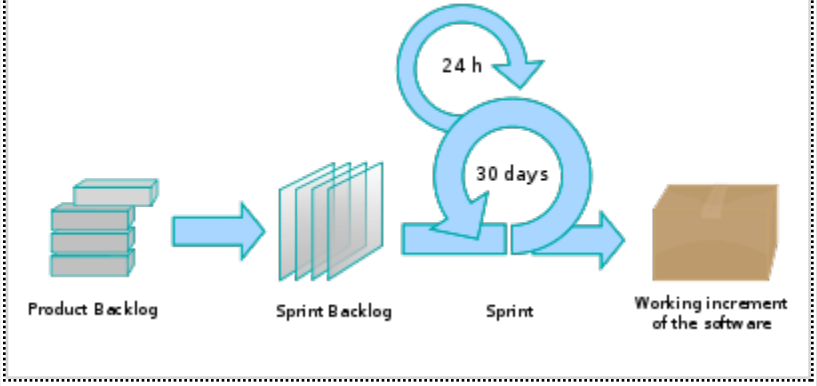
Scrum (development)

From Wikipedia, the free encyclopedia

Scrum is an iterative, incremental methodology for project management often seen in agile software development, a type of software engineering.

Although Scrum was intended for management of software development projects, it can be used to run software maintenance teams, or as a general project/program management approach.



The Scrum process.

Contents

Software development process

Activities and steps

Requirements · Specification
Architecture · Design
Implementation · Testing
Deployment · Maintenance

Methodologies

Agile · Cleanroom · Iterative
RAD · RUP · Spiral
Waterfall · XP · Lean
Scrum · V-Model · TDD

Supporting disciplines

Configuration management
Documentation
Quality assurance (SQA)
Project management
User experience design

Tools

Compiler · Debugger · Profiler
GUI designer · IDE

History

In 1986, Hirotaka Takeuchi and Ikujiro Nonaka described a new holistic approach that would increase speed and flexibility in commercial new product development.[1] They compared this new, holistic approach, in which the phases strongly overlap and the whole process is performed by one cross-functional team across the different phases, to rugby, where the whole team "tries to go the distance as a unit, passing the ball back and forth". The case studies came from the automotive, photo machine, computer, and printer industries.[1]

In 1991, DeGrace and Stahl, in "Wicked Problems, Righteous Solutions",[2] referred to this approach as scrum, a rugby term mentioned in the article by Takeuchi and Nonaka. In the early 1990s, Ken Schwaber used an approach that led to Scrum at his company, Advanced Development Methods. At the same time, Jeff Sutherland, John Scumniotales, and Jeff McKenna developed a similar approach at Easel Corporation and were the first to call it Scrum.[3]

In 1995, Sutherland and Schwaber jointly presented a paper describing Scrum at the Business Object Design and Implementation workshop held as part of OOPSLA ’95 in Austin, Texas, its first public appearance. 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 teamed up with Mike Beedle to describe the method in the book "Agile Software Development with Scrum".

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.[4]

Characteristics

Scrum is a process skeleton that contains sets of practices and predefined roles. The main roles in Scrum are:

1. the “**ScrumMaster**”, who maintains the processes (typically in lieu of a project manager)
2. the “**Product Owner**”, who represents the stakeholders and the business
3. the “**Team**”, a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.

During each “sprint”, typically a two to four week period (with the length being decided by the team), the team creates a potentially shippable product increment (for example, working and tested software). The set of features that go into a sprint come from the product “backlog”, which is a prioritized set of high level requirements of work to be done. Which backlog items go into the sprint 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 team then determines how much of this they can commit to complete during the next sprint.[4] During a sprint, no one is allowed to change the sprint backlog, which means that the requirements are frozen for that 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 software tools, such as spreadsheets to build and maintain artifacts such as the sprint backlog. There are also open-source and proprietary software 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.[5]

Roles

Main article: The Chicken and the Pig

A number of roles are defined in Scrum. All roles fall into two distinct groups—pigs and chickens—based on the nature of their involvement in the development process. These groups get their names from a joke [6] about a pig and a chicken opening a restaurant:[7]

> A pig and a chicken are walking down a road. The chicken looks at the pig and says, "Hey, why don't we open a restaurant?" The pig looks back at the chicken and says, "Good idea, what do you want to call it?" The chicken thinks about it and says, "Why don't we call it 'Ham and Eggs'?" "I don't think so," says the pig, "I'd be committed, but you'd only be involved."

So the "pigs" are committed to building software regularly and frequently, while everyone else is a "chicken"—interested in the project but really indifferent because if it fails they're not the pigs—that is, they weren't the ones that committed to doing it. The needs, desires, ideas and influences of the chicken roles are taken into account, but are not in any way allowed to get in the way of the actual Scrum project.

"Pig" roles

The Pigs are the ones committed to the project in the Scrum process—they are the ones with "their bacon on the line" and performing the actual work of the project.

ScrumMaster
: Scrum is facilitated by a ScrumMaster, also written as *Scrum Master*, whose primary job is to remove impediments to the ability of the team to deliver the sprint goal/deliverables. The ScrumMaster is not the leader of the team (as the team is self-organizing) but acts as a buffer

between the team and any distracting influences. The ScrumMaster ensures that the Scrum process is used as intended. The ScrumMaster is the enforcer of rules. A key part of the ScrumMaster's role is to protect the team and keep them focused on the tasks in hand.

Team
: The team has the responsibility to deliver the product. A team is typically made up of 5–9 people with cross-functional skills who do the actual work (design, develop, test, technical communication, etc.).

Product Owner
: The Product Owner represents the voice of the customer. He/she ensures that the Scrum Team works with the "right things" from a business perspective. The Product Owner writes customer-centric items (typically user stories), prioritizes them and then places them in the product backlog. A Product Owner can be a member of the Scrum Team but cannot be a ScrumMaster.[8] According to original Scrum, Product Owner is in a "pig" role. However, in the context of the Sprint and the daily Stand-Up meetings, the Product Owner is considered a "chicken" since he has no role in the implementation of Sprint tasks.

"Chicken" roles

Chicken roles are not part of the actual Scrum process, but must be taken into account. They are people for whom the software is being built.

Stakeholders (customers, vendors)
: These are the people who enable the project and for whom the project will produce the agreed-upon benefit[s], which justify its production. They are only directly involved in the process during the sprint reviews.

Managers
: People who will set up the environment for the product development organizations.

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 only "pigs" may speak
  - The meeting is timeboxed to 15 minutes
  - The meeting should happen at the same location and same time every day

  During the meeting, each team member answers three questions:[9]
  - What have you done since yesterday?
  - What are you planning to do today?
  - Do you have any problems preventing you from accomplishing your goal? (It is the role of the ScrumMaster to facilitate resolution of these impediments. Typically this should occur outside the context of the Daily Scrum so that it may stay under 15 minutes.)

Sprint Planning Meeting[10][11]

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) Product Owner + Team: dialog for prioritizing the Product Backlog
  - (2nd four hours) Team only: 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

[12]

- 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

[13]

- 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 a high-level list that is maintained throughout the entire project. It aggregates backlog items: broad descriptions of all potential features, prioritized as an absolute ordering by business value. It is therefore the "What" that will be built, sorted by importance. It is open and editable by anyone and contains rough estimates of both business value and development effort. Those estimates help the Product Owner to gauge the timeline and, to a limited extent prioritize. 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 property of the product owner. The associated development effort is however set by the Team.

Sprint backlog

The **sprint backlog** is the list of work the team must address during the next sprint. Features are broken down into tasks, which, as a best practice, should normally be between four and sixteen hours of work. With this level of detail the whole 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, according to the set priority and the team member skills. This promotes self-organization of the team, and developer buy-in.

The sprint backlog is the property of the team, and all included estimates are provided by the 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”.

Burn down

Main article: burn down chart

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**,[14] 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.

Adaptive project management

The following are some general practices of Scrum:

- "Working more hours" does not necessarily mean "producing more output"
- "A happy team makes a tough task look simple"

Terminology

The following terminology is used in Scrum:[15]

Roles

Product Owner
: The person responsible for maintaining the Product Backlog by representing the interests of the stakeholders.

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

Team
: A cross-functional group of people responsible for managing itself to develop the product.

Scrum Team

Product Owner, ScrumMaster and Team

Artifacts

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.

Others

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

Sprint
: A time period (typically 2–4 weeks) in which development occurs on a set of backlog items that the Team has committed to.

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.[16] 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.[17]

Point Scale
: Relates to an abstract point system, used to discuss the difficulty of the task, without assigning actual hours. Common systems of scale are linear (1,2,3,4...), Fibonacci (1,2,3,5,8...), Powers-of-2 (1,2,4,8...), and Clothes size (XS, S, M, L, XL).[17]

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.

Scrum modifications

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.[18]

To illustrate each stage of work, teams working in the same space often use post-it notes or a large whiteboard.[19] In the case of decentralized teams stage illustration software, such as Assembla, ScrumWorks, 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.[20]

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.[21] 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.[20]

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.[22]

Since Scrum-ban is such a new development model, there is not much reference material. Kanban, on the other hand, has been applied in software development at least by Microsoft and Corbis.[23]

Product development

Scrum as applied to product development was first referred to in "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) " (Harvard Business Review 86116:137–146, 1986) and later elaborated in "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=B5tPPUD6s-hBTlmi4cQLVYosoWs) " 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.

See also

- Kaizen
- List of software development philosophies

Other Agile methods

- Dynamic System Development Method
- Extreme programming (XP)
- Feature Driven Development
- Lean software development

References

1. ^ *a b* Takeuchi, Hirotaka; Nonaka, Ikujiro (January–February 1986). "The New 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.
2. ^ DeGrace, Peter; Stahl, Leslie Hulet (1990-10-01). *Wicked problems, righteous solutions*. Prentice Hall. ISBN 978-0-135-90126-7.
3. ^ 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.
4. ^ *a b* Schwaber, Ken (1 February 2004). *Agile Project Management with Scrum*. Microsoft Press. ISBN 978-0-735-61993-7.
5. ^ 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.
6. ^ "The Classic Story of the Pig and Chicken" (http://www.implementingscrum.com/2006/09/11/the-classic-story-of-the-pig-and-chicken/) . Implementing Scrum. 11 September 2006. http://www.implementingscrum.com/2006/09/11/the-classic-story-of-the-pig-and-chicken/. Retrieved 2010-04-03.
7. ^ Schwaber, p. 7
8. ^ "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.
9. ^ Schwaber, p. 135
10. ^ Schwaber, p. 133
11. ^ Sprint, Planning (January–February 2009). *Sprint Planning Rules* (http://www.sprintplanning.com /SprintPlanningRules.aspx) . http://www.sprintplanning.com/SprintPlanningRules.aspx. Retrieved 2009-03-30.
12. ^ Schwaber, p. 137
13. ^ Schwaber, p. 138
14. ^ Invented by Mike Cohn, more info can be found here (http://www.mountaingoatsoftware.com/pages/19-an-alternative-release-burndown-chart)
15. ^ Schwaber, pp. 141–143
16. ^ "Scrum: Developed and Sustained" (http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf) pp. 11 http://www.scrum.org/storage/scrumguides/Scrum%20Guide.pdf
17. ^ *a b* "Scrum Effort Estimation and Story Points" (http://scrummethodology.com/scrum-effort-estimation-

and-story-points) . http://scrummethodology.com/scrum-effort-estimation-and-story-points.
18. ^ p.5 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
19. ^ Leansoftwareengineering.com (http://leansoftwareengineering.com/wp-content/uploads/2008/04/scrumban-001.jpg)
20. ^ *a b* Leansoftwareengineering.com (http://leansoftwareengineering.com/ksse/scrum-ban/)
21. ^ p.18 - 19 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
22. ^ p.22 - 23 Crisp.se (http://www.crisp.se/henrik.kniberg/Kanban-vs-Scrum.pdf)
23. ^ Infoq.com (The video and the summary) (http://www.infoq.com/presentations/kanban-for-software)

Further reading

- "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/) by Ken Schwaber
- Scrum Alliance (http://www.scrumalliance.org/)
- Agile Alliance's Scrum library (http://cf.agilealliance.org/articles/article_list.cfm?CategoryID=17)
- A Scrum Process Asset Library (http://scrum.gem-up.com/)
- 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/)

Videos

- 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/)

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