Scrum (software development)

Scrum is a framework for managing software development. It is designed for teams of three to nine developers who break their work into actions that can be completed within fixed duration cycles (called "sprints"), track progress and re-plan in daily 15-minute stand-up meetings, and collaborate to deliver workable software every sprint.[1][2] Approaches to coordinating the work of multiple scrum teams in larger organizations include *Large-Scale Scrum*, *Scaled Agile Framework (SAFe)* and *Scrum of Scrums*, among others.

Contents

1 **Key ideas**

2 **History**

3 **Roles**
- 3.1 Product owner
- 3.2 Development team
- 3.3 Scrum master

4 **Workflow**
- 4.1 Sprint
- 4.2 Sprint planning
- 4.3 Daily scrum
- 4.4 Sprint review and retrospective
- 4.5 Extensions

5 **Artifacts**
- 5.1 Product backlog
- 5.2 Sprint backlog
- 5.3 Product increment
- 5.4 Extensions

6 **Limitations**

7 **Tools for implementation**

8 **Scrum values**

9 **Adaptations**
- 9.1 Scrumban
- 9.2 Scrum of scrums
- 9.3 Large-scale scrum

10 **See also**

11 **References**

12 **Further reading**

13 **External links**

Key ideas

Scrum is an iterative and incremental agile software development framework for managing product development.[3][4] It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal",[5] challenges assumptions of the "traditional, sequential approach"[5] 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.

motivation why Scrum/agile:

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[6]) 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.

Note on capitalization Many of the terms used in Scrum (e.g., scrum master) are typically written with leading capitals (i.e., Scrum Master) or as conjoint words written in camel case (i.e., ScrumMaster). To maintain an encyclopedic approach, however, this article uses normal sentence case for these terms—unless they are recognized marks (such as *Certified Scrum Master*).

While the trademark on the term Scrum itself has been allowed to lapse, so that it is deemed as owned by the wider community rather than an individual,[7] the leading capital is retained—except when used with other words (as in *daily scrum* or *scrum team*).

In the literature, this is occasionally seen in all capitals, as SCRUM.[8] While this is incorrect, as Scrum is not an acronym, it likely arose due to an early paper by Ken Schwaber which capitalized SCRUM in the title.[1][9]

History

Hirotaka Takeuchi and Ikujiro Nonaka introduced the term scrum in the context of product development in their 1986 *Harvard Business Review* article, "New New Product Development Game".[5] Takeuchi and Nonaka later argued in The Knowledge Creating Company[10] 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.[5] They called this the holistic or rugby approach, as the whole process is performed by one cross-functional team across multiple overlapping phases, where the team "tries to go the distance as a unit, passing the ball back and forth".[5] (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.[11])

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.[12] 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.[13] 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.[14]

In 2001 Schwaber worked with Mike Beedle to describe the method in the book, *Agile Software Development with Scrum*.[15] Scrum's approach to planning and managing product development involves bringing decision-making authority to the level of operation properties and certainties.[1]

In 2002 Schwaber with others founded the Scrum Alliance[16] and set up the *certified scrum* accreditation series. Schwaber left the Scrum Alliance in late 2009 and founded Scrum.org which oversees the parallel *professional scrum* accreditation series.[17]

Since 2010, there is a public document called *The Scrum Guide*[14] that defines sort of an official version of Scrum and is occasionally revised.

Roles

There are three core roles in the Scrum framework. These are ideally co-located to deliver potentially shippable product increments every sprint. Together these three roles form the scrum team. While many organizations have other roles involved with defining and delivering the product, Scrum defines only these three.[14]

Product owner

The product owner represents the product's stakeholders and the voice of the customer; and is accountable for ensuring that the team delivers value to the business. 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.[18] Scrum teams should have one product owner. 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 should not dictate how the team reaches a technical solution.[19] This role is equivalent to the customer representative role in some other agile frameworks such as extreme programming (XP).

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.[20][21]

As the face of the team to the stakeholders, the following are some of the communication tasks of the product owner to the stakeholders:[22]

- demonstrates the solution to key stakeholders who were not present at a sprint review;
- defines and announces releases;
- communicates team status;
- organizes milestone reviews;
- educates stakeholders in the development process;
- negotiates priorities, scope, funding, and schedule;
- ensures 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.[23]

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 is responsible for delivering potentially shippable product increments every sprint (the sprint goal).

The team has from three to nine members who carry out all tasks required to build the product increments (analysis, design, development, testing, technical writing, etc.).[18] Although there will be several disciplines represented in the team, its members are referred to generically as *developers*. To avoid potential confusion that this only refers to programmers, some organizations call this a *delivery team* and its members just *team members*.

The development team in Scrum is self-organizing, even though there may be interaction with other roles outside the team, such as a project management office (PMO).

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. 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[24] or servant-leader to reinforce these dual perspectives.

The core responsibilities of a scrum master include (but are not limited to):[25]

- 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
- 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 in the product on Scrum principles
- Coaching the development team in self-organization and cross-functionality

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. Scrum does not formally recognise the role of project manager, as traditional command and control tendencies would cause difficulties.[26]

Workflow

Sprint

A sprint (or iteration) is the basic unit of development in Scrum. The sprint is a timeboxed effort; that is, it is restricted to a specific duration.[27] The duration is fixed in advance for each sprint and is normally between one week and one month, with two weeks being the most common.[1]

Each sprint starts with a sprint planning event that aims to define a sprint backlog, identify the work for the sprint, and make an estimated forecast for the sprint goal. Each sprint ends with a sprint review and sprint retrospective,[12] that reviews progress to show to stakeholders and identify lessons and improvements for the next sprints.

Scrum emphasizes working product 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 shippable.[26]

Scrum framework

Sprint planning

At the beginning of a sprint, the scrum team holds a sprint planning event[27] to:

The Scrum process

- 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
- The recommended duration is four hours for a two-week sprint (pro-rata for other sprint durations) [14]
    - 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:


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)[28] 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). No detailed discussions should happen during the daily scrum.

Sprint review and retrospective

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)
- The team and the stakeholders collaborate 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 (pro-rata for other sprint durations)

At the sprint retrospective, the team:

- Reflects on the past sprint
- Identifies and agrees on continuous process improvement actions

Guidelines for sprint retrospectives:

- Two main questions are asked in the sprint retrospective: What went well during the sprint? What could be improved in the next sprint?
- The recommended duration is one-and-a-half hours for a two-week sprint (pro-rata for other sprint durations)
- This event is facilitated by the scrum master

Extensions

The following activities are commonly done, although not considered by all as a core part of Scrum:

Backlog refinement

Backlog refinement (once called backlog grooming) is the ongoing process of reviewing product backlog items and checking that they are appropriately prioritised and prepared 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; and dependencies, investigation, and preparatory work may be identified and agreed as technical spikes.

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.[29]

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.[14] 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 comprises an ordered list of requirements that a scrum team maintains for a product.[30] It consists of features, bug fixes, non-functional requirements, etc.—whatever must be done to successfully deliver a viable product. The product owner prioritizes those product backlog items (PBIs) based on considerations such as risk, business value, dependencies, size, and date needed.

Items added to a backlog are commonly written in story format. 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 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).[31]

The product backlog and the business value of each product backlog item is the responsibility of the product owner. The size (i.e. estimated complexity or effort) of each item is, however, determined by the development team, who contributes by sizing in story points or in estimated hours.

There is a common misunderstanding that only user stories are allowed in a product backlog. By contrast, Scrum is neutral on requirement techniques. As the Scrum primer[26] states,

> product backlog items are articulated in any way that is clear and sustainable. Contrary to popular misunderstanding, the product backlog does not contain "user stories"; it simply contains items. Those items can be expressed as user stories, use cases, or any other requirements approach that the group finds useful. But whatever the approach, most items should focus on delivering value to customers.

Scrum advocates that the role of product owner be assigned. 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 come together and write down everything that must be prioritized, and this becomes content for the first sprint—which is a block of time meant for focused work on selected items that can be accommodated within a timeframe. The product backlog can evolve as new information surfaces about the product and about its customers, and so later sprints may address new work.

The following items typically comprise a product backlog: features, bugs, technical work, and knowledge acquisition. A feature is wanted, while a bug is unintended or unwanted (but may not be necessarily something defective). An example of technical work could be to run a virus check on all developers' workstations. An example of knowledge acquisition could be to research Wordpress plugin libraries and making a selection.

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.[32]

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."[1]

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.[20]

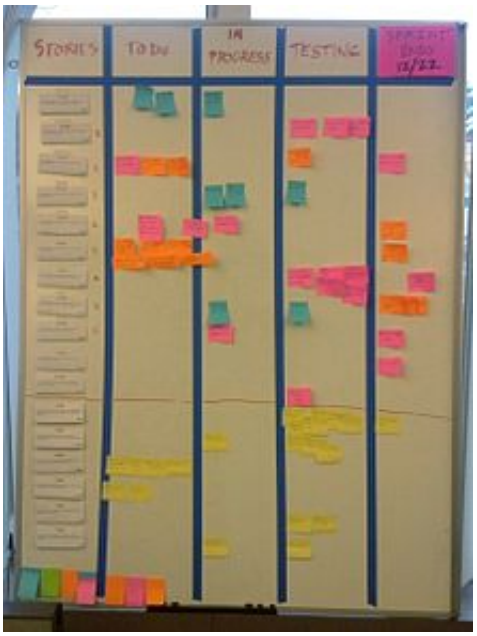
Sprint backlog

The sprint backlog is the list of work the development team must address during the next sprint.[30] 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 guide line of how much 'effort' they can complete.

The product backlog items may be broken down into tasks by the development team.[30] Tasks on the sprint backlog are never assigned; rather, tasks are signed up for by the team members as needed according to the set priority and the skills of the team. 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.



A Scrum task board

Product increment

The increment (or potentially shippable increment, PSI) is the sum of all the product backlog items completed during a sprint, integrated with the work of all previous sprints. At the end of a sprint, 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 release it.

Extensions

The following artifacts are commonly used, although not considered by all as a core part of Scrum:

Sprint burn-down chart

The sprint burn-down chart is a public displayed chart showing remaining work in the sprint backlog.[33] 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 burn-down chart shows the days in a sprint, while the vertical axis shows the amount of work remaining each day (typically representing estimate of hours of work remaining).


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

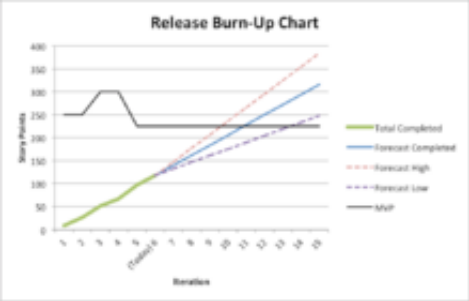
During sprint planning the ideal burndown chart is plotted. Then, during the sprint, each member picks up tasks from the sprint backlog and works on them. At the end of the day, they update the remaining hours for tasks to be completed. In such a way, the actual burndown chart is updated day by day.

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.

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.



A sample burn-up chart for a release, showing scope completed each sprint

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 should be successful. The definition of done may vary from one scrum team to another, but must be consistent within one team.[34]

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 likely achieve in a future sprint.

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.[35]

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 results in production quality code. It might just be a very narrow implementation of the functionality but is not throw away 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.[36]

Limitations

Scrum works less well in the following circumstances:[37][38]

- **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.[39]
- **Teams whose members have very specialized skills**: In Scrum, developers should be able to work on any task or pick up work that another developer has started. This can be managed 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 deliveries of software from 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.

From a business perspective, Scrum has many virtues, one of which is that it is designed to yield the best business solutions. However, the efficiency by which it does so in any given organization can vary widely, and is largely dependent on the ability of the organization to adhere to the implementation guidelines in this article. Every company has its own distinct organizational structure, culture, and set of business practices, and some are more naturally amenable to this methodology than others.

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.[40]

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.[18]

These three pillars require trust and openness in the team, which the following five values of Scrum enable:[14]

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 business case, requirements gathering and prioritization, initial high-level design, and budget and schedule forecasting.

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.[41][42] Such patterns have extended Scrum outside of the software development domain into Manufacturing,[43] Finance and Human Resources.

While the Scrum Guide[14] prescribes the essential elements of Scrum, it seems that many companies deviate significantly from it.[44] The least variation is in the sprints and sprint length, events, team size and requirements engineering. The most variation can be found in the roles, effort estimation and quality assurance.

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.[45]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[46] In the case of decentralized teams, stage-illustration software such as Assembla, JIRA or Agilo.

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.[45]

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,[47] 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.[47]

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),[48] which typically leads to greater coordination and collaboration between teams.[47]

This should run similar to a daily scrum, with each ambassador answering the following four questions:[49]

- 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,[47]

> 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 8 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."[50]

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.[51] 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.[52]

See also

- Disciplined agile delivery
- Lean software development
- Project management
- Scaled Agile Framework
- Unified Process
- High-performance teams

References

1. Schwaber, Ken (February 1, 2004). *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-7356-1993-7.
2. *Daily Scrum Meeting* (https://www.mountaingoatsoftware.com/agile/scrum/meetings/daily-scrum), Mountain Goat Software, retrieved July 26, 2017
3. "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.
4. Verheyen, Gunther. "Scrum: Framework, not methodology" (http://guntherverheyen.com/2013/03/21/scrum-framework-not-methodology/). *Gunther Verheyen*. Gunter Verheyen. Retrieved February 24, 2016.
5. Takeuchi, Hirotaka; Nonaka, Ikujiro (January 1, 1986). "New New Product Development Game" (https://cb.hbsp.harvard.edu/cbmp/product/86116-PDF-ENG). *Harvard Business Review*. Retrieved June 9, 2010. "Moving the

d.hbr.org/product/81104-PDF-ENG). *Harvard Business Review*. Retrieved June 6, 2016. "Moving the Scrum Downfield"

6. 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.
7. Johnson, Hillary Louise (January 13, 2011). "ScrumMaster vs scrum master: What do you think?" (http://www.agilelearninglabs.com/2011/01/scrummaster-vs-scrum-master/). *www.agilelearninglabs.com*. Retrieved May 10, 2017.
8. "Should "SCRUM" be written in all caps?" (http://stackoverflow.com/questions/6389423/should-scrum-be-written-in-all-caps). *stackoverflow.com*. Retrieved January 10, 2017.
9. Schwaber, Ken (2004). "SCRUM Development Process" (http://www.jeffsutherland.org/oopsla/schwapub.pdf) (PDF). *Advanced Development Methods*. MA.
10. *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.
11. Scrum, abbreviated form of scrummage, Oxford English Dictionary Online (http://www.oed.com/).
12. Sutherland, Jeff (October 2004). "Agile Development: Lessons learned from the first Scrum" (http://www.scrumalliance.org/resources/35) (PDF). Retrieved September 26, 2008.
13. Sutherland, Jeffrey Victor; Schwaber, Ken (1995). *Business object design and implementation: OOPSLA '95 workshop proceedings*. The University of Michigan. p. 118. ISBN 3-540-76096-2.
14. 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.
15. Schwaber, Ken; Beedle, Mike (2002). *Agile software development with Scrum*. Prentice Hall. ISBN 0-13-067634-9.
16. 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."
17. 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.
18. Morris, David (2017). *Scrum: an ideal framework for agile projects* (https://www.worldcat.org/oclc/951453155). In Easy Steps. pp. 178–179. ISBN 9781840787313. OCLC 951453155 (https://www.worldcat.org/oclc/951453155).
19. "The Product Owner's Role in Technical Matters" (https://www.scrumalliance.org/community/articles/2013/december/product-owner-should-not-interfere-in-techincal-as). *Scrum Alliance*. Retrieved July 6, 2015.
20. Pichler, Roman. *Agile Product Management with Scrum: Creating Products that Customers Love*. Upper Saddle River, NJ: Addison-Wesley, 2010.
21. 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."
22. "The Product Owner Role" (http://scrum-master.thinkific.com/pages/the-product-owner-role). *Scrum Master Test Prep*. Retrieved February 3, 2017.
23. Cohn, Mike. Succeeding with Agile: Software Development Using Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.
24. Leybourn, E. (2013). *Directing the Agile Organisation: A Lean Approach to Business Management*. London: IT Governance Publishing: 117–120.
25. "Core Scrum" (https://www.scrumalliance.org/why-scrum/core-scrum-values-roles). *Scrum Alliance*. Retrieved January 25, 2015.
26. 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.

27. 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.
28. Little, Joe (January 17, 2011). "Impediment Management" (http://agileconsortium.blogspot.com/2011/01/impediment-management.html). Agile Consortium.
29. Cho, L (2009). "Adopting an Agile Culture A User Experience Team's Journey". *Agile Conference*: 416. ISBN 978-0-7695-3768-9. doi:10.1109/AGILE.2009.76 (https://doi.org/10.1109%2FAGILE.2009.76).
30. 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.
31. 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.
32. "The product backlog: your ultimate to-do list" (https://www.atlassian.com/agile/backlogs). *Atlassian*. Retrieved July 20, 2016.
33. 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.
34. Ken Schwaber, *Agile Project Management with Scrum*, p.55
35. "Create a Spike Solution" (http://www.extremeprogramming.org/rules/spike.html). Extreme Programming.
36. 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.
37. "Limitations of Agile Software Processes" (http://www.se-rwth.de/~rumpe/publications/Limitations-of-Agile-Software-Processes.pdf) (PDF). Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, Springer Verlag page 43--46, 2002. Retrieved May 8, 2017.
38. "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, Volume 3, Issue 8, August 2012. Retrieved December 10, 2015.
39. 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.
40. Dubakov, Michael (2008). "Agile Tools. The Good, the Bad and the Ugly" (http://targetprocess.com/download/whitepaper/agiletools.pdf) (PDF). Retrieved August 30, 2010.
41. Bjørnvig, Gertrud; Coplien, Jim (June 21, 2008). "Scrum as Organizational Patterns" (https://sites.google.com/a/scrumorgpatterns.com/www/). Gertrude & Cope.
42. "Scrum Pattern Community" (http://www.scrumplop.org). *Scrum Pattern Languages of Programs*. ScrumPLoP.org. Retrieved July 22, 2016.
43. "WIKISPEED – Applying Agile software principles and practices for fast automotive development" (http://agilebusinessmanagement.org/content/wikispeed-%E2%80%93-applying-agile-software-principles-and-practices-fast-automotive-development). Agile Business Management Consortium. December 3, 2013. Retrieved September 11, 2015.
44. Diebold, Philipp; Ostberg, Jan-Peter; Wagner, Stefan; Zendler, Ulrich (2015). Lassenius, Casper; Dingsøyr, Torgeir; Paasivaara, Maria, eds. *What Do Practitioners Vary in Using Scrum?* (http://elib.uni-stuttgart.de/opus/volltexte/2015/9914/pdf/scrum_variations.pdf) (PDF). Lecture Notes in Business Information Processing. Springer International Publishing. pp. 40–51. ISBN 978-3-319-18611-5. doi:10.1007/978-3-319-18612-2_4 (https://doi.org/10.1007%2F978-3-319-18612-2_4).
45. Kniberg, Henrik; Skarin, Mattias (Dec 21, 2009). "Kanban and Scrum - Making the most of both" (https://www.infoq.com/minibooks/kanban-scrum-minibook) (PDF). InfoQ. Retrieved July 22, 2016.
46. Ladas, Corey (October 27, 2007). "scrum-ban" (http://leansoftwareengineering.com/ksse/scrum-ban/). Lean Software Engineering. Retrieved September 13, 2012.
47. "Scrum of Scrums" (http://guide.agilealliance.org/guide/scrumofscrums.html). Agile Alliance.

48. "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.
49. "Scrum of Scrums" (http://scrummastertest.com/scrum-of-scrums/). *Scrum Master Test Prep*. Retrieved May 29, 2015.
50. Larman; Vodde (2013). "Scaling Agile Development (Crosstalk journal, November / December 2013)" (http://www.crosstalkonline.org/storage/issue-archives/2013/201305/201305-larman.pdf) (PDF).
51. "Large-Scale Scrum (LeSS)" (http://less.works). 2014.
52. Grgic (2015). "Descaling organisation with LeSS (Blog)" (https://leanarch.eu/2015/05/09/descaling-organisation-with-less-2/).

Further reading

- Maria Almeida (2015). "How to be a Great Product Owner" (http://blog.jobbox.io/how-to-be-a-great-product-owner/). JOBBOX.io.
- Jeff Sutherland; Ken Schwaber (2013). "Scrum Guides" (http://www.scrumguides.org/). ScrumGuides.org. Retrieved July 26, 2017.
- N.S. Janoff; L. Rising (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.
- Deemer, Pete; Benefield, Gabrielle; Larman, Craig; Vodde, Bas (2009). "The Scrum Primer" (http://www.scrumprimer.org). Retrieved June 1, 2009.
- Kniberg, Henrik. "Scrum and XP from the Trenches" (http://www.infoq.com/minibooks/scrum-xp-from-the-trenches). Retrieved August 9, 2010.
- Münch, Jürgen; Armbrust, Ove; Soto, Martín; Kowalczyk, Martin (2012). "Software Process Definition and Management" (https://www.springer.com/computer/swe/book/978-3-642-24290-8). Retrieved July 16, 2012.
- Ambler, Scott (2013). "Going Beyond Scrum: Disciplined Agile Delivery" (http://disciplinedagileconsortium.org/Resources/Documents/BeyondScrum.pdf) (PDF). Retrieved February 4, 2014.
- "The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework" (Jeff Sutherland, www.Scruminc.com, April 2, 2012): http://jeffsutherland.com/ScrumPapers.pdf
- "Story Points: Why are they better than hours?" (Jeff Sutherland, www.Scruminc.com, September 30, 2012) http://Scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html (http://scrum.jeffsutherland.com/2010/04/story-points-why-are-they-better-than.html)
- Roman Pichler (2010). "Agile Product Management with Scrum - Creating Products That Customers Love" (http://www.romanpichler.com/blog/books/agile-product-management-with-scrum-creating-products-that-customers-love/). Addison-Wesley Professional.

External links

- Scrum Guide (http://www.scrumguides.org/)
- Scaled Scrum (https://www.scrum.org/Resources/The-Nexus-Guide) by Scrum.org
- Agile Alliance's Scrum library (http://cf.agilealliance.org/articles/article_list.cfm?CategoryID=17)
- A Scrum Process Description (http://epf.eclipse.org/wikis/scrum/) by the Eclipse Process Framework (EPF) Project (http://www.eclipse.org/epf)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=807528702"

This page was last edited on 28 October 2017, at 15:19.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.