New Project Structure—Agile & Scrum

Eljar Akhgarnush, Fabian Bruse, and Ben Hofer

1 Need for a New Project Structure

1.1 Cultural Change

This buzzword echoes throughout the literature and media, as it affects an increasing number of companies regardless of industry and size. Even more so, cultural change fuses into a crucial aspect of project management since it accounts for the success or failure of the outcome. Establishing the facilitating atmosphere, where the strengths of a company's culture are embraced, while weaknesses are mitigated, requires a project management approach which is prone to change. This, in turn, enables the teams involved to be adaptable to the business environment.

One of these methods—and the one that is most applied within the agile project management terminology—is Scrum (Petrova 2019). It breaks down rigid structures like conservative hierarchies, re-prioritizes from stability to functionality and agility, moves away from top-down to bottom-up and hybrid decision-making, emphasizes the market or customer to deal with rapid changes of the environment and introduces iterative processes as well as dynamism and scaling flexibility. The implementation of the latter three

E. Akhgarnush · F. Bruse (✉) · B. Hofer
ifb SE, Grünwald, Germany
e-mail: Fabian.Bruse@ifb-group.com

V. Liermann and C. Stegmann (eds.), *The Digital Journey of Banking and Insurance, Volume I*, https://doi.org/10.1007/978-3-030-78814-8_12

factors in particular ensures learning constantly and therefore deducing insights to optimize processes and, in the end, the outcome.

1.2 The Guiding Framework

Scrum consists of more than just trivial rules that supposedly lead the teams through projects. It is a framework consisting of work culture and ethics accompanied by tools and additional methodologies (Wijetunge 2019) that are meant as complementary additions to give Scrum a specific structure, enabling the realization of principles and values even, or rather especially, within highly complex projects. One of the main reasons why complex projects can be simplified within Scrum is the provision of a potentially shippable increment, a partitioning of the work packages, so to speak. Since Scrum is supposedly the most used agile methodology in projects (Lamelas 2018), this article takes a closer look at its content, describing how it works and providing practical experience.

2 Agile Principles and Values

The following section deals with principles and values included in the work culture and ethics of the Scrum methodology. They consist of the Core Scrum Values, the Agile Manifesto and the Scrum Pillars. The former represent the overall Scrum foundation and have been deduced from the Agile Manifesto. Therefore, it makes sense to introduce the latter first, then address the derived Core Values and afterwards briefly explain the link to the Scrum Pillars, which wrap up the other two.

2.1 The Agile Manifesto

Starting off with one of the most recent essential milestones of the agile movement, the Agile Manifesto from 2001: It lays out priorities upon which the Scrum Team should act (Beck et al. 2001) and refers to deductions taken from practical experience. They therefore represent best-practice maxims, so to speak. The following figure (Fig. 1) sums up these four maxims throughout the project development. It is worth noting that, in Fig. 1, all the aspects of each maxim mentioned are of significance, but the ones mentioned first (in blue) are considered more crucial for the success of the project.

Fig. 1 The Agile Manifesto (© ifb SE)

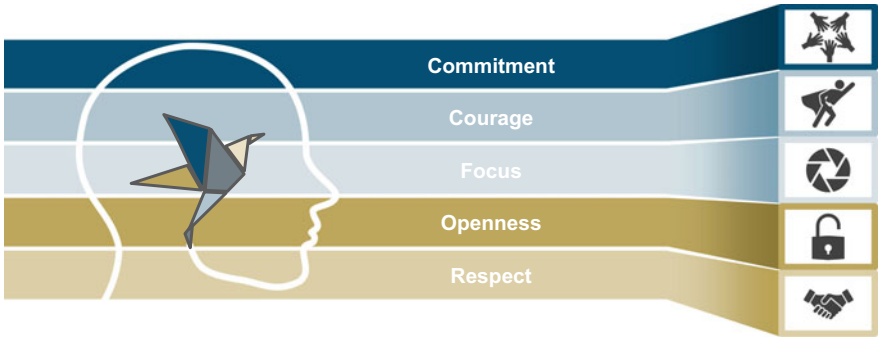

Fig. 2 The five Core Scrum Values (© ifb SE)

2.2 Scrum Values

Probably the most important part of the Scrum methodology, though, are its five Core Values (Schwaber and Sutherland 2017), without which it would be robbed of the foundation on which all the other ideas are based. The following figure (Fig. 2) illustrates the values mentioned enabling the Scrum participants to excel.

The Scrum Values of commitment, courage, focus, openness, and respect are mutually interdependent, with some weighing more than others. Furthermore, when the values are embraced and implemented by the Scrum Teams, the Scrum Pillars of transparency, inspection and adaptation take effect, ensuring trust within the teams. The Scrum Pillars, in this respect, represent the framework of the methodology.

While the realization of all the Scrum Values relies on the responsibility of each team member themselves, there are varying degrees of design, or rather influence, that some Scrum roles can exert more than others, facilitating the

successful implementation of values. So, what do each of these five terms represent exactly? Why do they exist? And how are they linked with each other?

- **Commitment**: True to the mantra that nothing beats diligence and grit, the methodology requires each team member to be highly engaged in the sense of being inherently motivated and showing unity as well as solidarity. This condition cannot be achieved by external force, for example by command or order. Instead, it is crucial to provide and establish the parameters under which all team members feel comfortable and the necessity to act self-directed. One way to maintain these framework conditions is to have regular conversations and listen carefully. But, of course, commitment does not stand alone here.
- **Courage**: Whenever individuals work closely together, courage needs to be their constant companion, as it is the beginning of the action, which leads to contentment in the end. It is often a fine line between courage and high spirits, which means that the Scrum Master as well as the other team members must maintain a healthy balance. Courage within the Scrum methodology—and probably beyond—means not only speaking the truth, but also risking mistakes and taking unsafe, unknown, or unpleasant paths, for example by saying "no" when necessary. The term courage enriches the value commitment with resilience, increasing the chances for success. However, commitment and courage are just a portion of the complete Scrum picture.
- **Focus**: As distraction can lead to the destruction of productivity, there is an overwhelmingly strong argument to include the value "focus" on the list. This term contains the inherent self-discipline of each team member as it does the external consideration to refrain from disturbing the team. Maintaining or enabling a maintaining of the focus within the team is one of the most significant factors the Scrum Master can control by, for example, keeping away any occurring impediments. Nevertheless, the values listed so far are barely enforceable without the next two.
- **Openness**: Smart solutions to complex challenges can be tapped with an open culture prevailing within teams. To delve deeper, establishing welcoming communication for topics like criticism, opinions, etc., enables all team members to be committed, have courage and stay focused. Openness is, in a sense, a tool for introducing honesty, fairness and information as well as knowledge exchange and that way improving the productivity of the team. After all, this term is closely related to—or rather a precondition

Fig. 3 The interplay between the Scrum layers (© ifb SE)

for—the success of every collaboration project and hence one of the most important values within Scrum, right after respect.

- **Respect**: Everything within the agile methodology comes down to respect. Being open, committed, focused, and courageous are in essence an act of respect toward the team members and oneself. Respect, in turn, stems from taking responsibility for one's own actions. In that sense, addressing and agreeing on guidelines or considering potential personal values of mutual respect in the very beginning of a new Scrum Team can facilitate later problem-solving considerably and prevent possible conflict situations from arising.

Now that the five Core Values of Scrum have been addressed, it is worth mentioning that the various principles and values are laid out as an interplay between each other without a rigid and self-containing structure. Therefore, in the following figure (Fig. 3), the Scrum architecture is summed up as a circulating and open illustration emphasizing the iterative and reciprocal interdependence of each Scrum layer.

In the next section, the theoretical foundation of the Scrum framework is introduced including different roles, events and tools encountered when using this method.

3 The Scrum Framework

3.1 Scrum Teams and Roles

Within the Scrum Team, only three roles are defined (where "Scrum Team" refers to everyone internal to the project team): the Product Owner, the

Scrum Master, and the Development Team. To stay in line with the philosophy of Scrum, one should not define any roles other than these three. A team member usually has exactly one of the three roles, even though there might be good reasons to fulfill more: a common example is a Scrum Master who also supports the Developers. All other people involved in the project, such as stakeholders or management, do not belong to the Scrum Team, as they are not considered internal to the project and only appear for certain events.

In order to be flexible, creative, and productive, i.e., to create an agile environment, Scrum Teams are supposed to have two key characteristics: self-organization and cross-functionality. This means that the team manages itself in terms of priorities, how things are done, who is doing what, etc. Additionally, it has all the expertise to finish tasks without help from outside the team (see Dräther et al. 2013; Rubin 2013; Stellman and Greene 2013).

3.1.1 Scrum Master

The Scrum Master is a servant leader and protector for the Scrum Team who fully understands the Scrum framework and helps the team to perform at their highest level. This role is in charge of the Scrum processes, coaches the team through them, and makes sure that they are implemented and understood correctly within the team and the company. The Scrum Master therefore ensures that important Scrum events take place (within their timebox) and tries to facilitate these events. Scrum Masters usually have a big toolbox of communication and planning techniques, as well as mediation experience to support and coach the Product Owner, if requested, or to remove impediments to the Development Team.

The Scrum Master is not limited to his own team. If a project does not require 100% of the role's attention, it is possible for the Scrum Master to work for several teams, or to help others in the organization to understand Scrum and how to interact with Scrum Teams.

3.1.2 Product Owner

As the name suggests, the Product Owner "owns" the product and is therefore responsible for the output of the project. Sometimes this role is described as the project manager because it is the most business-oriented, but in fact in Scrum there is no role like a traditional manager. However, the Product Owner's goal is to maximize the value of the work the rest of the team does.

Therefore, he is in close contact with the product's customers and stakeholders rather than having hands-on application or development knowledge. While in contact with customers, the Product Owner identifies items (so-called User Stories) that have to be implemented to satisfy their requirements. These are prioritized in a Product Backlog, one of the main artifacts and planning tools of Scrum. The Product Owner creates and maintains this Product Backlog but does not manage the daily activities of the team. His decisions for the project and the Product Backlog must be respected by the entire organization. This also means that not even the CEO should approach the Development Team directly to tell them what to deliver but rather discuss issues with the Product Owner.

3.1.3 Development Team

The Development Team (usually three to nine people) is responsible for delivering the Backlog Items entirely and for organizing itself to accomplish this task. The team therefore should consist of experts capable of doing everything from, for example, analyzing and programming in different systems to testing and documentation. This is called a cross-functional team. During the Sprint, specific team members might work on a task—a team decision, not a management order—but it is the team as a whole that will always be accountable and responsible for the task.

In order to plan and prioritize tasks, the team relies on Sprint Goals that have been discussed with the Product Owner. During that Sprint (a two- to four-week project period), the Development Team will focus on creating a potentially shippable increment piece of the product. That way, the new functionality becomes transparent to the customers quickly.

What about more specific roles? It might surprise you, but there are no other roles in Scrum. One should not try to tag team members with titles like tester or team leader. Everyone is just a Development Team member. This highlights the fact that the team needs to have a united and profound understanding of the tasks. This ensures that the Sprint Goal stays in everyone's focus, rather than specific roles or jobs.

3.1.4 Where Is the Management?

As you might have noticed, the tasks of this traditional role have been distributed among the three Scrum roles. Short-term planning and implementation is carried out by the Scrum Team, creating a product vision and

Table 1 Scrum Artefacts explained (© ifb SE)

Artefact	Description
Product Backlog	Ordered wish list of everything needed to finish the project. This list should be written in a non-technical way and always be up to date. The Product Backlog is "owned" by the Product Owner and can change dynamically. Backlog Items should also have a work estimate, e.g., Story Points, for comparative reasons
Sprint Backlog	Recreated in every Sprint Planning. This contains the Sprint Goal, selected stories from the Product Backlog for this Sprint and a plan for how to deliver them
Increment	The sum of all completed Product Backlog Items of the Sprint, plus the Increments of all previous Sprints. The Increment should be potentially shippable, but it is up to the Product Owner to release it. The very first increment is often the MVP (minimal viable product)
Definition of done	The company and the Scrum Team should have a common and transparent understanding of when a Story can be declared "done." What kind of tests have to be performed (integration testing, positive/negative test), the kind of documentation, update of specific charts and tables, etc.

taking decisions about the product, as well as communication with customers are carried out by the Product Owner, while coaching and dealing with problems is carried out by the Scrum Master.

However, managers do exist, but outside of the Scrum Team. They can help to energize people and promote the values of Scrum. They define the team's boundaries and are still responsible for the personal development of the team members (Table 1).

3.2 Scrum Events

We talked about a Scrum Team that has to self-organize and know a lot about the product, but how can this work if no customers or stakeholders are part of the team? And how can the team benefit from the key pillars mentioned: transparency, inspection, and adaptation? For this we must have a closer look at the Scrum lifecycle and the related events.

3.2.1 Sprint

The work, from the first project kick-off to the final product, is done in small increments in order to break down a complex product idea into manageable

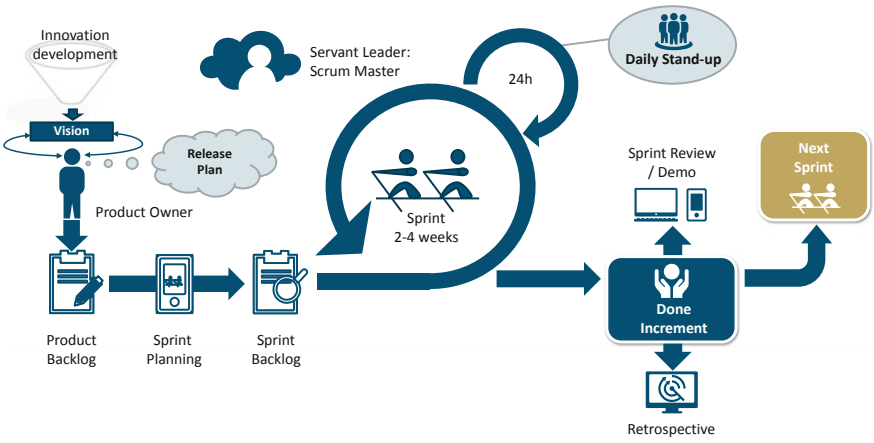

Fig. 4 A visualization of the Scrum process (© ifb SE)

chunks. This is one of the reasons why Scrum is said to be particularly useful for chaotic and complex projects, where either the "how should it be done?" or the "what should be done?" is unclear. Each Sprint (i.e., every two to four weeks), an increment of functionality should be potentially releasable and add value to the previously done increments for the customer. All desired functionalities are transparent in a Product Backlog that the Product Owner maintains and that evolves with the product. The increment of a Sprint will be the sum of completed Backlog Items plus all the increments of previous Sprints (for the whole process, see Fig. 4).

After the Sprint period has started and tasks have been selected for the Sprint (see Sect. 3.2.2 Planning), they should not be changed anymore, so that the team can focus. If questions arise while working on the topics they can of course be addressed and clarified. However, within the scope discussed, the story has to be 100% done at the end of the Sprint. It cannot become a part of the current increment if it is "almost" done.

If it should turn out that the Sprint Goal cannot be reached or has become obsolete, the Product Owner has the authority to cancel the Sprint entirely and restart. Otherwise, the Sprint is a frozen period where the Scrum Team focuses on reaching its Sprint Goal.

3.2.2 Planning

How do Backlog Items become part of the current Sprint and how does the Scrum Team know what to do? For this, Scrum has an event fittingly called

Planning. In this meeting, which all three Scrum roles attend, the Development Team first estimates the work capacity for the next Sprint period and then selects an appropriate number of Product Backlog Items into their Sprint Backlog. The Product Owner should have made sure beforehand that the Product Backlog is ordered by the value of the User Stories, so that the Development Team will work on the most important topics first. With the stories in mind, the Scrum Team should think about a Sprint Goal that allows the Development Team to plan their work within the Sprint and guide them on what they are building and why.

After the Backlog Items are selected, the team plans how to turn them into a done increment of the product in accordance with the Sprint Goal. The amount of detail discussed here should be enough to get the Development Team started for the first couple of days and longer; the rest can be prepared later.

3.2.3 Daily Scrum

During the Sprint, the Development Team meets every day for 15 minutes to inspect their work and to plan the day. This is called the "Daily Scrum." The Scrum Master should make sure that, firstly, the meeting will take place, and, secondly, it can be held at the same place and time to reduce complexity. During the meeting, it is suggested that each Development Team member answer "the three questions": What has been done since the last meeting? What will be done before the next meeting? Are there any obstacles blocking me proceeding toward the Sprint Goal? A lot of teams combine this with some kind of planning board, like a Kanban board, to make the current status of all User Stories transparent to everybody.

Please note that this should not be a traditional status meeting with stakeholders and customers. The Daily Scrum's purpose is to synchronize the work of the Development Team.

3.2.4 Sprint Review

At the end of a Sprint, it is time to inspect its outcome, the increment of done functionalities. Everyone—the team, stakeholders, and customers—meet for a maximum of four hours where the Scrum Team demonstrates its work. The idea is not to waste time preparing a labor-intensive presentation. For example, if you work with software, the increment is commonly shown live in the system.

The Review is the main event to gather feedback, adjust the strategy and update the Product Backlog with new requirements. The customer gets the chance to see real new product value early and to check it regularly. It is important to mention that only completely done User Stories that fulfill all the requirements from the Product Backlog, as well as general organizational requirements ("definition of done") can be presented and approved.

After the Sprint Review, everyone should have a common overview of the project status.

3.2.5 Sprint Retrospective

While the Sprint Review is an inspection and adaptation event for the product, the Retrospective is one for the Development Team. Directly after the Review, the team will hold another meeting to talk about things that went well or badly during the last Sprint. Think of it as a regular "lessons learned" workshop. This is the time to inspect the team's processes, tools and relationships and think about how to improve them. This meeting of up to three hours can become quite emotional and stressful, so remember the most important value of Scrum: respect. It is also the Scrum Master's time to shine, with the role's ability to mediate and a large toolbox of feedback and brainstorming techniques.

The output of the Retrospective should be at least one thing that the team will try to improve in the next Sprint, no matter how small the improvement might be ("kaizen").

4 Scrum—The Holy Grail of Agile Project Management?

Like with medicine, Scrum can cause a great improvement—if applied correctly—but can worsen things just as easily in some cases. So how can we know when to use Scrum? First of all, experience with different projects

and methods will help. Luckily, a lot of people have thought about this matter already. Consider the (Ralph Douglas) Stacey Matrix (Fig. 5) and answer the two questions: "How clear is it what has to be done in the project?" and "How clear is it how it should be done?" (Stacey 1996). If the answers are "it's all very clear," a traditional approach, or even no project management at all, probably suits you better. In that case we are in a simple project situation and can work with a straight plan. This can be the case for routine work, or another iteration of a project that has been done already.

If the situation is more complicated, one can differentiate: if the goal of the project (the what) is unclear, we are in the area of political decision-making. It is important to compromise and stay in close contact with the stakeholders. If we are unsure about how problems can be solved, we are in the area of judgmental decision-making and it is all about discussions with technical experts to find alternatives and best solutions. In these situations, traditional or agile methods can be equally beneficial.

If either the "how" or the "what" become completely unclear and—worse—requirements are constantly changing, we find ourselves in a complex situation. It becomes practically impossible to estimate the time and cost of the project. This is when one should not try to order the chaos; planning anything too far into the future here is a waste of time. This is the perfect

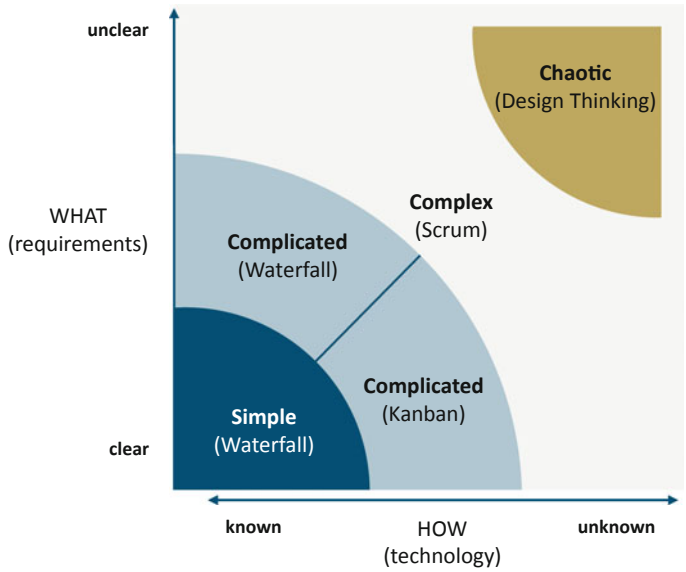

Fig. 5 The adapted Stacey matrix (© ifb SE)

environment for an iterative approach where small pieces of functionality can be inspected and adapted regularly. It is the perfect setting for using Scrum.

If nothing at all is clear, the situation is chaotic: a very risky endeavor to start a project. Try to clarify the goals of the project to order the chaos, for example with the Design Thinking method.

Apart from this theoretical analysis, there are other things to consider as well while choosing your project approach. The (potential) Scrum Team itself is key and needs to have a certain set of skills to support Scrum. A certain level of seniority is required for being able to self-manage and self-organize. With an inexperienced team, therefore, the correct implementation of Scrum might become more complex. Also, the Product Owner's availability for the project is an important factor. He creates the Product Backlog and prioritizes the User Stories, while needing to spend a lot of time communicating due to meetings with stakeholders and the developers. The Product Owner will decide how to "cut" the stories and therefore play a major role in how the increments are created. If the Product Owner is not available for the project, it is difficult to be successful and to keep the big picture in mind.

On the other hand, Scrum also needs to match the client's culture. Do they have experience with Scrum or are open to innovation and change? With Scrum, it will be difficult to plan the exact scope (or cost) of the project given the agile nature of the methodology. Trust is key so that everyone understands that the team will do its best to achieve its goals.

In some cases, it might be beneficial to use a hybrid method to mitigate some of the issues mentioned (see Beister and Zeljkovic 2021).

5 Common Mistakes and How to Avoid Them

With the introduction and use of any new method, misunderstandings and mistakes can never be avoided completely. For this reason, we would like to give you an understanding and some examples of the most common mistakes and challenges to deal with from our experience.

Most importantly, Scrum must not be mistaken as a universal remedy. An existing project or even an entire organization cannot simply be converted into an agile approach overnight. Why? Long-established processes, outdated structures, and thought patterns need to be broken up first and a rethinking process must be initiated. This process takes several weeks to months—possibly years—and requires a willingness for a change in attitude and mindset for the change to be successful.

One of the main challenges in introducing agile methods is creating a common sense for both principles and necessary changes. Only if everyone is willing to leave old habits behind and allow new things to happen will the introduction be successful. For this, efficient change management and training for both management and employees are essential. This importance becomes particularly clear when we compare the priority with those in aviation: of course, all passengers expect above all to reach their destination safely. Self-realization, creativity, and long-term commitment are not important for a single flight. To prioritize safety, the pilot should always first aviate, then navigate, then communicate—simple as that. To make a change successful, the sequence must be different: today, employees not only attach importance to a secure job and salary payments, but also to the working atmosphere, integration, and motivation. Hence communication is key. Only if everyone is well informed can a change or project manager safely navigate and lead the people involved, and then ask for their support to make it a success. We recommend paying special attention to the introduction of the framework, in particular the Scrum process, values, and roles behind it, in order to make the introduction of Scrum or other methods successful and sustainable.

You can find some other aspects and common mistakes in Table 2.

6 Methods and Tools of Effective Scrum Teams

6.1 Agile Metrics

Agile project management is often criticized for not allowing a clear schedule and budget. Agile Metrics can help solve this discrepancy. In particular, three Agile Metrics will be highlighted here that have proven themselves in practice for planning, controlling, and reporting.

6.1.1 Story Points

With constantly changing requirements and a high level of uncertainty, it is almost impossible to create a reliable and absolute estimate based on time expenditure only. For this reason, a second dimension—complexity—should always been considered.

In classic approaches, complexity is often added to an absolute estimate as a risk surcharge. With agile approaches, both dimensions are united in a so-called Story Point estimation. Story Points are always considered relatively,

Table 2 Common mistakes with prevention and improvement options (© ifb SE)

Aspect	Problem/common mistake	Prevention/improvement options
Agile Manifesto and values	Agile principles and values are not understood by all team members	Let the team take the Scrum Values as a basis for setting up their own rules. Next, make these visible to the Scrum Board and verify their compliance in the Retrospective
Meetings and sequencing	Meetings follow each other directly, without any breaks or set-up times. The team rates meetings as more disruptive than profitable and preparation suffers	Arrange 10–15 minutes' break time between meetings to ensure a clear separation of content and allow for set-up times and short preparation
Timeboxing	The set duration of meetings is regularly exceeded in order to achieve supposedly determined content goals	Ensure that the timebox is consistently adhered to, e.g., with a clock/countdown, and refer to follow-up meetings in which only affected stakeholders can participate
Moderation and coaching	The team repeatedly loses focus in meetings and drifts off into detailed discussions	The Scrum Master should act as a servant leader, identify such situations and help the team to regain its focus
Impediments	Impediments are either not documented or at least not tracked visibly. Measures are not taken consistently	Create an impediment backlog and make it accessible to everyone. Prioritize the impediments by degree of disability or severity of impact and log the date of first occurrence and progress

comparing two or more tasks to each other. In order to not fall back into sole estimation of time expenditure, Story Points are deliberately not assigned linearly, but along the Fibonacci number series (0, 1, 2, 3, 5, 8, 13, …). The more iterations the team runs through, the more precise the "gut feeling" for allocating Story Points will get—and thus the estimation itself.

Important: Story Point estimations are only valid for the team that carried out the estimation since each team establishes a unique estimation culture.

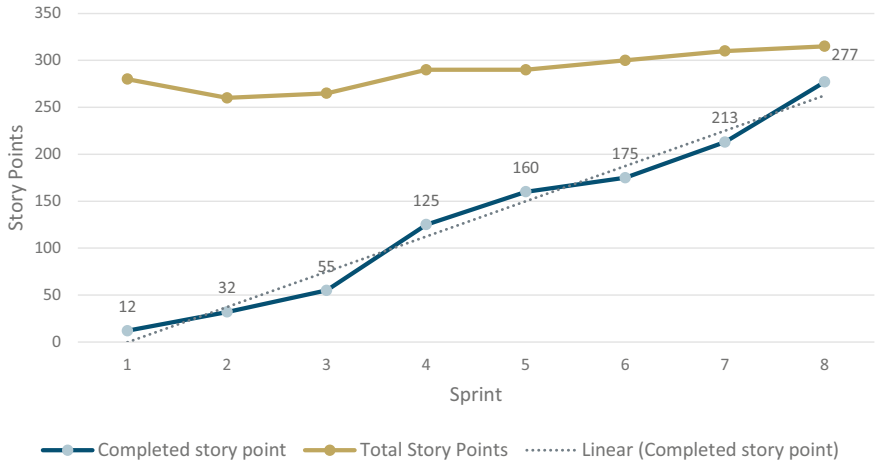

Fig. 6 Burnup Chart (© ifb SE)

Fun fact: Some teams have completely dropped estimation with numbers to get rid of the temptation to make absolute estimates. Instead, they estimate using animal sizes from ants and mice to dogs and elephants to blue whales—just to focus on the relativeness.

6.1.2 Burndown/Burnup

A Burndown Chart is a great tool to show progress during a Sprint. It shows the completion of work, usually measured in Story Points. The *x*-axis shows the time and the *y*-axis shows the work that still needs to be completed, measured in Story Points or hours. With this report, deviations can be identified immediately and measures can be taken to restore effectiveness to not endanger the Sprint target. The same is true for a Burnup Chart (Fig. 6), where the progress of the whole Product Backlog is shown in an inverted Burndown Chart. Deviations from the usual velocity can be detected, both positive, if the curve rises steeper, and negative, if the curve flattens.

6.1.3 Velocity

Velocity is defined as the average amount of work a Scrum Team completed during a Sprint, measured in Story Points or hours. This metric is very useful for forecasting. The Product Owner can use the velocity to predict how quickly a team can process upcoming work in the backlog, which has been

estimated by the team beforehand. The more iterations, the more accurate the estimations and the more accurate the forecast will be.

Say the Product Owner wants to complete 450 Story Points in the backlog. We know that the Development Team typically handles 45 Story Points per iteration. So, the Product Owner can pretty much safely assume that the team needs approximately 10 iterations to complete the required work.

It is important to monitor how the velocity develops over time. In new teams, an increase in velocity can be expected as long as the team establishes and optimizes both relationships and processes. Existing teams can track their velocity to ensure consistent performance over time. They can also use the velocity to identify whether or not a particular process change has brought improvements. A decrease in average velocity is usually a sign that part of the team's development process has become inefficient and should be discussed at the next Retrospective.

Be aware, the velocity of each team is unique. A difference in velocity between teams does not mean that the throughput in a specific team is higher. Therefore, a comparison of velocities is not possible. Effort and output of tasks should be measured based on the interpretation of Story Points unique to each team.

6.2 Additional Artefacts

6.2.1 Impediment Backlog

In addition to Product and Sprint Backlog it is advisable to also establish an Impediment Backlog. This backlog lists every impediment that negatively affects a Scrum Team but cannot be solved by the Scrum Team itself. The Scrum Master manages this list, tries to solve the issues or enables the team to solve them. Besides a description of the problem, the date of its first occurrence, a responsible person, and a status should be tracked.

6.2.2 Scrum Board

This auxiliary component serves as a clear display of current task statuses during a Sprint. The legitimate question about the difference between a Scrum Board and a Kanban Board is quickly answered by a look at the two methodologies: while Kanban focuses on continuous product improvement, Scrum is best suited to iterative product developments with defined

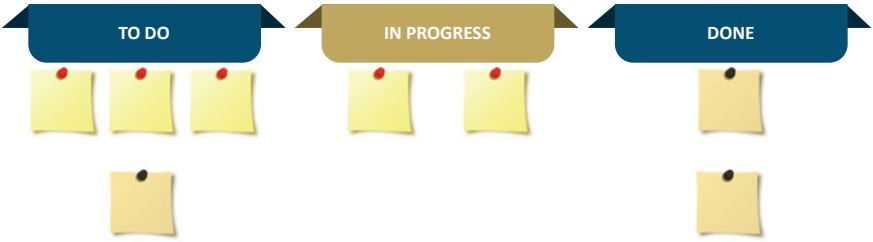

Fig. 7 Simple Scrum Board variation (© ifb SE)

durations. The figure (Fig. 7) depicts a simple Scrum Board variation with differently colored post-its (e.g., for different kinds of task).

6.3 Daily Routines

In order to make the Daily Meeting and team collaboration more efficient, agile teams have developed multiple routines. We recommend establishing these routines in your daily work to increase effectiveness:

- In the Daily Meeting, all team members stand up and use an object (pen, ball, etc.) to pass the word around. This helps to prevent talking at the same time.
- The central questions for the Daily Meeting are pinned somewhere visible for the whole team, so everybody can focus on answering them:
  - What did I do yesterday?
  - What will you do today?
  - Are there any impediments in your way?
- Colors are used on the Scrum Board to help categorize User Stories and tasks.
- The team establishes rituals like joint coffee breaks or leisure activities to strengthen team spirit and enable an informal exchange of information.

6.4 Estimation Methods

Could you estimate the exact length of the room you are currently in? Or the weight of a whale? Probably not. As it turns out, it is much easier if you have something to compare it to. Like how many tables could fit into your office? Or how many times bigger is the blue whale than the orca swimming right

next to it? Similar (relative) estimation approaches are used in agile methods to determine the complexity (not duration!) of a User Story.

6.4.1 Planning Poker

Every team member is given a special deck of Planning Poker cards with numbers that represent the complexity of a story (Story Points), plus additional cards to represent "no idea," "infinity," and "I need a break." The values used are the beginning of the Fibonacci series (as mentioned in Sect. 6.1.1) to indicate that it does not make sense to make an estimate in too much detail for high-complexity stories, as there will be a high amount of uncertainty involved. When it is time to estimate Product Backlog Items, all team members pick a card, depending on how complex they think the item could be. This evaluation is based on their personal experience with previous User Stories. The team members' choices are then disclosed simultaneously. If the values are too different, the person with the highest and the lowest number should share their opinion on the complexity level of the User Story. This is a playful way of detecting hidden stumbling blocks and ambiguities in a Backlog Item. The estimation is repeated until the team comes to a common understanding of the User Story.

6.4.2 T-Shirt Sizing

T-Shirt Sizing is another estimation technique to categorize User Stories. Rather than using numbers like in Planning Poker, items are classified into T-shirt sizes (XS, S, M, L, XL). This less numeric approach can help some teams to think in a more dynamic manner and avoid confusion between Story Points and story duration or effort. It is a great introduction to relative estimation for new teams.

6.5 Retrospective Methods

A good Scrum Master has a big toolbox of methods and games for Retrospectives that he can apply, depending on the project situation. Since a Retrospective is a key event for inspection and adaptation, the goal is to bring to light unspoken issues that are on the team's mind and to identify what worked well or didn't work well in the last Sprint. Instead of directly asking questions like "what do we have to do differently in the future?", it is more goal-oriented to split the Retrospective into five phases. Each phase

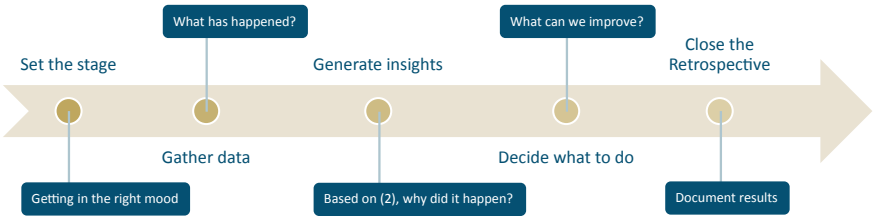

Fig. 8 Retrospective—five phases (© ifb SE)

consists of a task or a game to eventually reach a common understanding of potential improvement. Let's have a look at these phases (Fig. 8) and give one example of how this phase could be executed in your Retrospective.

- Setting the stage to enable the participants to mentally adjust to this event and to highlight that something other than their regular work is about to happen. Game idea "Weather Report": Prepare a flip chart with a drawing of storm, rain, clouds, and sunshine. Each participant marks their mood on the sheet.
- Gather data to remind people what has been done in the Sprint and to find out what they would like to talk about. Game idea "Tweet my Sprint": Ask participants to write three Tweets on sticky notes about the iteration they have just completed. Tweets could be about individual stories, a rant, or shameless self-promotion. Hashtags and emoticons are all welcome. Arrange the Tweets in a timeline and discuss themes, trends, etc. Now invite participants to mark their favorites, retweet, and write replies to the Tweets.
- Generate insights based on the gathered data to find the root cause of an issue. Game idea "Election": Split into political parties with two or three members. Work on a manifesto for change. What isn't working? How would they improve things? Present the manifestos to the group and summarize them with sticky notes, one color per party. What do the parties agree on? Which promises are unrealistic and which can you achieve?
- Decide what to do based on the generated insights. What is the top priority, what would you like to try? Game idea "Low Hanging Fruit": Draw a tree on the whiteboard and write the ideas and actions of the last game on cards. Read the shuffled cards one by one and place each "fruit" according to the assessment of how easy it is to achieve (lower placement) and how beneficial it would be (place it more to the left). The straightforward choice is to pick the bottom left fruit as action items. If there is no consensus, you can take a quick vote.

- Close the Retrospective. Like every good meeting, the Retrospective needs a closing. Document the results and what you would like to try in the next Sprint. Try the game "Retro Darts": Draw a dartboard on a flip chart. Write a statement next to each dartboard, e.g., "We talked about what's important to me," "I'm confident we'll improve next iteration." Participants mark their opinion with a sticky note.

Many more interesting games can be found on retromat.org (Baldauf, n.d.), a great and continuously expanding source of ideas in different languages.

6.6 Timeboxing

Timeboxing is the pre-allocation of a limited period of time that Scrum events may last at the most. This includes the following timeboxed events within the methodology: The Daily, Sprint Review, Sprint Retrospective, Sprint Planning, and the Sprint itself. Implementing a timebox aims to use resources—in this case "time"—efficiently and simultaneously motivates the team and increases their focus. Like in all project management frameworks, "time" is crucial within Scrum, too.

7 Other Agile Approaches

Besides the Scrum Framework, other agile approaches have become popular in recent years. Projects are increasingly aligned to these new approaches, with the goal of becoming faster and more flexible.

7.1 DevOps

The aim of DevOps is to combine software development (Dev) and operations (Ops) to shorten the development lifecycle and provide continuous delivery and integration but also to raise software quality. A toolset, also referred to as toolchain, consists of different categories from Coding, Building, and Testing to Packaging, Releasing, Configuring, and Monitoring. As neither academics nor practitioners have developed a unique definition of the term and method, it tends to be referred to as a set of practices (Loukides 2012).

7.2 Extreme Programming (XP)

Extreme Programming (XP) is a method of software development in which the development of the solution is given priority over the observance of formal rules (Beck, Extreme Programming Explained: Embrace Change 2000). Extreme Programming runs in recurring cycles, follows a structured approach and emphasizes teamwork, openness and constant communication between all participants. Communication is a basic pillar.

The main goal of Extreme Programming is faster provisioning, higher software quality, and customer satisfaction. The customer receives a ready-to-use product in the development of which he has actively participated. New functionalities are permanently developed, integrated, and tested. For each of the functionalities to be developed, the steps risk analysis, benefit analysis, the provision of a first executable version (prototyping), and an acceptance test are performed.

7.3 Feature-Driven Development (FDD)

FDD defines a process model and a role model that harmonizes well with existing project structures. For this reason, some companies prefer implementing FDD over XP or Scrum. In addition, FDD is very compact in terms of agile methods. It can be completely described on 10 pages.

The feature concept is crucial for the FDD process, where each defined feature represents added value for the customer.

Five steps have to be carried out in FDD (Palmer and Felsing 2002):

1. Develop overall model
2. Build feature list
3. Plan by feature
4. Design by feature
5. Build by feature.

The Chief Architect constantly monitors the overall architecture, keeps track of the functional core models and coordinates the Feature Plan. For larger teams, individual development teams are led by Chief Developers.

7.4 Kanban

Kanban is all about workflow visualization, limitation of current work, and meaningful feedback loops.

Kanban does not require any specific configuration and can be placed over an existing workflow or process to detect problems. Thus, Kanban can be easily introduced in every company because no comprehensive changes are necessary to get started. The method has been developed to contribute to continuous, incremental, and evolutionary changes in the current process with minimal resistance (Brechner 2015).

The only tool Kanban introduces is the Kanban Board, which is quite similar to the board often used in Scrum Teams. There are different vertical lanes (minimum: To Do, Doing, Done) and tasks cards can be placed in these lanes to transparently show the status (see Sect. 6.2.2).

Kanban also considers the potential value of existing processes, roles, responsibilities and titles, as well as their possible retention. The method does not prohibit changes, but it does not prescribe them either.

7.5 Lean Software Development

Lean software development is a translation of lean manufacturing principles and practices to the software development domain. Adapted from the Toyota Production System, it is emerging with the support of a pro-lean subculture within the agile community. Lean offers a solid conceptual framework, values, and principles, as well as good practices, derived from experience, that support agile organizations (Lean Software Development 2020).

Lean development can be summarized by seven principles, very close in concept to lean manufacturing principles (Poppendieck and Poppendieck 2003):

1. Eliminate waste
2. Amplify learning
3. Decide as late as possible
4. Deliver as fast as possible
5. Empower the team
6. Build integrity in
7. Optimize the whole.

8 Summary and Outlook

It becomes more and more apparent after reading this chapter about agile methods that each methodology has its perks and flaws depending on the situation and the project to which it is applied. The presented status quo is open to change and will do so continuously. In that sense, it is essential to assess each project and company individually, when considering which methodology to use and in what manner or rather in which combination. The next chapter will give more insights into this and illustrate how to mitigate disadvantages of agile methods by using such combinations with the help of "hybrid project management methods." Still, whatever method you choose, it is of utmost importance to keep the company structure, including its employees, open to change and flexible to quickly adapt to new models yet to come.

Literature

Baldauf, Corinna. n.d. *Retromat.* Accessed September 01, 2021. https://retromat.org.

Beck, Kent. 2000. *Extreme Programming Explained: Embrace Change.* Boston: Addison-Wesley.

Beck, Kent, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, et al. 2001. *Manifesto for Agile Software Development.* https://agilemanifesto.org.

Beister, Uwe, and Milica Zeljkovic. 2021. "Hybrid Project Management." In *The Digital Journey of Banking and Insurance, Volume I—Disruption and DNA*, edited by Volker Liermann and Claus Stegmann. New York: Palgrave Macmillan.

Brechner, Eric. 2015. *Agile Project Management with Kanban (Developer Best Practices).* US: Microsoft Press.

Dräther, Rolf, Holger Koschek, and Carsten Sahling. 2013. *Scrum - kurz & gut.* Köln: O'Reilly Verlag.

Lamelas, Ana. 2018. *Top 5 Main Agile Methodologies: Advantages and Disadvantages.* November 10. Accessed September 01, 2021. https://www.xpand-it.com/2018/10/11/top-5-agile-methodologies/#:~:text=Scrum,time%20for%20a%20software%20product.

Lean Software Development. 2020. *Lean Software Development*. October 5. Accessed September 01, 2021. https://en.wikipedia.org/wiki/Lean_software_development.

Loukides, Mike. 2012. *What Is DevOps?* June 7. http://radar.oreilly.com/2012/06/what-is-devops.html.

Palmer, S. R., and J. M. Felsing. 2002. *A Practical Guide to Feature-Driven Development.* São Paulo: Prentice Hall.

Petrova, Sandra. 2019. *Adopting Agile: The Latest Reports About the Popular Mindset.* January 18. https://adevait.com/blog/remote-work/adopting-agile-the-latest-reports-about-the-popular-mindset.

Poppendieck, Mary, and Tom Poppendieck. 2003. *Lean Software Development: An Agile Toolkit*, 13–15. Boston: Addison-Wesley.

Rubin, Kenneth S. 2013. *Essential Scrum—A Practical Guide to the Most Popular Agile Process*. Boston: Addison-Wesley.

Schwaber, Ken, and Jeff Sutherland. 2017. *The Scrum Guide*™. November. https://www.scrumguides.org/scrum-guide.html#values.

Stacey, Ralph. 1996. *Complexity and Creativity in Organizations*. San Francisco: Berrett-Koehler Publishers.

Stellman, Andrew, and Jennifer Greene. 2013. *Learning Agile: Understanding Scrum, XP, Lean, and Kanban.* Sebastopol: O'Reilly Media.

Wijetunge, Rumesh. 2019. *Scrum: As a Culture Rather Than a Process.* August 27. https://www.knowledgehut.com/blog/agile/scrum-as-a-culture-rather-than-a-process.