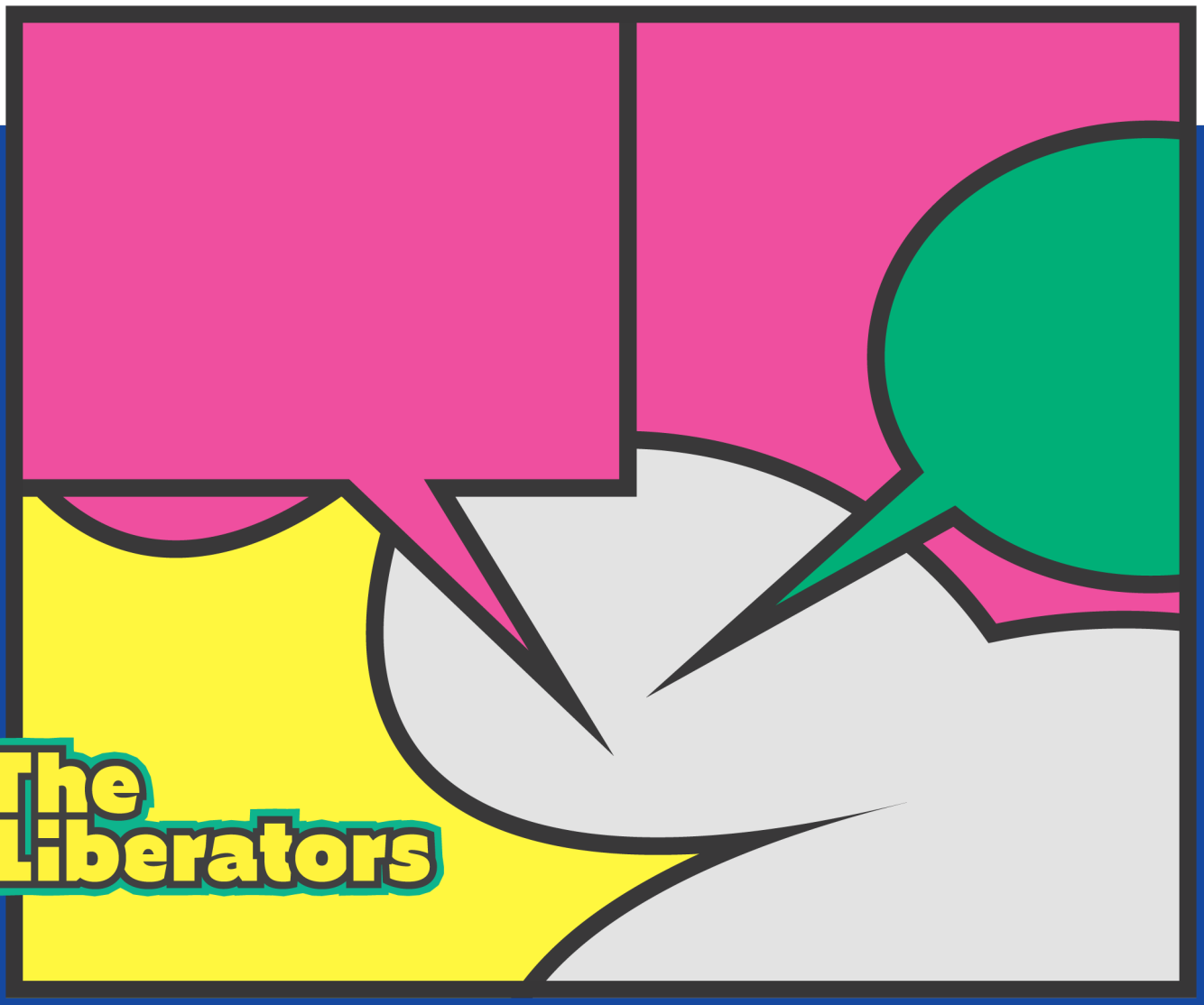
free download

Scrum: A Framework To Reduce Risk And Deliver Value Sooner

An overview of the Scrum framework, for people new to Scrum and those who'd like to refresh their understanding.



In this Whitepaper

Foreword

"We don't care about Scrum", is what we sometimes tell people. That certainly raises eyebrows. it's our way of saying that it's not about the Scrum framework, but about what it makes possible. When you take this perspective, many theoretical questions become obvious or even pointless. Like "Should all items on the Sprint Backlog relate to the Sprint Goal?", "Should a bug be on the Product Backlog?" or "Should I stop the Daily Scrum exactly at 15 minutes?". It's easy to get stuck in the minutia when you forget (or don't see) the bigger picture.

We love the Scrum framework for what it makes possible. The official Scrum Guide already does a great job of explaining it in a concise manner. We want to add to that with an explanation in our own words, just like we do in our work with Scrum Teams and as Professional Scrum Trainers for Scrum.org. Our aim was to write in a practical, down-to-earth manner from the perspective of what the Scrum framework makes possible. We hope that it is easy to read, clears up potential confusion, and deepens your understanding of Scrum.

We wrote this whitepaper together with Johannes Schartau for our book 'The Zombie Scrum Survival Guide'.

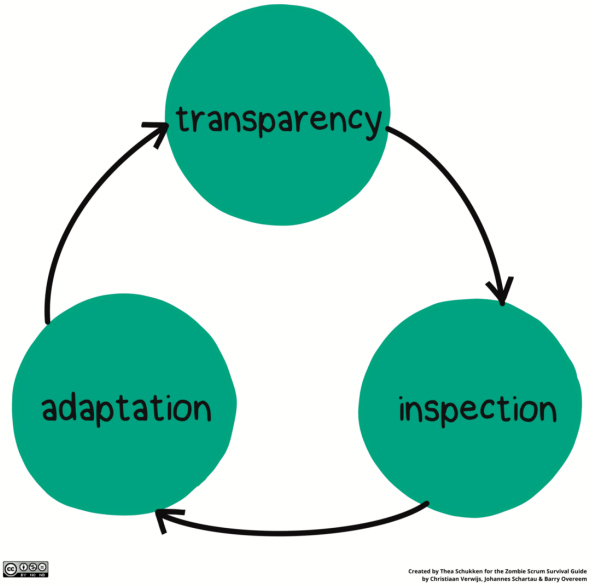
A Bit of History

The Scrum framework was developed by Ken Schwaber and Jeff Sutherland in the 1990s. It was first formalized in 1995 to address the inherent complexity of product and software development. More recently, the Scrum framework is being applied successfully to complex problems in a wide variety of domains. From marketing to organizational change. And from scientific research to software development. The Scrum framework, wherever you use it, is built on three pillars that allow empirical process control.

The Pillars of Empiricism

The Scrum framework is built on three pillars that allow empirical process control:

- **Transparency**: you gather data – like metrics, feedback, and other experiences – to find out what is going on;
- **Inspection**: you inspect the progress with everyone involved and decide what that means for your ambitions;
- **Adaptation**: you make changes that you hope will bring you closer to your ambitions;

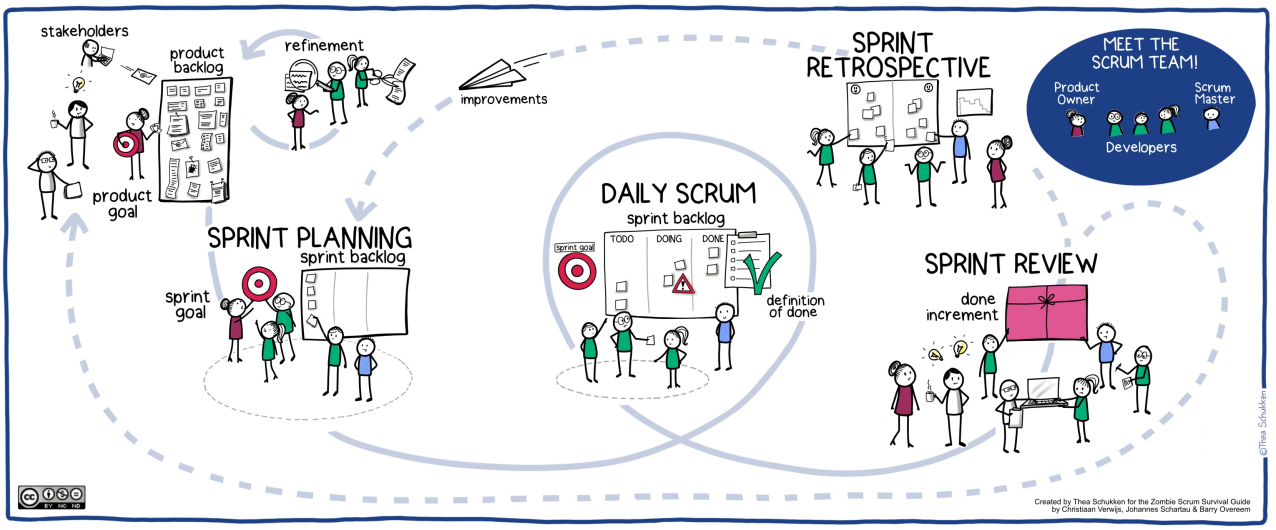


This cycle repeats as often as necessary to catch deviations, unexpected discoveries, and potential opportunities that emerge as the work is done. It happens not once a year or when the project is completed, but continuously on a daily, weekly, or monthly basis. Rather than making decisions based on assumptions about potential futures, you're instead making decisions on the data you've collected up to this point. This is empiricism. And in this post, you will discover how everything in the Scrum framework is designed around these pillars.

> "Rather than making decisions based on assumptions about potential futures, you're instead making decisions on the data you've collected up to this point."

Scrum: A Framework for Empirical Process Control

It's one thing to say that you need transparency in your work on a product, that you frequently need to inspect that work, and that the insights that emerge from this will drive adaptation. It's another to put that into practice in a tangible, concrete way. This is what makes Scrum a framework; it offers five repeating events to work on three artifacts, three accountabilities to support this plus several principles & rules to glue this together into a cohesive whole.



The Scrum Artifacts

The first pillar of empirical process control is 'Transparency'. In order to frequently inspect the progress of your work on a product and make decisions about what (else) is needed, we need to have something we can inspect. By making the work on a product 'transparent', we mean making it available in a form that allows the people that have a stake in the product to look at it, to validate assumptions with it, and to generate new ideas from it.

The Scrum framework requires teams to make at least three elements of their work on a product transparent. We call these 'artifacts' in Scrum, and they are the main vehicles for collecting the data and experiences we need to inform our decision-making about the future.

The Product Backlog

The first artifact is the Product Backlog. It makes transparent all the work that needs to be done on the product to achieve a Product Goal. The Product Goal gives focus to the work on the Product Backlog by allowing the Scrum Team to commit to a single long-term objective for the product. As Scrum Teams work towards the Product Goal, the Product Backlog continuously changes to reflect new insights and opportunities that emerge during that work. A new Product Goal should be set only when the previous one has been achieved or abandoned, otherwise focus and commitment will suffer.

Every idea, hypothesis, feature, bug, or task that the Scrum Team currently understands as necessary to achieve the Product Goal is represented on the Product Backlog by an individual item. Some will be very broad and unclear while others will be small and specific. All items are ordered according to their relevance to the ambitions of the product and its stakeholders.

A product has one Product Backlog with one Product Goal, regardless of how many teams are working on it. Otherwise, transparency and focus suffer as there is no single "source of truth" about the work that is needed and the order in which it should be done.

The Sprint Backlog

The second artifact is the Sprint Backlog. This is the selection of items that the Developers pull from the Product Backlog that they consider necessary to achieve the single Sprint Goal they committed to for that Sprint. The Sprint Backlog makes transparent all the work that a Scrum Team is working on or going to work on, in the current Sprint. Each Scrum Team has its own Sprint Backlog, even when they work with multiple teams on a single product. The Sprint Backlog is not static and changes as teams learn more during the Sprint. In close collaboration with the Product Owner, Developers may add or remove items based on how much time is left in the Sprint and the relevance or necessity of those items to the Sprint Goal.

The Increment

The third artifact is the Increment. Each Increment represents one step towards the overarching Product Goal. In a Sprint, an Increment is created whenever a Product Backlog Item from the Sprint Backlog is completed. To avoid confusion and different expectations, an item is considered "done" only when it is thoroughly verified and matches the Definition of Done that the Scrum Team has committed to. Scrum Teams may create multiple Increments during a Sprint and may even deliver them to their stakeholders before the Sprint Review.

The purpose of each Increment is to validate assumptions about the work that has been done to date. People that have a stake in the product can inspect it and determine if it meets their needs if they understand how it works and if it satisfies their expectations. The Increment is a driver of new ideas, as having something tangible to interact with is a great way for people to see new possibilities.

The Increment is the primary means by which Scrum Teams exercise empirical process control for complex problems. Every Increment starts with a hunch, a hypothesis, or a potentially valuable idea of how the product can be moved closer to its ambitions. This is what is captured in the Sprint Goal. The work necessary for achieving that goal is captured on the Sprint Backlog.

If multiple Scrum Teams work on a single product, they integrate their work into a single Increment every Sprint so that effective inspection can take place.

What Else?

Obviously, there are many other elements of the work of a team that we can make transparent. For example, you can create a stakeholder map to get a better sense of who and where they are. You can collect metrics on a dashboard and inspect them frequently. These are fine artifacts, and worthy of inspection. But for the purpose of building products empirically, the Scrum framework recommends that teams start by creating transparency with the three core artifacts: the Product Backlog, the Sprint Backlog, and the Increment.

The Scrum Events

In order to inform your decisions about what to do next, you have to frequently make sense of the Product Backlog, Sprint Backlog, and Increment with everyone who has a stake in it. This is what 'Inspection' is all about. It comes in many forms. We can make sense of the Product Backlog by looking at it and drawing conclusions – like 'it's too long', 'we shouldn't do this item before that one', or 'when can we deliver this to a stakeholder?'. We can make sense of the Increment by bringing in users and validating our assumptions with them – like 'do users understand this new feature?' or 'does this feature address the need?'. And we can make sense of the Sprint Backlog by making sure it is feasible and reflects our plan for the current Sprint.

Whatever form it takes, inspection offers the best foundation for decisions when it is based on something tangible and concrete. In the case of product development, inspecting a completed and deployed feature is the best way to truly validate your assumptions about that feature and how it is used. Looking at a presentation or a proposed design of a feature might seem useful, but everyone attending is still making plenty of – and probably different – assumptions of how it will work when it is implemented.

The Scrum framework proposes that in order to work empirically, teams should have at least five repeating moments for inspection to take place. Each of them has a specific timebox and offers a specific perspective on the work that is being done. Together they offer a complete picture. These are the five Scrum events. Although they are often referred to as 'meetings', they are not intended as meetings in the traditional sense of the word, where a group of people sits around a table being bored to tears. In fact, when the Scrum events are done with their purpose clearly in mind, the need for other meetings diminishes and makes place for natural collaboration and in-the-moment coordination as the team does its work.

The Sprint

The first event is the timebox of the Sprint itself. Much like you solve a complex jigsaw puzzle by starting on a smaller area, the purpose of each Sprint is to attempt to solve one part of a complex problem. This partial solution is represented by an incremental version of the product – the Increment – that can be inspected together to decide what should happen next. This Increment should at least be in a state where it can be released to stakeholders directly after the Sprint with the proverbial press of a button if the Product Owner decided to do so. Even better is a situation where teams can release throughout the Sprint, accelerating their ability to learn even further. Although the success of Sprints will vary – even a single Sprint involves a lot of complex, unpredictable work – much will be learned about the puzzle either way.

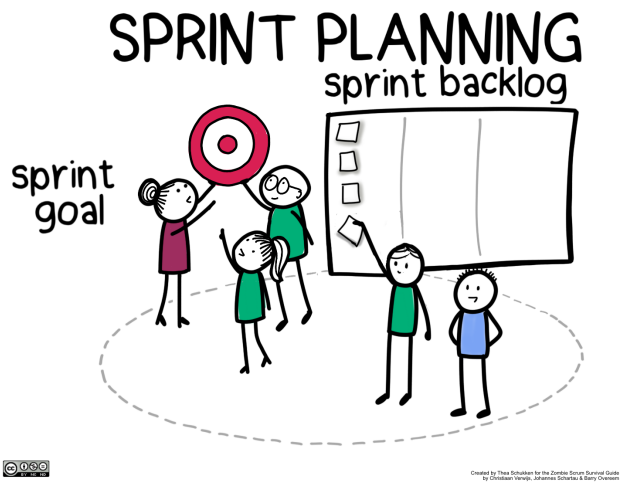
> "Much like you solve a complex jigsaw puzzle by starting on a smaller area, the purpose of each Sprint is to attempt to solve one part of a complex problem."

Sprints should always be the same length to create cadence and predictability for the team and its stakeholders. When a Sprint becomes too long, valuable opportunities are lost to validate assumptions and make sure that work is still progressing in the right direction. As the length of Sprints increases, so does the risk of wasting time building the wrong things. The Scrum framework does not specify a length for Sprints, except that they should be less than a month. It's up to the team to decide how fast they can, and want to, learn. Generally speaking, Sprints should be as short as possible while still allowing the Developers to deliver a meaningful new version of the product that achieves the goal of a Sprint.

So while the Sprint is a timeboxed opportunity to explore a particular part of the complex problem of developing a product, the other four Scrum events represent specific opportunities within the Sprint to promote inspection and adaptation.

Sprint Planning

Each Sprint in Scrum starts with a moment where the Scrum Team makes a basic plan for the upcoming Sprint. This is the Sprint Planning. The first and most important question is what the goal for this Sprint will be. Without a goal, there is no clear purpose that encourages members to unite and put their minds together during the Sprint. A goal for a Sprint can be to deliver a coherent set of features that solves a particular problem. It may be about addressing a need for a group of stakeholders. It may also be a hypothesis or critical assumption that a Product Owner wants to verify with the Increment that comes out of the Sprint.

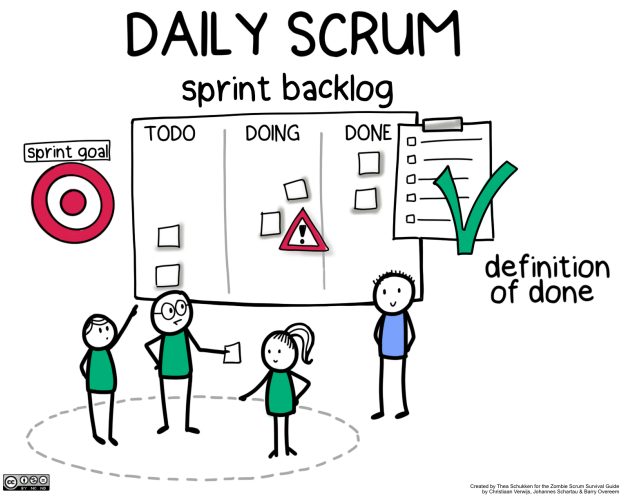


Although the Product Owner does well to come into Sprint Planning with an objective in mind, the Scrum Team works together to refine this into a goal that is both valuable and feasible within the timebox of a Sprint. The Developers work with the Product Owner to select the work from the Product Backlog that should take place within the Sprint to achieve that goal, reordering the Product Backlog as needed. This selection becomes the Sprint Backlog. Because the Developers do the actual work, they own and have the final say in what goes on the Sprint Backlog.

For Sprints of a month, Sprint Planning should take no more than eight hours. But the shorter the Sprint, the less time it will take. Sprint Planning completes when there is a rough outline for the coming Sprint and a more detailed plan for the first couple of days. This plan is usually captured in a decomposition of work for the first days of the Sprint. As the Sprint progresses, the Developers translate the rough outline – as set by the Sprint Goal and the Sprint Backlog – into a more concrete plan on how to work together to do this. At least one moment to do this is during the Daily Scrum.

Daily Scrum

The Daily Scrum takes place every 24 hours to allow the Developers to navigate the complexities involved in even a single Sprint. They inspect the progress of their work towards achieving the Sprint Goal, as made transparent through their Sprint Backlog, and make adjustments accordingly. For example, they may run into unexpected issues that require close collaboration or discover that they need to add work to the Sprint Backlog that is necessary to achieve the Sprint Goal. They may run into impediments; issues that are blocking them in achieving the Sprint Goal and are beyond their ability to resolve.



The Daily Scrum should take no more than 15 minutes. It is a short and minimal opportunity to coordinate collaboration for the next 24 hours. If more coordination is helpful, Developers can do so throughout the day. The Scrum framework does not prescribe how to do a Daily Scrum effectively, and Developers are encouraged to find the best way to make the Daily Scrum work for them.

Sprint Review

The Sprint Review happens at the end of a Sprint, before the Sprint Retrospective. Its purpose is to inspect the work that has been done to date and to decide what next steps make sense based on what was learned from that. The Sprint Review is at least one moment during the Sprint where the people building the product and the people that have a stake in it gather to inspect the outcomes of the Sprint. Together with current market conditions, organizational changes, budget, and timeline, they decide on the next steps together.

Sprint Reviews should take no more than four hours for a Sprint of a month. The shorter the Sprint, the shorter the Sprint Review tends to be. The output of the Sprint Review consists of adjustments to the Product Backlog based on what was learned. This may include new ideas that emerge during Sprint Reviews, discovered bugs, changes to items already on the Product Backlog, or re-ordering the Product Backlog itself. In a sense, the Sprint Review is about answering the question: "Based on what we learned this Sprint, what are the next steps?". This provides valuable input for Sprint Planning and potential Sprint Goals for coming Sprints.

For Sprint Reviews, a one-way presentation of what was done by the Developers does not constitute 'inspection'. Actually inspecting the increment means trying it out together and giving feedback. We prefer to describe Sprint Reviews as 'feedback parties' instead of 'demos'. Sprint Reviews should not be the first time that a Product Owner sees what the Developers have done. Instead, the Sprint Review is an important and repeating moment where the Product Owner invites stakeholders to inspect the product together.

> "For Sprint Reviews, a one-way presentation of what was done by the Developers does not constitute 'inspection'."

Sprint Retrospective

The Sprint Retrospective happens at the end of the Sprint, usually right after the Sprint Review. It is attended by the entire Scrum Team. Its purpose is to inspect how the Scrum Team worked together to achieve the Sprint Goal, and identify concrete steps that can be taken in the next Sprint to increase effectiveness and quality. Where the Sprint Review is more focused on the product and the content of the work that was done, the Sprint Retrospective is aimed at inspecting the process of how that work was done.

For Sprints of a month, the Sprint Retrospective should take no more than three hours. But the shorter the Sprint, the shorter the timebox tends to be. Based on what the Scrum Team has learned from the inspection of their collaboration, they may decide to make adjustments to work more effectively. This can involve an updated Definition of Done, researching new tools or technologies, a change in working agreements, or a different team composition. At least one actionable improvement goes straight to the Sprint Backlog for the next Sprint.

Product Backlog Refinement

So what about Product Backlog refinement? The Scrum Guide mentions this as something that needs to happen somewhere. Doesn't that make it an event? And yes, although it's definitely an essential part of the Scrum framework, it's not an event like the others. Rather, it's an ongoing activity. And while this may seem to be a play of words, it does capture an important difference. It all starts with understanding the many forms of Product Backlog refinement:

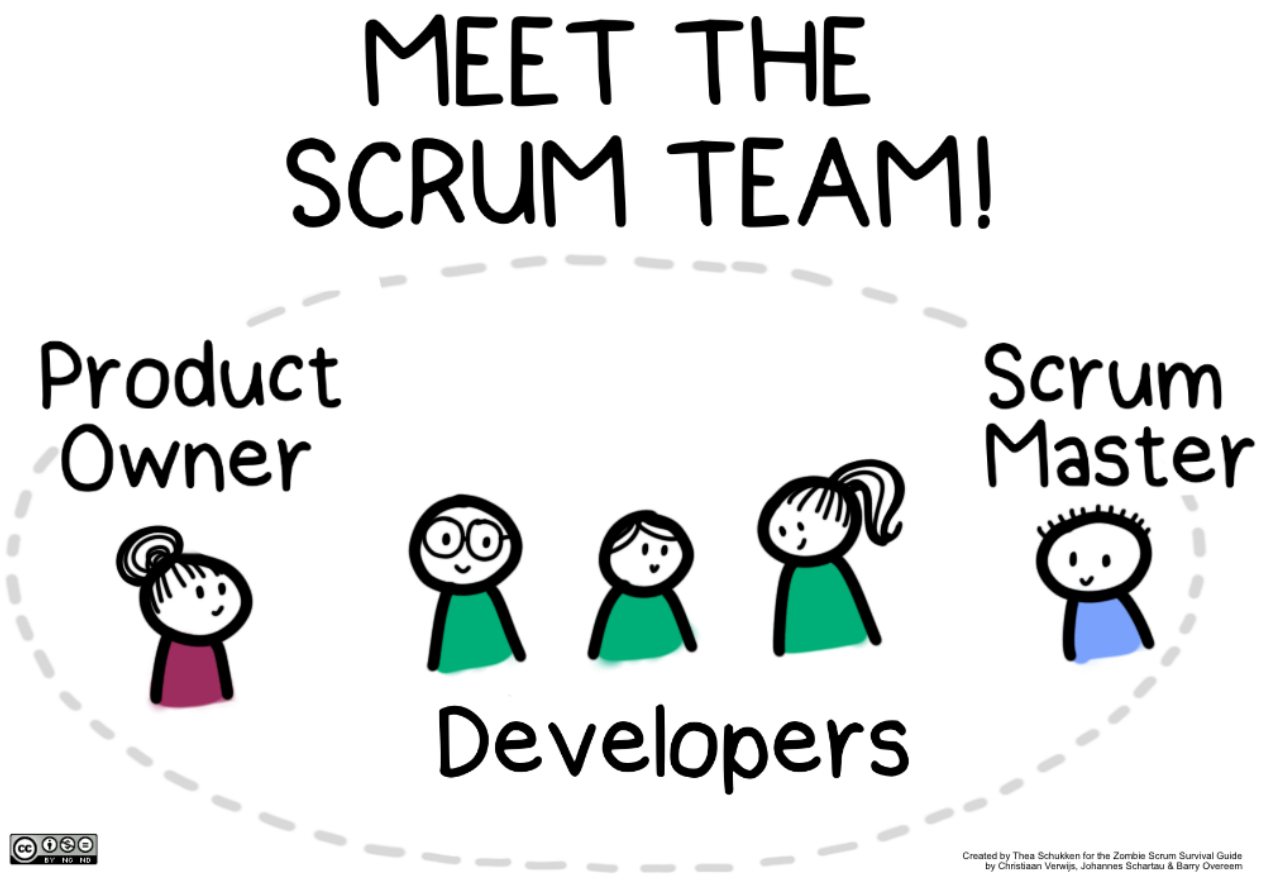
- Clarifying items on the Product Backlog that are too unclear to start work on. This is preferably done directly with the people you're building the items for (the stakeholders);
- Breaking down work on the Product Backlog that is too big to complete in a single Sprint (which generally also means that they're too unclear);
- Re-ordering work on the Product Backlog as needed to make the upcoming Sprints as smooth and valuable as possible;
- Adding or removing items from the Product Backlog as new insights emerge;
- Estimating the effort involved in implementing particular items. This does not have to be as 'formal' as assigning story points (an optional practice in Scrum), T-shirt sizes, or whatever sizing technique you use. A gut feeling ("Yeah, we know well enough what needs to be done and it feels doable in a Sprint") is fine too;

In a sense, items on a Product Backlog are reminders of "conversations that we will need to have in the future". And refinement is the ongoing process of having those conversations. Sometimes this means talking with stakeholders about an item that may end up in the next Sprint, while at other times it can be to clarify an item that the team is already working on. Some of these conversations are with the entire Scrum Team and others with smaller groups. Some refinements can even be done individually.

So rather than understanding Product Backlog refinement as a formalized meeting that happens once during the Sprint, attended by everyone in the team, it should be a series of ongoing activities to refine the work on the Product Backlog for the upcoming Sprints. It's an ongoing activity that happens in many different forms attended by different configurations of people. And it's entirely up to the Scrum Team to determine the best way to do this.

The Scrum Accountabilities

Collaboration between skilled professionals is essential for complex problems. Creative and outside-of-the-box solutions are more easily discovered when you bring together the perspectives of skilled professionals. In order to facilitate this collaboration and reduce complexity in communication and decision-making, the Scrum framework purposefully limits to three accountabilities (or "roles"). But rather than roles in the hierarchical sense, where one has authority over others, each represents a different perspective that needs to be included as a minimum to working empirically. Together, the three are sufficient to work empirically in any environment. No other accountabilities or roles are necessary (and would probably just get in the way).



Product Owner

The Product Owner includes the perspective of what is valuable (and what is not) when it comes to the ambitions for the product. As the Scrum Team spends its time and money working on the product, the Product Owner is there to make sure that this investment will return value to the stakeholders. Close collaboration with the people that have a stake in the product, as well as with the Developers, is important to decide what is valuable and what isn't.

> "The Product Owner includes the perspective of what is valuable (and what is not) when it comes to the ambitions of the product)"

One product has one Product Owner and one Product Backlog with one Product Goal. In order to keep the speed at which decisions can be made high, and adaptation can take place quickly, Product Owners need full mandate over the product. They have the ultimate say over what the ambitions for the product are, what goes on the Product Backlog and what doesn't, and how to spend the budget (or even set it).

The Product Owner ensures that there is a Product Goal, that there is an ordered Product Backlog, and that both are made available to the Scrum Team and its stakeholders. This doesn't mean that the Product Owner is the only person in the Scrum Team who does this work. In order to maximize the value of the work done by the Developers in each Sprint, it makes sense for the Product Owner to actively work with the Developers to write items, refine and order them. It's all about collaboration.

Developers

Developers are all the members of a Scrum Team who actively contribute to the delivery of an Increment that achieves the Sprint Goal. It doesn't matter what kind of work they do or what their formal job title is – for the Scrum framework they are all "Developers". Together, they include the perspective on how to do the work necessary to realize the ambitions for a product and keep its quality high.

Because product development is complex work, even the near future of a single Sprint is hard to predict. It is entirely likely that issues, challenges, and problems will pop up that impede the team in their ability to deliver that Increment. It is also likely that new ideas will pop up during the Sprint about what should be included or left out of the Increment. This unpredictable nature of even a single Sprint has three important requirements for how Developers work and organize themselves.

> "Developers include the perspective on how to do the work necessary to realize the ambitions for a product and keep its quality high."

The first requirement is that they work hard to minimize their dependencies on other people, departments, and skills that are not part of their team but are frequently necessary to deliver a done increment. Each dependency is something they have limited control over and can slow down or completely block them in their ability to deliver a done Increment every Sprint. This negatively impacts their ability to work empirically. Dependencies can be minimized in different ways, from automation (e.g. for deployment and testing) to training new skills or including people with the necessary skills into the Scrum Team.

The second requirement is that the Developers reduce bottlenecks in how skills are distributed within the team. Having one tester in a Scrum Team is better than not having one at all. But if the tester is bogged down with work, the entire team will slow down as a result. The same goes for other skills, like back-end development, design, and analysis. The team works hard to make sure that one or more of the other members within the team are able to do a particular kind of work. The goal here is not to have everyone be equally capable of doing that work – this is obviously unrealistic and disrespectful to the skills people have spent years developing – but to make sure that others can help or take over as needed. The Scrum framework encapsulates this in the statement that 'Scrum Teams are cross-functional'.

The third requirement is that the Developers frequently have to rely on their intelligence and creativity to overcome the many unexpected things that are bound to happen during a Sprint. This is why Scrum Teams should be self-managing insofar as they can make decisions about how to work and what to improve in how that work is done. In self-managing teams, there is no appointed 'boss' who makes these decisions for the team. All the people doing the work – meaning the Developers – work together to make decisions within the boundaries of the Scrum framework.

Scrum Master

The Scrum Master includes the perspective of empirical process control and the quality by which transparency, inspection, and adaptation are taking shape in and around the Scrum Team. The Scrum Master is there to make the elements of the Scrum framework come to life in the team and the broader organization. For this, Scrum Masters adopt a number of stances, depending on the situation they find themselves in:

- Teacher: they teach and explain the purpose of the Scrum framework as a means to work empirically. They work hard to make sure that everyone understands how the artifacts, events, roles, and principles promote empiricism and agility;
- Facilitator: they facilitate the pillars of the Scrum framework so that opportunities for transparency, inspection, and adaptation are maximized. One example of this is the facilitation of Scrum events as requested or needed by the Scrum Team. Another example is to help the Product Owner find and use techniques for managing the Product Backlog and stakeholders;
- Impediment Remover: they (help) remove issues that block the Scrum Team from achieving their Sprint Goals. Scrum Masters help Scrum Teams increase their ability to resolve problems on their own. This is something that teams have to learn, and the Scrum Master helps them do so. What may be considered an impediment during the first Sprint, may have become a problem that the team can easily resolve by itself during a future Sprint;
- Change Agent: they help remove impediments in the environments of Scrum Teams that impede empiricism and agility. For example, some organizations separate 'testing' and 'deployment' as different teams or departments. Or HR-practices reward individuals, not teams. The removal of these impediments is not something a Scrum Master can usually do on his or her own, so they will work together with other Scrum Masters, Product Owners, and others to make this possible;
- Coach & Mentor: they coach the Scrum Team by asking powerful open-ended questions. They are mentors to other Scrum Masters and help members of the Scrum Team find mentors that have the experience and skill to help them;

Scrum Masters are true leaders. Rather than drawing attention to themselves, they help others be as effective as possible. They do not manage or lead the team by telling them what to do or how to do it. The only area where Scrum Masters should take a strong position is when decisions are made that adversely impact the empirical process or the degree to which members of the team feel safe to take interpersonal risks.

> "The Scrum Master includes the perspective of empirical process control and the quality by which transparency, inspection, and adaptation are taking shape in and around the Scrum Team."

Together, the Scrum Team

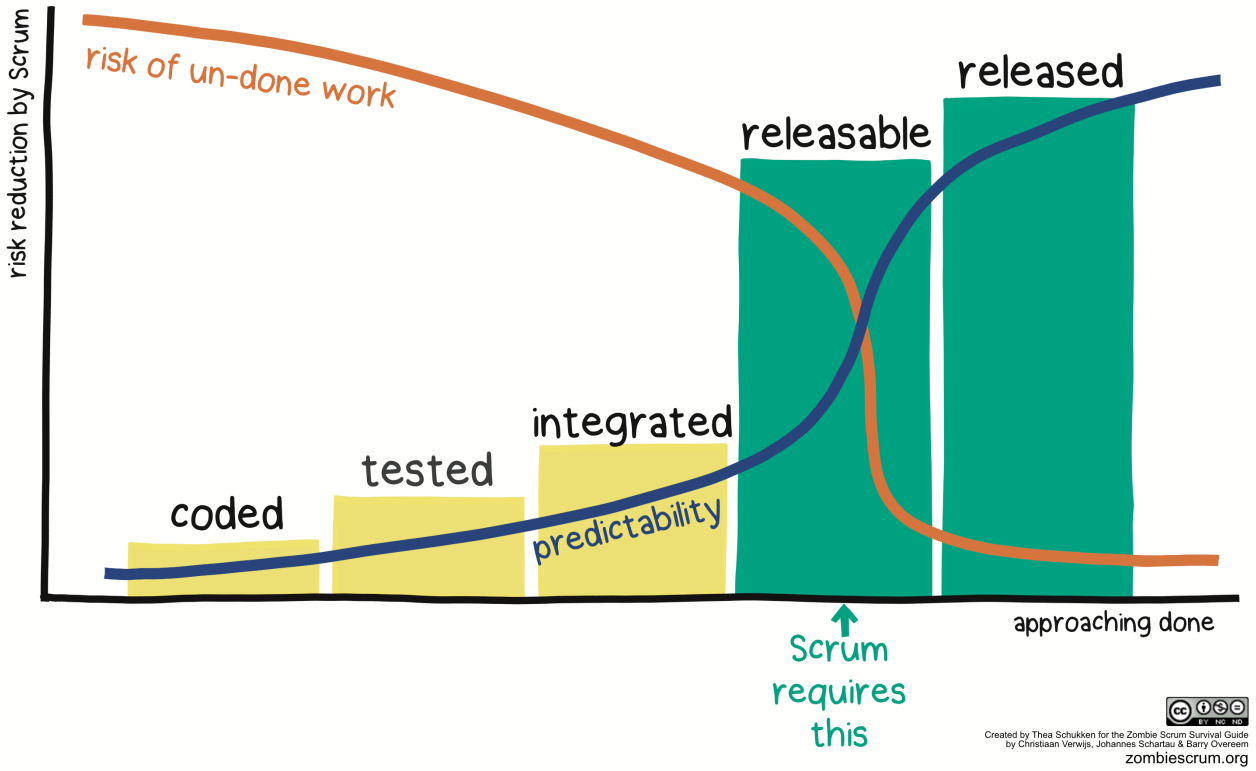
Together, the three accountabilities represent a Scrum Team. They have full authority to make all decisions relevant to their product. The Product Owner decides how to use the budget available to maximize the value for stakeholders. Developers decide how to create that product and collectively have all the skills they need to do so. And the Scrum Master guarantees that this is done in a way that maximizes empiricism. No other roles are necessary.

The Scrum framework is designed in a way that is beautifully captured in this quote by Gunther Verheyen; "in the face of complexity, simplicity is our path". It might be tempting to add roles, rules, practices, and structure to the Scrum framework because you believe your organization to be different. And while some situations may warrant that, maintaining a strong focus on keeping things as simple as possible really is the best way forward in complex environments. This also applies to scaling. Adding more teams adds complexity. Instead, explore ways to do more with fewer people.

Two Driving Principles

When people describe the Scrum framework, they often do so by explaining the events and the roles of the Scrum Framework. Although these elements are certainly important, their effectiveness is determined by how well the teams understand two driving principles.

Deliver a Done Increment Every Sprint



If you were to capture the purpose of the Scrum framework in a single sentence, it would be to work empirically by delivering a Done Increment at least once every Sprint. This is also why it is so important that Scrum Teams spend time clarifying what is involved in creating an increment that is Done. What work is required to do this? Which checks and tests are necessary to conform to our internal quality guidelines? Who needs to be involved in this? This shared understanding is called the Definition of Done, and it usually takes the form of a checklist.

The work on the Product Backlog tends to be chock full of assumptions until they are completed and put into the hands of users. For example, "Will implementing this item improve the experience of users?", "Will users understand how it works?", "Does it perform well enough?" Other assumptions will be about the work that needs to be done to implement the item, like "How easy will it be to test and deploy this item?", "What issues will we run into when working on this item?", "How well do we understand what to build?".

Every assumption represents a risk. The risk that you misunderstood what a user asked for and need to rework it. The risk that you run into technical limitations when you start writing the code. The risk that the feature performs much worse than expected. The risk that writing automated tests turns out to be much harder. Or the risk of spending money and time on a feature that ends up not being used. What these risks all share is that they result in more and unexpected work somewhere down the line. And this kind of "carry-over" work has a tendency to be invisible until it pops up when you least expect it, messing up what you are working on in that Sprint. This is called "Un-done work".

The best way to prevent the risk of un-done work is to make sure that each Sprint results - at least once - in a Done Increment that can be potentially released. That means that the Increment has been fully tested and is working. Texts and visuals are in their final state. Documentation and deployment packages are up-to-date. And performance and security conform to organizational standards. From here, the increment can be released to users with the proverbial press of a button. From this point onward, you know that there will be little to no potential un-done work that can mess up the focus and predictability of future Sprints. More importantly, it allows you to exercise empirical process control by releasing the increment to see if this moves your product closer to its ambitions, rather than assuming that it will.

> "As the ability of Scrum Teams to deliver Done Increments moves to the right, the risk of un-done work decreases. And predictability increases. Inspired by work by Gunther Verheyen."

Creating a Done Increment every Sprint is certainly challenging. But purposefully so. Because putting this amount of pressure on the system that is creating the product - to do all the work necessary to achieve this - shows clearly where improvements are necessary. Where skills, tools, and technologies are missing. Where bureaucracy is getting in the way. Where impediments need to be removed in order to actually work empirically and reduce the risk of complex work. By keeping a laser focus on delivering a Done Increment at least once every Sprint, the system will start hurting in the right spots and create opportunities for inspection and adaptation.

> "By keeping a laser focus on delivering a Done Increment at least once every Sprint, the system will start hurting in the right spots and create opportunities for inspection and adaptation."

Use a Shared Product Goal and Sprint Goal to Create Cohesion

The Scrum Guide mentions the word "goal" 40 times, which is more than any other element of the Scrum framework. Now, we don't want to use this kind of exegesis to drive decisions about how to work with Scrum. But it should tell you something about how important shared goals are.

The reason why the Product Goal and the Sprint Goal are so important goes back to what the Scrum framework is designed for; to more effectively navigate complex problems and reduce their risk. Goals help us navigate this complexity in three complementary ways.

The first is that they offer guidance to the Scrum Team when they have to make decisions about what to spend time on. Some work from the Sprint Backlog may be more important to achieving the Sprint Goal than others. The second is that this allows the team to keep their focus on what is important. When the going gets tough, as it often does, the team can decide to let go of certain work and prioritize other work. Or, when time permits, they can add work during the Sprint that they discover is necessary to achieve the goal. And third, Sprint Goals promote collaboration by giving the Scrum Team a clear and shared purpose to self-organize around instead of working on separate initiatives. Collaboration makes possible the kind of out-of-the-box thinking and team spirit you need when solving complex problems.

The Product and Sprint Goal also give color and purpose to the various Scrum events, elevating them beyond just being meetings about work. Sprint Planning is about determining the goal for the next Sprint and selecting the work that is needed for that. The Daily Scrum centers around how the Developers will spend the next 24 hours working towards this goal, and which impediments are blocking their ability to do so. The Sprint Review is about verifying the result of the Sprint Goal with stakeholders, how it relates to the Product Goal, and working with them to identify the next goals to focus on. The Sprint Retrospective is about discovering ways to work together more effectively to achieve Sprint Goals.

But the Product Goal and Sprint Goal also give focus to the three accountabilities of the Scrum framework. They give the Developers guidance and direction on what to self–organize around. They allow Product Owners to remain focussed on their ambitions for the product and how that translates into objectives for the various Sprints, rather than involving themselves with the details of how to implement that. And finally, Scrum Masters can increase the effectiveness of empirical process control by using the Product Goal and Sprint Goal as the proverbial canary in the coal mine. When teams struggle to create goals, that is a good reason for Scrum Masters to put on their Sherlock hats and go explore what is causing this. Teams may be too large or small, they may lack certain skills or people. The Product Owner may not have a mandate or vision. Or refinement is not taking place.

This explanation should make it crystal clear why there should be only one Product Goal per product and one Sprint Goal per Sprint. Multiple goals only confuse the focus, decrease commitment, and limit transparency. Of course, a new goal can be formulated whenever the previous one is achieved or abandoned.

The short story is that in complex work, shared goals like Product Goals and Sprint Goals are the lighthouses that help you reach the harbor through dense fog. Without them, you're likely to get lost and run ashore.

Five Core Values To Allow Empiricism

Up to this point, we've covered the mechanics and the principles of the Scrum framework. Although they work wonders to allow empirical process control, they wouldn't make much of a difference if there's no behavior that supports it. How honest can the inspection during a Sprint Review be when Scrum Teams are afraid to be open about the technical challenges they see and instead resort to a message that "all is well"? How effective can a Scrum Team be when their Product Owner is afraid to decline or reject items for the Product Backlog that don't fit with the goal? How much value will be generated by a Scrum Team that keeps distracting itself with the newest and shiniest technologies? How does changing the composition of teams without their knowledge or approval affect their willingness to commit to working as a team?

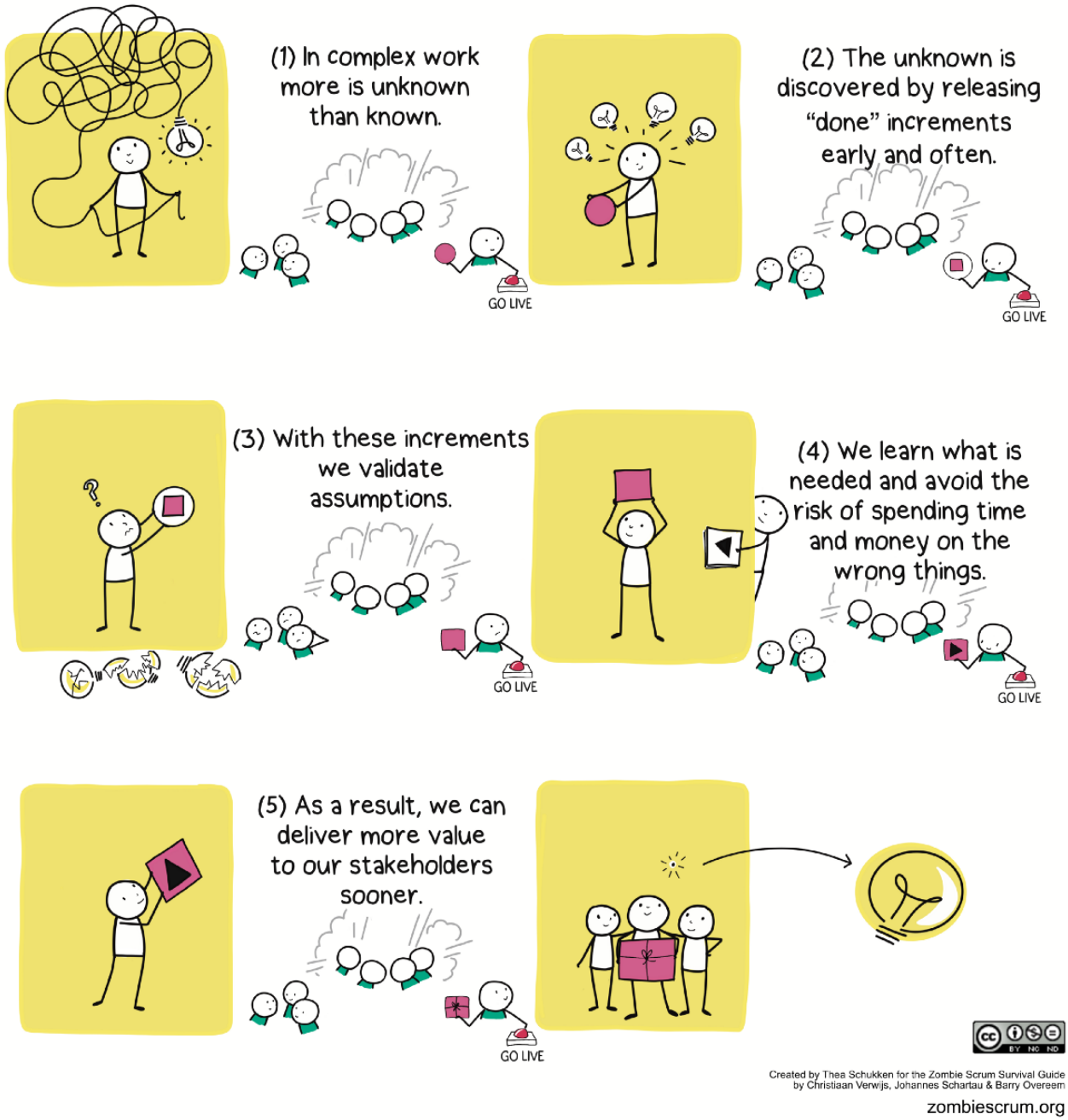
It is tough to work empirically, especially in organizations that are not used to it. In order to give teams and their stakeholders something to drive those decisions, the Scrum framework specifically emphasizes five guiding values. For every decision faced, teams can ask themselves how their options impact the five values:

- Openness: Be open about how things are going. What is going well? What is not? Where are the challenges and opportunities?
- Courage: Be courageous to do the right thing. Say "No" to things that impede the empirical process. Show courage by working on tough challenges together. Ask for, and give feedback about things you are not sure about. Ask questions and admit what you don't know or are uncertain about;
- Focus: Keep the focus on the Sprint Goal and the goals of the Scrum Team. Create a space where people can keep and sustain focus;
- Respect: respect the skills, expertise, and intelligence of the members of the Scrum Team. Trust their ability to self-organize around complex problems. And respect the uncertainty that is inherent to complex work;
- Commitment: Create an environment where people can personally commit to working together as a team towards the Sprint Goal;

Learning to espouse these values in day-to-day behavior is a lifelong journey. The more proficient teams become at them, the more effective frequent transparency, inspection, and adaptation will be. The great news is that the Scrum framework provides an excellent set of boundaries to learn and grow in espousing these values.

Together, A Simple Framework

Scrum is a simple framework for navigating complex, adaptive problems. It only prescribes what teams should do to work empirically, but not how they should do it. Because every team, every product, and every organization is different, teams have to find their own way to make this work for them. When they do, they can reduce the risk of complex work, start delivering value to their stakeholders sooner, and become more responsive. This journey will be easier for some Scrum Teams than for others, but change will follow when you persist.



Unleashing Teams All Over The World

We are The Liberators – Barry Overeem and Christiaan Verwijs. Our mission is to create data-driven products to unleash the superpowers of teams all over the world. We do this together with a growing community of patrons.

Awesome content for awesome teams

We unleash teams with our blogposts, our podcast, our newsletter, our videos, and our frequent meetups. While we offer most of this for free, we also have plenty of premium content in our webshop for you to explore.

Supported by the community

We are super proud that The Liberators is almost entirely funded by the community. If you appreciate our work too, you can already support us for 12 dollars/year by becoming a patron. In return, you gain free access to premium content, we share our work-in-progress and involve you in creating more awesome content.

Thank you for respecting our work!

We work hard to create high-quality content that puts you in a position to unleash your team. We're sure you appreciate that a lot of our time and money goes into this. At the same time, content like this is our main source of income. It's what pays our bills :)

If you purchased this content, we ask only that you treat our work with respect and don't share it with people outside your team. If you stumbled on it elsewhere without paying for it, and it offers you value, would you consider supporting us too? You can also check out our other offerings.