Table of Contents

* Agile Manifesto
* Agile Methodologies
* Scrum Overview
* Sprint
* User Story
* Scrum Roles
* Scrum Artifacts
* Dashboard
* Scrum Ceremonies
* Benefits
* History

Agile Manifesto

Agile Manifesto was created in February 2001 by 17 software developers that met at the Snowbird resort in Utah.

“*We are uncovering better ways of developing software by doing it and helping others do it; through this work, we have come to value:*

* ***Individuals and interactions****over processes and tools*
* ***Working software****over comprehensive documentation*
* ***Customer collaboration****over contract negotiation*
* ***Responding to change****over following a plan*

*That is, while there is value in the items on the right, we value the items on the left more.*”

According with the manifesto text above, the accent is put on the items from the left. As we can see also in the next chapters, **to be agile**means to:

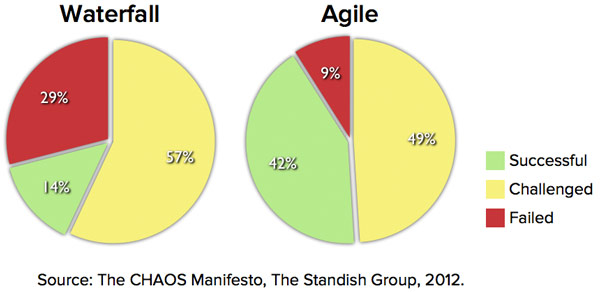
1. have a self organized team with better interactions and improved communication inside the team;
2. focus of development to be in creating a working software that delivers business values in each iteration;
3. have a real customers collaboration in order to offer them the possibility to provide continuous feedback during development;
4. be flexible and respond ASAP to scope changes.

Agile Methodologies

The main Agile methodologies are:

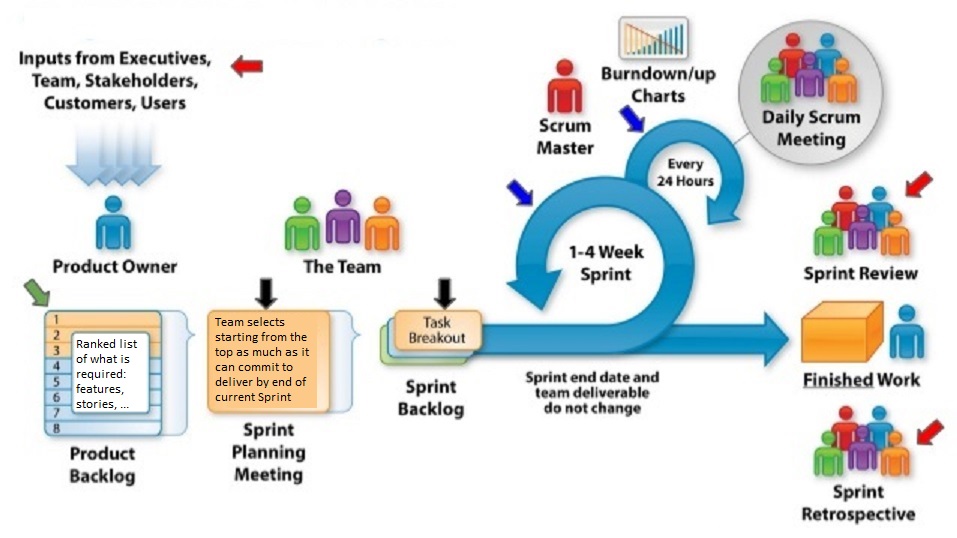
* Scrum
* Kanban
* XP (Extreme Programming)
* FDD (Feature Driven Development)
* AUP (Agile Unified Process)

By using Agile the projects became more successfully as you can see in the statistic below.



As you can see in the statistic above, there are big improvements in the case of projects that are develop by using an Agile methodology in comparison with the projects that are developed by using Waterfall classic methodology.

Scrum Overview



In the picture above I tried to present you an overview of the entire Scrum methodology into a single diagram. You can find details about Scrum Roles, Artifacts, Ceremonies (Meetings) and Sprints (Iterations) in the next chapters.

Sprint

The Scrum is an iterative methodology, and the Scrum name for iteration is "**Sprint**".

In the diagram below you can see a Scrum project development split into 24 iterations for a full year duration.

https://www.codeproject.com/KB/architecture/1201393/Sprints.png

**Sprint 0**– is the special iteration performed at the beginning of a project and forms the framework upon which each subsequent iteration builds. The duration of this iteration will vary depending on the complexity and existing knowledge of the desired capability. During this special Sprint perform enough architecture and system design in order to start development and get the first Sprint started. (Additional architecture and system engineering will be performed as necessary during subsequent iterations).

After Sprint 0 all other Sprints **must have the same length.** This length could be between 1 and 4 weeks. The recommended length is 2 weeks.

At the beginning of the Sprint, the team have a clear **commitment**on the **scope**, and the scope for current Sprint cannot be changed during the Sprint.

Short iterations enable Product Owners to get client feedback faster, developer estimates faster, and faster updates to plans, thereby shortening the whole feedback cycle. Any project should have at least three such occasions for stakeholders to provide their feedback before the release.

User Story

**User Story**(shortly **US**) - is a short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

**Acceptance criteria**-list of activities that must be fulfilled in order to consider US done;

**INVEST**helps to remember a widely accepted set of criteria, or checklist, to assess the quality of a user story.

* I (Independent)
* N (Negotiable)
* V (Valuable)
* E (Estimable)
* S (Small)
* T (Testable)

For a user story the next **template**should be used: ***As a <type of user>, I want <some goal> so that <some reason>.***

**Epic**–  a user story that covers large amounts of functionality. Because an epic is generally too large for an agile team to complete in one iteration, it is split into multiple smaller user stories before it is worked on.

**Examples**:  Epic1 *As a user, I can backup my entire hard drive, so that I can recover data after its loss.*

This epic can be broken in a set of user stories:

US1 *As a power user, I can specify files or folders to backup based on file size, created date and modified date, so that…*

…

US12 *As a user, I can indicate folders not to backup so that my backup drive isn't filled up with things I don't need saved, so that…*

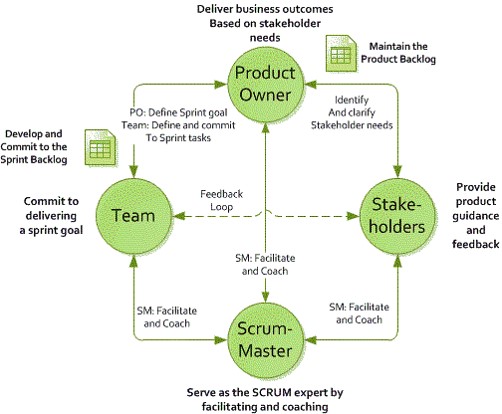
**Playing Poker**technique is used to estimate the User Stories in Story Points (SP) with the next possible values (Fibonacci numbers): **1, 2, 3, 5, 8, 13, 21**. By using cards, each user story is estimated by the entire team. The estimation for a user story is made in secret by the team, then the cards are revealed at the same time and the team members with higher and lower estimates must presents their reasons for the estimation. This estimation process for the user story is repeated until the entire team have the same estimation for the user story.

**Story Points**- are relative values, a story that estimated with 2 story points should be twice as much as items that are estimate with 1 story point. Story points represent the effort to develop a story and this includes everything that can affect the effort including amount, complexity and risks associated with this work.

**Hint**: each User Story must:

* be created by using the above template;
* respect INVEST rule;
* have a set of acceptance criteria;
* be estimated by the team in Story Points by using Playing Poker technique.

Scrum Roles



**Product Owner** - is one person responsible for Product Backlog (the scope) and for prioritizing the product features in each Sprint and over the course of the entire product development.

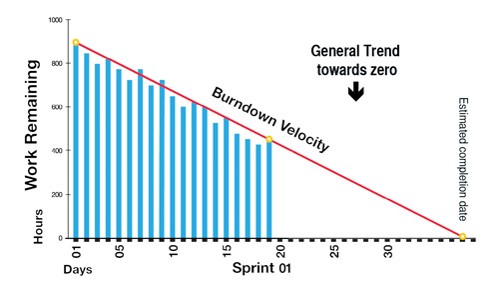
**Team**- self-organizing team that includes everyone involved with the design, development, test, and delivery of the finished product. The agile team’s goal is to deliver a quality product that meets the needs of the users.

**Scrum Master**- facilitates the Scrum process and creates an environment conducive to team self-organization; Can be seen as the coach for the entire team; Helps resolve impediments; Shields the team from external interference and distractions; Enforce time-boxes.

Scrum Artifacts

**Product Backlog**– is an always changing, dynamically prioritized list of requirements ordered by Business Value. Requirements are broken down into User Stories by the Product Owner. Definition of Done (DoD) at the Backlog level.

**Sprint Backlog**- contains all committed User Stories for the current Sprint broken down into Tasks by the Team. All items on the Sprint Backlog should be developed, tested, documented and integrated to fulfill the Team commitment.

**Burndown Chart**- shows the amount of work remaining per Sprint. It shows the correlation between work remaining at any point in time and the progress of the Team.

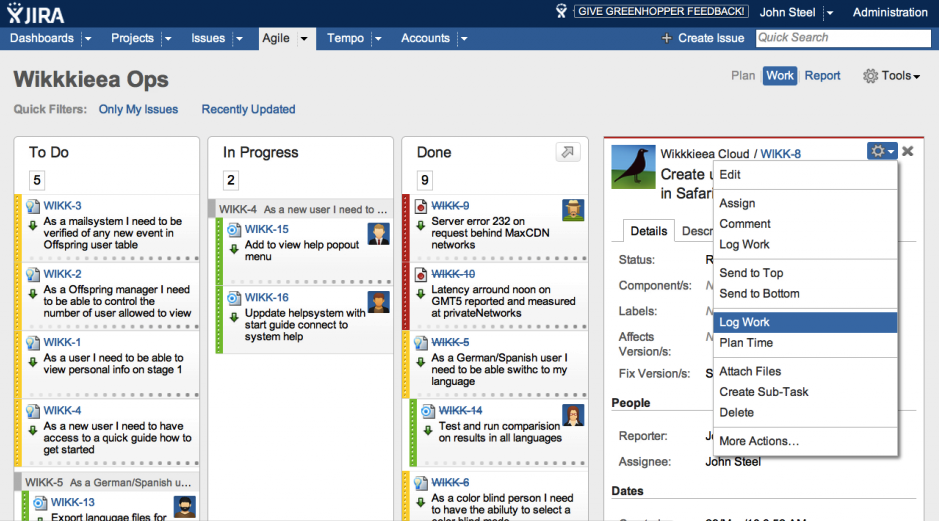
Burndwon Chart is normally generated by a SW tool, is used by the Scrum Master in order to track each Sprint and is useful for predicting when the work for the current Sprint will be completed.

**Sprint Velocity**- Number of Story Points completed per Sprint.

Dashboard

In Scurm the teams are using Dashboards in order to track the User Stories and Tasks for each Sprint and for each team member.

In the picture below you can see the Scrum Dashboard (from JIRA tool) used to filter all open issues for a user (team member).



You can see, in the above dashboard, the User Stories and their Tasks (for this user) grouped in 3 categories:

* To Do
* In Progress
* Done

Scrum Ceremonies

**Daily Scrum**– daily stand up meeting of the entire team including Product Owner no longer then **15 minutes.**Each participant should answer to the next  3 questions:

* *What did I do yesterday that helped the Team meet the Sprint Goal?*
* *What will I do today to help the Team meet the Sprint Goal?*
* *Do I see any impediment that prevents me or the Team from meeting the Sprint Goal?*

**Backlog Grooming**  - Backlog refinement meeting in order to ensure that the backlog remains populated with items that are relevant, detailed and estimated according with their priorities, and in keeping with current understanding the product and its objectives. The duration of this meeting should not be longer then **2 hours**. Could have a Backlog Grooming each week and must have a Backlog Grooming before to start next Sprint!

**Sprint Planning**-  is a time-boxed to a maximum of **8 hours for a 4 weeks Sprint**. The new sprint starts with this meeting. A set of user stories from the top of the Product Backlog are estimated into story points then decomposed into tasks and estimated in hours. Finally the Team, based on the Team Capacity, selects a set of these stories into the new Sprint Backlog (that may contains also unfinished user stories from the previous Sprint).

**Sprint Review/Demo** - At the end of each Sprint a Review/Demo meeting is held. During this meeting the Team shows what they accomplished during the Sprint. Typically this takes the form of a demo of the new features or underlying architecture. The duration of this meeting should be maximum **2 hours**.

**Sprint Retrospective** - meeting that helps the team to fine-tune the process. This is a time for each team member to reflect on what went right and areas for improvement. Clear action items should be defined. The duration of this meeting should be maximum **2 hours**.

Benefits

The main benefits of using Scrum in software development are:

* Increased product quality
* Enhanced transparency
* Increased flexibility
* Reduced risks
* Maximized productivity
* Improved communication
* Maximized cooperation

### Disadvantages of Agile

While the level of flexibility in Agile is usually a positive, it also comes with some trade-offs. It can be hard to establish a solid delivery date, documentation can be neglected, or the final product can be very different than originally intended.

Here are some of the disadvantages of Agile:

* **Planning can be less concrete**: It can sometimes be hard to pin down a solid delivery date. Because Agile is based on time-boxed delivery and project managers are often reprioritizing tasks, it’s possible that some items originally scheduled for delivery may not be complete in time. And, additional sprints may be added at any time in the project, adding to the overall timeline.
* **Team must be knowledgeable:** Agile teams are usually small, so team members must be highly skilled in a variety of areas. They also must understand and feel comfortable with the chosen Agile methodology.
* **Time commitment from developers:** Agile is most successful when the development team is completely dedicated to the project. Active involvement and collaboration is required throughout the Agile process, which is more time consuming than a traditional approach. It also means that the developers need to commit to the entire duration of the project.
* **Documentation can be neglected:**The Agile Manifesto prefers working software over comprehensive documentation, so some team members may feel like it’s less important to focus on documentation. While comprehensive documentation on its own does not lead to project success, Agile teams should find the right balance between documentation and discussion.
* **Final product can be very different:** The initial Agile project might not have a definitive plan, so the final product can look much different than what was initially intended. Because Agile is so flexible, new iterations may be added based on evolving customer feedback, which can lead to a very different final deliverable.

### Methodologies That Are Used to Implement Agile

Agile is a framework and there are a number of specific methods within the Agile movement. You can think of these as different flavors of Agile:

* **Extreme Programming (XP):** Also known as XP, Extreme Programming is a type of software development intended to improve quality and responsiveness to evolving customer requirements. The principles of XP include feedback, assuming simplicity, and embracing change.
* **Feature-driven development (FDD):** This iterative and incremental software development process blends industry best practices into one approach. There are five basic activities in FDD: develop overall model, build feature list, plan by feature, design by feature, and build by feature.
* **Adaptive system development (ASD):** Adaptive system development represents the idea that projects should always be in a state of continuous adaptation. ASD has a cycle of three repeating series: speculate, collaborate, and learn.
* **Dynamic Systems Development Method (DSDM):** This Agile project delivery framework is used for developing software and non-IT solutions. It addresses the common failures of IT projects, like going over budget, missing deadlines, and lack of user involvement. The eight principles of DSDM are: focus on the business need, deliver on time, collaborate, never compromise quality, build incrementally from firm foundations, develop iteratively, communicate continuously and clearly, and demonstrate control.
* **Lean Software Development (LSD):** Lean Software Development takes Lean manufacturing and Lean IT principles and applies them to software development. It can be characterized by seven principles: eliminate waste, amplify learning, decide as late as possible, deliver as fast as possible, empower the team, build integrity in, and see the whole.
* **Kanban:** Kanban, meaning “visual sign” or “card” in Japanese, is a visual framework to implement Agile. It promotes small, continuous changes to your current system. Its principles include: visualize the workflow, limit work in progress, manage and enhance the flow, make policies explicit, and continuously improve.
* **Crystal Clear:** Crystal Clear is part of the Crystal family of methodologies. It can be used with teams of six to eight developers and it focuses on the people, not processes or artifacts. Crystal Clear requires the following: frequent delivery of usable code to users, reflective improvement, and osmotic communication preferably by being co-located.
* **Scrum:** Scrum is one of the most popular ways to implement Agile. It is an iterative software model that follows a set of roles, responsibilities, and meetings that never change. Sprints, usually lasting one to two weeks, allow the team to deliver software on a regular basis.

Other Practices in Agile

There are many other practices and frameworks that are related to Agile. They include:

* **Agile Modeling (AM):**Agile modeling is used to model and document software systems and is a supplement to other Agile methodologies like Scrum, Extreme Programming (XP), and Rational Unified Process (RUP). AM is not a complete software process on its own. It can help improve models with code, but it doesn’t include programming activities.
* **Rational Unified Process (RUP):**Created by the Rational Software Corporation, a division of IBM, RUP is an iterative, adaptive framework for software development. According to Rational, RUP is like an online mentor that provides guidelines, templates, and examples for program development. The key aspects of RUP include a risk-driven process, use case focused development, and architecture-centric design.
* **Lean vs Agile:** Lean development focuses on eliminating and reducing waste (activities that don’t add any value). Lean development takes the principles from Lean manufacturing and applies them to software development. These principles are very similar to Agile, however Lean takes it one step further. In the development phase, you select, plan, develop, test, and deploy only one feature before you repeat the process for the next feature.
* **Test-Driven Development (TDD):** Test-driven development relies on repetitive, short development cycles. First, a developer writes an (initially failing) automated test case for a new feature and quickly adds a test with the minimum amount of code to pass that test. Then, he refactors the new code to acceptable standards.
* **Scaled Agile Framework (SAFe trademark logo):** The Scaled Agile Framework is a very structured method to help large businesses get started with adopting Agile. SAFe is based on Lean and Agile principles and tackles tough issues in big organizations, like architecture, integration, funding, and roles at scale. SAFe has three levels: team, program, and portfolio.
* **Rapid Application Development (RAD):**RAD’s approach to software development puts more emphasis on development than planning tasks. It follows an incremental model, where each component is developed in parallel. The phases in RAD are: business modeling, data modeling, process modeling, application generation, and testing and turnover.
* **Empirical Control Method:** With Agile software development, you can use an Empirical Control Method, which means that you make decisions based on the realities you observe in the actual project. The empirical model of process control has three parts: visibility, inspection, and adaption.