WIKIPEDIA

Scrum (software development)

Scrum is an agile framework for developing, delivering, and sustaining complex products,[1] with an initial emphasis on software development, although it has been used in other fields including research, sales, marketing and advanced technologies.[2] It is designed for teams of ten or fewer members, who break their work into goals that can be completed within timeboxed iterations, called *sprints*, no longer than one month and most commonly two weeks. The Scrum Team track progress in 15-minute time-boxed daily meetings, called daily scrums. At the end of the sprint, the team holds sprint review, to demonstrate the work done, and sprint retrospective to improve continuously.

Contents

Name

The software development term *scrum* was first used in a 1986 paper titled "The New New Product Development Game". The term is borrowed from rugby, where a scrum is a formation of players. The term *scrum* was chosen by the paper's authors because it emphasizes teamwork.[3]

Scrum is occasionally seen written in all-capitals, as *SCRUM*.[4] While the word itself is not an acronym, its capitalized styling likely comes from an early paper by Ken Schwaber[5] that capitalized *SCRUM* in its title.[6][7]

While the trademark on the term *Scrum* itself has been allowed to lapse, it is deemed as owned by the wider community rather than an individual,[8] so the leading capital for *Scrum* is retained in this article.

Many of the terms used in Scrum are typically written with leading capitals (e.g., *Scrum Master*, *Daily Scrum*). However, to maintain an encyclopedic tone, this article uses normal sentence case for these terms (e.g., *scrum master*, *daily scrum*) – unless they are recognized marks (such as *Certified Scrum Master*).

Key ideas

Scrum is a lightweight, iterative and incremental framework for managing complex work.[9][10] The framework challenges assumptions of the traditional, sequential approach to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines involved.

A key principle of Scrum is the dual recognition that customers will change their minds about what they want or need (often called *requirements volatility*[11]) and that there will be unpredictable challenges—for which a predictive or planned approach is not suited.

As such, Scrum adopts an evidence-based empirical approach – accepting that the problem cannot be fully understood or defined up front, and instead focusing on how to maximize the team's ability to deliver quickly, to respond to emerging requirements, and to adapt to evolving technologies and changes in market conditions.

History

Hirotaka Takeuchi and Ikujiro Nonaka introduced the term *scrum* in the context of product development in their 1986 *Harvard Business Review* article, 'The New New Product Development Game'.[12] Takeuchi and Nonaka later argued in *The Knowledge Creating Company*[13] that it is a form of "organizational knowledge creation,

[...] especially good at bringing about innovation continuously, incrementally and spirally".

The authors described a new approach to commercial product development that would increase speed and flexibility, based on case studies from manufacturing firms in the automotive, photocopier and printer industries.[12] They called this the holistic or rugby approach, as the whole process is performed by one cross-functional team across multiple overlapping phases, in which the team "tries to go the distance as a unit, passing the ball back and forth".[12] (In rugby football, a scrum is used to restart play, as the forwards of each team interlock with their heads down and attempt to gain possession of the ball.[14])

The Scrum framework was based on research by Schwaber with Tunde Babatunde at DuPont Research Station and University of Delaware. Tunde advised that attempts to develop complex products, such as software, that weren't based in empiricism were doomed to higher risks and rates of failure as the initial conditions and assumptions change. Empiricism, using frequent inspection and adaptation, with flexibility and transparency is the most suitable approach.

In the early 1990s, Ken Schwaber used what would become Scrum at his company, Advanced Development Methods; while Jeff Sutherland, John Scumniotales and Jeff McKenna developed a similar approach at Easel Corporation, referring to it using the single word scrum.[15]

Ken and Jeff worked together to integrate their ideas into a single framework, Scrum. They tested Scrum and continually improved it, leading to their 1995 paper, contributions to the Agile Manifesto[16] in 2001, and the worldwide spread and use of Scrum since 2002.

In 1995, Sutherland and Schwaber jointly presented a paper describing the Scrum framework at the Business Object Design and Implementation Workshop held as part of Object-Oriented Programming, Systems, Languages & Applications '95 (OOPSLA '95) in Austin, Texas.[17] Over the following years, Schwaber and Sutherland collaborated to combine this material—with their experience and evolving good practice—to develop what became known as Scrum.[18]

In 2001, Schwaber worked with Mike Beedle to describe the method in the book, *Agile Software Development with Scrum*.[19] Scrum's approach to planning and managing product development involves bringing decision-making authority to the level of operation properties and certainties.[6]

In 2002, Schwaber with others founded the Scrum Alliance[20] and set up the *Certified Scrum* accreditation series. Schwaber left the Scrum Alliance in late 2009 and founded Scrum.org[21] which oversees the parallel *Professional Scrum* accreditation series.[22]

Since 2009, a public document called *The Scrum Guide*[18] has been published and updated by Schwaber and Sutherland. It has been revised 5 times, with the current version being November 2017.

Roles

There are three roles in the Scrum framework.[23] These are ideally co-located to ensure optimal communication among team members. While many organizations have other roles involved with defining and delivering the product, Scrum defines only these three.[18]

Product owner

The product owner, representing the product's stakeholders and the voice of the customer (or may represent the desires of a committee[24]), is responsible for delivering good business results.[25] Hence, the product owner is accountable for the product backlog and for maximizing the value that the team delivers.[24] The product owner defines the product in customer-centric terms (typically user stories), adds them to the Product Backlog, and prioritizes them based on importance and dependencies.[26] A scrum team should have only one product owner (although a product owner could support more than one team)[27] This role should not be combined with that of the scrum master. The product owner should focus on the business side of product development and spend the majority of their time liaising with stakeholders and the team. The product owner should not dictate how the team reaches a technical solution, but rather will seek consensus among the team members.[28][29][30] This role is crucial and requires a deep understanding of both sides: the business and the engineers (developers) in the scrum team. Therefore a good product owner should be able to communicate what the business needs, ask why they need it (because there may be better ways to achieve that), and convey the message to all stakeholders including the Development Team using a technical language, as required. The Product Owner uses Scrum's empirical tools to manage highly complex work, while controlling risk and achieving value.

Communication is a core responsibility of the product owner. The ability to convey priorities and empathize with team members and stakeholders is vital to steer product development in the right direction. The product owner role bridges the communication gap between the team and its stakeholders, serving as a proxy for stakeholders to the team and as a team representative to the overall stakeholder community.[31][32]

As the face of the team to the stakeholders, the following are some of the communication tasks of the product owner to the stakeholders:[33]

- Define and announce releases.
- Communicate delivery and team status.
- Share progress during governance meetings.
- Share significant RIDAs (risks, impediments, dependencies, and assumptions) with stakeholders.
- Negotiate priorities, scope, funding, and schedule.
- Ensure that the product backlog is visible, transparent and clear.

Empathy is a key attribute for a product owner to have—the ability to put one's self in another's shoes. A product owner converses with different stakeholders, who have a variety of backgrounds, job roles, and objectives. A product owner must be able to see from these different points of view. To be effective, it is wise for a product owner to know the level of detail the audience needs. The development team needs thorough feedback and specifications so they can build a product up to expectation, while an executive sponsor may just need summaries of progress. Providing more information than necessary may lose stakeholder interest and waste time. A direct means of communication is the most preferred by seasoned agile product owners.[27]

A product owner's ability to communicate effectively is also enhanced by being skilled in techniques that identify stakeholder needs, negotiate priorities between stakeholder interests, and collaborate with developers to ensure effective implementation of requirements.

Development team

The development team has from three to nine members who carry out all tasks required to build increments of valuable output every sprint.[26]

While team members are referred to as *developers* in some literature[18], the term refers to anyone who plays a role in the development and support of the system or product, and can include researchers, architects, designers, data specialists, statisticians, analysts, engineers, programmers, and testers, among others.[23] However, due to the confusion that can arise when some people do not feel the term 'developer' applies to them, they are often referred to just as *team members*.

The team is self-organizing. While no work should come to the team except through the product owner, and the scrum master is expected to protect the team from too much distraction, the team should still be encouraged to interact directly with customers and/or stakeholders to gain maximum understanding and immediacy of feedback.[26]

Scrum master

Scrum is facilitated by a scrum master, who is accountable for removing impediments to the ability of the team to deliver the product goals and deliverables.[34] The scrum master is not a traditional team lead or project manager but acts as a buffer between the team and any distracting influences. The scrum master ensures that the scrum framework is followed. The scrum master helps to ensure the team follows the agreed processes in the Scrum framework, often facilitates key sessions, and encourages the team to improve. The role has also been referred to as a team facilitator or servant-leader to reinforce these dual perspectives.

The core responsibilities of a scrum master include (but are not limited to):[35]

- Helping the product owner maintain the product backlog in a way that ensures the needed work is well understood so the team can continually make forward progress
- Helping the team to determine the definition of done for the product, with input from key stakeholders
- Coaching the team, within the Scrum principles, in order to deliver high-quality features for its product[36]
- Promoting self-organization within the team
- Helping the scrum team to avoid or remove impediments to its progress, whether internal or external to the team
- Facilitating team events to ensure regular progress
- Educating key stakeholders on Agile and Scrum principles
- Coaching the development team in self-organization and cross-functionality

The scrum master helps people and organizations adopt empirical and lean thinking, leaving behind hopes for certainty and predictability.

One of the ways the scrum master role differs from a project manager is that the latter may have people management responsibilities and the scrum master does not. A scrum master provides a limited amount of direction since the team is expected to be empowered and self-organizing.[37] Scrum does not formally recognise the role of project manager, as traditional command and control tendencies would cause difficulties.[38]

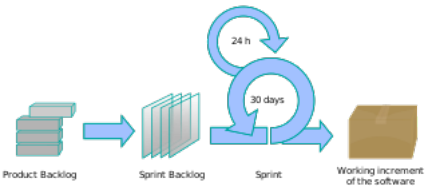
Workflow

Sprint

A sprint (also known as *iteration* or *timebox*) is the basic unit of development in Scrum. The sprint is a timeboxed effort; that is, the length is agreed and fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.[6]

Scrum framework

Each sprint starts with a *sprint planning* event that establishes a sprint goal and the required product backlog items. The team accepts what they agree is ready and translate this into a sprint backlog, with a breakdown of the work required and an estimated forecast for the sprint goal. Each sprint ends with a *sprint review* and *sprint retrospective*, that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.[15]



The Scrum process

Scrum emphasizes valuable, useful output at the end of the sprint that is really done. In the case of software, this likely includes that the software has been fully integrated, tested and documented, and is potentially releasable.[38]

Sprint planning

At the beginning of a sprint, the scrum team holds a sprint planning event[39] to:

- Mutually discuss and agree on the scope of work that is intended to be done during that sprint
- Select product backlog items that can be completed in one sprint
- Prepare a sprint backlog that includes the work needed to complete the selected product backlog items
- Agree the **sprint goal,** a short description of what they are forecasting to deliver at the end of the sprint.
- The recommended duration is four hours for a two-week sprint (pro-rata for other sprint durations) [18]
  - During the first half, the whole scrum team (development team, scrum master, and product owner) selects the product backlog items they believe could be completed in that sprint
  - During the second half, the development team identifies the detailed work (tasks) required to complete those product backlog items; resulting in a confirmed sprint backlog
    - As the detailed work is elaborated, some product backlog items may be split or put back into the product backlog if the team no longer believes they can complete the required work in a single sprint
- Once the development team has prepared their sprint backlog, they forecast (usually by voting) which tasks will be delivered within the sprint.

Daily scrum

Each day during a sprint, the team holds a daily scrum (or stand-up) with specific guidelines:[40][6]

A daily scrum in the computing room. This centralized location helps the team start on time.

- All members of the development team come prepared. The daily scrum:
  - starts precisely on time even if some development team members are missing
  - should happen at the same time and place every day
  - is limited (timeboxed) to fifteen minutes
- Anyone is welcome, though only development team members should contribute.
- During the daily scrum, each team member typically answers three questions:
  - What did I complete yesterday that contributed to the team meeting our sprint goal?
  - What do I plan to complete today to contribute to the team meeting our sprint goal?
  - Do I see any impediment that could prevent me or the team from meeting our sprint goal?

Any impediment (e.g., stumbling block, risk, issue, delayed dependency, assumption proved unfounded)[41] identified in the daily scrum should be captured by the scrum master and displayed on the team's scrum board or on a shared risk board, with an agreed person designated to working toward a resolution (outside of the daily scrum). While the currency of work status is the whole team's responsibility, the scrum master often updates the sprint burndown chart.[42] Where the team does not see the value in these events, it is the responsibility of the scrum master to find out why.[43] This is part of the responsibility of educating the team and stakeholders about the Scrum principles.[36]

No detailed discussions should happen during the daily scrum. Once the meeting ends, individual members can get together to discuss issues in detail; such a meeting is sometimes known as a 'breakout session' or an 'after party'.[42]

Sprint review

At the end of a sprint, the team holds two events: the sprint review and the sprint retrospective.

At the sprint review, the team:

- reviews the work that was completed and the planned work that was not completed
- presents the completed work to the stakeholders (a.k.a. the demo)
- collaborates with the stakeholders on what to work on next

Guidelines for sprint reviews:

- Incomplete work cannot be demonstrated.
- The recommended duration is two hours for a two-week sprint (proportional for other sprint-durations).[18]

Sprint retrospective

At the sprint retrospective, the team:

- reflects on the past sprint
- identifies and agrees on continuous process improvement actions

Guidelines for sprint retrospectives:

- Three main questions arise in the sprint retrospective:
  - What went well during the sprint?
  - What did not go well?
  - What could be improved for better productivity in the next sprint?
- The recommended duration is one-and-a-half hours for a two-week sprint (proportional for other sprint duration(s)).
- The scrum master facilitates this event.

Backlog refinement

Backlog refinement (formerly called grooming) is the ongoing process of reviewing product backlog items and checking that they are appropriately prepared and ordered in a way that makes them clear and executable for teams once they enter sprints via the sprint planning activity. Product backlog items may be broken into multiple smaller ones. Acceptance criteria may be clarified. Dependencies may be identified and investigated.

Although not originally a core Scrum practice, backlog refinement has been added to the Scrum Guide and adopted as a way of managing the quality of product backlog items entering a sprint, with a recommended investment of up to 10% of a team's sprint capacity.[18][44]

The backlog can also include technical debt (also known as design debt or code debt). This is a concept in software development that reflects the implied cost of additional rework caused by choosing an easy solution now instead of using a better approach that would take longer.

Cancelling a sprint

The product owner can cancel a sprint if necessary.[18] The product owner may do so with input from the team, scrum master or management. For instance, management may wish the product owner to cancel a sprint if external circumstances negate the value of the sprint goal. If a sprint is abnormally terminated, the next step is to conduct a new sprint planning, where the reason for the termination is reviewed.

Artifacts

Product backlog

The product backlog is a breakdown of work to be done[45] and contains an ordered list of product requirements that a scrum team maintains for a product. Common formats include user stories and use cases.[38] The requirements define features, bug fixes, non-functional requirements, etc.—whatever must be done to deliver a viable product. The product owner prioritizes product backlog items (PBIs) based on considerations such as risk, business value, dependencies, size, and date needed.

The product backlog is what will be delivered, ordered into the sequence in which it should be delivered. It is visible to everyone but may only be changed with the consent of the product owner, who is ultimately responsible for ordering product backlog items for the development team to choose.

The product backlog contains the product owner's assessment of business value and the development team's assessment of development effort, which are often, but not always, stated in story points using the rounded Fibonacci scale. These estimates help the product owner to gauge the timeline and may influence the ordering of product backlog items; for example, if two features have the same business value, the product owner may schedule earlier delivery of the one with the lower development effort (because the return on investment is higher) or the one with higher development effort (because it is more complex or riskier, and they want to retire that risk earlier).[46]

The product backlog and the business value of each product backlog item is the responsibility of the product owner. The effort to deliver each item is estimated by the development team in story points, or time. By estimating in story points, the team reduces the dependency in individual developers; this is useful especially in dynamic teams where developers are often assigned to other projects after sprint delivery. For instance, if a user story is estimated as a 5 in effort (using Fibonacci sequence), it remains 5 regardless of how many developers are working on it

Story points define the effort in a time-box, so they do not change with time. For instance, in one hour an individual can walk, run, or climb, but the effort expended is clearly different. The gap progression between the terms in the Fibonacci sequence encourages the team to deliver carefully considered estimates. Estimates of 1, 2 or 3 imply similar efforts (1 being trivial), but if the team estimates an 8 or 13 (or higher), the impact on both delivery and budget can be significant. The value of using story points is that the team can reuse them by comparing similar work from previous sprints, but it should be recognized that estimates are relative to the team. For example, an estimate of 5 for one team could be a 2 for another having senior developers and higher skills.

Every team should have a product owner, although in many instances a product owner could work with more than one team.[27] The product owner is responsible for maximizing the value of the product. The product owner gathers input and takes feedback from, and is lobbied by, many people, but ultimately makes the call on what gets built.

The product backlog:

- Captures requests to modify a product—including new features, replacing old features, removing features, and fixing issues
- Ensures the development team has work that maximizes business benefit to the product owner

Typically, the product owner and the scrum team work together to develop the breakdown of work; this becomes the product backlog, which evolves as new information surfaces about the product and about its customers, and so later sprints may address new work.

Management

A product backlog, in its simplest form, is merely a list of items to work on. Having well-established rules about how work is added, removed and ordered helps the whole team make better decisions about how to change the product.[47]

The product owner prioritizes product backlog items based on which are needed soonest. The team then chooses which items they can complete in the coming sprint. On the scrum board, the team moves items from the product backlog to the sprint backlog, which is the list of items they will build. Conceptually, it is ideal for the team to only select what they think they can accomplish from the top of the list, but it is not unusual to see in practice that teams are able to take lower-priority items from the list along with the top ones selected. This normally happens because there is time left within the sprint to accommodate more work. Items at the top of the backlog, the items to work on first, should be broken down into stories that are suitable for the development team to work on. The further down the backlog goes, the less refined the items should be. As Schwaber and Beedle put it "The lower the priority, the less detail until you can barely make out the backlog item."[6]

As the team works through the backlog, it must be assumed that change happens outside their environment—the team can learn about new market opportunities to take advantage of, competitor threats that arise, and feedback from customers that can change the way the product was meant to work. All of these new ideas tend to trigger the team to adapt the backlog to incorporate new knowledge. This is part of the fundamental mindset of an agile team. The world changes, the backlog is never finished.[48]

Sprint backlog

The sprint backlog is the list of work the development team must address during the next sprint.[49] The list is derived by the scrum team progressively selecting product backlog items in priority order from the top of the product backlog until they feel they have enough work to fill the sprint. The development team should keep in mind its past performance assessing its capacity for the new-sprint, and use this as a guideline of how much 'effort' they can complete.

A scrum task board

The product backlog items may be broken down into tasks by the development team.[49] Tasks on the sprint backlog are never assigned (or pushed) to team members by someone else; rather team members sign up for (or pull) tasks as needed according to the backlog priority and their own skills and capacity. This promotes self-organization of the development team and developer buy-in.

The sprint backlog is the property of the development team, and all included estimates are provided by the development team. Often an accompanying task board is used to see and change the state of the tasks of the current sprint, like to do, in progress and done.

Once a sprint backlog is committed, no additional work can be added to the sprint backlog except by the team. Once a sprint has been delivered, the product backlog is analyzed and reprioritized if necessary, and the next set of functionality is selected for the next sprint.

Increment

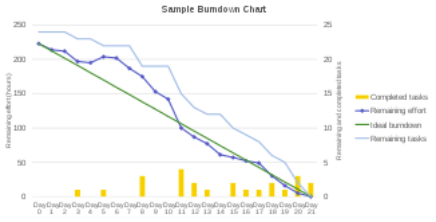
The *increment* is the potentially releasable output of the sprint that meets the sprint goal. It is formed from all the completed sprint backlog items, integrated with the work of all previous sprints. The increment must be complete, according to the scrum team's *definition of done* (DoD), fully functioning, and in a usable condition regardless of whether the product owner decides to actually deploy and use it.

Extensions

The following artifacts and techniques can be used to help people use Scrum.[18]

Sprint burndown chart

The sprint burndown chart is a publicly displayed chart showing remaining work in the sprint backlog.[50] Updated every day, it gives a simple view of the sprint progress. It also provides quick visualizations for reference. The horizontal axis of the sprint burndown chart shows the days in a sprint, while the vertical axis shows the amount of work remaining each day (typically representing the estimate of hours of work remaining).



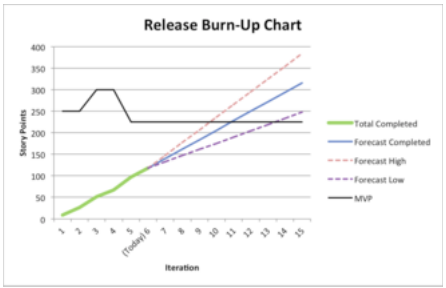
A sample burndown chart for a completed sprint, showing remaining effort at the end of each day.

During sprint planning, the ideal burndown chart is plotted. Then, during the sprint, each member picks up tasks from the sprint backlog and works on them. At the end of the day, they update the remaining hours for tasks to be completed. In such a way, the actual burndown chart is updated day by day.

It should not be confused with an earned value chart.

Release burn-up chart

The release burn-up chart is a way for the team to provide visibility and track progress toward a release. Updated at the end of each sprint, it shows progress toward delivering a forecast scope. The horizontal axis of the release burn-up chart shows the sprints in a release, while the vertical axis shows the amount of work completed at the end of each sprint (typically representing cumulative story points of work completed). Progress is plotted as a line that grows up to meet a horizontal line that represents the forecast scope; often shown with a forecast, based on progress to date, that indicates how much scope might be completed by a given release date or how many sprints it will take to complete the given scope.



A sample burn-up chart for a release, showing scope completed each sprint (MVP = Minimum Viable Product)

The release burn-up chart makes it easy to see how much work has been completed, how much work has been added or removed (if the horizontal scope line moves), and how much work is left to be done.

Definition of ready (DoR)

The start criteria to determine whether the specifications and inputs are set enough to start the work item, i.e. a user story.

Definition of done (DoD)

The exit-criteria to determine whether a product backlog item is complete. In many cases, the DoD requires that all regression tests be successful. The definition of done may vary from one scrum team to another but must be consistent within one team.[51]

Velocity

The total effort a team is capable of in a sprint. The number is derived by evaluating the work (typically in user story points) completed in the last sprint. The collection of historical velocity data is a guideline for assisting the team in understanding how much work they can achieve.

Spike

A time-boxed period used to research a concept or create a simple prototype. Spikes can either be planned to take place in between sprints or, for larger teams, a spike might be accepted as one of many sprint delivery objectives. Spikes are often introduced before the delivery of large or complex product backlog items in order to secure budget, expand knowledge, or produce a proof of concept. The duration and objective(s) of a spike is agreed between product owner and development team before the start. Unlike sprint commitments, spikes may or may not deliver tangible, shippable, valuable functionality. For example, the objective of a spike might be to successfully reach a decision on a course of action. The spike is over when the time is up, not necessarily when the objective has been delivered.[52]

Tracer bullet

Also called a drone spike, a tracer bullet is a spike with the current architecture, current technology set, current set of best practices that result in production quality code. It might just be a very narrow implementation of the functionality but is not throwaway code. It is of production quality, and the rest of the iterations can build on this code. The name has military origins as ammunition that makes the path of the bullet visible, allowing for corrections. Often these implementations are a 'quick shot' through all layers of an application, such as connecting a single form's input field to the back-end, to prove the layers connect as expected.[53]

Limitations

The benefits of Scrum may be more difficult to achieve when:[54][55]

- **Teams whose members are geographically dispersed or part-time**: In Scrum, developers should have close and ongoing interaction, ideally working together in the same space most of the time. While recent improvements in technology have reduced the impact of these barriers (e.g., being able to collaborate on a digital whiteboard), the Agile manifesto asserts that the best communication is face to face.[56]

- **Teams whose members have very specialized skills**: In Scrum, developers should have T-shaped skills, allowing them to work on tasks outside of their specialization. This can be encouraged by good Scrum leadership. While team members with very specific skills can and do contribute well, they should be encouraged to learn more about and collaborate with other disciplines.
- **Products with many external dependencies**: In Scrum, dividing product development into short sprints requires careful planning; external dependencies, such as user acceptance testing or coordination with other teams, can lead to delays and the failure of individual sprints.
- **Products that are mature or legacy or with regulated quality control**: In Scrum, product increments should be fully developed and tested in a single sprint; products that need large amounts of regression testing or safety testing (e.g., medical devices or vehicle control) for each release are less suited to short sprints than to longer waterfall releases.

Tools for implementation

Like other agile methods, effective adoption of Scrum can be supported through a wide range of tools.

Many companies use universal tools, such as spreadsheets to build and maintain artifacts such as the sprint backlog. There are also open-source and proprietary software packages for Scrum—which are either dedicated to product development using the Scrum framework or support multiple product development approaches including Scrum.

Other organizations implement Scrum without software tools and maintain their artifacts in hard-copy forms such as paper, whiteboards, and sticky notes.[57]

Scrum values

Scrum is a feedback-driven empirical approach which is, like all empirical process control, underpinned by the three pillars of transparency, inspection, and adaptation. All work within the Scrum framework should be visible to those responsible for the outcome: the process, the workflow, progress, etc. In order to make these things visible, scrum teams need to frequently inspect the product being developed and how well the team is working. With frequent inspection, the team can spot when their work deviates outside of acceptable limits and adapt their process or the product under development.[26]

These three pillars require trust and openness in the team, which the following five values of Scrum enable:[18]

1. Commitment: Team members individually commit to achieving their team goals, each and every sprint.
2. Courage: Team members know they have the courage to work through conflict and challenges together so that they can do the right thing.
3. Focus: Team members focus exclusively on their team goals and the sprint backlog; there should be no work done other than through their backlog.
4. Openness: Team members and their stakeholders agree to be transparent about their work and any challenges they face.
5. Respect: Team members respect each other to be technically capable and to work

with good intent.

Adaptations

The hybridization of Scrum with other software development methodologies is common as Scrum does not cover the whole product development lifecycle; therefore, organizations find the need to add in additional processes to create a more comprehensive implementation. For example, at the start of product development, organizations commonly add process guidance on the business case, requirements gathering and prioritization, initial high-level design, and budget and schedule forecasting.[58]

Various authors and communities of people who use Scrum have also suggested more detailed techniques for how to apply or adapt Scrum to particular problems or organizations. Many refer to these methodological techniques as 'patterns' - by analogy with design patterns in architecture and software.[59][60]

Scrumban

Scrumban is a software production model based on Scrum and Kanban. Scrumban is especially suited for product maintenance with frequent and unexpected work items, such as production defects or programming errors. In such cases the time-limited sprints of the Scrum framework may be perceived to be of less benefit, although Scrum's daily events and other practices can still be applied, depending on the team and the situation at hand. Visualization of the work stages and limitations for simultaneous unfinished work and defects are familiar from the Kanban model. Using these methods, the team's workflow is directed in a way that allows for minimum completion time for each work item or programming error, and on the other hand ensures each team member is constantly employed.[61]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[62] In the case of decentralized teams, stage-illustration software such as Assembla, JIRA or Agilo can be used.

The major differences between Scrum and Kanban is that in Scrum work is divided into sprints that last a fixed amount of time, whereas in Kanban the flow of work is continuous. This is visible in work stage tables, which in Scrum are emptied after each sprint, whereas in Kanban all tasks are marked on the same table. Scrum focuses on teams with multifaceted know-how, whereas Kanban makes specialized, functional teams possible.[61]

Scrum of scrums

The scrum of scrums is a technique to operate Scrum at scale, for multiple teams working on the same product, allowing them to discuss progress on their interdependencies, focusing on how to coordinate delivering software,[63] especially on areas of overlap and integration. Depending on the cadence (timing) of the scrum of scrums, the relevant daily scrum for each scrum team ends by designating one member as an ambassador to participate in the scrum of scrums with ambassadors from other teams. Depending on the context, the ambassadors may be technical contributors or each team's scrum master.[63]

Rather than simply a progress update, the scrum of scrums should focus on how teams are collectively working to resolve, mitigate, or accept any risks, impediments, dependencies, and assumptions (RIDAs) that have been identified. The scrum of scrums tracks these RIDAs via a backlog of its own, such as a risk board (sometimes known as a *ROAM board* after the initials of resolved, owned, accepted, and mitigated),[64] which typically leads

to greater coordination and collaboration between teams.[63]

This should run similar to a daily scrum, with each ambassador answering the following four questions:[65]

- What risks, impediments, dependencies, or assumptions has your team resolved since we last met?
- What risks, impediments, dependencies, or assumptions will your team resolve before we meet again?
- Are there any new risks, impediments, dependencies, or assumptions slowing your team down or getting in their way?
- Are you about to introduce a new risk, impediment, dependency, or assumption that will get in another team's way?

As Jeff Sutherland commented,[63]

> Since I originally defined the Scrum of Scrums (Ken Schwaber was at IDX working with me), I can definitively say the Scrum of Scrums is not a 'meta Scrum'. The Scrum of Scrums as I have used it is responsible for delivering the working software of all teams to the Definition of Done at the end of the sprint, or for releases during the sprint. PatientKeeper delivered to production four times per Sprint. Ancestry.com delivers to production 220 times per two-week Sprint. Hubspot delivers live software 100-300 times a day. The Scrum of Scrums Master is held accountable for making this work. So the Scrum of Scrums is an operational delivery mechanism.

Large-scale Scrum

Large-scale Scrum (LeSS) is a product development framework that extends Scrum with scaling rules and guidelines without losing the original purposes of Scrum.

There are two levels to the framework: the first LeSS level is designed for up to eight teams; the second level, known as 'LeSS Huge', introduces additional scaling elements for development with up to hundreds of developers. "Scaling Scrum starts with understanding and being able to adopt standard real one-team Scrum. Large-scale Scrum requires examining the purpose of single-team Scrum elements and figuring out how to reach the same purpose while staying within the constraints of the standard Scrum rules."[66]

Bas Vodde and Craig Larman evolved the LeSS framework from their experiences working with large-scale product development, especially in the telecoms and finance industries. It evolved by taking Scrum and trying many different experiments to discover what works. In 2013, the experiments were solidified into the LeSS framework rules.[67] The intention of LeSS is to 'descale' organization complexity, dissolving unnecessary complex organizational solutions, and solving them in simpler ways. Less roles, less management, less organizational structures.[68]

Criticisms

Ceremonial Scrum meetings have been reported to be hurting productivity and wasting time that could be better spent on actual productive tasks.[69][70]

Scrum practices, when not correctly implemented in the spirit of the Agile Manifesto, have a tendency to become a form of micromanagement.[71]

Scrum also assumes that the amount of effort required for completing certain tasks can be accurately quantified using metrics, although most of the time this can be quite unpredictable. [72]

See also

- Agile testing
- Disciplined agile delivery
- High-performance teams
- Lean software development
- Project management
- Scrumedge
- Unified Process

References

1. Schwaber, Ken; Sutherland, Jeff (November 2017), *The Scrum Guide: The Definitive Guide to Scrum: The Rules of the Game* (https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-US.pdf) (PDF), retrieved May 13, 2020
2. "Lessons learned: Using Scrum in non-technical teams" (https://www.agilealliance.org/resources/experience-reports/lessons-learned-using-scrum-in-non-technical-teams/). *Agile Alliance*. Retrieved April 8, 2019.
3. "Scrum, What's in a Name? - DZone Agile" (https://dzone.com/articles/scrum-whats-in-a-name). *dzone.com*.
4. "Should "SCRUM" be written in all caps?" (https://stackoverflow.com/q/6389423). *stackoverflow.com*. Retrieved January 10, 2017.
5. Schwaber, Ken. "Scrum.org Ken Schwaber" (https://www.scrum.org/team/ken-schwaber).
6. Schwaber, Ken (February 1, 2004). *Agile Project Management with Scrum* (https://archive.org/details/agileprojectmana0000schw). Microsoft Press. ISBN 978-0-7356-1993-7.
7. Schwaber, Ken (2004). "SCRUM Development Process" (http://www.jeffsutherland.org/oopsla/schwapub.pdf) (PDF). *Advanced Development Methods*.
8. Johnson, Hillary Louise (January 13, 2011). "ScrumMaster vs scrum master: What do you think?" (http://www.agilelearninglabs.com/2011/01/scrummaster-vs-scrum-master/). *agilelearninglabs.com*. Retrieved May 10, 2017.
9. "What is Scrum?" (https://www.scrumalliance.org/why-scrum). *What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance*. Scrum Alliance. Retrieved February 24, 2016.
10. Verheyen, Gunther (March 21, 2013). "Scrum: Framework, not methodology" (http://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/). *Gunther Verheyen*. Gunther Verheyen. Retrieved February 24, 2016.

11. J. Henry and S. Henry. Quantitative assessment of the software maintenance process and requirements volatility. In Proc. of the ACM Conference on Computer Science, pages 346–351, 1993.
12. Takeuchi, Hirotaka; Nonaka, Ikujiro (January 1, 1986). "The New New Product Development Game" (https://cb.hbsp.harvard.edu/cbmp/product/86116-PDF-ENG). *Harvard Business Review*. Retrieved June 9, 2010. "Moving the Scrum Downfield"
13. *The Knowledge Creating Company* (https://books.google.com/books?id=B-qxrPaU1-MC&dq=The+Knowledge+Creating+Company&printsec=frontcover). Oxford University Press. 1995. p. 3. ISBN 9780199762330. Retrieved March 12, 2013.
14. "Scrum" (https://www.lexico.com/definition/Scrum). *Lexico UK Dictionary*. Oxford University Press.
15. Sutherland, Jeff (October 2004). "Agile Development: Lessons learned from the first Scrum" (https://web.archive.org/web/20140630020607/http://www.scrumalliance.org/resource_download/35). Archived from the original (https://www.scrumalliance.org/resource_download/35) (PDF) on June 30, 2014. Retrieved September 26, 2008.
16. "Manifesto for Agile Software Development" (https://agilemanifesto.org). Retrieved October 17, 2019.
17. Sutherland, Jeffrey Victor; Schwaber, Ken (1995). *Business object design and implementation: OOPSLA '95 workshop proceedings*. The University of Michigan. p. 118. ISBN 978-3-540-76096-2.
18. Ken Schwaber; Jeff Sutherland. "The Scrum Guide" (http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf) (PDF). Scrum.org. Retrieved October 27, 2017.
19. Schwaber, Ken; Beedle, Mike (2002). *Agile software development with Scrum*. Prentice Hall. ISBN 978-0-13-067634-4.
20. Maximini, Dominik (January 8, 2015). *The Scrum Culture: Introducing Agile Methods in Organizations* (https://books.google.com/books?id=ShojBgAAQBAJ). Management for Professionals. Cham: Springer (published 2015). p. 26. ISBN 9783319118277. Retrieved August 25, 2016. "Ken Schwaber and Jeff Sutherland presented Scrum for the first time at the OOPSLA conference in Austin, Texas, in 1995. [...] In 2001, the first book about Scrum was published. [...] One year later (2002), Ken founded the Scrum Alliance, aiming at providing worldwide Scrum training and certification."
21. "Home" (https://www.scrum.org/index). *Scrum.org*. Retrieved January 6, 2020.
22. Partogi, Joshua (July 7, 2013). "Certified Scrum Master vs Professional Scrum Master" (http://blog.leanagile.in/post/54764080535/certified-scrum-master-vs-professional-scrum). Lean Agile Institute. Retrieved May 10, 2017.
23. Rad, Nader K.; Turley, Frank (2018). *Agile Scrum Foundation Courseware, Second Edition*. 's-Hertogenbosch, Netherlands: Van Haren. p. 26. ISBN 9789401802796.
24. McGreal, Don; Jocham, Ralph (June 4, 2018). *The Professional Product Owner: Leveraging Scrum as a Competitive Advantage* (https://books.google.com/books?id=cEBbDwAAQBAJ&pg=PT173&dq=scrum+product+owner+accountable+backlog#q=scrum%20product%20owner%20accountable%20backlog). Addison-Wesley Professional. ISBN 9780134686653.
25. Rubin, Kenneth (2013), *Essential Scrum. A Practical Guide to the Most Popular Agile Process*, Addison-Wesley, p. 173, ISBN 978-0-13-704329-3

26. Morris, David (2017). *Scrum: an ideal framework for agile projects*. In Easy Steps. pp. 178–179. ISBN 9781840787313. OCLC 951453155 (https://www.worldcat.org/oclc/951453155).
27. Cohn, Mike. Succeeding with Agile: Software Development Using Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.
28. "The Scrum Guide" (http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf) (PDF).
29. *The Scrum guide*. http://www.scrumguides.org/docs/scrumguide/v2016/2016-Scrum-Guide-US.pdf. p. 6.
30. "The Role of the Product Owner" (https://www.scrumalliance.org/learn-about-scrum/community-webinars/webinar-replays/scrum-fundamentals/the-role-of-the-product-owner). *Scrum Alliance*. Retrieved May 26, 2018.
31. Pichler, Roman (March 11, 2010). *Agile Product Management with Scrum: Creating Products that Customers Love*. Addison-Wesley Professional. ISBN 978-0-321-68413-4.
32. Ambler, Scott. "The Product Owner Role: A Stakeholder Proxy for Agile Teams" (http://agilemodeling.com/essays/productOwner.htm). agilemodeling.com. Retrieved July 22, 2016. "[...] in practice there proves to be two critical aspects to this role: first as a stakeholder proxy within the development team and second as a project team representative to the overall stakeholder community as a whole."
33. "The Product Owner Role" (http://scrum-master.thinkific.com/pages/the-product-owner-role). *Scrum Master Test Prep*. Retrieved February 3, 2017.
34. Carroll, N, O'Connor, M. and Edison, H. (2018). The Identification and Classification of Impediments to Software Flow, The Americas Conference on Information Systems (AMCIS 2018), August 16–18, New Orleans, Louisiana, USA.
35. "Core Scrum" (https://www.scrumalliance.org/why-scrum/core-scrum-values-roles). *Scrum Alliance*. Retrieved January 25, 2015.
36. Drongelen, Mike van; Dennis, Adam; Garabedian, Richard; Gonzalez, Alberto; Krishnaswamy, Aravind (2017). *Lean Mobile App Development: Apply Lean startup methodologies to develop successful iOS and Android apps*. Birmingham, UK: Packt Publishing Ltd. p. 43. ISBN 9781786467041.
37. Cobb, Charles G. (2015). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach*. Hoboken, NJ: John Wiley & Sons. p. 37. ISBN 9781118991046.
38. Pete Deemer; Gabrielle Benefield; Craig Larman; Bas Vodde (December 17, 2012). "The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum (Version 2.0)" (http://www.infoq.com/minibooks/Scrum_Primer). InfoQ.
39. Gangji, Arif; Hartman, Bob (2015). "Agile SCRUM For Denver Web Development" (http://www.neonrain.com/agile-scrum-web-development). Neon Rain Interactive. Retrieved September 25, 2015.
40. "What is a Daily Scrum?" (https://www.scrum.org/resources/what-is-a-daily-scrum). *Scrum.org*. Retrieved January 6, 2020.
41. Little, Joe (January 17, 2011). "Impediment Management" (http://agileconsortium.blogspot.com/2011/01/impediment-management.html). Agile Consortium.
42. Flewelling, Paul (2018). *The Agile Developer's Handbook: Get more value from your software development: get the best out of the Agile methodology*. Birmingham, UK: Packt Publishing Ltd. p. 91. ISBN 9781787280205.

43. McKenna, Dave (2016). *The Art of Scrum: How Scrum Masters Bind Dev Teams and Unleash Agility*. Aliquippa, PA: CA Press. p. 126. ISBN 9781484222768.
44. Cho, L (2009). "Adopting an Agile Culture A User Experience Team's Journey". *2009 Agile Conference*. *Agile Conference*. pp. 416–421. doi:10.1109/AGILE.2009.76 (https://doi.org/10.1109%2FAGILE.2009.76). ISBN 978-0-7695-3768-9. S2CID 11201935 (https://api.semanticscholar.org/CorpusID:11201935).
45. Sedano, Todd; Ralph, Paul; Péraire, Cécile. "The Product Backlog" (https://www.researchgate.net/publication/330823863). IEEE.
46. Higgins, Tony (March 31, 2009). "Authoring Requirements in an Agile World" (http://www.batimes.com/articles/authoring-requirements-in-an-agile-world.html). BA Times.
47. "The product backlog: your ultimate to-do list" (https://www.atlassian.com/agile/backlogs). *Atlassian*. Retrieved July 20, 2016.
48. Pichler, Roman. *Agile Product Management with Scrum: Creating Products that Customers Love*. Upper Saddle River, NJ: Addison-Wesley, 2010.
49. Russ J. Martinelli; Dragan Z. Milosevic (January 5, 2016). *Project Management ToolBox: Tools and Techniques for the Practicing Project Manager* (https://books.google.com/books?id=SbA7CwAAQBAJ&pg=PA304). Wiley. p. 304. ISBN 978-1-118-97320-2.
50. Charles G. Cobb (January 27, 2015). *The Project Manager's Guide to Mastering Agile: Principles and Practices for an Adaptive Approach* (https://books.google.com/books?id=vHjTBQAAQBAJ&pg=PA378). John Wiley & Sons. p. 378. ISBN 978-1-118-99104-6.
51. Ken Schwaber, *Agile Project Management with Scrum*, p.55
52. "Create a Spike Solution" (http://www.extremeprogramming.org/rules/spike.html). Extreme Programming.
53. Sterling, Chris (October 22, 2007). "Research, Spikes, Tracer Bullets, Oh My!" (http://www.gettingagile.com/2007/10/22/research-spikes-tracer-bullets-oh-my/). *Getting Agile*. Retrieved October 23, 2016.
54. Turk, Dan; France, Robert; Rumpe, Bernhard (2014) [2002]. "Limitations of Agile Software Processes". *Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering*: 43–46. arXiv:1409.6600v1 (https://arxiv.org/abs/1409.6600v1).
55. "Issues and Challenges in Scrum Implementation" (http://www.ijser.org/researchpaper/Issues-and-Challenges-in-Scrum-Implementation.pdf) (PDF). *International Journal of Scientific & Engineering Research*. **3** (8). August 2012. Retrieved December 10, 2015.
56. Kent Beck; James Grenning; Robert C. Martin; Mike Beedle; Jim Highsmith; Steve Mellor; Arie van Bennekum; Andrew Hunt; Ken Schwaber; Alistair Cockburn; Ron Jeffries; Jeff Sutherland; Ward Cunningham; Jon Kern; Dave Thomas; Martin Fowler; Brian Marick (2001). "Principles behind the Agile Manifesto" (http://agilemanifesto.org/principles.html). Agile Alliance. Retrieved August 7, 2017.
57. Dubakov, Michael (2008). "Agile Tools. The Good, the Bad and the Ugly" (http://targetprocess.com/download/whitepaper/agiletools.pdf) (PDF). Retrieved August 30, 2010.
58. Hron, M.; Obwegeser, N. (January 2018). "Scrum in practice: an overview of Scrum adaptations" (http://pure.au.dk/portal/files/116906219/Hron_Obwegeser_Scrum_in_practice_An_overview_of_Scrum_adaptations.pdf) (PDF). *Proceedings of the 2018 51st Hawaii International Conference on System Sciences (HICSS), January 3-6, 2018*.
59. Bjørnvig, Gertrud; Coplien, Jim (June 21, 2008). "Scrum as Organizational Patterns" (https://sites.google.com/a/scrumorgpatterns.com/www/). Gertrude & Cope.

60. "Scrum Pattern Community" (http://www.scrumplop.org). *ScrumPLoP.org*. Retrieved July 22, 2016.
61. Kniberg, Henrik; Skarin, Mattias (December 21, 2009). "Kanban and Scrum - Making the most of both" (https://www.infoq.com/minibooks/kanban-scrum-minibook) (PDF). InfoQ. Retrieved July 22, 2016.
62. Ladas, Corey (October 27, 2007). "scrum-ban" (http://leansoftwareengineering.com/ksse/scrum-ban/). Lean Software Engineering. Retrieved September 13, 2012.
63. "Scrum of Scrums" (https://guide.agilealliance.org/guide/scrumofscrums.html). Agile Alliance. December 17, 2015.
64. "Risk Management – How to Stop Risks from Screwing Up Your Projects!" (http://www.allaboutagile.com/risk-management-how-to-stop-risks-from-screwing-up-your-projects/). Kelly Waters.
65. "Scrum of Scrums" (http://scrummastertest.com/scrum-of-scrums/). *Scrum Master Test Prep*. Retrieved May 29, 2015.
66. Larman; scrumyear=2013. "Scaling Agile Development (Crosstalk journal, November / December 2013)" (http://www.crosstalkonline.org/storage/issue-archives/2013/201305/201305-larman.pdf) (PDF).
67. "Large-Scale Scrum (LeSS)" (http://less.works). 2014.
68. Grgic (2015). "Descaling organisation with LeSS (Blog)" (https://leanarch.eu/2015/05/09/descaling-organisation-with-less-2/).
69. Jenson, John (March 8, 2019). "Meetings: The productivity killer for developers" (https://www.tandemseven.com/blog/meetings-productivity-killer-for-developers/). *TandemSeven - The Experience Innovation Company*. Retrieved June 5, 2020.
70. Matters, Business (December 4, 2019). "Not all developers like agile, and here are 5 reasons why" (https://www.bmmagazine.co.uk/in-business/not-all-developers-like-agile-and-here-are-5-reasons-why/). *Business Matters*. Retrieved June 5, 2020.
71. on, Isaak Tsalicoglou. "How to transition from Agile to micromanagement | Hacker Noon" (https://hackernoon.com/how-to-transition-from-agile-to-micromanagement-de9247c26e11). *hackernoon.com*. Retrieved June 5, 2020.
72. Cagle, Kurt. "The End of Agile" (https://www.forbes.com/sites/cognitiveworld/2019/08/23/the-end-of-agile/). *Forbes*. Retrieved June 5, 2020.

Further reading

- Vacaniti, Daniel (February 2018). "The Kanban Guide for Scrum Teams" (https://scrumorg-website-prod.s3.amazonaws.com/drupal/2018-02/2018%20Kanban%20Guide%20for%20Scrum%20Teams_0.pdf) (PDF). *scrum.org*. Retrieved March 12, 2018.
- Sutherland, Jeff; Schwaber, Ken (2013). "Scrum Guides" (http://www.scrumguides.org/). ScrumGuides.org. Retrieved July 26, 2017.
- Verheyen, Gunther (2013). *Scrum - A Pocket Guide (A Smart Travel Companion)* ISBN 978-9087537203.
- Münch, Jürgen; Armbrust, Ove; Soto, Martín; Kowalczyk, Martin (2012). *Software Process Definition and Management*. ISBN 978-3-642-24291-5.
- Deemer, Pete; Benefield, Gabrielle; Larman, Craig; Vodde, Bas (2009). "The Scrum Primer" (http://www.scrumprimer.org). Retrieved June 1, 2009.
- Janoff, N.S.; Rising, L. (2000). "The Scrum Software Development Process for Small

Teams" (http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/scrum/s4026.pdf) (PDF). Retrieved February 26, 2015.

External links

- Agile Alliance's Scrum library (https://cf.agilealliance.org/articles/article_list.cfm?CategoryID=17)
- A Scrum Process Description (https://web.archive.org/web/20170605114816/http://epf.eclipse.org/wikis/scrum/) by the Eclipse Process Framework (EPF) Project