Course "Softwareprozesse"

Scrum, Kanban

Lutz Prechelt
Freie Universität Berlin, Institut für Informatik

- **Scrum**
  - Scrum Team: Developers, Scrum Master, Product Owner
  - Sprint: Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective
  - Product Backlog & Goal, Sprint Backlog & Goal, Increment & Definition of DONE

- **Kanban**
  - limit WiP, manage flow, visualize, explicit policies
  - feedback loops, improve via experiments
  - flow system with Kanban board and WiP limits
Learning objectives

• Understand how the few elements of Scrum can
  • drive a process forward
  • keep it from falling apart

• Understand how Kanban ideas can
  • maximize flow and so
  • reduce lead times
Scrum

  - first formulates Scrum ideas
    (case studies of photo copiers, car, PC, cameras)

- Ken Schwaber, Jeff Sutherland: "The Scrum Guide", version of Nov. 2020
  - the definitive version of Scrum (15 pages)
    - our reference, updated at irregular intervals
    - Lots of differences over time. Beware!

Other sources, e.g.:
- https://www.agilealliance.org
Scrum? What a strange word!

'Scrum' is a standard situation in Rugby

(Scrum's "Daily Scrum" resembles it)
What Scrum is and is not

- Scrum is an approach for managing a development process
  - Any knowledge work, not only software development
  - Scrum thus does not describe technical development activities

- Scrum's goal is facilitating the self-organization of the Scrum Team so that it can adapt to
  - the specifics of the project and
  - their changes over time

- Scrum is the most-used agile "method"
"In a nutshell, **Scrum** requires a **Scrum Master** to foster an environment where:

1. A **Product Owner** orders the work for a complex problem into a **Product Backlog**.

2. The **Scrum Team** turns a selection of the work into an **Increment** of value during a **Sprint**.

3. The **Scrum Team** and its stakeholders inspect the results and adjust for the next **Sprint**.

4. Repeat"
Scrum Guide: "Scrum Theory"

- "Scrum employs an iterative, incremental approach to optimize predictability and to control risk."
- "Scrum combines four formal events for inspection and adaptation within a containing event, the Sprint."
- "Scrum is founded on empiricism and lean thinking."
  - "[The] empirical Scrum pillars [are] transparency, inspection, and adaptation."
  - Transparency: "important decisions are based on the perceived state of [...] three formal artifacts"
    - "The Scrum Team and its stakeholders are open about the work and the challenges."
  - "Inspection enables adaptation."
  - "if the resulting product is unacceptable, [it] must be adjusted. [...] Adaptation becomes more difficult when the people involved are not empowered or self-managing."
Roles in a Scrum Team

- **Product Owner**
  - Represents all customers, manages the *Product Backlog*
  - Sets priorities, selects stories for a *Sprint*

- **Scrum Master**
  - Responsible for ensuring a smooth execution of the Scrum process (as teacher and coach, *not* as a manager)
    - This role targets Developers and Product Owner
  - Responsible for removing organizational obstacles ("impediments")

- **Developers**
  - The developers (typically 3-9), viewed as a self-organizing group of technical and process experts
    - The role is Developers (formerly *Development Team*), *not* Developer!

- **Scrum Team**
  - All of the above together
  - The *Scrum Team* as a whole is responsible for product delivery
  - Larger projects can use multiple Scrum Teams
Structure of the Scrum process:
Anatomy of an iteration ("Sprint")

(Scrum Master supports all activities)

- Product Owner
- Developers

Sprint Planning → (development) → Sprint Review → Sprint Retrospective

- Product Backlog
- Sprint Goal
- Sprint Backlog
- Increment
- Definition of "Done"

process flow: →
data flow: ←
support: →
participation: ←
Role Product owner

**Product Owner**

- Core responsibilities:
  - Developing and explicitly communicating the Product Goal
    (Sprint Goals are defined together with the Developers)
  - Needs to understand the application domain
  - Should have contact to end users and sponsors
  - Creating and clearly communicating Product Backlog items
    and supporting the Developers in their value production
  - Has the *sole* authority over the Product Backlog
    - "*The Product Owner is one person, not a committee.*"
    - Orders the Product Backlog items
    - Ensures "*that the Product Backlog is transparent, visible and understood*"
Role Scrum Master

- **Scrum Master**
  - Can be a technical or (quite often) a non-technical person:
    - "Coaches [...] in self-management and cross-functionality"
    - Needs to understand software development, but does not need to be able to perform it.
    - Responsible "that all Scrum events take place and are positive, productive, and kept within the timebox."
  - *Servant leader* to the Developers, the Product Owner, the Scrum Team, and the organization
  - The "removing obstacles" role is particularly important if the surrounding organization is not yet agile:
    - Protect the Development Team from management interference
    - Management should target the Product Owner only

Lutz Prechelt, prechelt@inf.fu-berlin.de
Role Developers

• Developers
  • planning: create the Sprint Backlog
    • by turning conceptual Product Backlog items into technical tasks
  • all required skills must be present in the team
  • overall: "hold each other accountable as professionals"
    • everyone participates in self-organization
    • everyone adheres to the Definition of Done
Structure of the Scrum process:
Anatomy of an iteration ("Sprint")

Lutz Prechelt, prechelt@inf.fu-berlin.de
Artifacts and their commitments

• Commitments are the reference for inspection

• **Product Backlog**:  
  • list of user requirements  
    • often as "stories"  
  • commitment: **Product Goal**  
    • long-term vision

• **Increment**:  
  • "*a concrete stepping stone toward the Product Goal*"  
    • 1 per Product Backlog item  
    • perhaps several in 1 Sprint  
    • might be released  
  • commitment: **Definition of Done**  
    • quality gate

• **Sprint Backlog**:  
  • initially contains stories  
  • which get refined into tasks during Sprint Planning  
    • by Product Owner and Developers  
  • and further refined into tasks and subtasks during development  
    • by Developers  
    • during and outside the Daily Scrum  
  • commitment: **Sprint Goal**  
    • set by Product Owner and Developers together  
    • fixed for the Sprint  
    • defines its "topic" only
Definition of "Done"

- The "Definition of Done" defines when a feature or story is considered complete.
- It describes acceptable levels of e.g.(!)
  - automated tests, code review
  - technical debt
  - known defects and user-visible weaknesses
  - acceptance testing, documentation
  - deployment
  - and anything else relevant for this particular team

- Maintaining and strictly obeying a Definition of Done is Scrum's key element of product quality assurance

- [https://www.agilealliance.org/glossary/definition-of-done](https://www.agilealliance.org/glossary/definition-of-done)
Structure of the Scrum process:
Anatomy of an iteration ("Sprint")

(Scrum Master supports all activities)

Product Owner

Developers

Sprint Planning

Sprint Backlog

Product Owner

Sprint Goal

Sprint Backlog

Sprint Backlog

Sprint

Goal

Sprint

Backlog

Sprint

Planning

Daily

Scrum

Sprint

Review

Sprint

Retrospective

Definition of "Done"

Increment

(support)

(activity)

(artifact)

process flow:  

data flow:  

participation:  

Lutz Prechelt, prechelt@inf.fu-berlin.de
Activity: Sprint

• "Sprints are the heartbeat of Scrum, where ideas are turned into value."
• "During the Sprint:
  • No changes are made that would endanger the Sprint Goal
  • Quality does not decrease
  • The Product Backlog is refined as needed
  • Scope may be clarified and renegotiated with the Product Owner as more is learned"

• When explaining Scrum to classical-view inhabitants, use the second form, not the first
Activity: **Sprint Planning**

- **Goals:**
  1. "Why is this Sprint valuable?"
     - Define Sprint Goal
  2. "What can be **Done** this Sprint?"
     - Selecting Product Backlog items for the next sprint
     - not too few, not too many
     - involves estimation
     - Obey the Sprint Goal!
  3. "How will the chosen work get done?"
     - Plan technical tasks: Decompose items into pieces of 1 day or less

- **Procedure**
  - as everywhere, Scrum is procedure-agnostic
  - often done like the XP Planning Game,
  - but teams have many variations and extensions

- [https://www.agilealliance.org/glossary/sprint-planning](https://www.agilealliance.org/glossary/sprint-planning)
Activity: Daily Scrum

What gave Scrum its name

• A stand-up meeting
  • to keep it short (15 min.):
    • same time every day
    • everybody must attend

• Intended as catalyst for self-organization
  • coordination of development:
    • day plan, problems, help needs, collaboration needs
    • DONE progress
    • make joint decisions

• Problem: Purpose often not understood by teams:
  • They do status reports, not coordination

https://www.agilealliance.org/glossary/daily-meeting

a.k.a "The daily standup"
Product discussion activity: Sprint Review

Sprint Review:

- Presents, demos, explains, and inspects the Increment(s)
- Reviews Product Backlog
  - may modify it
- Reviews strategy
  - timeline, budget, potential capabilities, and marketplace for anticipated releases
Process discussion activity: Sprint Retrospective ("Retro")

- **Topics:**
  - What went well? How can we optimize it?
  - What went not well? How should we improve it?
  - What did not work at all? Why? What do we do about that?
  - What is the status of difficult changes decided in earlier retros?

- **Many teams suffer from the "Elephant in the room" effect:**
  - Initial retros are helpful and produce progress,
  - but after easy problems are solved, nobody talks about the most important problem,
    - it is just too hard.

- **Result:** Little process improvement
  - and self-organization at a tactical level only

- **Retrospectives practices** may help
  - complicated enough for a whole course of its own
Activities: How long? (Time-boxing)

- All activities are *time-boxed*
  - which means: They can be shorter, but not longer
  - end time is predictable, content/result is not

- Sprint length:
  - Scrum guide: "one month or less"
    - (MLS: month-long Sprint)
  - most teams use 1 week or 2 weeks
  - short Sprints require shortening planning, review, and retrospective
  - but have *many advantages*

- Sprint Planning:
  - time-boxed to 8 hrs / MLS

- Sprint Review:
  - time-boxed to 4 hrs / MLS

- Sprint Retrospective:
  - time-boxed to 3 hrs / MLS

- Daily Scrum:
  - time-boxed to 15 minutes

- These are maximums
  - Scrum Team should self-organize to find proper lengths

Lutz Prechelt, prechelt@inf.fu-berlin.de
Scrum technical practices

None

- In practice, different sets of technical practices are used by Scrum teams
  - often too few
Kanban

- Based on ideas from the *Toyota Production System*
  - and later *Lean Production*
- Plenty of chic terminology & jargon
  - for any type of knowledge work, not only SW dev.
  - ➔ only mgmt. practices (just like Scrum)
- Combinable with Scrum
  - **Scrumban**

- **Key ideas:**
  - Start with what you are already doing and pursue evolutionary change
  - Manage the value-creation flow; allow self-organization

- **Our source:**
  *D. Anderson, A. Carmichael: "Essential Kanban Condensed", 2016*
Some of the chic jargon: Classification, values, "agendas"

- Kanban is
  - a method for delivering and improving knowledge work via a **flow system**;
  - a catalyst for rapid and focused change;
  - based on making intangible aspects visible

- 9 values:
  - **transparency**; balance; collaboration; **flow**; customer focus; leadership; understanding; agreement; respect
  - (Scrum has 5: Commitment, Focus, Openness, Respect, and Courage)

- 3 agendas (purpose of change):
  - inward: sustainability
  - outward: service orientation
  - futureward: survivability
6 Kanban principles

• Change management:
  1. Start with what you do now
  2. improve through evolutionary change
  3. encourage leadership at every level

• Service delivery:
  4. focus on customer needs
  5. manage work (not people); let people self-organize around it
  6. evolve policies
The flow system

- The heart of Kanban is "managing flow"
  - flow: How a valuable function progresses from idea to delivered SW
  - lead time: How long this takes
- Low lead times mean high agility
  - can react to a customer need fast
- Lead times get higher when too many items are being worked on at any time
  - also more half-products, hence more waste

Flow system ideas:
- Partition flow into N successive steps
- Limit the capacity for each step appropriately
  - "work in progress" (WiP) limit
- Visualize current status on a "Kanban board"
- As a result...
  - ...work is pulled by consumers, not pushed by producers
  - ...blockages are easily detected (and removed)
  - so work flows smoothly
A team must find suitable steps, sub-states, and WIP limits.
Work-in-progress limitation in action

Lutz Prechelt, prechelt@inf.fu-berlin.de
Full strip at https://blog.crisp.se/2009/06/26/henrikkniberg/1246053060000
Visualize the workflow:
A real Kanban board

WiP limits??

Lutz Prechelt, prechelt@inf.fu-berlin.de
6 Kanban practices

1. **limit WiP**
2. **manage flow:**
   - goals: delivery rates high and smooth, lead time minimal
   - methods: identify bottlenecks; apply cost-of-delay archetypes
3. **visualize:**
   - e.g. WiP limits and step policies on kanban board
   - e.g. history by data plots
4. **make policies explicit:**
   - sparse, simple, visible, well-defined, always applied, and (importantly) readily changeable;
   - topics: WiP limits, capacity allocation, definition of Done, ...
5. **implement feedback loops:**
   - e.g. by cadences (cyclical meetings)
6. **improve collaboratively, evolve by experiments:**
   - "it can be useful to employ models and the scientific method"

Lutz Prechelt, prechelt@inf.fu-berlin.de
"Cadences": meetings that establish feedback loops

Other aspects are beyond our consideration
## Kanban benefits found in empirical research

<table>
<thead>
<tr>
<th>#</th>
<th>Reported benefit</th>
<th>Primary study</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Improve visibility and transparency</td>
<td>P1, P2, P3, P4, P5, P6, P7, P11, P13, P14, P15, P17, P19, P20, P22, P23</td>
</tr>
<tr>
<td>2</td>
<td>Better control of project activities and tasks</td>
<td>P1, P2, P5, P9, P10, P11, P13, P15, P19, P20, P22, P23</td>
</tr>
<tr>
<td>3</td>
<td>Identify impediments to flow</td>
<td>P1, P2, P3, P5, P9, P15, P17, P20, P22, P23</td>
</tr>
<tr>
<td>4</td>
<td>Improve workflow</td>
<td>P2, P4, P6, P11, P16, P19, P20</td>
</tr>
<tr>
<td>5</td>
<td>Faster time-to-market</td>
<td>P6, P7, P10, P16, P23</td>
</tr>
<tr>
<td>6</td>
<td>Improve prioritisation of products and tasks</td>
<td>P1, P3, P15, P17</td>
</tr>
<tr>
<td>7</td>
<td>Decrease defects and bugs</td>
<td>P2, P7, P14, P21</td>
</tr>
<tr>
<td>8</td>
<td>Improve quality</td>
<td>P6, P7, P16, P17</td>
</tr>
<tr>
<td>9</td>
<td>A lightweight intuitive method</td>
<td>P14, P15, P16, P17</td>
</tr>
<tr>
<td>10</td>
<td>Improve communication and collaboration</td>
<td>P1, P4, P6, P7, P9, P14, P17</td>
</tr>
<tr>
<td>11</td>
<td>Improve team motivation</td>
<td>P4, P6, P11, P16, P17, P19</td>
</tr>
<tr>
<td>12</td>
<td>Team building and cohesion</td>
<td>P5, P7, P17, P20, P23</td>
</tr>
<tr>
<td>13</td>
<td>Increase customer satisfaction</td>
<td>P6, P7, P14, P15, P17, P20</td>
</tr>
<tr>
<td>14</td>
<td>Promoting a culture of continuous learning</td>
<td>P7, P10, P16, P20</td>
</tr>
<tr>
<td>15</td>
<td>Strategic alignment</td>
<td>P3, P5, P7</td>
</tr>
</tbody>
</table>

But compared to what? Beware of varying frames of reference!

M.O. Ahmad, D. Dennehy, K. Conboy, M. Oivo: [Kanban in software engineering: A systematic mapping study](https://doi.org/10.1016/j.jss.2018.12.031), Journal of Systems and Software 2018
Summary

• Scrum is an iterative process with
  • repeated requirements engineering (→Product Backlog)
  • some planning (Sprint Planning, Sprint Backlog, Daily Scrum)
  • efficient coordination (small team, Daily Scrum)
  • strategic product planning (Sprint Review)
  • continuous process improvement (Sprint Retrospective)
  • fine-grained quality assurance (Definition of DONE)

• Scrum emphasizes self-organization

• Kanban is a pull-based process focusing on
  • managing (WIP limits) and visualizing (Kanban board) flow
  • reducing lead times

• Kanban emphasizes continuous improvement
  • (Scrumban is an intermediate step from Scrum to Kanban)
Thank you!

further slides: short information on several other methods
Other method-oids

Agile development in the large

• Jutta Eckstein: "Agile Softwareentwicklung im Großen: Ein Eintauchen in die Untiefen erfolgreicher Projekte", dpunkt Verlag 2004
  • "Agile Software Development in the Large: Diving into the Deep", Dorset House B&T 2004

• http://www.jeckstein.de/
• http://www.agilebuch.de/

• How to scale agile devmt. to 30-200 people:
  • Using explicit "communication teams"
  • Coping with virtual and distributed teams
  • Handling the surrounding organization
    • see next slide

Jutta Eckstein
Agile development in the large (2)

- Handling the surrounding organization:
  - Talk early to people unfamiliar with Agile Development, such as
    - project planning and control departments,
    - the Method Police (process quality assurance group),
    - the Tool Support group
    - if relevant: Human Resources, Legal, Marketing
  - Integrate the QA department (if any) into the project
  - Integrate the Operations department into the project
  - Larger organizations tend to have higher fractions of below-average developers
    - To compensate for that, work towards a Learning Organization
  - Make learning materials part of the project deliverables
    - always to be kept consistent, part of acceptance testing
  - Handle insourcing, outsourcing, part-time employees
- The book ends with a case-story of a complex project
  - Perhaps the most useful part of the book!
Rational Unified Process: Agile variants

Agile variants of **Rational Unified Process (RUP)**:
- include technical and management practices

- **Project-specific variants**
  - formed by leaving out many RUP process elements and executing the rest with an agile mindset

- **dX**
  - RUP in XP mode: A minimal version of RUP resembling XP

- **Agile modeling**
  - Not a full process, just an approach to modeling
  - Based on 11 practices in four categories: Iterative and Incremental Modeling, Teamwork, Simplicity, Validation

- ...
The Pragmatic Programmer

- Andrew Hunt, David Thomas: "The Pragmatic Programmer: From Journeyman to Master", Addison-Wesley 1999
- 70 "tips" of advice and practices for individuals, not teams
- Management advice has to do a lot with discipline, attitude, and practical responsibility

Andy Hunt  
Dave Thomas
The Pragmatic Programmer (2)

• Principles:
  • Take responsibility for what you do.
    • Think in terms of solutions, not of excuses.
  • Don't just accept bad design or coding – improve them
  • Actively introduce process changes where necessary
  • Create software that delights your customer – and then stop
  • Automate
  • Broaden your knowledge. Learn. Improve yourself.
  • Improve your self and your communication skills
Scaling Scrum

- Ken Schwaber claims he has coached a project using Scrum that took 2.5 years and had 3500 participants overall.

- The technique to do this is the "Scrum of Scrums":
  - One participant of each daily Scrum is sent of the daily Scrum-of-Scrums on a second project-level.
  - This scales Scrum from 10 up to 100 participants. A third level could scale up to 1000.
  - It is difficult to get this to work.
    - see PasLasHei12
Still other methods

For large projects:

- **SAFe**: Scaled Agile Framework
  - orchestrates many teams
  - somewhat popular
- **DSDM**: Dynamic Systems Development Method
  - strict time-boxing
- **FDD**: Feature-driven development
  - modeling, parallel feature teams, individual code ownership, ...

For small projects:

- **Crystal Clear**
  - minimal framework for easy introduction and maximum freedom for the team
Crystal Clear
Goals and Practices

http://alistair.cockburn.us/index.php/Crystal_Clear_distilled

• "Crystal Clear is a highly optimized way to use a small, colocated team,
  • prioritizing for safety in delivering a satisfactory outcome,
  • efficiency in development, and
  • habitability of the working conventions."

• Brief description of Crystal Clear:
  • "The lead designer and two to seven other developers
  • ... in a large room or adjacent rooms,
  • ... using information radiators such as whiteboards or flip charts,
  • ... having easy access to expert users,
  • ... distractions kept away,
  • ... deliver running, tested, usable code to the users
  • ... every month or two (quarterly at worst),
  • ... reflecting and adjusting their working conventions periodically"
"The people set in place the safety properties below using the techniques they feel appropriate."

- The first three properties are required in Crystal Clear;
- The next four get the team further into the safety zone.

1. Frequent Delivery
2. Reflective Improvement
3. Osmotic Communication
4. Personal Safety
5. Focus
6. Easy Access to Expert Users
7. A Technical Environment with Automated Tests, Configuration Management, and Frequent Integration"
Crystal process improvement technique: Reflection workshop

- Hang a flipchart
- Fill in the chart
  - 30 minutes
- Hang the chart in a public, visible, frequently seen place!

- Try the ideas
- Repeat each month or after each iteration

<table>
<thead>
<tr>
<th>Keep these</th>
<th>Try these</th>
</tr>
</thead>
<tbody>
<tr>
<td>test lock-down</td>
<td>pair testing</td>
</tr>
<tr>
<td>quiet time</td>
<td>fines for interruptions</td>
</tr>
<tr>
<td>daily meetings</td>
<td>programmers help testers</td>
</tr>
</tbody>
</table>

Problems
- too many interruptions
- shipping buggy code

(Headings are part of the chart. Entries are **examples** only.)