Scrum (software development)

From Wikipedia, the free encyclopedia

Scrum is an iterative and incremental agile software development framework for managing product development.[1][2] It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal",[3] challenges assumptions of the "traditional, sequential approach"[3] 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.

Wasserfallmodell

A key principle of Scrum is its recognition that during product development, the customers can change their minds about what they want and need (often called requirements volatility[4]), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, Scrum adopts an evidence-based empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly, to respond to emerging requirements and to adapt to evolving technologies and changes in market conditions.

Contents

History

Hirotaka Takeuchi and Ikujiro Nonaka introduced the word 'scrum' as a term in the context of product development in 1986 in their article on the *New New Product Development Game*.[3] Takeuchi and Nonaka later argued in The Knowledge Creating Company[5] 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.[3] 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".[3] (In rugby football, a *scrum* refers to a tight-packed formation of players with their heads down who attempt to gain possession of the ball.[6])

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*.[7] In 1995 Sutherland and Schwaber jointly presented a paper describing the *Scrum methodology* at the Business Object Design and Implementation Workshop held as part of Object-Oriented Programming, Systems, Languages & Applications '95 (OOPSLA '95) in Austin, Texas.[8] 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.[9]

In 2001 Schwaber worked with Mike Beedle to describe the method in the book *Agile Software Development with Scrum*.[10] Scrum's approach to planning and managing product development involves bringing decision-making authority to the level of operation properties and certainties.[11] Although the word Scrum is not an acronym, some companies implementing the process have been known to spell it with capital letters as 'SCRUM'. This may be due to one of Ken Schwaber's early papers, which capitalized SCRUM in the title.[11]

In 2002 Schwaber with others founded the Scrum Alliance[12] and set up the Certified Scrum Master program and its derivatives. Schwaber left the Scrum Alliance in late 2009 and founded Scrum.org.

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.

These three pillars require trust and openness in the team, which the following five values of Scrum enable:[9]

Commitment

Team members individually commit to achieving their team goals, each and every Sprint.

Courage
: Team members know they have the courage to work through conflict and challenges together so that they can do the right thing.

Focus
: Team members focus exclusively on their team goals and the Sprint Backlog; there should be no work done other than through their backlog.

Openness
: Team members and their stakeholders agree to be transparent about their work and any challenges they face.

Respect
: Team members respect each other to be technically capable and to work with good intent.

Roles

There are three core roles[13] in the Scrum framework. These core roles are ideally colocated to deliver potentially shippable Product Increments. They represent the *Scrum Team*. Although other roles involved with product development may be encountered, Scrum does not define any team roles other than those described below.[9]

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 writes customer-centric items (typically user stories), prioritizes them based on importance and dependencies, and adds them to the Product Backlog. 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.[14] 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. Product Owners bridge the communication gap between the team and its stakeholders. They serve as a proxy for stakeholders to the team and as a team representative to the overall stakeholder community.[15][16]

As the face of the team to the stakeholders, the following are some of the communication tasks of the Product Owner to the stakeholders:

- 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.[17]

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 increments (PSIs) of product at the end of each Sprint (the Sprint goal). A team is made up of 3–9 individuals who do the actual work (analyse, design, develop, test, technical communication, document, etc.). Development Teams are cross-functional, with all of the skills as a team necessary to create a Product Increment. The Development Team in Scrum is self-organizing, even though there may be some interaction with a project management office (PMOs).

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

The core responsibilities of a Scrum Master include (but are not limited to):[19]

- 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[20]

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.[21]

Workflow

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.[22] 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.[11]

Scrum framework

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 commitment for the Sprint goal. Each Sprint ends with a Sprint Review and Sprint Retrospective,[7] 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.[21]

The Scrum process

Planning

At the beginning of a Sprint, the Scrum Team holds a *Sprint Planning event*[22] to:

- Communicate 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 the Sprint Backlog that details the work needed to finish the selected Product Backlog Items
- Time-boxed to a four-hour limit for a two-week Sprint (pro rata for other Sprint durations) [9]
  - During the first half, the whole Scrum Team (Development Team, Scrum Master, and Product Owner) selects the Product Backlog Items might be achievable in that Sprint
  - During the second half, the Development Team decomposes the work items (tasks) required to deliver those Product Backlog Items; resulting in a confirmed *Sprint Backlog*
    - Some Product Backlog Items may be split or deprioritized if the identified work is not achievable in that Sprint

Once the Development Team prepares the *Sprint Backlog*, they commit (usually by voting) to deliver tasks within the Sprint.

Daily Scrum

Each day during a Sprint, the team holds a *Daily Scrum* (or *stand-up*) with specific guidelines:

- 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 normally only Scrum Team roles contribute.
- During the Daily Scrum, each team-member answers three questions:
  - What did I do yesterday that helped the Development Team meet the Sprint goal?
  - What will I do today to help the Development Team meet the Sprint goal?
  - Do I see any impediment that prevents me or the Development Team from meeting the Sprint goal?

Any impediment (e.g., stumbling block, risk, issue, delayed dependency, assumption proved unfounded) identified in the Daily Scrum should be captured by the Scrum Master and displayed on the team's Scrum board or a shared Resolved/Owned/Accepted/Mitigated (ROAM) 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.

A Daily Scrum in the computing room. This centralized location helps the team start on time.

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*)

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 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.[23]

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,[24] 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.[24]

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 ROAM board,[25] which typically leads to greater coordination and collaboration between teams.[24]

This should run similar to a Daily Scrum, with each ambassador answering the following four questions:[26]

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

As Jeff Sutherland commented,[24]

> *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.*

Artifacts

Product Backlog

The *Product Backlog* comprises an ordered list of *requirements* that a Scrum Team maintains for a product. It consists of features, bug fixes, non-functional requirements, etc.—whatever must be done to successfully deliver a viable product. The Product Owner orders the *Product Backlog Items* (PBIs) based on considerations such as risk, business value, dependencies, 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 a rounded Fibonacci sequence. 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).[27]

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*[21] 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.[28]

The Product Owner prioritizes which Product Backlog Items are most needed. 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."[11]

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.[15]

Sprint Backlog

The *Sprint Backlog* is the list of work the Development Team must address during the next Sprint. 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.

A Scrum task board

The Product Backlog Items may be broken down into tasks by the Development Team. 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 Development Team member skills. 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.

Product Increment

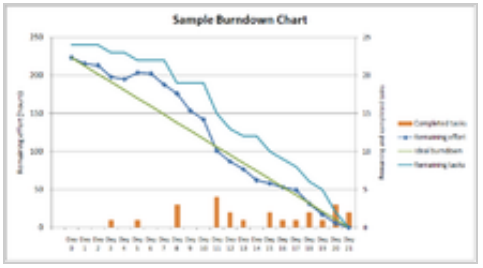
The *increment* (or *potentially shippable increment*, PSI) is the sum of all the Product Backlog Items completed during a Sprint, integrated with the work of all previous Sprints. At the end of a Sprint, the increment must be complete, according to the Scrum Team's definition of done (DoD), fully functioning, 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. 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).



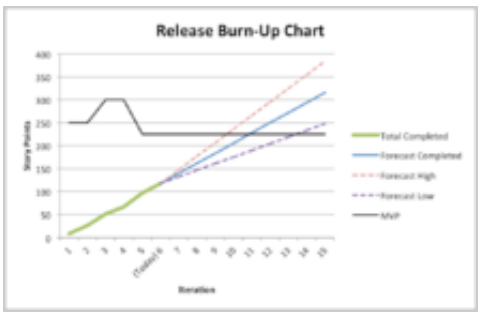
A sample burn-down chart for a completed Sprint, showing remaining effort at the end of each day.

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.



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

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.

Terminology

The following terms are often used in product development using the Scrum framework:

Scrum Team
: Product Owner, Scrum Master and Development Team

Product Owner
: The person responsible for maintaining the Product Backlog by representing the interests of the stakeholders, and ensuring the value of the work the Development Team does.

Scrum Master
: The person responsible for the Scrum framework, making sure it is used correctly and maximizing its benefits.

Development Team
: A cross-functional group of people responsible for delivering potentially shippable increments of

product at the end of every Sprint.

Sprint burn-down chart

Daily progress for a Sprint over the Sprint's length.

Release burn-down chart

Feature level progress of completed Product Backlog Items in the Product Backlog.

Product Backlog (PBL) list

A prioritized list of high-level requirements.

Sprint Backlog (SBL) list

A prioritized list of tasks to complete during the Sprint.

Sprint

A time period (typically 1–4 weeks) in which development occurs on a set of Product Backlog Items that the team has committed to—commonly referred to as a time-box or iteration

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.[29]

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.[30]

Tasks

Work items added to the Sprint Backlog in Sprint Planning, or during the Sprint, with an estimate of hours to complete. Generally, each task should be small enough to be easily completed within a single day.

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.[31]

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.

Impediment

Anything that prevents a team member from performing work as efficiently as possible.[32]

Sashimi

A term used to describe one or more user stories, indicating that they are thin slices of a product feature or capability.

Abnormal termination

The Product Owner can cancel a Sprint if necessary.[9] The Product Owner may do so with input from the team, Scrum Master or management. For instance, management may wish 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.

ScrumBut

ScrumBut (or Scrum but) is a term to describe the approach of a team who have heavily adapted the Scrum framework to their own needs in some way contradictory to the principles of Scrum.[33][34]

Limitations

Scrum works less well in the following circumstances:[35][36]

- 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.
- Teams whose members have very specialised skills. In Scrum, developers should be able to work on any task or pick up work that another developer has started.
- 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.

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 are 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.[37]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[38] In the case of decentralized teams, stage-illustration software such as Assembla, Targetprocess, Eylean Board, JIRA or Agilo for Trac.

In their simplest, the tasks are categorized into the work stages:

- Unstarted
- Ongoing
- Completed

If desired, though, the teams can add more stages of work (such as 'defined', 'designed', 'tested', and 'delivered'). These additional phases can be of assistance if a certain part of the work becomes a bottleneck and the limiting values of the unfinished work cannot be raised. A more specific task division also makes it possible for employees to specialize in a certain phase of work.[38]

There are no set limiting values for unfinished work. Instead, each team has to define them individually by trial and error; a value too small results in workers standing idle for lack of work, whereas values too high tend to accumulate large amounts of unfinished work, which in turn hinders completion times.[37] A rule of

thumb worth bearing in mind is that no team member should have more than two simultaneous selected tasks, and that on the other hand not all team members should have two tasks simultaneously.[38]

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.[37]

Since Scrumban is such a new development model, there is not much reference material. Kanban, on the other hand, has been applied by Microsoft and Corbis.[39]

Tools for implementation

Like other agile methods, Scrum can be implemented 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]

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[9] 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.

See also

- Extreme programming
- Kanban
- Lean software development
- Project management
- Unified Process

- Matrix management, which shares with Scrum the idea of having different manager roles for different aspects of team management
- Disciplined Agile Delivery
- Large-Scale Scrum (LeSS)
- Scaled Agile Framework

References

1. "What is Scrum?". *What is Scrum? An Agile Framework for Completing Complex Projects - Scrum Alliance*. Scrum Alliance. Retrieved February 24, 2016.
2. Verheyen, Gunther. "Scrum: Framework, not methodology". *Gunther Verheyen*. Gunter Verheyen. Retrieved February 24, 2016.
3. Takeuchi, Hirotaka; Nonaka, Ikujiro (January 1, 1986). "New New Product Development Game". *Harvard Business Review*. Retrieved June 9, 2010. "Moving the Scrum Downfield"
4. 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.
5. *The Knowledge Creating Company*. Oxford University Press. 1995. p. 3. ISBN 9780199762330. Retrieved March 12, 2013.
6. *Scrum*, abbreviated form of *scrummage*, *Oxford English Dictionary Online* (http://www.oed.com/).
7. Sutherland, Jeff (October 2004). "Agile Development: Lessons learned from the first Scrum" (PDF). Retrieved September 26, 2008.
8. 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.
9. Ken Schwaber, Jeff Sutherland. "The Scrum Guide" (PDF). Scrum.org. Retrieved July 22, 2016.
10. Schwaber, Ken; Beedle, Mike (2002). *Agile software development with Scrum*. Prentice Hall. ISBN 0-13-067634-9.
11. Schwaber, Ken (February 1, 2004). *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-7356-1993-7.
12. Maximini, Dominik (2015). *The Scrum Culture: Introducing Agile Methods in Organizations*. Management for Professionals. Cham: Springer. 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."
13. Gauthier, Alexandre (August 17, 2011). "What is scrum". Planbox.
14. "The Product Owner's Role in Technical Matters". *Scrum Alliance*. Retrieved July 6, 2015.
15. Pichler, Roman. *Agile Product Management with Scrum: Creating Products that Customers Love*. Upper Saddle River, NJ: Addison-Wesley, 2010.
16. Ambler, Scott. "The Product Owner Role: A Stakeholder Proxy for Agile Teams". agilemodeling.com. Retrieved July 22, 2016.
17. Cohn, Mike. Succeeding with Agile: Software Development Using Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.
18. Leybourn, E. (2013). Directing the Agile Organisation: A Lean Approach to Business Management. London: IT Governance Publishing: 117–120.
19. "Core Scrum". *Scrum Alliance*. Retrieved January 25, 2015.
20. Scrum Alliance, 2016
21. 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)". InfoQ.
22. Gangji, Arif; Hartman, Bob (2015). "Agile SCRUM For Denver Web Development". Neon Rain Interactive. Retrieved September 25, 2015.
23. Cho, L (2009). "Adopting an Agile Culture A User Experience Team's Journey". *Agile Conference*: 416. doi:10.1109/AGILE.2009.76. ISBN 978-0-7695-3768-9.
24. "Scrum of Scrums". Agile Alliance.
25. "Risk Management – How to Stop Risks from Screwing Up Your Projects!". Kelly Waters.
26. "Scrum of Scrums". *Scrum Master Test Prep*. Retrieved May 29, 2015.
27. Higgins, Tony (March 31, 2009). "Authoring Requirements in an Agile World". BA Times.
28. "The product backlog: your ultimate to-do list". *Atlassian*. Retrieved July 20, 2016.
29. "Create a Spike Solution". Extreme Programming.

30. Sterling, Chris (October 22, 2007). "Research, Spikes, Tracer Bullets, Oh My!". *Getting Agile*. Retrieved October 23, 2016.
31. Ken Schwaber, *Agile Project Management with Scrum*, p.55
32. Little, Joe (January 17, 2011). "Impediment Management". Agile Consortium.
33. "ScrumButs and Modifying Scrum". Scrum.org. Retrieved March 18, 2013.
34. Bloomberg, Jason (July 31, 2012). "The Scrum But Paradox". DevX.com. Retrieved March 18, 2013.
35. "Limitations of Agile Software Processes" (PDF). Proceedings of the Third International Conference on Extreme Programming and Flexible Processes in Software Engineering, Springer Verlag page 43--46, 2002. Retrieved December 10, 2015.
36. "Issues and Challenges in Scrum Implementation" (PDF). International Journal of Scientific & Engineering Research, Volume 3, Issue 8, August 2012. Retrieved December 10, 2015.
37. Kniberg, Henrik; Skarin, Mattias (Dec 21, 2009). "Kanban and Scrum - Making the most of both" (PDF). InfoQ. Retrieved July 22, 2016.
38. Ladas, Corey (October 27, 2007). "scrum-ban". Lean Software Engineering. Retrieved September 13, 2012.
39. Anderson, David (March 26, 2009). "A Kanban System for Software Engineering". Infoq. Retrieved September 13, 2012.
40. Dubakov, Michael (2008). "Agile Tools. The Good, the Bad and the Ugly" (PDF). Retrieved August 30, 2010.
41. Bjørnvig, Gertrud; Coplien, Jim (June 21, 2008). "Scrum as Organizational Patterns". Gertrude & Cope.
42. "Scrum Pattern Community". *Scrum Pattern Languages of Programs*. ScrumPLoP.org. Retrieved July 22, 2016.
43. "WIKISPEED – Applying Agile software principles and practices for 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?* (PDF). Lecture Notes in Business Information Processing. Springer International Publishing. pp. 40–51. doi:10.1007/978-3-319-18612-2_4. ISBN 978-3-319-18611-5.

Further reading

- Maria Almeida (2015). "How to be a Great Product Owner". JOBBOX.io.
- Jeff Sutherland; Ken Schwaber (2013). "Scrum Guides". ScrumGuides.org. Retrieved October 2013. Check date values in: `|access-date=` (help)
- N.S. Janoff; L. Rising (2000). "The Scrum Software Development Process for Small Teams" (PDF). Retrieved February 26, 2015.
- Deemer, Pete; Benefield, Gabrielle; Larman, Craig; Vodde, Bas (2009). "The Scrum Primer". Retrieved June 1, 2009.
- Kniberg, Henrik. "Scrum and XP from the Trenches". Retrieved August 9, 2010.
- Münch, Jürgen; Armbrust, Ove; Soto, Martín; Kowalczyk, Martin (2012). "Software Process Definition and Management". Retrieved July 16, 2012.
- Ambler, Scott (2013). "Going Beyond Scrum: Disciplined Agile Delivery" (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". 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/articl

Wikimedia Commons has media related to ***Scrum (development)***.

es/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)

Categories: Agile software development | Management | Production and manufacturing | Project management | Software development process

---