Aalto University
School of Science

CSE-C3610
Software Engineering (5 cr)

Agile Software Development

Prof. Casper Lassenius

SoberIT
Casper Lassenius

Aalto University
School of Science

Agile Development

T-76.3601

Prof. Casper Lassenius

History and Principles of Agile Development

Aalto University
School of Science

Agile methods

- Dissatisfaction with the overheads involved in software design methods of the 1980s and 1990s led to the creation of agile methods. These methods:
  - Focus on the code rather than the design
  - Are based on an iterative approach to software development
  - Are intended to deliver working software quickly and evolve this quickly to meet changing requirements.
- The aim of agile methods is to reduce overheads in the software process (e.g. by limiting documentation) and to be able to respond quickly to changing requirements without excessive rework.

The Manifesto for Agile Software Development

"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."

Kent Beck et allii

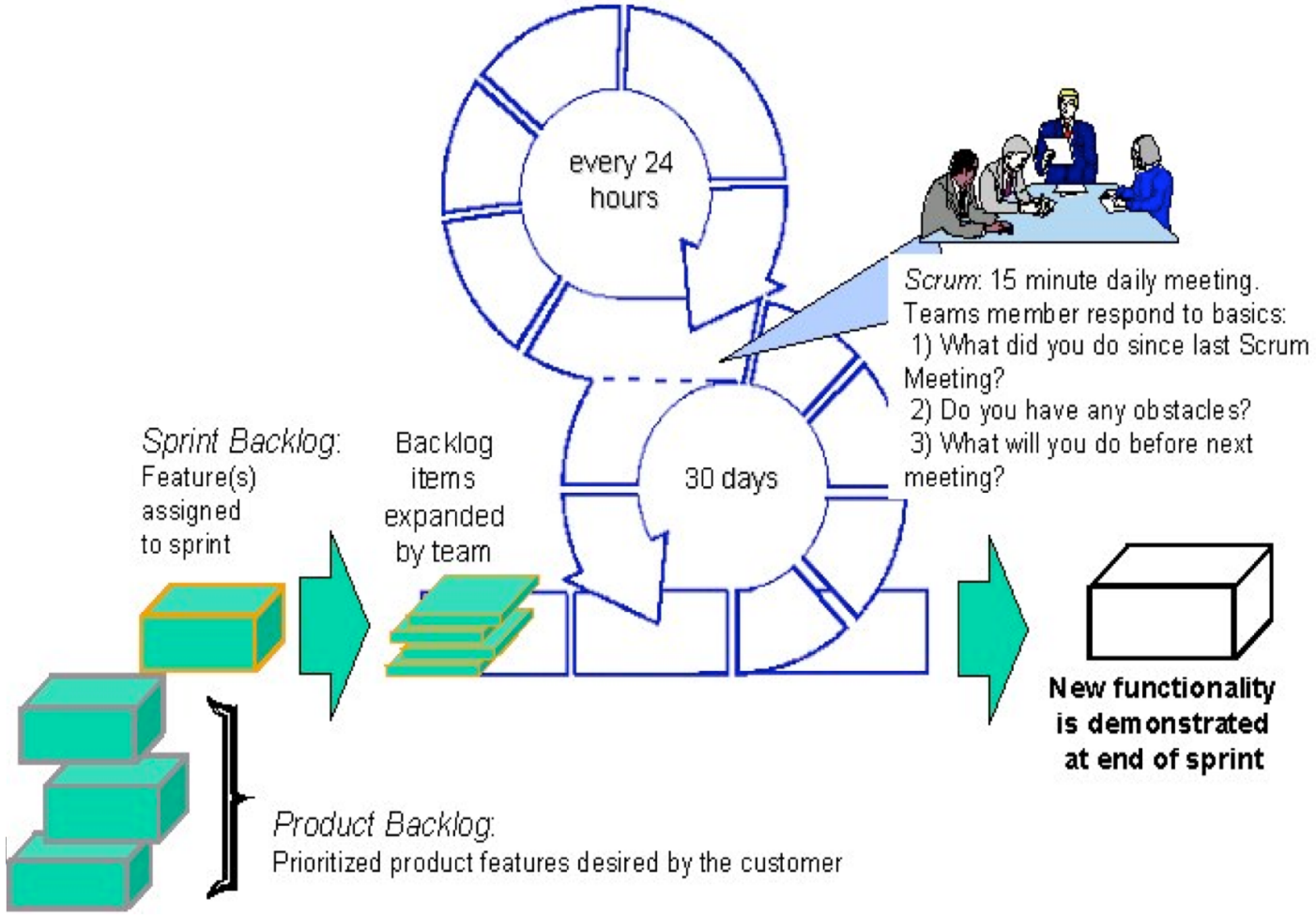
The principles of agile methods

Principle	Description
Customer involvement	Customers should be closely involved throughout the development process. Their role is provide and prioritize new system requirements and to evaluate the iterations of the system.
Incremental development	The software is developed in increments with the customer specifying the requirements to be included in each increment.
People not process	The skills of the development team should be recognized and exploited. Team members should be left to develop their own ways of working without prescriptive processes.
Embrace change	Expect the system requirements to change and so design the system to accommodate these changes.
Maintain simplicity	Focus on simplicity in both the software being developed and in the development process. Wherever possible, actively work to eliminate complexity from the system.

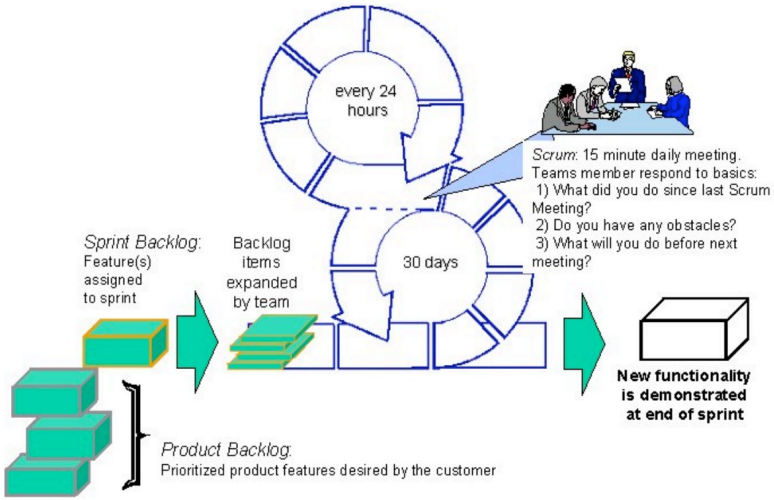
Scrum

Aalto University
School of Science

Scrum

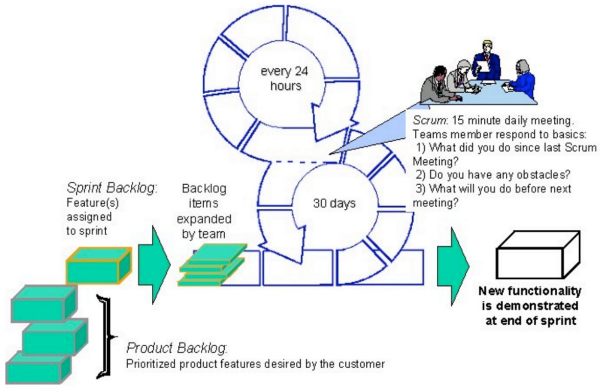

every 24 hours
30 days
Scrum: 15 minute daily meeting.
Teams member respond to basics:
1) What did you do since last Scrum Meeting?
2) Do you have any obstacles?
3) What will you do before next meeting?
Sprint Backlog:
Feature(s) assigned to sprint
Backlog items expanded by team
New functionality is demonstrated at end of sprint
Product Backlog:
Prioritized product features desired by the customer

The Sprint cycle



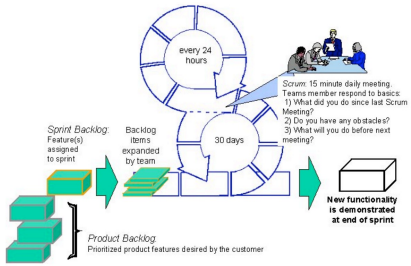
- Sprints are fixed length, normally 2–4 weeks.
- The starting point for planning is the product backlog, which is the list of work to be done on the project.
- The selection phase involves all of the project team who work with the customer to select the features and functionality to be developed during the sprint.

The Sprint cycle



- Once these are agreed, the team organize themselves to develop the software.
- The role of the Scrum master is to guard the process and protect the development team from external distractions.
- At the end of the sprint, the work done is reviewed and presented to stakeholders. The next sprint cycle then begins.

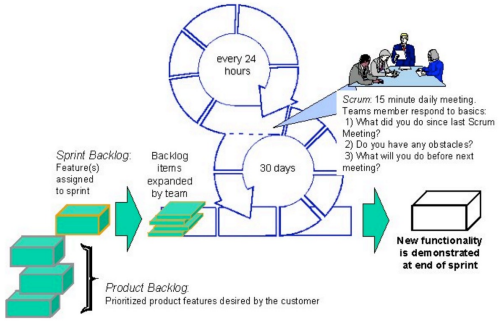
Teamwork in Scrum



- The 'Scrum master' is a facilitator who arranges daily meetings, tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with customers and management outside of the team.

- The whole team attends short daily (stand-up) meetings (Scrum meeting) where all team members share information:
  - 1) describe their progress since the last meeting,
  - 2) problems that have arisen and
  - 3) what is planned for the following day.
  - This means that everyone on the team knows what is going on and, if problems arise, can re-plan short-term work to cope with them.

Scrum benefits



- The product is broken down into a set of manageable and understandable chunks.
- Unstable requirements do not hold up progress.
- The whole team have visibility of everything and consequently team communication is improved.
- Customers see on-time delivery of increments and gain feedback on how the product works.
- Trust between customers and developers is established and a positive culture is created in which everyone expects the project to succeed.

Scrum problems

- How can product backlog when different size work items are in the same
  - Small task (individual bugs, simple features)
  - Very large tasks (complete renovation of the products user interface)
- Others, see problems with agile methods in general.

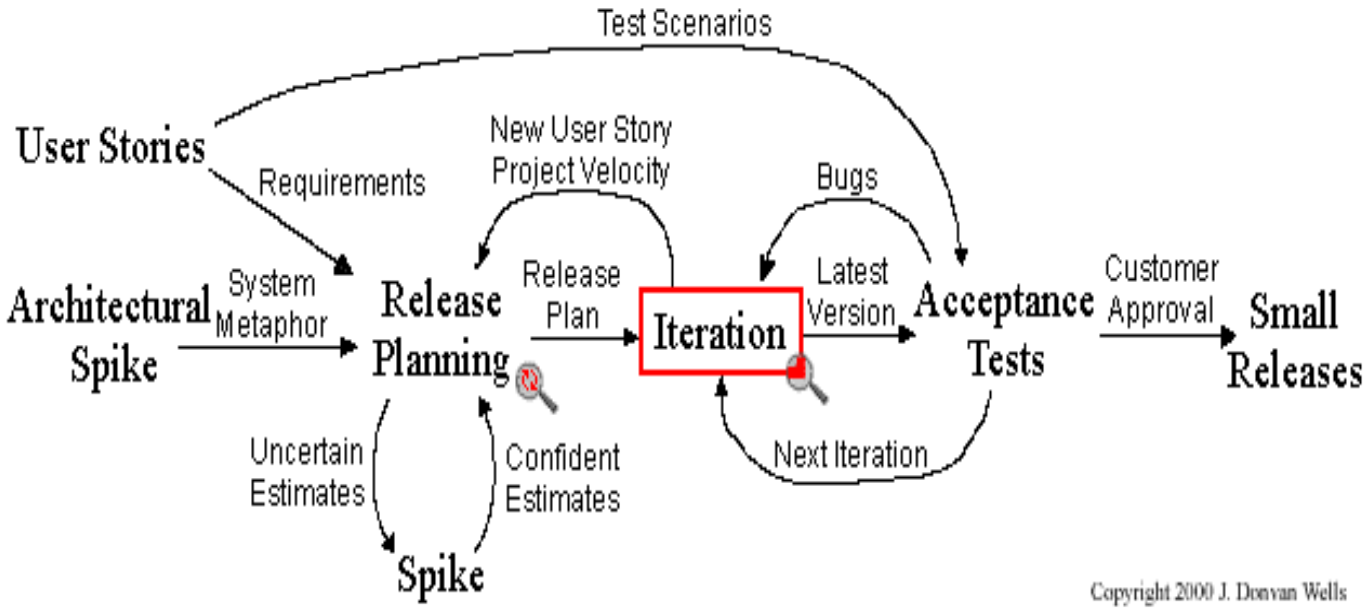
eXtreme Programming
XP

Aalto University
School of Science

Extreme programming

- Perhaps the best-known and ~~most widely used~~ agile method.
- One of few methodologies that actual takes a stand on software engineering “details”
  - Pair-programming, Test-driven development
- Extreme Programming (XP) takes an ‘extreme’ approach to iterative development.
  - New versions may be built several times per day;
  - Increments are delivered to customers every 2 weeks;
  - All tests must be run for every build and the build is only accepted if tests run successfully.

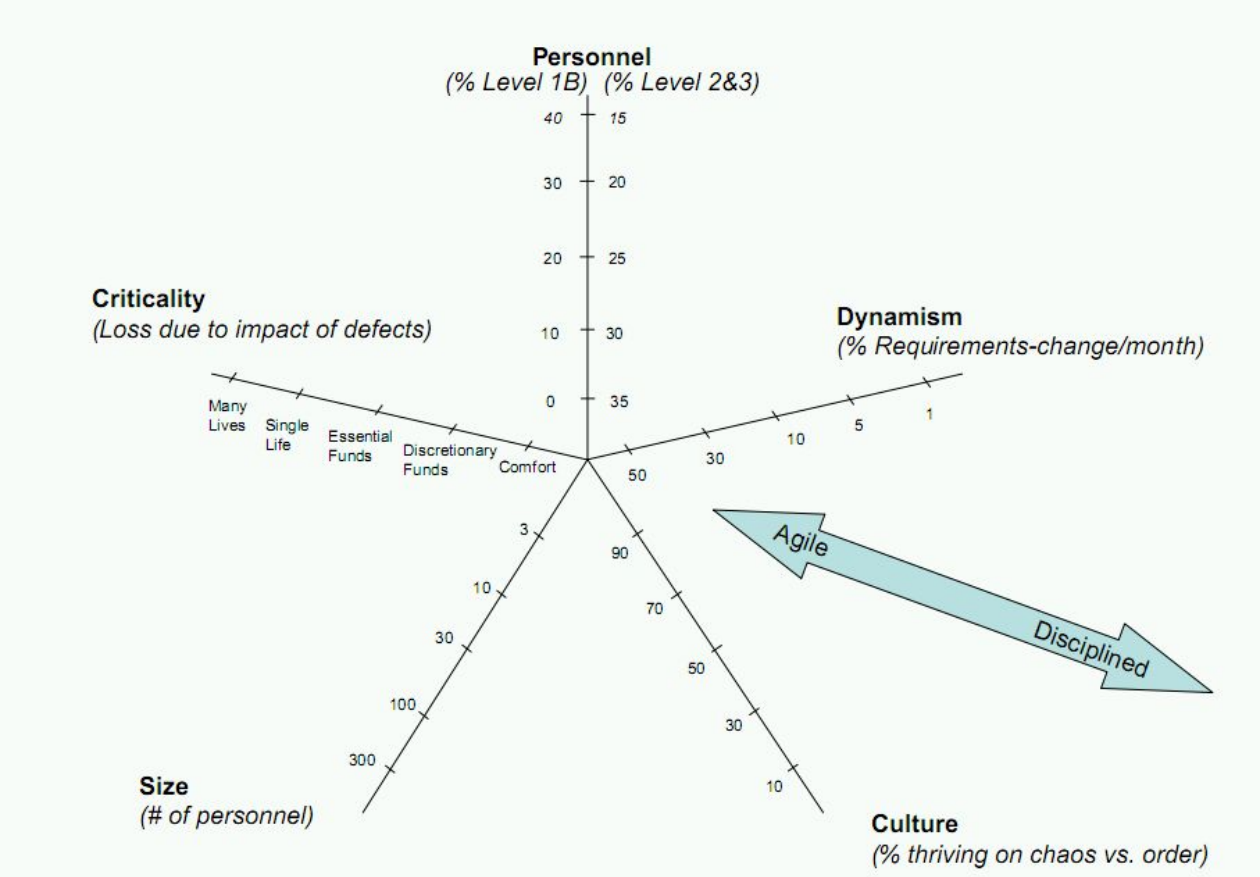
Extreme Programming Project



The Homeground of Agile Development

Aalto University
School of Science

Dimensions Affecting Process Model Selection



Boehm, B. and Turner, R. (2003), *IEEE Computer*, 36(6), pp 64-69)

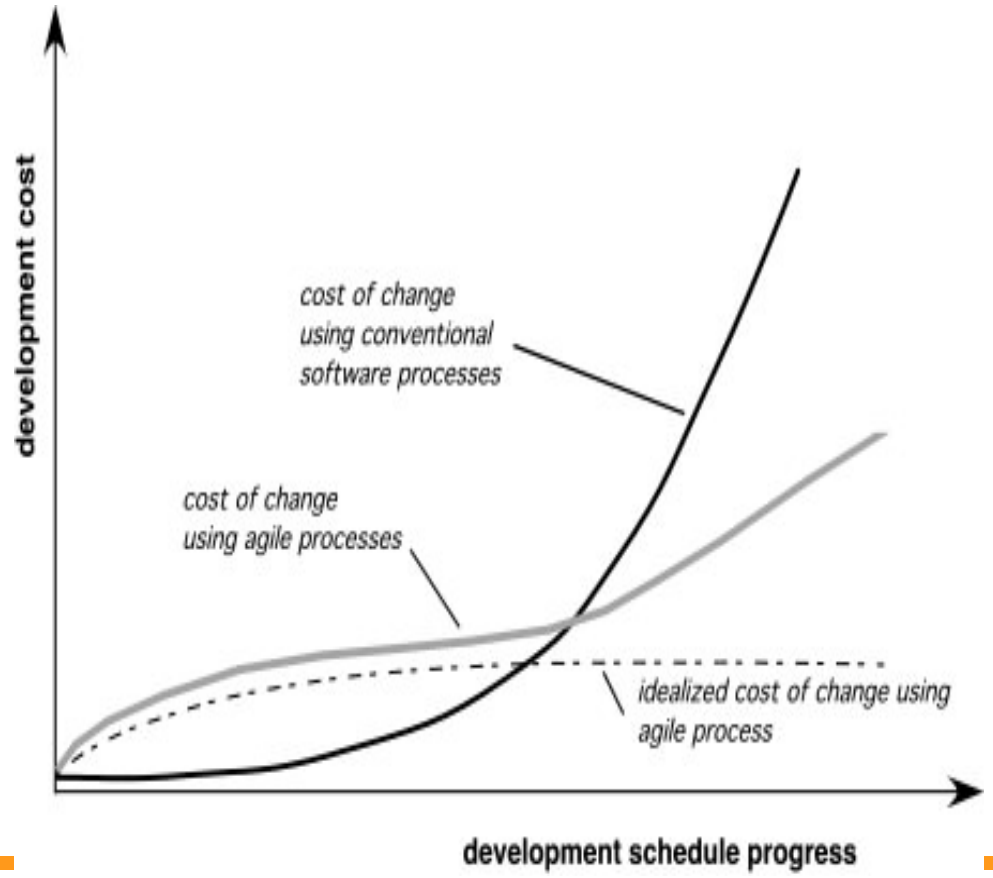
Agile method applicability

- Product development where a software company is developing a small or medium-sized product for sale.
- Custom system development, where there is a clear commitment from the customer to become involved in the development process and where there are not a lot of external rules and regulations that affect the software.
- Because of their focus on small, tightly-integrated teams, there are problems in scaling agile methods to large systems
  - Scaling Agile is a hot research topic

Problems with agile methods

- It can be difficult to keep the interest of customers who are involved in the process.
- Prioritising changes can be difficult where there are multiple stakeholders.
- Over-reliance on automated testing
- No up-front planning / architectural design -> unsuitable to complex / large systems
- Team members may be unsuited to the intense involvement that characterises agile methods.
- Contracts may be a problem as with other approaches to iterative development.

Agility and the Cost of Change



Time-boxing

- Fixing the iteration end date and not allowing it to change
  - 1-6 weeks is normal length for a timeboxed iteration
  - 3-6 month is usually too long and misses the point and value
  - Shorter steps have lower complexity and risk, better feedback, and higher productivity and success rates

- The scope is reduced, rather than slip the iteration end date
  - lower priority requests back on the wish list

- Should not be used to pressure developers to work long hours to meet the deadline -> If normal pace of work is insufficient, do less

Impossible to fix all 4 dimensions

- When plans do not work you have to one of the following
  - Time is extended
    - typical approach
  - Scope is reduced
    - time-boxing / agile approach
  - Quality is reduced
    - usually very bad idea, but still often practiced
  - Resources are added
    - Does not work very well in Software Engineering: “Adding more people to a project that is late will only increase the delay (Brooks, Mythical man-month)
    - Software engineering is snow shuffeling

Scaling up to large systems

- For large systems development, it is not possible to focus only on the code of the system. You need to do more up-front design and system documentation
- Cross-team communication mechanisms have to be designed and used. This should involve regular phone and video conferences between team members and frequent, short electronic meetings where teams update each other on progress.
- Continuous integration, where the system is built every time any developer checks in a change, is practically **difficult**. However, it is essential to maintain frequent system builds and regular releases of the system.

Scaling out to large companies

- Project managers who do not have experience of agile methods may be reluctant to accept the risk of a new approach.
- Large organizations often have quality procedures and standards that all projects are expected to follow and, because of their bureaucratic nature, these are likely to be incompatible with agile methods.
- Agile methods seem to work best when team members have a relatively high skill level. However, within large organizations, there are likely to be a wide range of skills and abilities.
- There may be cultural resistance to agile methods, especially in those organizations that have a long history of using conventional systems engineering processes.

Key points

- Agile methods are incremental development methods that focus on rapid development, frequent releases of the software, reducing process overheads and producing high-quality code. They involve the customer directly in the development process.
- The decision on whether to use an agile or a plan-driven approach to development should depend on the type of software being developed, the capabilities of the development team and the culture of the company developing the system.
- Extreme programming is a well-known agile method that integrates a range of good programming practices such as frequent releases of the software, continuous software improvement and customer participation in the development team.
- Scaling up to large systems and large organizations is difficult

Questions?