the next Sprint. We just need to leave the Story in a ‘good state’ so that we can restart work on it when the next Sprint starts, should we decide to do so.

   The Team is also not worried about how much will get completed in the Sprint. The Team is focused on getting work to Done without lollygagging or gold-plating; it is **not** focused on the end of the Sprint. When the Sprint ends, the current Increment (collection of Done work) gets reviewed. The Team is always looking for both useful feedback and ways to improve.

2) If something really, truly, has to get completed by a given date, there are two choices:

   - If you can, make sure there is plenty of buffer built in so that the Story’s Standard of Care does not need to be compromised to meet the date. The

basic guidance is: *"if you want it 'Done' soon, you better start it early…"* or

- If there was no chance to start work on the Story early enough, the Team Captain, Business Owner, Product Champion, and Stakeholders must agree on a new, compromised, definition of Done for the Story that allows the Team to get the Story to the new Done on time.
    - This typically requires an agreement to do the Story 'correctly' later, to the proper definition of Done.
    - For example, I (Dan) worked on Retail Software, and we often had to get a feature demonstrable for a Trade Show, but couldn't get it actually finished in time. So, we switched to 'Trade Show Done', which was good enough to 'Show and Sell', but not good enough to actually Deliver. This left big messes in the CodeBase that had to be cleaned up later – after the Trade Show – but the compromise was owned and sanctioned by the proper Product Ownership (the CEO).

3) The Sprint is often used as a common cadence within Organizations to:
    - Coordinate the integration of the Results of many Teams, and
    - Instill good habits and behaviors. Once a Team has matured and developed such behaviors, it

should continue to use the Sprint as a cadence for continuous, ongoing, improvement.

4) In order to minimize inventory for the Scrum Team, which would be recommended when using lean thinking, you should use a limit on the number of Capabilities that are represented in the Work Backlog. You could make a practice of moving a Capability from the Results Backlog to the Work Backlog only when a (refined) Capability has been completed, and all its extracted Stories have been Done. This would be a formalization of what it means to 'pull' work to the Work Backlog, which you should be doing in any case.

Interrupt Each Sprint

At the end of the Sprint the Team has four Ceremonies/Events/Discussions/Meetings in order to have Feedback, Improvement, and Re-Planning.

Product Review

At the end of a Sprint, the Team, the Business Owner, and appropriate Product Champions (if necessary) have a Product Review with Clients in order to obtain Meaningful Feedback for the Team about the Team's Results (proposed and partial Deliverables). This is at most a three-hour discussion for a four-week Sprint, and is usually shorter for shorter Sprints. This discussion is 'owned' by the Business Owner, and consists of discussions driven by questions like:

"What do you like about what we did?", *"What don't you like?"*, *"What would you like to change?"*, and *"What would you like to see next?"* The whole Scrum Team, the Business Owner, the Product Champions, and the Clients should be involved in these discussions.

Commentary:

1) This discussion is about getting feedback on the Team's Results; the idea is for the Business Owner and Product Champions to gain the feedback they need in order to prioritize the Capabilities that the Team will work on in the next Sprint. Sometimes, there are discussions about the work itself, or the Standard of Care, and those discussion are owned by the Team Captain.

   The Product Review is not the only feedback the Team should be getting from your Clients. In fact, we often use the Review as a 'recruitment exercise' to get people to come work with the Team as Subject Matter Experts. I've often used a sentence like the following: *"I can see that you'd like this changed; could you please help us get it right in the next Sprint? Could you send somebody?"*

2) Since a Team can be working on multiple Products, each with its own set of Stakeholders, there may be a need to have multiple Product Reviews – to review multiple Products. The main point is that the Team must receive meaningful feedback about its Results

every Sprint; it is not about the meeting, it is about the feedback…

For example, I (Dan) once had two major Clients (call them WalWay and SafeMart) who received their own tailored versions of our Product. It would be ludicrous to think we could get them in the same room at the same time, and we tried two different strategies at different times:

- We held two different reviews for the Clients, and
- We did a single review, and included the Sales-persons who were their Product Champions'

In the second case, the salespersons had to work as go-betweens, and this often led to additional Reviews (as Stories) within the next Sprint. Frankly, we never could decide which method produced better results…

Bottom Line: You've got to think about your Product Reviews; you've got to figure out how best to do them. They are not simply about 'following the process', they are about 'getting meaningful feedback'. You may need to have multiple meetings, do some of them as Stories within the Sprint, or whatever, in order to get this feedback.

In any case, you should keep them short; the total of ALL your Product Reviews should add up to no more than the appropriate amount of time (2-3 hours), and keeping

them short will likely help turn them into the SME recruitment exercises we mentioned above.

Progress Review

The Team Captain, Business Owner, and Product Champions (augmented by the Agile Actuary) have a Progress Review with Customers about 'how the work is going' — information to help update the Delivery Forecast(s), evaluate risks, and set expectations with the Customers. This discussion should take no more than one-hour for a four-week Sprint, and is usually shorter for shorter Sprints. This discussion is 'owned' by the Business Owner, and leverages the expertise of the Agile Actuary. The discussion is largely about risks, Dates, and Dollars. Discussions involve questions like: *"How fast are we going?"*, *"When will we be finished?"*, *"What's going to jump up and bite us?"*, and *"How do we adapt Cost, Scope, and Schedule to match the realities we see?"*

***Commentary**:*

Even though the Agile Actuary is heavily involved in this discussion, any changes in plans are owned by the Business Owner (or Product Champions). The Agile Actuary understands the relationship between Cost, Scope, and Schedule – and may offer options about how to keep them balanced and in sync – but any decisions about what to do (change Scope, get additional Funding, and so on) belong to Business Ownership.

Since the Team could be working on multiple deliveries with multiple Delivery Forecasts and different Customers, there may need to be multiple Progress Reviews. As with the Product Reviews, you've got to figure out how best to do them. Because this is a Product Ownership thing, and need not involve the rest of the Team, it is easy to do them during the Sprint. In any case, keep the total time of the Progress Reviews short, with a total amount of time maxed out at an hour. Basically, this is just an informational meeting, with the Agile Actuary supplying 'expert' advice about Dates and Dollars that will lead to further discussions about what to do…

Team Retrospective

The Scrum Team has a Retrospective at the end of a Sprint (facilitated by the Team Facilitator with possible support from a Team Coach) in order to discuss and agree upon ways they could improve their Practices, teamwork, environment, or Organization for the next Sprint. This Discussion should take three-hours for a four-week Sprint, and is usually shorter for shorter Sprints. This discussion is 'owned' by the Team Facilitator, and is driven by the questions: *"What did we do well"* and *"What could we improve?"* One result of the Retrospective should be to propose a Kaizen (or two) that could be worked on in the next Sprint.

Commentary:

The Retrospective is about Team improvement. Scrum expects a Team to do Kaizen, which is the process of continuous improvement, and to do this the team selects a Kaizen (one single improvement to work on) every Sprint. Technically, the Sprint's Kaizen is chosen during Sprint Planning, but identifying potential Kaizens is part of the Retrospective.

The last of these discussions 'between the Sprints' is actually the beginning of the next Sprint...

Sprint Planning

Sprint Planning is a discussion the Team has at the beginning of the Sprint in order to:

- Establish **Sprint End** (when the interrupt for the four Ceremonies will take place),
- Select a **Kaizen** to accomplish,
- Make sure there are enough **Stories** Ready or 'In Progress' so the Team can get back to work, and
- Determine a **Sprint Goal**.

Commentary:

1) Sprint End usually consists of dates and times for the Product and Progress Reviews. However, the Sprint End could also be state (or milestone) based. For

example, the Sprint End could be defined by statements like the following:

- We'll have a review as soon as Randy can get a meeting with the Clients; or
- Let's review this thing after we finish these first four Stories.

2) There is no reason that all four Ceremonies need to be held at the same time; they just need to be held at least once a Sprint.
   - We have had teams that Retrospected every week, but had their Reviews every two weeks. It's about getting frequent meaningful feedback, not following a process.
   - When there are multiple Products involved, the Team will probably need multiple Reviews (both Product and Progress). Often, these Reviews will have to be held as Stories in the next Sprint.

3) Sprint Planning can be accomplished in many ways: some like to get 'just enough' Stories as defined above; some like to fill the Sprint, some like to do two-pass planning; some do one-pass planning; and so on. Sprint Planning is certainly a 'self-organization thing'… Consequently, the length of Sprint Planning is highly variable. For two-week Sprints we have seen it as short as 15 minutes, and as long as four hours.

In $Scrum_P$, the Team's Sprint Commitment was usually to complete the Sprint Backlog. In $Scrum_G$ and $Scrum_H$, the Sprint Commitment is to a single Sprint Goal. This change in Sprint Commitment was made so that there will be 'wiggle room' for the Team to meet their commitment without being rushed and (possibly) compromise the Standard of Care.

4) Even though the BO (or TC) may present the Team with an objective the BO (or TC) **hopes** the Team will meet, the Sprint Goal is what they are committed to – and it is determined by the Team. The Team can choose just about anything to be its Sprint Goal. Here are a few examples of appropriate Sprint Goals:
    - No head works alone this Sprint (a Goal about Process),
    - Bring the new people, Joe and Gina, up to speed (a Goal about the People),
    - Clean up Module ABC (a Goal about a Chore),
    - No 'cheating' on the Code Reviews (A Goal about their Kaizen), or
    - Have a releasable version of <Buy an e-Ticket> (a Goal about the Product).

Whatever the Sprint Goal is, the Team is actually committed to it; they will do 'anything they have to do' in order to meet it. The Sprint Goal is not aspirational – it is

not a hope or dream – it is an actual commitment; it is something *"that will be met within the Sprint".*

These are the ceremonies that happen every Sprint. Once a Sprint starts, we expect the Team to do work until the end of the Sprint, but they sometimes need to stop early.

Stopping a Sprint

There are two ways the Sprint can be stopped before its planned end.

Sprint Cancellation

The Team Captain may cancel a Sprint at any time, often because the Business Owner requests it. The Reason for this cancellation is usually because the BO's (or TC's) objective for the Sprint isn't going to be met, or the objective is no longer what is needed. In any case, the Sprint's work is evaluated to see what can be kept, and whether or not a re-planning is called for. This is not considered abnormal, but is simply an agile reaction to something that has happened. It should not be considered a 'bad thing' – it is merely a 'thing.'

Abnormal Termination

Not only can the Team Captain cancel a Sprint at any time, but the Team Facilitator can stop the Sprint at any time at the behest of the Team – this is called an Abnormal Termination. The Abnormal Termination has been a part of

Scrum from the very beginning, and it has always been a very important part of a Team's self-organization – the Team needs to have the power to cancel its own Sprint. If an Abnormal Termination is invoked, the Sprint is re-planned and all work in the Sprint is discarded, making it a 'very big deal'.

Commentary:

The Abnormal Termination is rarely invoked; it is usually used as a threat. The threat of an Abnormal Termination alerts management (in a very noticeable way) that something is going wrong and needs to be addressed. The most common use of the threat of the Abnormal Termination is when Product Ownership is doing something the Team considers 'out of line'. In essence, the threat is saying *"if you don't play nice, we'll go on strike."*

In any case, Teams should not threaten to use the Abnormal Termination unless they are willing to actually use it. The Abnormal Termination is quite disruptive, but does give the Team the ability to re-start a Sprint with a clean slate.

The Abnormal Termination was inexplicably left out of the Scrum Guide. We believe this was a mistake, and put it back in in the Handbook. This is the only substantive difference between the Scrums described in $Scrum_G$ and $Scrum_H$.

Scrum Glossary

***Scrum uses many terms that are confusing, so we present this Glossary**, which contains terms that are found in the Scrum Guide, the Analysis of the Scrum Guide, the Handbooks, and other sources. It is not a comprehensive list, but it's a start.*

Abnormal Termination | A stopping of the Sprint by the Scrum Master at the behest of the Team. This is a 'self-organization thing' and is often threatened but seldom invoked – it is usually used by the Team as a way of saying *'you didn't play nice, so we are going on strike.' (see Self-Organized)*

Acceptance-Based Story | A Story whose Acceptance Criteria does not include a Time-Box; the Time an Acceptance-Based Story takes is a byproduct of getting to Done*. (see Time-Boxed Story, Acceptance Criteria, Standard of Care, Done)*

Acceptance Criteria | The description of the objective criteria the Team will use to determine whether or not a Story achieves the Value it represents; the part of Done that is separate from the Standard of Care and the General Agreements. *(see Time-Boxed Story, Acceptance-Based Story, Standard of Care, General Agreement, Done)*

Accountable/Accountability | An accountable person is a person who is ultimately answerable for an activity or decision; the accountable person can be *held to account* for the results of the activity or the making of the decision. There can only be one person accountable for any particular activity or decision. A person is *always* accountable for their commitments. *(see Commitment, compare to Responsible)*

Actionable Story | Synonym for **Ready Story**. *(see Ready Story)*

Agile | 1) Having a decision-making turning radius tight enough to deal with complexity; 2) An umbrella term that encompasses a family of processes known for being 'agile' (Scrum, eXtreme Programming (XP), DSDM, Crystal, Feature Driven Development, Agile2, Kanban, and others). *(see agility)*

Agile Actuary | A professional who does qualitative and quantitative analysis to help Business Owners and Product Champions mitigate risk, support decision-making, evaluate options, and produce the Delivery Forecast. *(see Business Owner, Product Champion, Product Ownership, Delivery Forecast)*

Agile Analysis | Any iterative and incremental method or practice that produces Epics and/or Stories for the Backlog.

Agile Coach | A person who helps people improve their 'agile' skills. The skills might be technical or non-technical, and improvements are typically achieved through training and/or mentoring. The Agile Coach often works with the Team Facilitator to: 1) help the Team Members understand, implement, and improve their use of Scrum and agility; and 2) help the Team identify its Kaizen every Sprint. Every Team must have access to an Agile Coach as needed; this usually requires one Agile Coach for every 2-10 Teams, with five being the 'sweet spot'. *(see Scrum Mastering, Team Facilitator, Kaizen, Team Facilitator)*

Agile Improvement | When an Organization improves its agility, the 'Improvement Stories' compete with other work for prioritization. *(see Improvement Story, compare to Agile Transformation,)*

Agile Transformation | An Organization transforms when it makes radical changes that are more than simply improvement; and Organization in transformation has the Transformation as its main goal, any other work is done simply to determine if the transformation is working. *(compare to Agile Improvement)*

agility | The act of adapting to, and exploiting, the realities we see, as opposed to being predictive or plan-driven. Agility has two primary facets: Physical Agility and Mental Agility. *(see Physical Agility, Mental*

Agility)

Agreement-Based Planning | An alternative to Capacity-Driven methods for Sprint Planning. With Agreement-Based Planning Product Ownership and the Team work together to add stories to the Sprint, one at a time, until the Team agrees that the Sprint is 'full'. Things like who is available, technical debt, and the story's definition of Done, are all taken into account, as they impact what the team can and cannot do. *(see Sprint Planning, compare to Capacity-Driven Planning)*

Alignment | People are in alignment about something when their understandings of the essence of that something are consistent; we often say they *"are on the same page".* The primary goal of Product Ownership is to have Goals, Strategies, Plans, and Action in alignment from top to bottom – from Stakeholders to Production Team Members. *(note: this concept was originally described by General Helmuth von Moltke (the elder) (1800-1891), and the German term for this concept is "Auftragsklärung")*

Analysis Story | A Story that finds Items or Stories; a Story that conducts Agile Analysis. The most common Analysis Stories find functional Stories by one of various methods (working with SMEs, studying Change Requests, conducting Usability Analysis, etc.); however, there can also be risk analysis Stories (finding risks and fears that need be dealt with), process analysis Stories (finding process improvements), and so on. *(see Agile Analysis, Backlog Refinement)*

Architecturally Significant Story | A Functional Story that causes the Team to make one or more architectural decisions, which is then validated by the fact that there is existing, working functionality using the decisions. *(see Functional Story, Architecture)*

Architecture | The collection of decisions about how a system will be built – from Grady Booch in the early 80's.

Back Burner | Synonym for **Work Backlog.** *(see Work Backlog)*

Backlog | An ordered list of Backlog Items to be worked on. *(see Backlog Item, Product Backlog, Work Backlog, Results Backlog)*

Backlog Item | A single unit of work on the Backlog, an Item is either an Epic or a Story. *(see Backlog, Epic, Story)*

Backlog Maintenance | Synonym for **Backlog Refinement.** *(see Backlog Refinement)*

Backlog Refinement | The act of preparing a Backlog to make it ready for Planning. This includes: extracting Stories from Epics, refining Stories to make them Ready, and ordering the Stories. There could be Refinement Stories, Sessions, or both. Synonyms include Grooming, Backlog Maintenance, and Story Time. *(see Backlog, Planning, Order, Epic, Story, Ready)*

Bug | A simple change that does not require an acceptance test; examples include correcting a misspelling in a dialog box or moving an interface element on the screen. Often used (incorrectly, in my view) as a synonym for Defect. *(see Defect)*

BuildUp | A BuildUp graph is any graph that shows the completion of Backlog as a function of Time.

BurnDown | A BurnDown graph is any graph that shows the amount of remaining Backlog (Items or Tasks) as a function of Time. Many people and tools use BurnDowns, but they have been largely deprecated from Scrum as they are inherently predictive, and not agile.

BurnUp | Synonym for **BuildUp**. *(see BuildUp)*

Business Organization | The organization the Team belongs to, or works for. The Business Organization provides services the Team needs in order to exist, such as Human Resources, Logistics, Facilities, Contracting, Training, and so on. The Business Organization is typically referred to as either *"the Business"* or *"the Organization"*, depending on context.

Business Owner | A Product Ownership role that represents a person who is accountable to the Business Organization for maximizing the overall value of Deliverable Results, which could represent one, or many, Products. The Business Owner owns the Results Backlog, and

is also called the Business's Product Owner. *(see Product Ownership, Business Organization, Results Backlog)*

Business Value (BV) | A property of an Item that simply indicates that some external Stakeholder wants it; it is very hard to quantify, even though we continue to try to do so.

Business's Product Owner | Synonym for **Business Owner**. *(see Business Owner)*

Cadence | Synonym for **Rhythm**. *(see Rhythm)*

Cancelling a Sprint | The Team's Product Owner (Team Captain) may cancel a Sprint at any time, usually because the Business Owner's objective for the Sprint isn't going to be met or because the objective is no longer what is needed.

Capability | A Story that provides value to an external Stakeholder; an Item that has Business Value. *(see Story, compare to Chore)*

Capacity | An estimate or prediction of the rate that a Team or Organization *will be able* to develop Product; it is often used in Release Planning. Often confused with Velocity and WorkRate. *(see Velocity, WorkRate)*

Capacity-Driven Planning – Any planning method based on an assumption of future Capacity. *(see Capacity, compare to Agreement-Based Planning)*

'Catch' Feedback | Feedback obtained by actively listening to responses from a stakeholder during review of an artifact (such as a completed story or product increment). *(see 'Pull' Feedback)*

Change Agent | A person who helps the Organization adopt, implement, and sustain Scrum, and understand how best to support and work with Scrum Teams – this includes organizational design. There needs to be at least one Change Agent per Organization. The Change Agent is usually also an Agile Coach (with expertise in organizational behavior), and is considered the 'senior Scrum Master' in the Organization. *(see Scrum Mastering, Agile Coach)*

Chief Scrum Master (CSM) | The Team Captain of a Scrum Master Team; we expect to see a CSM for every Pod, sub-Pod, Group and sub-Group. *(see Team Captain, Scrum Master, Scrum Master Team, Pod, Group)*

Chore | A Story that provides value to the Team or Product, as opposed to an external Stakeholder; an Item whose value is other than Business Value. *(see Story, compare to Capability)*

Clean Code | 1) Code that is easy to change: that is extensible, modifiable, and maintainable. 2) Code that has little or no Technical Debt. *(see Technical Debt)*

CleanUp Story | A Story that *apologizes* to the Code Base about something bad that happened, and *promises* to fix it. It usually documents what is wrong and indicates what needs to be accomplished to fix it. A Cleanup Story is a story that tells us where the mess is and what we have to do to clean it up.

Clients | The people who will be using, consuming, or working with the Product/Results the Team is developing.

Co-Located | A Team 'being co-located' means that its Team Members must be close enough together (either physically or virtually) that they can talk to each other within seconds or minutes, rather than hours or days.

Code Complete | A Product Increment is code complete when the development team agrees that no entirely new source code (including automated tests) needs to be added.

Coding Story | A Story that has Code as its primary result.

Commitment | 1) One of the least understood of the Scrum Values; the Team commits to living their Values and doing their due diligence to get Stories Done; 2) an obligation a person has promised to accomplish, 3) often used as a synonym for Sprint Commitment. *(see Scrum Values, Done, Sprint Commitment)*

Complex Problem | Any problem that is too complicated for any one person to fully grok (understand deeply).

Conflict Resolution Process | A process that resolves issues resulting from disagreements, disputes, violations of Team Norms or Values, and so on. A Conflict Resolution Process should: 1) be fair and legal; 2) have an escalating series of steps (negotiations, agreements, discussions, sanctions, punishments, etc.) that 3) ultimately lead to removal from the Team, expulsion from the Organization, or worse. *(see Team Norms, Team Values)*

'Continuous Development' Strategy | A development strategy for a Sprint where the Backlog represents a Continuous Flow of work for the Team, and the Sprint simply defines the duration until 'inspecting and adapting' the Done Results. *(compare to 'Small Project' Strategy)*

Continuous Flow | 1) Moving products through a production system without separating them into lots; 2) In Scrum, doing all work on an as-needed, just-in-time, basis, with each chunk of work being completed without interruptions. The work the Scrum Team does every day includes the Daily Scrum, Production, Maintaining the Results Backlog, and Backlog Refinement.

Continuous Planning | Just-in-Time planning that supports the 'Continuous Development' Strategy by providing the agility required to meet the needs of a chaotic requirements environment. The Team Captain's decision-making inherent in Continuous Planning is often informed by GOs that were provided by a Business Owner. *(see Planning, 'Continuous Development' Strategy, Team Captain, GO, Business Owner)*

Coordinator | In a Team Swarm, the Coordinator is the Team Member who is 'in charge' of the Story being worked on. *(see Team Swarm, Stay-At-Home, Swarmer)*

Courage | Team Members must have the courage to make reality visible, do the right thing, and work on tough problems. *(see Scrum Values)*

Crew Chief | A journeyman who directs work at a work site; this term is used to refer to a **Team Captain** when the Team Captain is clearly also a member of the DevTeam and is also Scrum Mastering. *(see*

Product Ownership, Scrum Mastering, Business Owner, Team's Product Owner, Squad Leader, Team Leader, Team Representative, DevTeam)

Cross-Cutting Workgroup | In a **Team-of-Teams**, a Cross-Cutting Workgroup is a Virtual Scrum Team, consisting of Team Members from multiple Teams, that has a specific mission or problem to solve. *(see Team-of-Teams, Virtual Team)*

Cross-Functional | Synonym for **Self-Contained**. *(see Self-Contained)*

CSM | Acronym for **Chief Scrum Master**. *(see Chief Scrum Master)*

CURB | An acronym that stands for Complex, Unknown, Risky or Big. CURB is a mnemonic to help you remember how to determine if a functional Story is actually an Epic. *(see Epic)*

Customers | The people who will be paying for the Product, or Results.

Daily Scrum | Scrum's daily 'inspect and adapt' ceremony; its purpose is to collect 'today's reality', compare it to the Sprint's Backlog, Goal, Objective, and Kaizen, and 'inspect and adapt' to determine what to do today. *(see Kaizen, Sprint Backlog, Sprint Goal, Sprint Objective)*

Daily Standup | Synonym for **Daily Scrum.** *(see Daily Scrum)*

Defect | 1) Anything about a Product that is seen as 'wrong' by a Stakeholder; defects usually result in a new Item being added to the Backlog; 2) A bug, failure, flaw or error in an application or program that results in unexpected, incorrect or unintended ways. *(see Bug)*

Definition of Done (DoD) | 1) Synonym for **Done**; 2) often used erroneously as a synonym for **Standard of Care**. *(see Done, Standard of Care)*

Definition of Ready (DoR) | Synonym for **Ready.** *(see Ready Story)*

Deliverable Results | Scrum Teams and Organizations produce (proposed and partial) Deliverable Results for Stakeholders iteratively and incrementally; these Deliverable Results are often called Product. *(see Product, Increment)*

Delivery Forecast | The Delivery Forecast is 'owned' by the Product Champion, and predicts Scope, Delivery Dates, and Costs. It should be based on Data, not Hope. It is important to note that the Team saying they will deliver on time is not Data, and telling the Team what it will deliver, and when, is not Data. *(see Product Champion)*

Developer | Synonym for **Team Member**. *(see Team Member)*

Development Team | The subset of the Scrum Team that is actually producing Results; this may, or may not, include the Scrum Team's Captain and Facilitator. *(see Scrum Team, Team Captain, Team Facilitator)*

DevTeam | Synonym for **Development Team**. *(see Development Team)*

Do Work | Every day during the Sprint, the Team is said to 'Do Work', which includes: 1) The Team Swarms with its Subject Matter Experts in order to get Stories that are 'In Progress' to Done; 2) As Stories get to Done, the Team Captain determines which Ready Stories should be moved from the Work Backlog to become Work In Progress; 3) The Team makes progress on its Kaizen; and 4) The Team discusses and removes Impediments. *(see Subject Matter Expert, Team Captain, Ready Story, Work Backlog, Work in Progress, Kaizen, Impediment)*

DoD | 1) Acronym for **Definition of Done**; 2) often used as a synonym for **Standard of Care**. *(see Done, Standard of Care)*

Done | 1) A piece of work is Done when it satisfies the (pre-existing) agreement between Stakeholders and Developers about what it takes for the work to be complete; 2) Done for a Story usually consists of Acceptance Criteria (which could include a TimeBox), a Standard of Care, and General Agreements; 3) A Software Increment is Done when *"the increment consists of thoroughly tested, well-structured, and well-written code that has been built into an executable and that the user operation of the functionality is documented… This is the definition of a Done increment."* (per Ken Schwaber, *Agile Project Management with Scrum*, Microsoft Press, 2004, pg. 12). The concept of Done has often been extended to Epics, Sprints, Releases, and so on… *(see Acceptance-Based Story, Time-Boxed Story, Acceptance*

Criteria, Standard of Care, General Agreement, compare to UnDone)

Doneness Agreement | Synonym for **Story Agreement**. *(see Story Agreement)*

Due Diligence | A person, Team, or Organization is doing its 'due diligence' when it is taking necessary steps to avoid harm to people or property.

Effort [for a Story] | The amount of time it actually took to complete a Story (get the Story to Done), and is usually measured in Hours or PersonDays. *(see Done, compare to Size [of a Story])*

EffortPoint | A relative measure of the actual effort it will take to 'do' a Story. Often confused with StoryPoint. *(see StoryPoint, Ideal Engineering Hour/Day)*

Empirical Process | An empirical process is a process based on empiricism, which asserts that knowledge comes from experience and decisions are made based on what is known. *(synonym for agility)*

Environmental Variables | Factors affecting Effort that are not related to the actual Story. These include, but are not limited to, Technical Debt, Organizational Noise, SME Availability, and Team Ability. *(see Technical Debt, Organizational Noise, SME Availability, Team Ability)*

Epic | 1) a Backlog Item that is too Complex, Unknown, Risky, or Big (CURB) for the Team to agree to do all at once; 2) a named Container of other Epics and Stories. *(see CURB)*

Estimation Game | Any of a variety of consensus-based methods of estimating.

Executive Review | A Review for the business to discuss project, process, or people issues. This is not a part of Scrum, but is often necessary for legitimate business reasons. *(see Project Review, Sprint Review, Product Review, Progress Review)*

Exemplar Story | An example Item used as a reference point for Estimation. For example, we could have exemplar Small, Medium,

and Large Stories as reference points for Estimating Story Size or Effort. *(see Effort [for a Story], Size [of a Story])*

Extracting Stories | The Team does Backlog Refinement with its Subject Matter Experts and/or Product Champions, its Business Owner, and its Stakeholders to assure that there are Ready Stories in the Work Backlog. This includes Extracting Stories from the Results Backlog to the Work Backlog. *(see Backlog Refinement)*

eXtreme Programming (XP) | An agile development process whose practices largely focus on the production of Clean Code. *(see Clean Code)*

Feature | Something a software product enables a user to do. *(see Capability)*

Feature Complete | A state of software indicating that no more features need to be added.

Flow Management Team | Synonym for **Pod Flow Management Team** *(see Pod Management Team)*

Focus | Everything the Team does must have a focus, and the Team Members must focus on what is important in everything they do. *(see Scrum Values)*

Forecast | Short for **Sprint Forecast.** *(See Sprint Forecast)*

Freezer | The portion of the Backlog that contains Items that are 'out of scope.'

Fridge | Synonym for **Results Backlog.** *(see Results Backlog)*

Front Burner | The portion of the Backlog that the Team has agreed to work on 'now'. Depending on your 'flavor' or Scrum, this can either be the Sprint Backlog or the Work in Progress (WIP). *(see Sprint Backlog, Work in Progress)*

Full-Time | Each Team Member is full-time; he or she *belongs* to a single Team. However, the Team, because it is self-organized, may 'loan out' one or more of its Team Members to work elsewhere on Virtual Teams or as a Subject Matter Experts. *(see Virtual Team, Subject*

Matter Expert)

Functional Story | A Story that produces working code that has actual, demonstrable user value. A Functional Story's Acceptance Criteria usually consists of a single Acceptance Test, and verifying that the Test passes is part of the Story itself.

General Agreements | The Part of a Story's definition of Done that contains information about which SMEs will be involved, who will be the Story Coordinator, what is 'out of scope' for the Story, and so on. *(see Done)*

GO | acronym for **Guidance and Objectives**. GOs are typically developed by Business Owners to help Team Captains and Product Champions with their decision-making. *(see Guidance, Objective, Business Owner, Team Captain, Product Champion)*

Gold-Plating | 1) The incorporation of costly and unnecessary features or refinements into a product or structure; 2) making changes to a to a well-defined unit of work that are outside of the original agreed-upon scope. *(see Acceptance Criteria)*

Governance | An Organization's Governance Mechanism is the method it uses to manage and make decisions.

Grooming | Synonym for **Backlog Refinement.** *(see Backlog Refinement)*

Group | A Group is a collection of Pods and sub-Groups along with a Group Leadership Team. *(see Pod, Group Leadership Team)*

Group Leadership Team | A Virtual Scrum Team that consists of the Group Owner (as its Team Captain) and its subordinate Pod and sub-Group Owners. *(see Virtual Team, Team Captain, Group Owner, Pod Owner)*

Group Owner | A Business Owner who is accountable for maximizing the Value produced by a Group; the Group Owner is the Team Captain of the Group Leadership Team. *(see Group, Team Captain, Group Leadership Team)*

Guidance | Advice or information, given by someone in authority, aimed at helping other people achieve objectives or resolve problems. *(see Accountable, GO, Objective)*

Hardening Sprint | A Hardening Sprint is a specialized sprint dedicated to stabilizing the code base so that it is robust enough for release; it is often necessary because the Team failed to use an appropriate Standard of Care when it did its work. Using a Hardening Sprint is not recommended, and the need for it should be eliminated by improving engineering practice. If the need for hardening exists, it should be accomplished a little at a time by using Cleanup Stories within Sprints, not by dedicating a complete Sprint to hardening.

High-Ceremony Agility | Any Agile process that has many meeting, artifacts, or practices.

Humor | Everyone needs a sense of humor; if we can't laugh at some of the things we do, we'd have to cry. *(see Team Values)*

Hybridized Agile | An approach to creating a better breed of agile by combining various agile and existing processes; this often leads to Paradigm Induced Blindness. *(see Paradigm Induced Blindness)*

Ideal Effort | The amount of effort it would take to build something if conditions were as they *should* be; there are no impediments of any kind, and you don't require any magic or miracles. This is actually a measure of Size because Ideal Effort would not differ from Team to Team. *(see Ideal Engineering Hour/ Day, Impediment)*

Ideal Engineering Hour/Day | An estimate of actual effort that ignores disruptions or disasters; an Ideal Hour/Day is an Hour/Day that has no interruptions, but does not 'wish away' other Impediments. *(see Ideal Effort, Impediment)*

Impediment | Anything that is causing the Team to not be at its best. These could be fears, risks, or problems.

Improvement Backlog | A Backlog of Improvement Stories; the Improvement Backlog is usually created and 'owned' by a Scrum Master Team. *(see Improvement Story, Scrum Master Team)*

Improvement Story | A Story whose goal is to improve an Organization's or Team's agility. *(see Story, Improvement Backlog)*

Improvement Team | another term for Pod/Group Scrum Master Team. The Improvement Team creates and manages the Pod/Group's Improvement Backlog. *(see Improvement Backlog)*

InBox | Items in the Backlog that have not yet been prioritized.

Increment | The Team completes each Story, it continuously produces Results (proposed and partial Deliverables) that require Stakeholder feedback. These Results are accumulated to produce a Product Increment, and the Increment should always be Done and in reviewable condition. *(see Product)*

Informed Process | An informed process is one in which decision-making is informed by gathering observations in progressive steps. *(synonym for Agile Process)*

Interested Bystander | People who think that they are Stakeholders, but actually 'don't matter' to you, are called Interested Bystanders.

Intraspective | A discussion by the Scrum Team about its Practices or Teamwork that occurs *within* the Sprint: it is often precipitated by an event that 'didn't go well.' *(compare to Retrospective)*

Intrinsic Difficulty | The Intrinsic Difficulty of a Story is inherent in the Acceptance Criteria, and, for a Functional Story, is based on the complexity involved in the design activities themselves and the complexity of the resulting designs and algorithms. *(see Size [of a Story], Functional Story)*

Item | Short for **Backlog Item**, synonym for **Story**. *(see Backlog Item, Story)*

Kaizen | 1) A philosophy of continuous improvement of working practices, personal efficiency, etc, and 2) a single improvement for a person, Team, or Organization.

Kanban Method | A method of organizing and managing the work for delivering services to customers. The main strength of Kanban (from a Scrum point of view) is that its Planning is continuous, which makes it more likely to keep up with reality, and hence more agile.

Kanban Values | The Kanban Values are: Respect, Values, Focus, Transparency, Understanding, Leadership, Agreement, Collaboration, and Flow. *(see Values)*

Kanban(ish) Variant | The Kanban(ish) Variant of Scrum is one in which the standard (batch) Sprint Planning is replaced by continuous Planning throughout the Sprint. $Scrum_H$ is a Kanban(ish) variant of Scrum. *(see $Scrum_H$)*

Leadership | 1) The act of being followed, 2) The interplay between decision-making and team direction.

Leadership Team | synonym for **Group Leadership Team**. *(see Group Leadership Team)*

Lean Principles | Lean Principles focus on creating value while eliminating waste, thus making a Value Stream (process flow) more efficient. Two of the Lean Principles that are built into *good* implementations of Scrum are 'Pull, don't Push' and 'Minimize Inventory.'

Lollygagging | To waste time aimlessly, to waste an excessive amount of time. Note: the 'right' amount of time is called Slack. *(see Slack)*

Management Team | Synonym for **Pod Flow Management Team** *(see Pod Flow Management Team)*

Mental Agility | Having situational awareness and using feedback to make the decisions necessary to be agile. *(see agility, Physical Agility)*

Mission | A short statement of why the Team exists – what it does, for whom, and why – the Mission is often obvious from looking at the Team's name.

Mission Protection | Ongoing, public commitment from the Business Organization to safeguard the Scrum Team's ability to achieve their Mission. *(see Mission, Business Organization)*

Modern Scrum | Any Scrum that is consistent with the definition of the *Scrum Guide*. *(see Scrum Guide)*

MTS | Acronym for **Multi-Team Scrum**. *(see Multi-Team Scrum)*

Multi-Team Scrum | A type of Scaled Scrum that involves Teams-of-Teams, and is documented in the *Scrum Handbook: Multi-Team Scrum (MTS). (see Team-of-Teams)*

Objective | A thing aimed at or sought; something that is wanted, but not guaranteed or committed to. *(see GO)*

One Big Thing (OBT) | In any model, any entity in the model should be defined by 'One Big Thing'; its reason for being. If you can't find the One Big Thing for any given entity, then either the model is defective, or you don't understand the model.

Openness | There should be no secrets between/amongst Team Members about things relevant to the work and their ability to do the work; this requires some measure of Psychological Safety. *(see Scrum Values, Psychological Safety)*

Order | Refers to the order of a Backlog; the order that the appropriate Product Owner wants the Items worked on. *(see Priority)*

Organizational Noise | An Organization that empowers and nurtures its Teams is said to be a 'quiet' Organization, while one with many procedures, meetings, interruptions, and the like, is said to be 'noisy.' *(see Environmental Variables)*

Pairing | Pairing is a Swarming Pattern often associated with eXtreme Programming (XP), is when each Story is worked on by two Developers, working side-by-side at one computer, collaborating on

the same design, algorithm, code or test. Many Teams have found it useful to rotate Pairs every 1-2 hours, which is referred to as Polygamous Pairing. *(see Team Swarm)*

Paradigm Induced Blindness | When a person follows a process 'blindly' because the process is so convoluted it just overloads the person's head.

Physical Agility | A Team, Project, or Organization has Physical Agility if its processes provide opportunities to obtain the feedback necessary to enable agility. *(see agility, Mental Agility)*

PlaceHolder Story | A PlaceHolder Story is a Story that represents a 'known unknown' (or contains resources to be used for 'known unknowns'). One of the most common issues for Scrum Teams is what to do about work that it expects to have to do during a Sprint, but doesn't actually know the details about yet, such as fixing bugs in existing systems, or expected sales support efforts. Using PlaceHolder Stories is a form of buffer and is used as part of contingency planning.

Plan of Action | A tentative plan, developed by the Team, of how the Sprint might be carried out. The purpose of the Plan of Action is not to have a plan, *per se*, but to enable the Team to justify to itself that doing the work is possible. *(see Sprint Planning)*

Planning | A session with the whole Scrum Team (including the Team Captain) to decide which Stories should be Done next. *(see Sprint Planning, Team Captain, Done)*

Planning Day | Planning day takes place 'between' Sprints, and includes the Sprint Review (broken into the Product and Progress Reviews when necessary), the Retrospective, and Sprint Planning for the next Sprint.

Pod | A Pod is a Team-of-Teams consisting of Production Teams, sub-Pods, a Pod Management Team, and other (physical and virtual) Teams, that produces similar Products for similar Product Champions. *(see Product Champion, Pod Management Team, Team-of-Teams, Production Team)*

Pod Flow Management Team | A Virtual Scrum Team that has the Pod Owner as its Team Captain, and its (immediate) subordinate Team Captains, Product Champions, and sub-Pod Owners as Team Members. The Pod Management Team manages the flow of work within the Pod. *(see Virtual Team, Team Captain, Product Champion, Pod Owner)*

Pod Owner | A Business Owner who is accountable for maximizing the Value of the Results produced by a Pod; the Pod Owner is the Team Captain of the Pod Flow Management Team. *(see Pod, Team Captain, Business Owner, Pod Management Team)*

Potentially Releasable | Synonym for **Done**. *(See Done)*

Potentially Shippable | Synonym for **Done**. *(See Done)*

Priority | The priority of an Item is based on how important it is. Normally, the importance of an Item would be based on its Business Value, but in Scrum, an Item's priority is determined by when it will be undertaken (where it is in the Backlog), not by how valuable it is. *(see Order)*

Product | 1) (in Scrum) Something that a Team or Organization produces for delivery to its Stakeholders; a Product consists of Deliverable Results; 2) (in popular use) A specific marketable/ sellable/usable unit, such as 'website ABC' or the '123 Counting Program.' *(see Deliverable Results, Team-Product, Product-Product)*

Product Backlog | A Product Backlog is an ordered list of everything that is known to be needed in a Product. This term is usually used when the Product is actually a Product-Product. When the Product is a Team-Product, its Product backlog is imbedded in the Team's Work Backlog. *(see Product-Product, Team-Product, Work Backlog)*

Product Backlog Item | Synonym for **Backlog Item** *(see Backlog Item)*

Product Champion | A Product Ownership role that represents the interests of a cohesive set of Stakeholders focused on the same Product. The Product Champion is a Business Owner who is restricted to a single Product, represents that Product's Backlog, owns the

Product Vision, and provides/ identifies Subject Matter Experts to the Organization to aid in development and reviews of that Product. Also called the Product's Product Owner. *(see Product, Product Ownership, Business Owner, Product Backlog, Product-Product Owner)*

Product Increment | Synonym for **Increment** *(see Increment)*

Product Owner | Depending on who you are talking to, the term 'Product Owner' could refer to any (or all) of the primary Product Ownership roles (Team Captain, Product Champion, Business Owner), but usually refers to a Product Champion. *(see Product Ownership, Product Champion, Team Captain, Business Owner)*

Product Owner Team | A Virtual Team made up of Product Owners (and others) that provide guidance to Organizations consisting of a Team-of-Teams *(see Virtual Team, Team-of-Teams, Leadership Team, Flow Management Team)*

Product Ownership | The collection of Responsibilities and Accountabilities that sit between the Stakeholders and the Developers; these responsibilities range from creating a strategic vision to delivery management to determining what work will be worked on next. *(see Team Captain, Business Owner, Product Champion, Agile Actuary)*

Product-Product | A Product that an Organization delivers to its Stakeholders; it has an associated Product Backlog and Product Champion. In some circles, a Product-Product is referred to as a 'Real' Product, as opposed to a Team's Product. *(see Product Backlog, Product Champion, 'Team-Product)*

Product-Product Owner | Synonym for **Product Champion**. *(see Product Champion)*

Product Review | A Team session at the end of a Sprint where the Team (along with its Product Ownership and Stakeholders) reviews their Increment in order to get feedback on what to do next, what to do better, and so on. *(see Increment, Product Ownership, Sprint Review, Progress Review)*

Product Vision | The Product Vision is a quick summary expressing how the product supports the Organization and/or Stakeholders.

Product's Product Owner | Synonym for **Product Champion**. *(see Product Champion)*

Production Team | A Scrum Team that *develops Product* to be used by Customers, Clients, and the like.

Progress Review | A meeting including Product Ownership and Customers about 'how the work is going'. Its purpose is to gather information and set expectations with the Customers. *(See Executive Review, Sprint Review, Product Review, Project Review)*

Project Review | Another name for **Progress Review** when the work is organized into a Project and, therefore, has a Delivery Forecast (Project Plan). *(see Progress Review, Delivery Forecast)*

Psychological Safety | the belief that one will not be punished or humiliated for speaking up with ideas, questions, concerns, or mistakes. *(see Openness)*

'Pull' Feedback | Feedback obtained by actively engaging a stakeholder during review of any product artifact (completed story or product increment). *(see 'Catch' Feedback)*

Quality Code | Synonym for **Clean Code** *(see Clean Code)*

Ready | A Ready Story is small, well-defined, and ready to take to Planning. Generally, this means that the Story's definition of Done is a '10 minute discussion' away from being agreed to. *(see Well-Defined Story)*

'Real' Product | Synonym for **Product-Product**. *(see Product-Product)*

Refactoring | Rewriting existing source code in order to improve its readability, reusability or structure without affecting its meaning or behavior.

Refinement | Synonym for **Backlog Refinement.** *(see Backlog Refinement)*

Regular Sync-Up | A re-plan/plan the Team has on a regular basis (typically daily) in order to collect together an understanding of 'changes in reality' in order to deal with it. *(see Daily Scrum)*

Regulators | Stakeholders, either inside or outside your Business Organization, who can regulate, or constrain, the Product/Results the Team is producing and/or the process that the Team uses. Examples include things like Cyber Security, CISA, ITAR Compliance, SOX, Medical, etc. *(see Stakeholder)*

Regulatory Constraint | An official standard, rule, or regulation that can affect either what is built (e.g., it must be bi-lingual), how it is built (e.g., the Code must be completely protected by Tests), or both. *(see Regulators)*

Release | A movement of the Team's Deliverable Results from the development environment to some other environment, for some other reason than development. Examples include alpha releases, beta releases, go-live releases, releases to a test lab, and so on. Releases are not a part of Scrum; releasing product must be done with Stories – there is no 'release' ceremony in Scrum.

Release Sprint | A specialized Sprint whose purpose is to Release Deliverable Results; it contains Stories specific to Release Activities and finishing UnDone Work. A Release Sprint usually contains no additional development. *(see UnDone)*

Release Strategy | A term used to refer to all types of Release Planning, Release Monitoring, and the like. It is not a part of Scrum, but a Release Strategy is often needed with Scrum – in part to decide what work will be left UnDone until the Release Sprint.

Representative | Synonym for **Team Representative**. *(see Team Representative)*

Resolution Process | Short for **Conflict Resolution Process**. *(see Conflict Resolution Process)*

Respect | Respect is the belief that people are always doing the best they can do at any given moment. *(see Scrum Values)*

Responsible | Responsible people are the individual(s) who actually do the work; responsibility can be shared. The degree of responsibility is determined by the person with the Accountability; and Responsibility is often confused with Accountability. *(see Accountable)*

Results | Short for **Deliverable Results.** *(see Deliverable Results)*

Results Backlog | A prioritized list of Deliverable Results that a Business Owner hopes to deliver to Stakeholders. The Stakeholders and Business Owner maintain the Results Backlog by adding new Deliverables and prioritizing/re-prioritizing. *(see Business Owner)*

Retrospective | Short for **Team Retrospective.** *(see Team Retrospective)*

Review | Short for **Sprint Review.** *(see Sprint Review)*

Rhythm | Movement or procedure with uniform or patterned recurrence. Scrum has two primary Rhythms: the Daily rhythm of work, and the Sprintly rhythm of feedback and planning.

Safety | Short for **Psychological Safety**. *(see Psychological Safety)*

Scale-Ready | A single-team process or practice is said to be 'scale-ready' when it can be easily applied or extended to an organization of many teams.

Scaling | The changes in Structure and Governance that enable successful growth (or reduction) of production. In general, the increase or decrease in one or more dimensions of an organization in order to improve success.

Scaling Scrum with Scrum® (SSwS) | A Scaling method that leverages what we already know about Scrum rather than developing new concepts. SSWS is the basis for Multi-Team Scrum. *(See Multi-Team Scrum)*

Scenario | An interaction with the System that consists of a single thread, and is represented by a single Acceptance Test. *(see Use Case)*

Scrum | An agile framework, model, or philosophy for Product Development, *not* Project Management. There have been, are, and will be, many variants of Scrum; it is more of a concept than a prescription.

Scrum$_P$ | Scrum$_P$ is a variant of Scrum in which the Product Owner 'lives with' the Stakeholders and is their representative to the Team. In Scrum$_P$ the Product Owner may not change priorities during the Sprint. Scrum$_P$ is defined by the Scrum Primer at scrumprimer.org, and the 'P' in Scrum$_P$ stands for 'Primer'.

Scrum$_G$ | Scrum$_G$ is a variant of Scrum in which the Product Owner is a member of the Team who consults with the Stakeholders. In Scrum$_G$ the Team's Product Owner may work with the Team to change priorities during the Sprint. Scrum$_G$ is defined by the Scrum Guide at scrumguides.org, and the 'G' in Scrum$_G$ stands for 'Guide'.

Scrum$_H$ | Scrum$_H$ is a variant of Scrum in which Product Ownership is shared between a Team Captain on the Team and Business Owners and Product Champions outside the Team. Scrum$_H$ is described in *The Scrum Handbook: Single-Team Scrum*, and the 'H' in Scrum$_H$ stands for 'Handbook.'

Scrum-of-Scrums (SoS) | In a multi-scrum-team environment: 1) a meeting held after all the individual Scrum Teams' Daily Scrums, consisting of a representative/ambassador from each Scrum Team, in order to achieve cross-team collaboration; 2) a virtual Scrum Team composed of the Scrum Masters from each Scrum Team; and 3) any virtual Scrum Team composed of representatives from various Scrum Teams.

Scrum Board | Synonym for **Story Board.** *(see Story Board)*

Scrum Guide | Any of *The Scrum Guides* published by Schwaber and Sutherland at scrumguides.org; Scrum Guides are usually distinguished by their publication date, like *The 2017 Scrum Guide*.

Scrum Master | Depending on who you are talking to, the term 'Scrum Master' could refer to somebody playing any (or all) of the Scrum Mastering roles (Team Facilitator, Change Agent, Team Coach), but usually refers to the Team Facilitator. *(see Scrum Mastering, Team Facilitator, Change Agent, Team Coach)*

Scrum Master Community | The collection of people doing Scrum Mastering within an Organization; this includes Team Facilitators. Agile Coaches, and Change Agents. This group is responsible for 'making Scrum better' within the Organization. They have an Improvement Backlog (often virtual or invisible) of changes they would like to have in the Organization to make it more amenable to Scrum. *(see Scrum Mastering, Team Facilitator, Agile Coach, Change Agent)*

Scrum Master Team | The Scrum Master Team for a Pod or Group is a Virtual Scrum Team consisting of all the Scrum Masters of the subordinate Teams, Pods, and Groups; with the Scrum Master of the Ldrsp/Mgmt Team as its Team Captain. *(see Virtual Team, Pod, Group, Management Team, Leadership Team)*

Scrum Mastering | Enabling, Empowering, and improving People, Teams, and Organizations in order to allow them to do their jobs better. People playing Scrum Master roles apply Servant Leadership principles to teamwork and practices in order to facilitate (making an action or process easy or easier) them becoming more effective and enjoyable. *(see Team Facilitator, Agile Coach, Change Agent)*

Scrum Team | A Scrum Team (commonly called the 'Team') is a small, co-located, self-organized, self-contained, value-driven, group of full-time Team Members who are organized around a Mission. Their job is to produce High-Quality Results at a Sustainable Pace.

Scrum Values | The Scrum Values are: Openness, Focus, Commitment, Respect, and Courage. *(see Values)*

Self-Contained | A self-contained (also called Cross-Functional) team is one that contains all the knowledge and skills necessary to accomplish its objectives and goals: in software development this

means that there are people who can test, there are people who can code, there are people who do analysis, there are people who write documentation, and so on.

Self-Organized | A self-organized team is one that chooses how best to accomplish its work, rather than being directed (micro-managed) by others outside the team. *(see Tactical Agility)*

Single Item Flow | Single item flow (also called 'single piece flow' or 'one piece flow') is a lean manufacturing concept that says that each individual Item will move through the manufacturing process *all at once* with no waiting between steps. On Scrum Teams, this means Stories don't wait for people who have skills they need – the people are available when they're needed. *(see Team Swarm)*

Single-Team Scrum (STS) | The Scrum that is described in the *Scrum Handbook: Single-Team Scrum (STS).*

Size [of a Story] | A measure of 'how much' Product was (or will be) produced by the Story. The Size of a Functional Story is typically based on the Story's Intrinsic Difficulty, Ideal Effort, number of 'moving parts', or some such. *(see Intrinsic Difficulty, Ideal Effort, compare to Effort [for a Story])*

Slack | Time people use to think, innovate, and improve themselves, the right amount of time to 'waste' (from Tom DeMarco), *(see lollygagging)*

Small [size of a Team] | A Team needs to be small enough to have one (or a few conversations) at the same time, so that everybody can be 'on the same page' all (or most of) the time. A Team must be small enough to remain nimble and be large enough to complete significant work in a Sprint. Typically, this means that a Team has between three and nine Members – the 'sweet spot' for software development seems to be about five.

'Small Project' Strategy | A development strategy that treats a Sprint as a Small Project, with a forecasted, anticipated, Result**.** *(compare to 'Continuous Development' Strategy)*

SME | Acronym for **Subject Matter Expert.** *(see Subject Matter Expert)*

SME Availability | An Environmental Variable that indicates whether or not there are Subject Matter Experts available who have the knowledge or expertise you need, when you need it. *(see Environmental Variables)*

SoC | Acronym for **Standard of Care.** *(see Standard of Care)*

Spike | An XP (eXtreme Programming) term that describes Stories that figure out answers to tough technical or design problems. Spikes address only the problem under consideration and ignore all other concerns. Most Spikes get thrown away, which differentiates them from Architecturally Significant Stories. *(see Architecturally Significant Story)*

Sprint | A fixed period of time (less than a month) in which a Team produces an Increment for review. The Sprint length is defined by the interval between Product Reviews, is usually consistent across Sprints, and must not be changed once the Sprint has started. *(see Increment)*

Sprint Backlog | A 'living document' that consists of the Stories the Team has brought into the Sprint, along with their definitions of Done and (possibly) Tasks. *(see Work in Progress, Sprint Forecast, Done, Task)*

Sprint Cancellation | Synonym for **Cancelling a Sprint**. *(see Cancelling a Sprint)*

Sprint Commitment | The Team commits to its Sprint Goal and to doing its due diligence to assure that all completed Stories are actually Done. *(see Done, Sprint Goal, Sprint Backlog)*

Sprint End | The time when work in the Sprint is ended in order to Review, Retrospect, and Plan. The Sprint End is usually a set date and time, but may be pre-determined by an event, as *"We'll have Sprint End when Doug gets back from Europe…" (see Sprint Planning, Sprint Review, Team Retrospective)*

Sprint Forecast | 1) when using the 'Small Project' Strategy for a Sprint, the Team's anticipated Result for the Sprint; or 2) The Team's best guess about how many Stories, Tasks or Story Points will be accomplished in the Sprint. There is no requirement for a Team to have a Sprint Forecast and, In any case, the Team is not accountable for achieving the Sprint Forecast; it is, at most, a 'best guess' – it is not a plan, commitment, or promise. *(see Accountable, Sprint Goal)*

Sprint Goal | The Sprint Goal is something that the Scrum Team members agree to accomplish *together* within the Sprint. The Sprint Goal defines success for the Sprint, and the Team will do 'whatever it takes' to meet it. Committing to the Sprint Goal, rather than the Sprint Backlog, allows the Team the 'wiggle room' needed to avoid compromising Quality while it works. Not to be confused with Sprint Objective or Forecast. *(see Sprint Objective, Sprint Forecast)*

Sprint Interrupt | At the end of the Sprint the Scrum Team interrupts the continuous flow of work and has four Ceremonies/Discussions/ Meetings in order to allow for Feedback, Improvement, and Re-Planning. *(see Sprint Planning, Team Retrospective, Product Review, Progress Review)*

Sprint Objective | An objective brought to Sprint Planning by the Team Captain. The Team Captain may be accountable for the Sprint Objective – but the Team is not – and the Team is under no obligation to choose the Sprint Objective as its Sprint Goal. *(see Accountable, Sprint Goal, Sprint Planning)*

Sprint Planning | A Team session at the beginning of a Sprint in which the Team Members (including the Team Captain) discuss and negotiate amongst themselves in order to: 1) Establish Sprint End; 2) Select a Kaizen to accomplish; 3) Make sure there are enough Stories Ready or 'in progress' so the Team can get back to work; and 4) Commit to a Sprint Goal. *(see Sprint End, Kaizen, Sprint Goal)*

Sprint Retrospective | Synonym for **Team Retrospective.** *(see Team Retrospective)*

Sprint Review | The Sprint Review consists of a Product Review and a Progress Review, and may consist of other Reviews as well. These reviews often require different participants, so the Sprint Review is often a series of separate Reviews. *(see Executive Review, Project Review, Progress Review, Product Review)*

Sprint Team | The Scrum Team *along with* any external SMEs who are (either officially or unofficially) members of the Team during the Sprint.

Sprint Zero | A deprecated synonym for **Startup Sprint**. *(see Startup Sprint)*

Squad Leader | A sergeant who is in charge of a squad of soldiers and lower-ranking sergeants; often used to refer to a **Team Captain** who is also a member of the DevTeam. *(see Product Ownership, Business Owner, Team's Product Owner, Team Leader, Crew Chief, Team Representative)*

Stakeholder | 1) A person with a *legitimate* interest in the Product, Process, or Team. 2) Someone who the Scrum Team ignores 'at their peril.' 3) A person who reviews the Team's Increment at the Product Review. 4) A person who is involved with, affected by, or has an effect on, the Team. *Note: While Team Members are stakeholders, the word Stakeholder [uppercase] is usually reserved for external stakeholders.*

Standard of Care (SoC) | The Standard of Care is the part of Done that describes the objective criteria the Team uses to determine whether or not they used the prudence, caution, processes, and procedures that they should have when doing their work. The Standard of Care depends on the work being done; each item of work (Story, Increment, Release, etc.) could have its own Standard of Care. The Standard of Care is used to 'guarantee' sufficient technical quality to be acceptable. Failure to meet the Standard of Care is negligence, and the Team is accountable for any damages that result. The Standard of Care is often (erroneously) referred to as the Definition of Done (DoD). *(see Done, Definition of Done)*

Startup Sprint | A specialized Sprint used to get a Team 'up and running'

quickly, rather than dragging their feet *getting ready* to start development. A Startup Sprint usually includes Analysis, Team Training, Infrastructure and Environmental Work – and the development of something 'real' – and its purpose is to limit the amount of 'up front' work that takes place before actual Product is developed. It is a Sprint whose Sprint Goal is *"We will be producing real Results by the end of this Sprint"* – it is not about 'getting ready'; it is about 'getting moving'.

Stay-At-Home | In swarming, a person who 'stays with' the Story, and probably works as the Story's Coordinator. When working on Coding Stories, it is common to have a Coder as the Stay-At-Home in order to avoid context-switching. *(see Team Swarm, Coordinator)*

Story | 1) A request from a stakeholder for something of value (it could be an Epic); 2) A unit of work that is 'small enough' to be agreed to by the Team; a Backlog Item that is not an Epic. 3) (by others) A synonym for Backlog Item. 3) (by others) a synonym for User Story. *(see User Story, Backlog Item)*

Story Agreement |An agreement between the Product Owner (Team Captain) and the rest of the Team that defines when a Story will be complete (or Done). The Story Agreement typically consists of the Acceptance Criteria, the Standard of Care, and (possibly) additional General Agreements. This is a synonym for Done. *(see Done, Acceptance Criteria, Standard of Care, General Agreements)*

Story Board | A Team tool that shows the tasks that are needed in the Sprint, organized by Story. The Story Board is a 'living' document that the Team updates to reflect its current thinking; it is often updated at the Daily Scrum.

Story Owner | A Team Member (or SME) who represents the Stakeholder's interests in the Story to the rest of the Team during Planning and Development. *(see Story)*

Story Size | Synonym for **Size [of a Story].** *(see Size [of a Story])*

Story Time | Synonym for **Backlog Refinement.** *(see Backlog Refinement)*

Storyotype | A stereotype or template for a Story or Epic. Storyotypes are used to capture re-useful information common to many Stories; in particular, Storyotypes are used to capture common Standards of Care. *(see Standard of Care)*

StoryPoint | A relative measure of the size of a Story. Often confused with EffortPoint. *(see EffortPoint, Velocity, Size [of a Story])*

Strategic Agility | Agility that changes 'what' the Team or Organization does in order to maximize Value or ROI. Strategic Agility is a Product Ownership responsibility. *(compare to Tactical Agility)*

Structure | An Organization's Structure shows how the Teams, Pods, Groups, and Individuals are connected and related to each other.

STS | Acronym for **Single-Team Scrum**. *(see Single-Team Scrum)*

Subject Matter Expert (SME) | Somebody with specialized knowledge or talent that is needed by the Team; this includes SMEs on the product, the environment, development practices, and so on. The term usually refers to SMEs that are 'outside' the Team, but not always.

Sustainable Pace | The rate at which a Team can work without burning itself out. Originally called '40 Hour Week' by Kent Beck as an XP practice.

Swarm | Short for **Team Swarm**. *(see Team Swarm, Swarming)*

Swarmer | In a Team Swarm, a Swarmer is a person who is moving from Story to Story, working with those Story's Coordinators and other Swarmers, in order to offer his or her expertise and efforts wherever they are needed. *(see Team Swarm, Swarming)*

Swarming | Having several people work together on a piece of work. Common swarming patterns in software development include: *Pair Programming*, *Three-People-at-a-Whiteboard*, and *Team-at-a-Table*. *(see Team Swarm. Three-People-at-a-Whiteboard, Team-at-a-Table)*

Tactical Agility | Agility that changes 'how' a Team works in order to get to Done; this is embodied in the Team's Self-Organization. *(compare*

to Strategic Agility)

Task | A small, undivided, 'chunk' of work to be accomplished, usually within the Team. Tasks organize the team's work plan on how they will get Stories to Done.

Team | 1) The Scrum Team; 2) (by others) The Development Team; 3) (Ken Schwaber) A role, taken on by a group of people, that means that they are a Well-Formed Team. *(see Well-Formed Team)*

Team-at-a-Table | A Swarming Pattern used for planning, general goal-setting, and so on. As its name implies, this involves a group, sitting around a table, having open discussions. It is frequently facilitated by a Scrum Master. *(see Team Swarm)*

Team Ability | This is an Environmental Variable, and includes the capabilities of individual Team Members, the Team's frame of mind, and how well the Team synergizes. *(see Environmental Variables)*

Team Captain | The Team Member accountable for maximizing the value of the Scrum Team's Work. Each Team must have its own Team Captain to own the 'play calling' that determines what work the Team should do next. Also commonly called the Team's Product Owner. Some Team Captains are also Team Leaders, Squad Leaders, Crew Chiefs, or virtual. *(see Product Ownership, Business Owner, Team's Product Owner, Team Leader, Squad Leader, Crew Chief, Team Representative)*

Team Facilitator | A Team Member who facilitates (makes easier) the Team's self-organization, growth, maturation, and improvement (on a daily basis) as the Team 1) does its work, 2) removes **Impediments** to progress, and 3) achieves improvement objectives, or **Kaizens**. Every Team must have a Team Facilitator, who is often a technical contributor, as well *(see Scrum Mastering, Impediment, Kaizen, Team Representative, Scrum Master)*

Team Leader | A Scrum Team has a Team Captain who is merely a Team Leader when the rest of the Team does not see the Team Captain as a full-fledged member of the Team.. *(see Product Ownership, Business Owner, Team's Product Owner, Squad Leader, Crew Chief,*

Team Representative)

Team Member | Any member of the Scrum Team, including the Team Captain (Product Owner) and Team Facilitator (Scrum Master).

Team Norms | Team norms are a set of rules or guidelines that a team establishes to shape the interaction of its members with each other and with people external to the team. Team norms are used to help guide the behavior of team members and to assess how well they are behaving. *(see Conflict Resolution Process)*

Team-of-Teams | An Organization that is made up of Scrum Teams; it is often seen as a big Team whose members are Scrum Teams.

Team-Product | The Deliverable Results produced by a single Team. *(see Product, Product-Product)*

Team Representative | A Scrum Team has a Team Captain that is merely a Team Representative when its internal Product Ownership is achieved through self-organization, and there is no identifiable person playing the Team Captain role *inside* the Team. In this case, the Team selects a Team Representative to represent the Team *outside* the Team when necessary. *(see Product Ownership, Business Owner, Team's Product Owner, Squad Leader, Crew Chief, Team Leader)*

Team Retrospective | A Team session at the end of a Sprint when the Team Members (facilitated by their Team Facilitator) discuss and agree upon ways they could improve their Practices, teamwork, environment, or Organization for the next Sprint.

Team Swarm | A method of working where a Team works on just a few things at a time. Each Story is finished as quickly as possible by having many people work on it together, rather than having a series of handoffs. The ultimate is Single Item Flow, where the whole Team works on one Story at a time, and finishes it completely before moving on to the next one. *(see Swarmer, Swarming, Coordinator, Stay-At-Home)*

Team Values | The Team Values are: Openness, Focus, Commitment,

Respect, Courage, Visibility, Humor, and Accountability. *(see Values, Scrum Values, Kanban Values)*

Team's Product Owner | Synonym for **Team Captain.** *(see Team Captain)*

TeamLet | The Team Members and SMEs who are Swarming on a particular Story. The typical swarm involves 2-3 people at a time. *(see Team Swarm)*

Technical Debt | This is an Environmental Variable that includes deficiencies in the code, technical documentation, development environments, 3rd-party tools, and development practices, which makes it hard for the Team to modify, update, repair, or deliver the Product. *(see Environmental Variables)*

Three-People-at-a-Whiteboard | A Swarming Pattern used for problem-solving, design, and so on. As its name implies, it consists of three people, with complementary skills and knowledge, working together to solve a hard problem. This pattern is generally acknowledged to be the most powerful tool in engineering. *(see Team Swarm)*

Time-Boxed Story | Time-Boxed Stories have their Acceptance Criteria (at least partially) defined by a time-box, and the actual Results produced are limited to what can be completed (using the Story's Standard of Care) within that time-box. *(see Acceptance-Based Story, Standard of Care, Done)*

Transformation Owner | A combination Business Owner and Scrum Master who is accountable for implementing an Agile Transformation. The Transformation Owner is often the Team Captain of a Transformation Team. *(see Transformation Team, Business Owner, Scrum Master, Agile Transformation)*

Transformation Team | A Team that shepherds an Organization's Agile Transformation. A Transformation Team contains Agile Coaches and 'People with Power' who can 'get things done', and is (often) external to the Organization being transformed. *(see Agile Coach, Agile Transformation)*

UnDone | The phrase 'UnDone work' is often used to describe the work needed to move something from Done to Releasable; in other words, it is work that *maybe should have been* Done, but wasn't. Deciding what work to leave UnDone is a delicate issue. *(see Done)*

Use Case | A Capability that represents an interaction between a User and the System in order to achieve a Goal. A Use Case consists of multiple Scenarios, and usually requires many Stories to implement, so a Use Case is usually an Epic. *(see Scenario, Epic)*

User Story | A Story whose value is for the User of the software; popularized by eXtreme Programming (XP). *(see Story)*

Validation | Validation is assuring that a Result (Capability) is fit for use; that it does what it *needs* to do. *(compare to Verification)*

Value-Driven | A Team is value-driven when the Team Members value working together; they are constantly improving themselves, their Team, their environment, and their tools; and they strive to live an appropriate set of Values. *(see Values)*

Values | The word Values, in common use, refers either to values in general, the Team Values, the Kanban Values, or the Scrum Values. *(see Team Values, Kanban Values, Scrum Values)*

Velocity | The rate that a Team or Organization *has been producing* Product; usually calculated as completed StoryPoints per Sprint. It is often used as an approximation for Capacity and is often confused with WorkRate. *(see StoryPoint, Capacity, WorkRate)*

Verification | Verification is assuring that something has met its specification; that it does as it was *intended* to do. *(compare to Validation)*

Virtual Team | A Team containing Team Members who actually 'live' on other Teams.

Visibility | The Team makes the current state of the Team's Product/ Results visible to Stakeholders and the Business. *(see Values)*

Walking Skeleton | A subset of the System that demonstrates the basic

architectural decisions; it is the result of many Architecturally Significant Stories. *(see Architecturally Significant Story)*

Well-Defined Story | A Story whose Acceptance Criteria are known. *(see Acceptance Criteria, Ready Story)*

Well-Formed Team (WFT) | A Well-Formed Team (WFT) is more than just a Team; it's a 'real Team' – a Team that knows its job, does its job, and looks good doing it. A WFT is a Team with heart and soul; where Team Members value working together to be the best Team they can be. A WFT is a team that is self-organized, self-contained, and value-driven. A Scrum Team is a well-formed team that has both a Product Owner and a Scrum Master, and it is a primary teaching of Scrum that all teams (especially those working in complex domains) should be well-formed. *(see Self-Organized, Self-Contained, Value-Driven)*

WFT | Acronym for **Well-Formed Team.** *(see Well-Formed Team)*

Whole Team | The Scrum Team along with its Subject Matter Experts and its Stakeholders; this is the 'whole team' involved in production.

WIBNI (wib'·nee) **|** stands for **W**ouldn't **I**t **B**e **N**ice **I**f, and represents things that we wish were true, but aren't – so we must *get over* them; example is *"wouldn't it be nice if we had more testers…"*

WIP | Acronym for **Work in Progress.** *(see Work in Progress)*

WIP Limit | The maximum number of Stories allowed in the Work in Progress at any given time. *(see Work in Progress)*

Work Backlog | The Team's Work Backlog is owned by the Team Captain and consists of Stories that are being refined to become Ready for Planning. These Stories are of two types: Capabilities and Chores. The Capabilities in the Work Backlog can be referred to as the Team-Product's Product Backlog. *(see Product Backlog, Ready Story, Capability, Chore, Team-Product)*

Work in Progress (WIP) | The Stories that the Team is currently working on. *(see Backlog)*

Work Item | The Work Items the Scrum Team works on are called Stories, and as the Team completes the Stories, they produce Results in an iterative and incremental manner, thus providing opportunities for meaningful feedback from Stakeholders.

Workgroup | Synonym for **Cross-Cutting Workgroup**. *(see Cross-Cutting Workgroup)*

WorkRate | The rate that a Team expends effort; usually calculated as EffortPoints per Sprint, Ideal Engineering Hours/Days per Sprint, or something similar. It is used as an aid in Sprint Planning, and is often confused with Velocity. *(see EffortPoints, Ideal Engineering Hours/Days, Velocity)*

Work Results | Whatever the Team produces; consists of the Increment (partial and potential deliverables) and non-deliverables, such as Chores. *(see Increment, Chore)*

XP | Acronym for **eXtreme Programming.** *(see eXtreme Programming)*

Zombie Scrum | Any of the Scrums that the *Scrum Guide* replaced; they don't have a Team Captain and/or they commit to completing their Sprint Backlogs. *(see Scrum Guide, Sprint Backlog, Team's Product Owner)*

Appendices

Nine Zones of Scrum (A Taxonomy)

There are two major discriminators when looking at types of Scrum: the Product Ownership, and Sprint Planning. There are three types of each, leading to Nine Zones of Scrum, as we see in the picture.

	Locked-Down Sprint	**Adaptable Sprint**	**Continuous Planning**
(both) Team's & Product's Product Owners	Scrum 2.5	Scrum 2.75	Scrum 3.0 ($Scrum_H$)
(only) Team's Product Owner	Scrum 1.75	Scrum 2.0 ($Scrum_G$)	Scrum 2.25
(only) Product's Product Owner	Scrum 1.0 ($Scrum_P$)	Scrum 1.25	Scrum 1.5

In our Scrum journeys, we have seen Scrums in each of these 9 zones; each type of Scrum proves to be useful in the appropriate situations. As is noted in the picture, the three Scrums we mention in this Handbook are along the diagonal, as instances of Scrum 1.0, 2.0, and 3.0.

The Scrum we describe in this handbook, $Scrum_H$, is an instance of Scrum 3.0, the most flexible, and capable, kind of Scrum. The types of Scrums that are 'grayed-out' are what we call 'zombie' Scrums (Scrums that the *Scrum Guide* 'killed'), while the four Scrums in the upper-right corner are what we call the 'modern' Scrums (Scrums that are compliant with the Scrum Guide).

The Agile Manifesto

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

Twelve Principles of Agile Software

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Three Recursive Steps for Improvement

Acknowledgements

We have both been involved with Scrum for over 30 years, and we'd like to thank Ken Schwaber and Jeff Sutherland for bringing Scrum to the software community. Without them, there would be no Scrum.

We would also like to thank all the teams we have worked with and observed, and all our students who have brought us news and information about their Scrum teams. We know that "*Scrum is what successful Scrum Teams do,"* and without this feedback about what successful Scrum Teams do, there would also be no Scrum.

Good luck, and happy scrumming!

About the Authors

Dan Rawsthorne

Dan has developed software in an agile way since 1983. He has worked in many different domains, from e-commerce to military avionics. He has a PhD in Mathematics (number theory), is a retired Army Officer, and is a Professional Bowler and Coach. Dan is very active in the Agile/Scrum community and speaks at conferences and seminars. He is a transformation agent, helping Organizations become more successful through agility. His non-software background helps him immeasurably: his mathematics back-ground tells him to look for underlying problems rather than focus on symptoms; his military career gave him experience in teamwork and empowerment; and his work with bowlers helps him understand that coaching is a two-way street.

Doug Shimp

Doug has worked in the technology field since 1992 and has played many key roles on software teams, including Coder, Tester, Analyst, Team Leader, Manager, Coach, and Consultant. Doug's passion is for team learning to improve product development, and he is a leader in the area of Agile/Scrum transitions and applied practices. He believes that the core basis for applied agility is that 'You must see the result for it to be real; otherwise, it is all just theory…' Much of his experience with teamwork and agility comes from outside the software field, including an earlier career as an owner/manager of a painting company – which enables him to learn about small-team dynamics in a very hands-on way.

For More Information

To learn more about 3Back, LLC and our Scrum-related services contact us at info@3back.com. Follow @Scrum_Coach on Twitter, and to subscribe to our newsletter, visit: https://3back.com/blog/.

We Make Teams Better

We don't just talk Agile, we live Agile. Our 3Back Team is a well-formed, Agile team; applying $Scrum_H$ in our own workplace. From our hands-on support staff to our seasoned consultants and trainers, every member of the 3Back Team is, at a minimum, a CSM (Certified Scrum Master). Within every level of the 3Back Team, we bring a real-world appreciation and understanding of your team's needs.

For Help With Your Company's Scrum: info@3Back.com

For Training: training@3Back.com

Managing Work

At 3Back we are a fully distributed team. We actively build and manage Get To Done (gettodone.com), an online Scrum software development tool. Get To Done helps us train as a pedagogical tool, explore new ways of collaborating and focus our most precious resource – attention – on work being done.