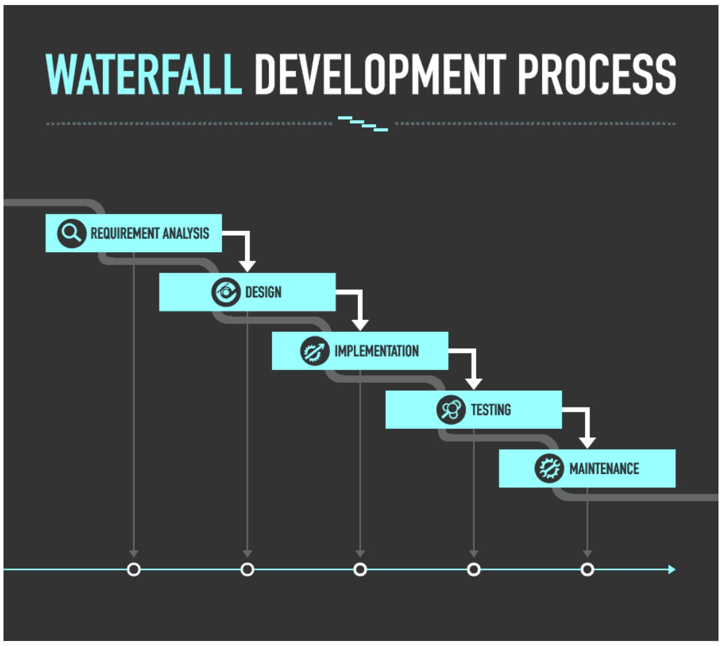
Agile and Scrum

History

- Throughout the history of the Software engineering there were several models of software development.

Waterfall model

- Created in the 1970's



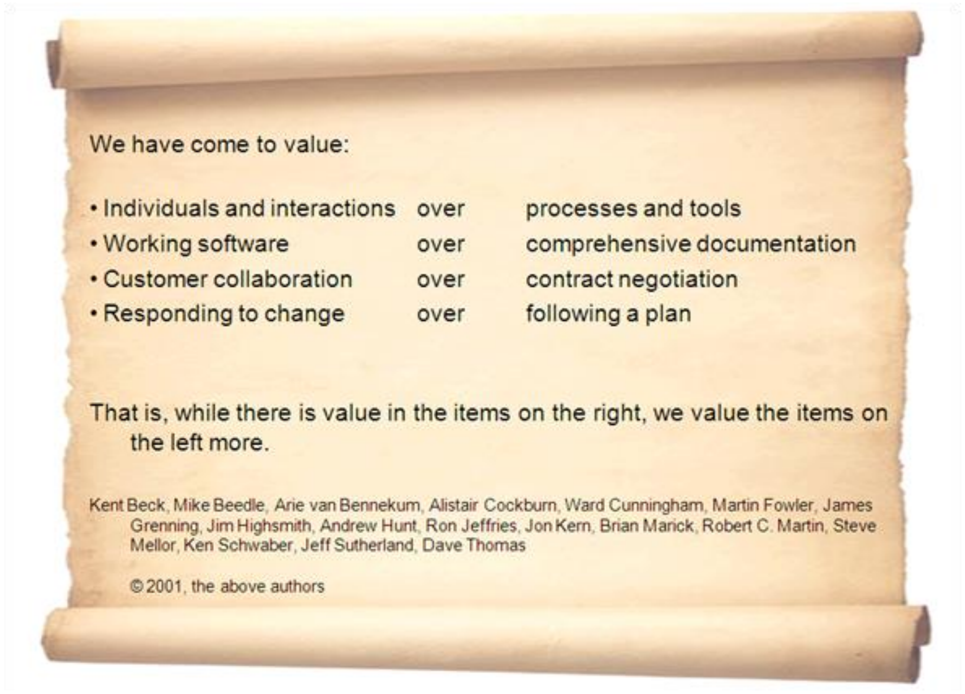
- Pros:
  - Many design errors are identified early
  - Measuring progress is easy
  - Accurate cost estimates can be provided
- Cons:
  - Change is difficult
  - Projects often run late
  - It assumes you know all requirements

Iterative development

- Created in the 1990's
- Requirements are implemented and tested separately and then presented to the costumer, he can object to some aspects and then change can be made quickly and cheaply.

Agile

- In 2002 the agile manifesto was created:



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.

Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, Dave Thomas

©2001, the above authors

Agile Principles

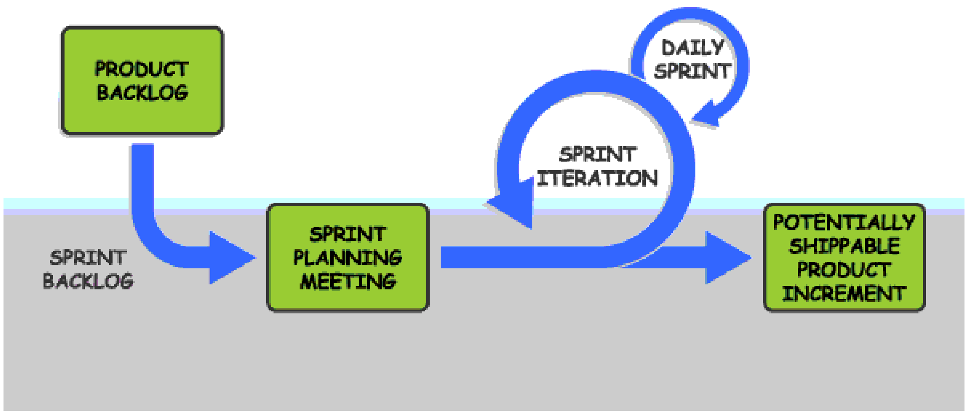
The Agile Manifesto also describes twelve principles of agile development:

1. Provide customer satisfaction through early and continuous software delivery
2. Accommodate changing requirements throughout the development process
3. Supply frequent delivery of working software
4. Collaborate between the business stakeholders and developers throughout the project
5. Support, trust, and motivate the people involved
6. Enable face-to-face interactions
7. Measure progress through working software
8. Use Agile processes to support a consistent development pace
9. Enhance agility through attention to technical detail and design
10. Keep things simple by developing just enough to get the job done for right now
11. Self-organize teams to encourage great architectures, requirements, and designs
12. Reflect regularly on how to become more effective

Agile frameworks

Scrum

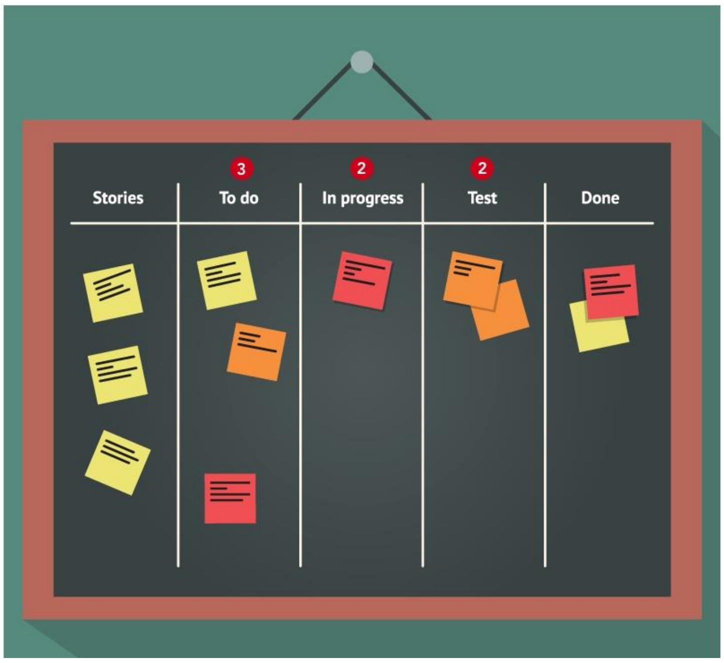
- A team works in short cycles (called *sprints*) that are typically 2-4 weeks long.
- A prioritized list of requirements called the *product backlog* is created.
- Before each sprint, a number of features are chosen from the product backlog to be part of the cycle. The team will choose a list of features they believe they can complete during that sprint.
- Each day, the team meets briefly in a *stand-up* meeting to discuss progress.
- At the end of the sprint, the completed work should be in a state that is ready for release. The team then reviews the sprint and reflects on what they have learned and what they can improve upon for the next cycle.



Kanban

Kanban is an Agile methodology that encourages **flow** and seeks to keep work items from being stuck, blocked, or delayed.

The idea is that the team works on fewer items at a time focusing on reducing the time spent on each stage of development. This way there is not a lot of time between when tasks or features start and finish.



Implementation guidlines:

- **Limit "works in progress."** The key to Kanban is to limit the number of items in development (tasks or features), not so that you do less but rather that you start and complete more items.
- **Visualize workflow.** Put all work items on a wall and use columns to denote their status. This allows the team and the whole office to see the progress.
- **Manage flow.** By analyzing the point at which items get stuck, blocked, or slowed down you can identify (and then remove) bottlenecks.
- **Improve collaboratively.** Continuous improvement and teamwork are vital concepts in Kanban

Extreme Programming (XP)

Extreme Programming or XP is an Agile framework that focuses a lot on the quality of practice and the habits of the software practitioner (i.e., the developers on the team). Its main guidelines are as follows:

- Developers will adhere to coding standards, all writing code the same way.
- Use test-driven development. This is a process where developers write the code for a test that a feature should pass (or validate) before proceeding. It is a key part of XP.
- Developers write code in pairs. Usually, one developer writes the test code and the other writes feature code.
- Work is done in short iterations (usually two weeks) and planning happens before each iteration.
- Just enough design and architecture are involved to build the features for the current iteration.
- Code is frequently checked against the master code base so errors can be instantly detected (this is called continuous integration of code with the code base).