Scrum (development)

From Wikipedia, the free encyclopedia

Scrum is an iterative and incremental agile software development method for managing software projects and product or application development. Scrum has not only reinforced the interest in project management[*citation needed*], but also challenged the conventional ideas about such management. Scrum focuses on project management institutions where it is difficult to plan ahead. Mechanisms of *empirical process control*, where feedback loops that constitute the core management technique are used as opposed to traditional command-and-control oriented management.[*citation needed*] It represents a radically new approach for planning and managing projects, bringing decision-making authority to the level of operation properties and certainties.[1]

Scrum as applied to product development was first referred to in "New New Product Development Game"[2] (Harvard Business Review 86116:137–146, 1986) and later elaborated in "The Knowledge Creating Company"[3] both by Ikujiro Nonaka and Hirotaka Takeuchi (Oxford University Press, 1995). Today there are records of Scrum used to produce financial products, Internet products, and medical products by ADM.

Contents

History

In 1986, Hirotaka Takeuchi and Ikujiro Nonaka 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.[4] 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".[4]

In rugby football, a scrum refers to the manner of restarting the game after a minor infraction. 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 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.

In 2001, Schwaber worked with Mike Beedle to describe the method in the book *Agile Software Development with Scrum*.[7]

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.[1]

Roles

There are three core roles[8] and a range of ancillary roles—core roles are often referred to as *pigs* and ancillary roles as *chickens* (after the story The Chicken and the Pig).

Core roles

The 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**.

Product Owner
: 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 customer-centric items (typically user stories), 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, it is recommended that this role not be combined with that of Scrum Master.[9]

Development Team
: The Development Team is responsible for delivering potentially shippable product increments at the end of each Sprint. A Development Team is made up of 3–9 people with cross-functional skills who do the actual work (analyse, design, develop, test, technical communication, document, etc.). The Development Team in Scrum is self-organizing, even though they may interface with project management organizations (PMOs).

Scrum Master
: Scrum is facilitated by a Scrum Master, sometimes written as *ScrumMaster*, who is accountable for removing impediments to the ability of the team to deliver the sprint goal/deliverables. The Scrum Master is not the team leader, 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 is the enforcer of rules. A key part of the

Scrum Master’s role is to protect the Development Team and keep it focused on the tasks at hand. The role has also been referred to as a *servant-leader* to reinforce these dual perspectives.

Ancillary roles

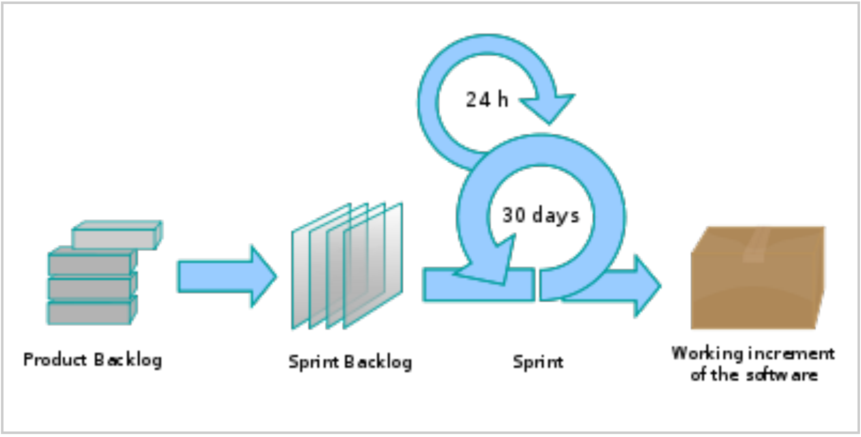
The ancillary roles in Scrum teams are those with no formal role and infrequent involvement in the Scrum process —but nonetheless, they must be taken into account.

Stakeholders
: The stakeholders are the customers, vendors. They are people who enable the project and for whom the project produces the agreed-upon benefit[s] that justify its production. They are only directly involved in the process during the sprint reviews.

Managers
: People who control the environment.

Sprint

A sprint is the basic unit of development in Scrum. Sprints last between one week and one month,[1] and are a "timeboxed" (i.e. restricted to a specific duration) effort of a constant length.[10]



The Scrum process.

Each sprint is preceded by a planning meeting, where the tasks for the sprint are identified and an estimated commitment for the sprint goal is made, and followed by a review or retrospective meeting,[11] where the progress is reviewed and lessons for the next sprint are identified.

During each sprint, the team creates finished portions of a product. The set of features that go into a sprint come from the product *backlog*, which is a prioritized list of requirements. Which backlog items go into the sprint (the sprint goals) is determined during the sprint planning meeting. During this meeting, the Product Owner informs the team of the items in the product backlog that he or she wants completed (the ones with the highest priority). The team then determines how much of this they can commit to complete during the next sprint, and records this in the sprint backlog.[1] The sprint backlog is property of the development team, i.e. during a sprint, no one is allowed to edit the sprint backlog except for the development team. The sprint goals should not be changed during the sprint. Development is timeboxed such that the sprint must end on time; if requirements are not completed for any reason they are left out and returned to the product backlog. After a sprint is completed, the team demonstrates how to use the software.

Scrum enables the creation of self-organizing teams by encouraging co-location of all team members, and verbal communication between 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.

Like other agile development methodologies, 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 tools, and maintain their artifacts in hard-copy forms such as paper, whiteboards, and sticky notes.[12]

Meetings

Daily Scrum

Each day during the sprint, a project status meeting occurs. This is called a *daily scrum*, or *the daily standup*. This meeting has specific guidelines:

- The meeting starts precisely on time.
- All are welcome, but normally only the core roles speak
- The meeting length is set (timeboxed) to 15 minutes
- The meeting should happen at the same location and same time every day

A daily sprint meeting in the computing room. This place choice lets the team start in time.

During the meeting, each team member answers three questions:[13]

- What have you done since yesterday?
- What are you planning to do today?
- Any impediments/stumbling blocks?

  It is the role of the Scrum Master to facilitate resolution of these impediments, although the resolution should occur outside the Daily Scrum itself to keep it under 15 minutes.

Backlog grooming: storytime

The team should spend time during a sprint doing product backlog grooming. This is the process of estimating the existing backlog using effort/points, refining the acceptance criteria for individual stories, and breaking larger stories into smaller stories.

- Meetings should not be longer than an hour
- Meeting does not include breaking stories into tasks
- The team can decide how many meetings are needed per week.

The most commonly used method is the planning poker.

Scrum of Scrums

Each day normally after the Daily Scrum.

- These meetings allow clusters of teams to discuss their work, focusing especially on areas of overlap and integration.
- A designated person from each team attends.

The agenda will be the same as the Daily Scrum, plus the following four questions:

- What has your team done since we last met?
- What will your team do before we meet again?
- Is anything slowing your team down or getting in their way?
- Are you about to put something in another team's way?

Sprint planning meeting[14][15]

At the beginning of the sprint cycle (every 7–30 days), a "Sprint planning meeting" is held.

- 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 the current sprint
- Eight-hour time limit
  - (1st four hours) Entire team [16]: dialog for prioritizing the Product Backlog
  - (2nd four hours) Development Team [17] : hashing out a plan for the Sprint, resulting in the Sprint Backlog

At the end of a sprint cycle, two meetings are held: the "Sprint Review Meeting" and the "Sprint Retrospective"

Sprint review meeting[18]

- Review the work that was completed and not completed
- Present the completed work to the stakeholders (a.k.a. "the demo")
- Incomplete work cannot be demonstrated
- Four-hour time limit

Sprint retrospective[19]

- All team members reflect on the past sprint
- Make continuous process improvements
- Two main questions are asked in the sprint retrospective: What went well during the sprint? What could be improved in the next sprint?
- Three-hour time limit

Artifacts

Product Backlog

The **product backlog** is an ordered list of "requirements" that is maintained for a product. It contains Product Backlog Items that are ordered by the Product Owner based on considerations like risk, business value, dependencies, date needed, etc. The features added to the backlog are commonly written in story format (See terminology below). The product backlog is the "What" that will be built, sorted in the relative order it should be built in. It is open and editable by anyone, but the Product Owner is ultimately responsible for ordering the stories on the backlog for the Development Team. The product backlog contains rough estimates of both business value and development effort, these values are often stated in story points using a rounded Fibonacci sequence. Those 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 one with the smallest development effort will probably have higher priority, because the ROI (Return on Investment) is higher.

The Product Backlog, and business value of each listed item is the responsibility of the Product Owner. The estimated effort to complete each backlog item is, however, determined by the Development Team. The team contributes by estimating Items and User-Stories, either in Story-points or in estimated hours.

Sprint Backlog

The **sprint backlog** is the list of work the Development Team must address during the next sprint. The list is derived by selecting stories/features 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 stories/features to the sprint backlog. The Development Team should keep in mind the velocity of its previous Sprints (total story points completed from each of the last sprints stories) when selecting stories/features for the new sprint, and use this number as a guide line of how much "effort" they can complete.

A scrum task board.

The stories/features are broken down into tasks by the Development Team, which, as a best practice, should normally be between four and sixteen hours of work. With this level of detail the Development Team understands exactly what to do, and potentially, anyone can pick a task from the list. Tasks on the sprint backlog are never assigned; rather, tasks are signed up for by the team members as needed during the daily scrum, according to the set priority and the Development Team member skills. This promotes self-organization of the Development Team, and developer buy-in.

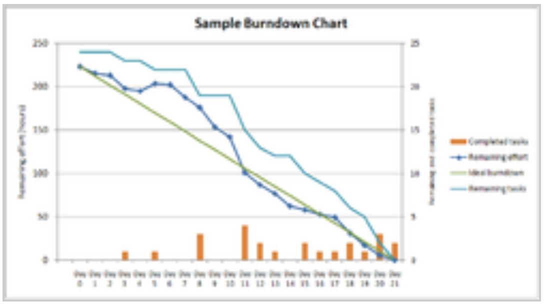
The sprint backlog is the property of the Development Team, and all included estimates are provided by the Development Team. Often an accompanying **task board** is used to see and change the state of the tasks of the current sprint, like “to do”, “in progress” and “done”.

Increment

The **increment** 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 done according to the Scrum Team's definition of done. The increment must be in usable condition regardless of whether the Product Owner decides to actually release it.

Burn down

Main article: burn down chart



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.

The sprint burn down chart is a publicly 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. There are also other types of burndown, for example the **release burndown chart** that shows the amount of work left to complete the target commitment for a Product Release (normally spanning through multiple iterations) and the **alternative release burndown chart**,[20] which basically does the same, but clearly shows scope changes to Release Content, by resetting the baseline.

It should not be confused with an earned value chart.

Terminology

The following terminology is used in Scrum:[21]

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.

Product backlog
: A prioritized list of high-level requirements.

Sprint backlog
: 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.

(User) Story
: A feature that is added to the backlog is commonly referred to as a story and has a specific suggested structure. The structure of a story is: "As a <user type> I want to <do some action> so that <desired result>" This is done so that the development team can identify the user, action and required result in a request and is a simple way of writing requests that anyone can understand. Example: As a wiki user I want a tools menu on the edit screen so that I can easily apply font formatting.

A story is an *independent, negotiable, valuable, estimatable, small, testable requirement* ("INVEST Acronym"). Despite being independent i.e. they have no direct dependencies with other requirements, stories may be clustered into epics when represented on a product roadmap or further down in the backlog.

Theme
: A theme is a top-level objective that may span projects and products. Themes may be broken down into sub-themes, which are more likely to be product-specific. Themes can be used at both program and project level to drive strategic alignment and communicate a clear direction.

Epic
: An epic is a group of related stories, mainly used in product roadmaps and the backlog for features that have not yet been analyzed enough to break it down into it's component stories, which should be done before bringing it into a sprint so to reduce uncertainty. Epics can also be used at a both program and project level.

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 epics or user stories 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 Delivery 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.

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 rest of the iterations can build on this code.

Point Scale/Effort/Story points
: Relates to an abstract point system, used to discuss the difficulty of the story, without assigning actual hours. The most common scale used is a rounded Fibonacci sequence (1,2,3,5,8,13,20,40,100), although some teams use linear scale (1,2,3,4...), powers of two (1,2,4,8...), and clothes size (XS, S, M, L, XL).[22]

Tasks
: Added to the story at the beginning of a sprint and broken down into hours. Each task should not exceed 12 hours, but it's common for teams to insist that a task take no more than a day to finish.

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.

Velocity
: The total effort a team is capable of in a sprint. The number is derived by adding all the story points from the last sprint's stories/features. This is a guideline for the team and assists them in understanding how many stories they can do in a sprint.

Impediment
: Anything that prevents a team member from performing work as efficiently as possible.[23]

Sashimi
: A report that something is "done". The definition of "done" may vary from one Scrum team to another, but must be consistent within one team.

Abnormal Termination
: The Product Owner can cancel a Sprint if necessary.[24] 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 meeting, where the reason for the termination is reviewed.

Planning Poker
: In the Sprint Planning Meeting, the team sits down to estimate its effort for the stories in the backlog. The Product Owner needs these estimates, so that he or she is empowered to effectively prioritize items in the backlog and, as a result, forecast releases based on the team's velocity.[22]

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 user stories or programming errors. In such cases the time-limited sprints of the Scrum model are of no appreciable use, but Scrum's daily meetings 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 user stories 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 user story or programming error, and on the other hand ensures each team member is constantly employed.[25]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[26] In the case of decentralized teams, stage-illustration such as Assembla, ScrumWorks, Rational Team Concert or JIRA in combination with GreenHopper can be used to visualize each team's user stories, defects and tasks divided into separate phases.

In their simplest, the tasks or usage stories 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.[27]

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.[28] 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.[27]

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.[29]

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.[30]

Project management tools that support scrum

- Banana Scrum (http://www.bananascrum.com/)
- CollabNet ScrumWorks Pro
- Hansoft (http://www.hansoft.se/)
- Kunagi (http://kunagi.org/)
- IBM Rational Team Concert
- Ice Scrum (http://www.icescrum.org)
- JIRA using Green Hopper plugin
- Mingle (http://www.thoughtworks-studios.com/) by ThoughtWorks Studios
- OneDesk
- PangoScrum (http://pangoscrum.com/)
- Pivotal Tracker
- Planbox (http://www.planbox.com/)
- Rally Software
- Redmine and ChiliProject, with a plug-in (several are available)
- ScrumDo (http://www.scrumdo.com/)
- ScrumPad Pro (http://www.scrumpad.pro/)
- Scrumwise (http://www.scrumwise.com/)
- Scrumy (http://scrumy.com)
- Sprintometer (http://sprintometer.com)
- Tinypm
- VersionOne (http://versionone.com)
- Visual Studio 2010, Microsoft Team Foundation Server
- Yodiz (http://www.yodiz.com/)

See also

- Kaizen
- Code monkey
- Extreme programming (XP)
- Test-driven development

- Feature-driven development
- Lean
- Kanban

References

1. ^ *a b c d* Schwaber, Ken (1 February 2004). *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-7356-1993-7.
2. ^ New New Product Development Game (http://cb.hbsp.harvard.edu/cb/web/product_detail.seam;jsessionid=8D8BACDD4CC4F2F87B3CD58D2ED10332?R=86116-PDF-ENG&conversationId=22181&E=48834)
3. ^ The Knowledge Creating Company (http://books.google.ru/books?hl=en&id=B-qxrPaU1-MC&dq=The+Knowledge+Creating+Company&printsec=frontcover&source=web&ots=XfRLlzreeT&sig=B5tPPUI hBTlmi4cQLVYosoWs)
4. ^ *a b* Takeuchi, Hirotaka; Nonaka, Ikujiro (January–February 1986). "The New Product Development Game" (http://hbr.org/product/new-new-product-development-game/an/86116-PDF-ENG) (PDF). *Harvard Business Review*. http://hbr.org/product/new-new-product-development-game/an/86116-PDF-ENG. Retrieved 2010-06-09.
5. ^ Sutherland, Jeff (2004-10). "Agile Development: Lessons learned from the first Scrum" (http://www.scrumalliance.org/resources/35) (PDF). http://www.scrumalliance.org/resources/35. Retrieved 2008-09-26.
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. ^ Schwaber, Ken; Beedle, Mike (2002). *Agile software development with Scrum*. Prentice Hall. ISBN 0-13-067634-9.
8. ^ Gauthier, Alexandre (August 17th, 2011). "What is scrum" (http://www.planbox.com/resources/agile-artifacts#roles) . Planbox. http://www.planbox.com/resources/agile-artifacts#roles.
9. ^ "Scrum, Scrum Developer Courses, Scrum Knowledge Assessment, Scrum Guide, Ken Schwaber - Scrum Guides" (http://www.scrum.org/scrumguides/) . Scrum.org. 2009. http://www.scrum.org/scrumguides/. Retrieved 2010-04-03.
10. ^ Sprint, Planning (January-February 2009) (html). *Sprint Planning Rules* (http://www.sprintplanning.com/SprintPlanningRules.aspx) . http://www.sprintplanning.com/SprintPlanningRules.aspx. Retrieved 2009-03-30.
11. ^ Sutherland, Jeff (2004-10). "Agile Development: Lessons learned from the first Scrum" (http://www.scrumalliance.org/resources/35) (PDF). http://www.scrumalliance.org/resources/35. Retrieved 2008-09-26.
12. ^ Dubakov, Michael (2008). "Agile Tools. The Good, the Bad and the Ugly." (http://targetprocess.com/download/whitepaper/agiletools.pdf) (PDF). http://targetprocess.com/download/whitepaper/agiletools.pdf. Retrieved 2010-08-30.
13. ^ Schwaber, p. 135
14. ^ Schwaber, p. 133
15. ^ Sprint, Planning (January–February 2009). *Sprint Planning Rules* (http://www.sprintplanning.com/SprintPlanningRules.aspx) . http://www.sprintplanning.com/SprintPlanningRules.aspx. Retrieved 2009-03-30.
16. ^ "The Scrum Guide" (http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf) . p. 9. http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf.
17. ^ "The Scrum Guide" (http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf) . pp. 9-10. http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf.
18. ^ Schwaber, p. 137
19. ^ Schwaber, p. 138
20. ^ Invented by Mike Cohn, more info can be found here (http://www.mountaingoatsoftware.com/pages/19-an-alternative-release-burndown-chart)
21. ^ Schwaber, pp. 141–143
22. ^ *a b* "Scrum Effort Estimation and Story Points" (http://scrummethodology.com/scrum-effort-estimation-and-story-

22. "Scrum Effort Estimation and Story Points" (http://scrummethodology.com/scrum-effort-estimation-and-story-points) . http://scrummethodology.com/scrum-effort-estimation-and-story-points.
23. ^ Little, Joe (January 17, 2011). *Impediment Management* (http://agileconsortium.blogspot.com/2011/01/impediment-management.html) . Agile Consortium. http://agileconsortium.blogspot.com/2011/01/impediment-management.html
24. ^ "The Scrum Guide" (http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf) . pp. 8. http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%202011.pdf.
25. ^ p.5 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
26. ^ Leansoftwareengineering.com (http://leansoftwareengineering.com/wp-content/uploads/2008/04/scrumban-001.jpg)
27. ^ *a b* Leansoftwareengineering.com (http://leansoftwareengineering.com/ksse/scrum-ban/)
28. ^ p.18 - 19 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
29. ^ p.22 - 23 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
30. ^ Infoq.com (The video and the summary) (http://www.infoq.com/presentations/kanban-for-software)

Further reading

- "The Scrum Guide" (http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf) . 2011. http://www.scrum.org/storage/scrumguides/Scrum_Guide.pdf. Retrieved 2011-12-3.
- "The Scrum Software Development Process for Small Teams" (http://members.cox.net/risingl1/Articles/IEEEScrum.pdf) . 2000. http://members.cox.net/risingl1/Articles/IEEEScrum.pdf. Retrieved 2007-03-15.
- Deemer, Pete; Benefield, Gabrielle; Larman, Craig; Vodde, Bas (2009). "The Scrum Primer" (http://scrumtraininginstitute.com/home/stream_download/scrumprimer) . http://scrumtraininginstitute.com/home/stream_download/scrumprimer. Retrieved 2009-06-01.
- Kniberg, Henrik. "Scrum and XP from the Trenches" (http://www.infoq.com/minibooks/scrum-xp-from-the-trenches) . http://www.infoq.com/minibooks/scrum-xp-from-the-trenches. Retrieved 2010-08-09.

External links

- Scrum.org (http://www.scrum.org/)
- Scrum Alliance, non-profit (http://www.scrumalliance.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)
- BPMN process diagram of Scrum (http://www.ariscommunity.com/users/sstein/2010-08-09-bpm-view-scrum)
- A six-page illustrated Scrum reference (http://ScrumReferenceCard.com/)
- Burn Down Chart Tutorial: Simple Agile Project Tracking (Scrum) (http://joel.inpointform.net/software-development/burn-down-charts-tutorial-simple-agile-project-tracking/)
- Scrum Development Interview Questions and Answers (http://www.questions-interviews.com/software-development-testing-models/scrum.aspx)

Videos and slides

- Jeff Sutherland in *Scrum Tuning: Lessons learned from Scrum implementation at Google* (http://video.google.com/videoplay?docid=8795214308797356840) Retrieved 2007-12-15
- Ken Schwaber in *Scrum et al.* (http://video.google.com/videoplay?docid=2531954797594836634) Retrieved 2008-01-19
- Jeff Sutherland in *Hyperproductive Distributed Scrum Teams* (http://www.youtube.com/watch?v=Ht2xcIJrAXo)
- Jeff Sutherland in *Self-Organization: The Secret Sauce for Improving your Scrum team* (http://www.youtube.com/watch?v=M1q6b9JI2Wc)

- Bruno Sbille and his team in *Scrum applied on a real-world project* (HD) (http://www.vimeo.com/4587652) Retrieved 2009-05-19
- Scrum at Large: Managing 100 People and More (http://www.tvagile.com/2009/07/24/scrum-at-large-managing-100-people-and-more/)
- Andrea Tomasini's keynote about Agile in an Enterprise (http://www.slideshare.net/lourenh1/adopting-scrum-an-enterprise-transformation)
- Scrum Training Series: Scrum Tutorial (http://ScrumTrainingSeries.com)

Retrieved from "http://en.wikipedia.org/w/index.php?title=Scrum_(development)&oldid=498917370"
Categories: Agile software development | Management | Production and manufacturing | Project management