Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
العربية
Български
Català
Dansk
Deutsch
Eesti
Ελληνικά
Español
فارسی
Français
한국어
Italiano
עברית
Lietuvių
Magyar
Nederlands
日本語
Norsk bokmål
Polski
Português
Română
Русский
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
Svenska
தமிழ்
ไทย
Türkçe
Українська
中文
Edit links

Scrum (software development)

From Wikipedia, the free encyclopedia

Scrum is an iterative and incremental agile software development methodology for managing product development. It defines "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal", challenges assumptions of the "traditional, sequential approach" to product development, and enables teams to self-organize by encouraging physical co-location or close online collaboration of all team members, as well as daily face-to-face communication among all team members and disciplines in the project.

A key principle of scrum is its recognition that during a project the customers can change their minds about what they want and need (often called "requirements churn"), and that unpredicted challenges cannot be easily addressed in a traditional predictive or planned manner. As such, scrum adopts an empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements.

Software development process

Core activities
Requirements · Design · Construction · Testing · Debugging · Deployment · Maintenance

Methodologies
Waterfall · Prototype model · Incremental · Iterative · V-Model · Spiral · **Scrum** · Cleanroom · RAD · DSDM · UP · XP · Agile software development · Lean · Dual Vee Model · TDD · BDD · FDD · DDD · MDD

Supporting disciplines
Configuration management · Documentation · Software Quality assurance (SQA) · Project management · User experience

Tools
Compiler · Debugger · Profiler · GUI designer · Modeling · IDE · Build automation

v · t · e

Contents [hide]

History [edit]

Scrum was first defined as "a flexible, holistic product development strategy where a development team works as a unit to reach a common goal" as opposed to a "traditional, sequential approach" in 1986 by Hirotaka Takeuchi and Ikujiro Nonaka in the *New Product Development Game*.[1] Takeuchi and Nonaka later argued in *The Knowledge Creating Company*[2] 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.[4])

In the early 1990s, Ken Schwaber used what would become scrum at his company, Advanced Development Methods, and Jeff Sutherland, with John Scumniotales and Jeff McKenna, developed a similar approach at Easel Corporation, and were the first to refer to it using the single word *scrum*.[5] 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, its first public presentation.[6] Schwaber and Sutherland collaborated during the following years to merge the above writings, their experiences, and industry best practices into what is now known as scrum.[7]

In 2001, Schwaber worked with Mike Beedle to describe the method in the book *Agile Software Development with Scrum*.[8] Its approach to planning and managing projects is to bring decision-making authority to the level of operation properties and certainties.[9] Although the word 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.[9]

Later, Schwaber with others founded the Scrum Alliance and created the Certified Scrum Master programs and its derivatives. Schwaber left the Scrum Alliance in the fall of 2009, and founded Scrum.org to further improve the quality and effectiveness of scrum.

Roles [edit]

There are three core roles[10] with a range of ancillary roles. These core roles are those committed to the project in the scrum process—they are the ones producing the product (objective of the project). They represent the *scrum team*. Although other roles may be encountered in real projects, scrum does not define any team roles other than those described below.[11]

Product owner [edit]

The product owner represents the stakeholders and is the voice of the customer. He or she is accountable for ensuring that the team delivers value to the business. The product owner writes (or has the team write) customer-centric items (typically user stories), ranks and prioritizes them, and adds them to the product backlog. Scrum teams should have one product owner, and while they may also be a member of the development team, this role should not be combined with that of the scrum master. This role is equivalent to the customer representative role in some other agile frameworks.

Role of product owner in defining and communicating product requirements [edit]

Communication is a main function of the product owner. The ability to convey priorities and empathize with team members and stakeholders are vital to steer the project in the right direction. Product owners bridge the communication gap between the team and their stakeholders. As Figure 1 shows, they serve as a proxy stakeholder to the development team and as a project team representative to the overall stakeholder community.[12]

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 in a normal iteration demo
- 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 will be conversing with different stakeholders in the project – different people, with a variety of backgrounds, job roles, and objectives. A product owner needs to be able to see from these different points of view. To be effective, it would also be wise for a product owner to know the level of detail the audience needs from him or her. The development team would need thorough feedback and specifications so they 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. There is also significant evidence that face-to-face communication around a shared sketching environment is the most effective way to communicate information instead of documentation. A direct means of communication is then most preferred by seasoned agile product owners.[13]

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 [edit]

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 level of interface with project management offices (PMOs).

Scrum master [edit]

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 process is used as intended. The scrum master helps ensure the team follows the agreed scrum processes, often facilitates key sessions, and encourages the team to improve. The role has also been referred to as a *team facilitator*[14] or *servant-leader* to reinforce these dual perspectives.

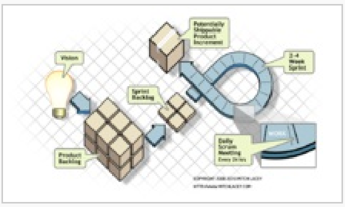
The core responsibilities of a scrum master include (but are not limited to):[15]

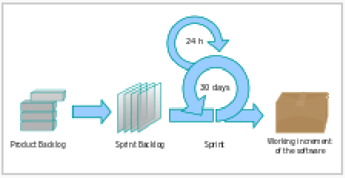
- Helping the product owner maintain the product backlog in a way that ensures the project is well defined and the team can continually advance forward on the project at any given time
- Helping the team to determine the *definition of done* for the product, with input from key stakeholders
- Coaching the team, within the scrum principles, in order to deliver quality valuable features for their product
- Promoting self-organization within the team
- Helping the scrum team to avoid or remove impediments to their progress, whether internal or external to the team
- Facilitating team events to ensure regular progress
- Educating key stakeholders in the product on scrum principles

One of the ways in which the scrum master role differs from that of a project manager is that the latter may have people management responsibilities while the scrum master does not. Scrum does not formally recognise the role of project manager, as traditional command and control tendencies would cause difficulties,[16] however, it is possible for scrum teams to work effectively with agile project managers, especially on large-scale programs.[17]

Events [edit]

Sprint [edit]


Potentially Shippable Product Increment
Vision
2-4 Week Sprint
Sprint Backlog
Product Backlog
Daily Scrum Meeting Every 24 Hrs
Scrum Framework



The scrum process

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.[18] 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.[9]

Each sprint starts with a sprint planning event, the aim of which is to define a sprint backlog, where the work for the sprint is identified and an estimated commitment for the sprint goal is made. Each sprint ends with a sprint review and a sprint retrospective,[5] where the progress is reviewed and shown to stakeholders and improving lessons for the next sprints are identified.

Scrum emphasizes working product at the end of the sprint that is really *done*; in the case of software, this would likely include that the software has been integrated, fully tested, end-user documented, and is potentially shippable.[16]

Events [edit]

Sprint planning [edit]

At the beginning of a sprint, the team holds a *sprint planning event*:[18]

- Select what work is to be done
- Prepare the sprint backlog that details the time it will take to do that work, with the entire team
- Identify and communicate how much of the work is likely to be done during that sprint
- Four-hour time limit for a two-week sprint (pro rata for other sprint durations) [11]
  - During the first half, the whole team (development team, scrum master, and product owner) agree what product backlog items they will consider for that sprint

- During the second half, the development team establish the work (tasks) required to deliver those backlog items, resulting in the *sprint backlog*

Daily scrum [edit]

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

Each day during a sprint, the team hold 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.
- The daily scrum should happen at the same location and same time every day.
- The daily scrum length is constrained (timeboxed) to fifteen minutes.
- Anyone is welcome, although normally only the 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 (stumbling block, risk, or issue) identified in the daily scrum should be captured by the scrum master, displayed on the team's scrum board, and someone agreed to be responsible for working toward a resolution outside of the daily scrum. No detailed discussions should happen during the daily scrum.

Sprint review and sprint retrospective [edit]

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 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 [edit]

Backlog refinement [edit]

Backlog refinement (which used to be 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*.

Backlog refinement is not a core scrum practice but has been adopted as a way of managing the quality of backlog items entering a sprint, with a recommended duration of up to 10% of the sprint capacity.[19]

Scrum of scrums [edit]

The *scrum of scrums* is a technique to scale scrum up for multiple teams working on the same product, allowing teams to discuss progress on their interdependencies, focusing on how to coordinate delivering software,[20] 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.[20]

This should run similar to a daily scrum, with each ambassador answering the following four questions:[21]

- What impediments have my team resolved since we last met?
- What impediments will my team resolve before we meet again?

- Are there any impediments slowing my team down or getting in our way?
- Are we about to put something in another team's way?

Resolution of impediments is expected to focus on the challenges of coordination between the teams, and may entail agreeing to interfaces between teams, negotiating responsibility boundaries, etc.

The scrum of scrums will track these working items via a backlog of its own, such as a ROAM board,[22] where each item contributes to improving cross-team coordination.[20]

As Jeff Sutherland commented,[20]

> *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 [edit]

This section **needs additional citations for verification**. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. *(March 2013)*

Product backlog [edit]

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 needs doing in order 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 open and editable by anyone, but the product owner is ultimately responsible for ordering the items on the backlog 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 backlog items; for example, if the "add spellcheck" and "add table support" 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).[23]

The product backlog and the business value of each backlog item is the responsibility of the product owner. The size (i.e. estimated complexity or effort) of each backlog item is, however, determined by the development team, who contributes by sizing items, either 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*[16] 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 is used to:

- capture requests for modifying a product. This can include adding new features, replacing old features, removing features and fixing issues
- ensure the development team is given work which maximizes the business benefit to the owner of the product

Typically, the product owner and the scrum team come together and write down everything that needs to 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 scrum backlog: features, bugs, technical work, and knowledge acquisition. Web development can entail confusion as to the difference between a feature and a bug: technically a feature is "wanted", while a bug is a feature that is "unintended" or "unwanted" (but may not be necessarily a defective thing). An example of technical work would be: "run virus check on all developers' workstations". An example of knowledge acquisition could be a scrum backlog item about researching Wordpress plugin libraries and making a selection.

Managing the product backlog between product owner and scrum team [edit]

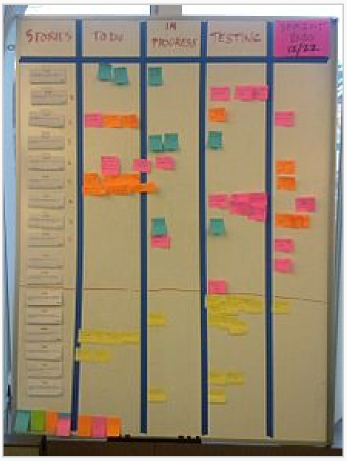
A backlog, in its simplest form, is merely a list of items to be worked on. Having well-established rules about how work is added, removed and ordered helps the whole team make better decisions about how to change the product.[*citation needed*]

The product owner prioritizes which of the items in the product backlog 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 that are going to be worked 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."[9]

As the team works through the backlog, it needs to be assumed that "changes in the world can happen"—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.[12]

Sprint backlog [edit]

The *sprint backlog* is the list of work the development team must address during the next sprint. The list is derived by selecting product backlog items from the top of the product backlog until the development team feels it has enough work to fill the sprint. This is done by the development team asking "Can we also do this?" and adding product backlog items to the sprint backlog. 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 are 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 functionality 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.

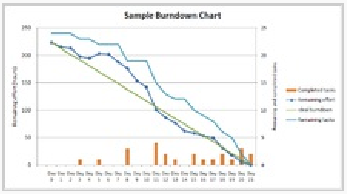
Product increment [edit]

The *increment* (or *potentially shippable increment*, PSI) is the sum of all the product backlog items completed during a sprint and all previous sprints. At the end of a sprint, the increment must be complete, according to the scrum team's Definition of Done (DoD), and in a usable condition regardless of whether the product owner decides to actually release it.

Sprint burn-down chart [edit]

Main article: Burn down chart

The sprint burndown 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. 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 way the actual burndown chart is updated day by day.



A sample burn down chart for a completed iteration, showing remaining effort and tasks for each of the 21 work days of the 1-month iteration

It should not be confused with an earned value chart.

Release burn-down chart [edit]

The release burndown chart is the way for the team to track progress and provide visibility. The release burndown chart is updated at the end of each sprint by the scrum master. The horizontal axis of the release burndown chart shows the sprints; the vertical axis shows the amount of work remaining at the start of each sprint. The release burndown chart makes it easy to drill down into a sprint and understand what is the remaining work, what work has been added, what work has been done, what work will have to be done. You can see what the team has completed as well as how scope changed.

Terminology [edit]

The following terms are often used in a scrum process.

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 process, 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 be completed during the sprint.

Sprint
A time period (typically 1–4 weeks) in which development occurs on a set of backlog items that the team has committed to. Also commonly referred to as a time-box or iteration.

Spike
A time boxed period used to research a concept and/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, and/or produce a proof of concept. The duration and objective(s) of a spike will be agreed between the 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.[24]

Tracer Bullet
The tracer bullet is a spike with the current architecture, current technology set, current set of best practices which 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 will connect as expected.[*citation needed*]

Tasks
Work items added to the sprint backlog at the beginning of a sprint and broken down into hours. Each task should not exceed 12 hours (or two days), but it's common for teams to insist that a task take no more than a day to finish.[*citation needed*]

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.[25]

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 from the last sprint's backlog items. The collection of historical velocity data is a guideline for assisting the team in understanding how much work they can do in a future sprint.

Impediment

Anything that prevents a team member from performing work as efficiently as possible.[26]

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.[11] 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 adapted the scrum process to their own needs in some way contradictory to supposed *pure* scrum.[27][28]

Scrum-ban [edit]

Main article: Scrum ban

Scrum-ban is a software production model based on scrum and kanban. Scrum-ban is especially suited for maintenance projects or (system) projects with frequent and unexpected work items or programming errors.[29] In such cases the time-limited sprints of the scrum model are of no appreciable use, but scrum's daily events and other practices can 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.[30]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[31] In the case of decentralized teams, stage-illustration software such as Assembla, TargetProcess, Eylean Board, JIRA or Agilo for Scrum.

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" or "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.[32]

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.[33] 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.[32]

The major differences between scrum and kanban are derived from the fact that, in scrum, work is divided into sprints that last a certain amount of time, whereas in Kanban the workflow is continuous. This is visible in work stage tables, which in scrum are emptied after each sprint. 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.[33]

Since scrum-ban is such a new development model, there is not much reference material. Kanban, on the other hand, has been applied by Microsoft and Corbis.[34]

Tools for implementation [edit]

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 packages dedicated to management of products under the scrum process. Other organizations implement scrum without the use of any software tools, and maintain their artifacts in hard-copy forms such as paper, whiteboards, and sticky notes.[35]

Adaptations [edit]

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 the project, organizations commonly add process guidance on 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.[36][37]

See also [edit]

- Disciplined Agile Delivery
- Extreme programming
- Kanban
- Lean software development
- Project management
- Unified Process

References [edit]

1. ^ "New Product Development Game". Harvard Business Review 86116:137–146, 1986. January 1, 1986. Retrieved March 12, 2013.
2. ^ *The Knowledge Creating Company*. Oxford University Press. 1995. p. 3. ISBN 9780199762330. Retrieved March 12, 2013.
3. ^ *a* *b* Takeuchi, Hirotaka; Nonaka, Ikujiro (January–February 1986). "The New Product Development Game" (PDF). *Harvard Business Review*. Retrieved June 9, 2010.
4. ^ *Scrum*, abbreviated form of *scrummage*, *Oxford English Dictionary Online*.
5. ^ *a* *b* Sutherland, Jeff (October 2004). "Agile Development: Lessons learned from the first Scrum" (PDF). Retrieved September 26, 2008.
6. ^ 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.
7. ^ The latest synthesis is published as "The Scrum Guide™ / The Definitive Guide to Scrum: The Rules of the Game", in various versions at http://www.scrumguides.org/
8. ^ Schwaber, Ken; Beedle, Mike (2002). *Agile software development with Scrum*. Prentice Hall. ISBN 0-13-067634-9.
9. ^ *a* *b* *c* *d* Schwaber, Ken (February 1, 2004). *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-7356-1993-7.
10. ^ Gauthier, Alexandre (August 17, 2011). "What is scrum". Planbox.
11. ^ *a* *b* *c* Ken Schwaber, Jeff Sutherland. "The Scrum Guide" (PDF). Scrum.org. Retrieved October 28, 2013.
12. ^ *a* *b* Pichler, Roman. Agile product management with Scrum : creating products that customers love. Upper Saddle River, NJ: Addison-Wesley, 2010.
13. ^ Cohn, Mike. Succeeding with agile : software development using Scrum. Upper Saddle River, NJ: Addison-Wesley, 2010.
14. ^ Leybourn, E. (2013). Directing the Agile Organisation: A Lean Approach to Business Management. London: IT Governance Publishing: 117–120.
15. ^ "Core Scrum". *Scrum Alliance*. Retrieved 1-25-15. Check date values in: `|accessdate=` (help)
16. ^ *a* *b* *c* Pete Deemer, Gabrielle Benefield, Craig Larman, Bas Vodde (2012-12-17). "The Scrum Primer: A Lightweight Guide to the Theory and Practice of Scrum (Version 2.0)". InfoQ.
17. ^ Cohn, Mike (22 October 2013). "What is Agile Project Management?". Mountain Goat Software. Retrieved 30 March 2015.
18. ^ *a* *b* "Agile SCRUM For Denver Web Development". `|first1=` missing `|last1=` in Authors list (help)
19. ^ 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.
20. ^ *a* *b* *c* *d* "Scrum of Scrums". Agile Alliance.
21. ^ "Scrum of Scrums". *Scrum Master Test Prep*. Retrieved 29 May 2015.
22. ^ "Risk Management – How to Stop Risks from Screwing Up Your Projects!". Kelly Waters.
23. ^ Higgins, Tony (March 31, 2009). "Authoring Requirements in an Agile World". BA Times.
24. ^ "Create a Spike Solution". Extreme Programming.
25. ^ Ken Schwaber, *Agile Project Management with Scrum*, p.55
26. ^ Little, Joe (January 17, 2011). "Impediment Management". Agile Consortium.
27. ^ "ScrumButs and Modifying Scrum". *Scrum.org*. Retrieved March 18, 2013.
28. ^ Bloomberg, Jason (July 31, 2012). "The Scrum But Paradox". *DevX.com*. QuinStreet. Retrieved March 18, 2013.
29. ^ Ian Mitchell (November 11, 2013). "Scrumban - or How to Get Leaner by Sprinting Less". Retrieved March 17, 2014.
30. ^ "p.5 Crisp.se" (PDF). June 29, 2009. Retrieved September 13, 2012.
31. ^ "Leansoftwareengineering.com". Retrieved September 13, 2012.
32. ^ *a* *b* "Leansoftwareengineering.com". Leansoftwareengineering.com. October 27, 2007. Retrieved September 13, 2012.
33. ^ *a* *b* "Kanban vs. Scrum: How to Make the Most of Both" (PDF). June 29, 2009. Retrieved September 13, 2012.
34. ^ "(The video and the summary)". Infoq.com. March 26, 2009. Retrieved September 13, 2012.
35. ^ Dubakov, Michael (2008). "Agile Tools. The Good, the Bad and the Ugly" (PDF). Retrieved August 30, 2010.
36. ^ https://sites.google.com/a/scrumorgpatterns.com/www/
37. ^ Scrum Pattern Community ScrumPlop

Further reading [edit]

- Maria Almeida (2015). "How to be a Great Product Owner". JOBBOX.io.
- Jeff Sutherland; Ken Schwaber (2013). "Scrum Guides". ScrumGuides.org. Retrieved October 2013.
- 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.
- SCRUMstudy (2013). "A Guide to the Scrum Body of Knowledge". VMEdu,Inc. Retrieved December 1, 2013.
- "The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework" (Jeff Sutherland, www.Scruminc.com, 2 Apr 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

External links [edit]

- Agile Alliance's Scrum library

- A Scrum Process Description by the Eclipse Process Framework (EPF) Project

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

v · t · e **Software engineering** [hide]

Fields	Computer programming · Requirements engineering · Software deployment · Software design · Software maintenance · Software testing · Systems analysis · Formal methods	
Concepts	Data modeling · Enterprise architecture · Functional specification · Modeling language · Orthogonality · Programming paradigm · Software · Software archaeology · Software architecture · Software configuration management · Software development methodology · Software development process · Software quality · Software quality assurance · Software verification and validation · Structured analysis	
Orientations	Agile · Aspect-oriented · Object orientation · Ontology · Service orientation · SDLC	
Models	Developmental	Agile · EUP · Executable UML · Incremental model · Iterative model · Prototype model · RAD · UP
	Other	SPICE · CMMI · Data model · ER model · Function model · Information model · Metamodeling · Object model · Systems model · View model
	Languages	IDEF · UML · SysML
Software engineers	Kent Beck · Grady Booch · Fred Brooks · Barry Boehm · Ward Cunningham · Tom DeMarco · Edsger W. Dijkstra · Martin Fowler · C. A. R. Hoare · Watts Humphrey · Michael A. Jackson · Ivar Jacobson · Stephen J. Mellor · Bertrand Meyer · David Parnas · Winston W. Royce · James Rumbaugh · Niklaus Wirth · Edward Yourdon · Victor Basili	
Related fields	Computer science · Computer engineering · Project management · Risk management · Systems engineering	

Category · Commons

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