



Scrum

Arian Treffer
arian.treffer@hpi.de

Software Engineering II
WS 2015/16

Prof. Plattner, Dr. Uflacker
Enterprise Platform and Integration Concepts group

Scrum



- 1. The Case for Agile**
2. The Scrum Process
3. Scaling Scrum

Why Traditional Projects Fail



- People are afraid of making changes
- Unofficial changes are carried out
- Documents get out of sync
- ...

How Traditional Projects Fail



- Delivering late
- Delivering over budget
- Delivering the wrong thing
- Unstable in production
- Costly to maintain

Why Traditional Projects Fail



- Smart people trying to do good work
- Stakeholders are well intended

Process in traditional projects



- Much effort for
 - Documents for formalized hand-offs
 - Templates
 - Review committees

Why Traditional Projects Fail



“The later we find a defect, the more expensive it is to fix it!”

Does front-loading a software development process make sense?

Reality shows:

- Project plans are wonderful
- Adjustments & assumptions are made during analysis, design, code
- Replanning takes place
- Example: Testing phase at the end
 - Tester raises a defect
 - Programmer claims he followed the specification
 - Architect blames business analyst etc.
 - Exponential cost

A Self-Fulfilling Prophecy



- We conduct the front-loaded process to minimize exponential costs of change
 - Project plan
 - Requirements specification
 - High-level design documents
 - Low-level design documents
- This process causes the exponential costs of change!
 - ➔ A self-fulfilling prophecy

*This makes sense for a bridge, ship, or a building
but software (and Lego) are easy to change!*

The Agile Manifesto



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. <http://agilemanifesto.org/>

How Agile Methods Address Project Risks



No longer late or over budget

- Tiny iterations
- Easy to calculate budget
- High-priority requirements first

No longer delivering the wrong thing

- Strong stakeholder communication
- Short feedback cycles

How Agile Methods Address Project Risks



No longer unstable in production

- Delivering each iteration
- High degree of automation

No longer costly to maintain

- Maintenance mode starting with Sprint 2
- Maintenance of multiple versions during development

The Cost of Going Agile



Outcome-based planning

- No complete detailed project plan

Streaming requirements

- A new requirements process

Evolving design

- No complete upfront design → flexible

Changing existing code

- Need for refactoring

The Cost of Going Agile



Frequent code integration

- Continuous integration

Continual regression testing

- Add nth feature; test n-1 features

Frequent production releases

- Organizational challenges

Co-located team

- Keep momentum

Discuss!



Pros and Cons

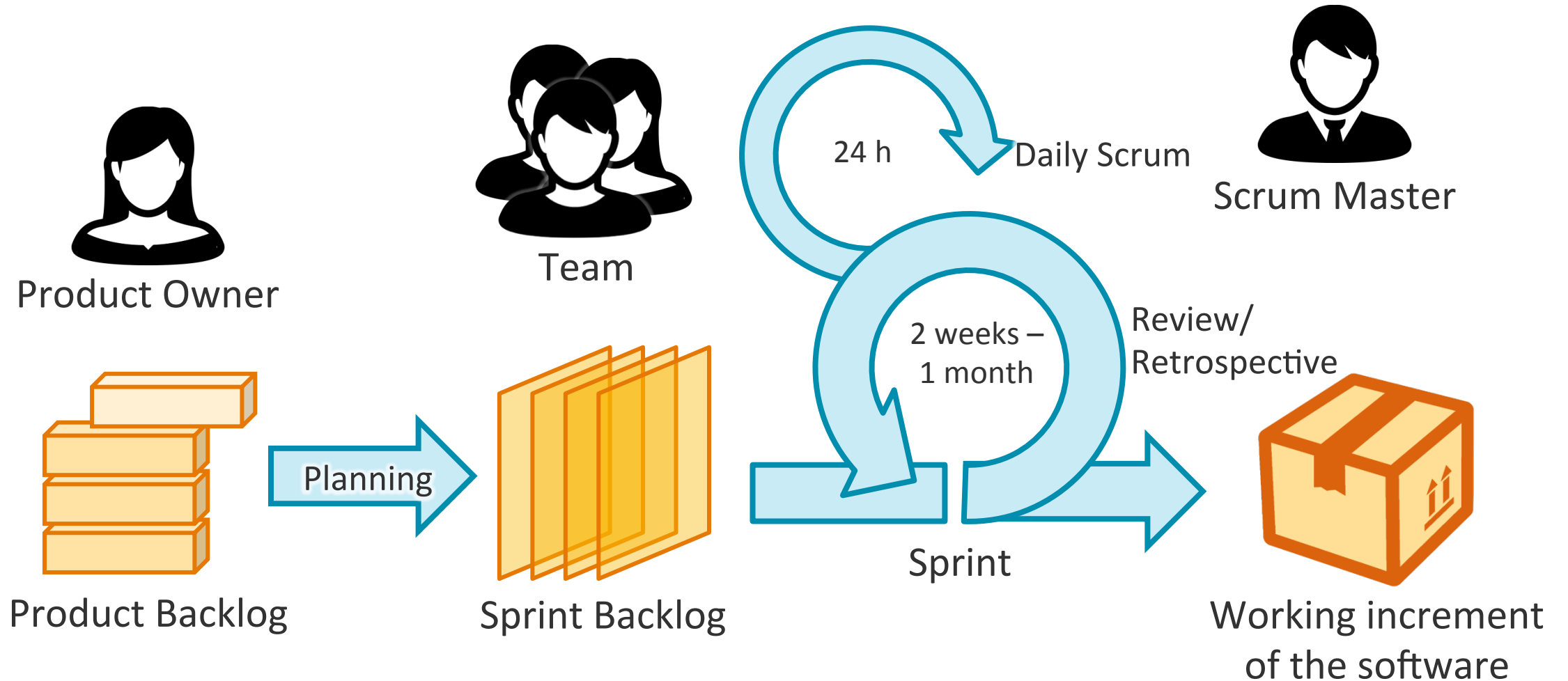
- Short planning horizon
- No up-front design
- Stories instead of requirement documents
- Extreme ideology

Scrum

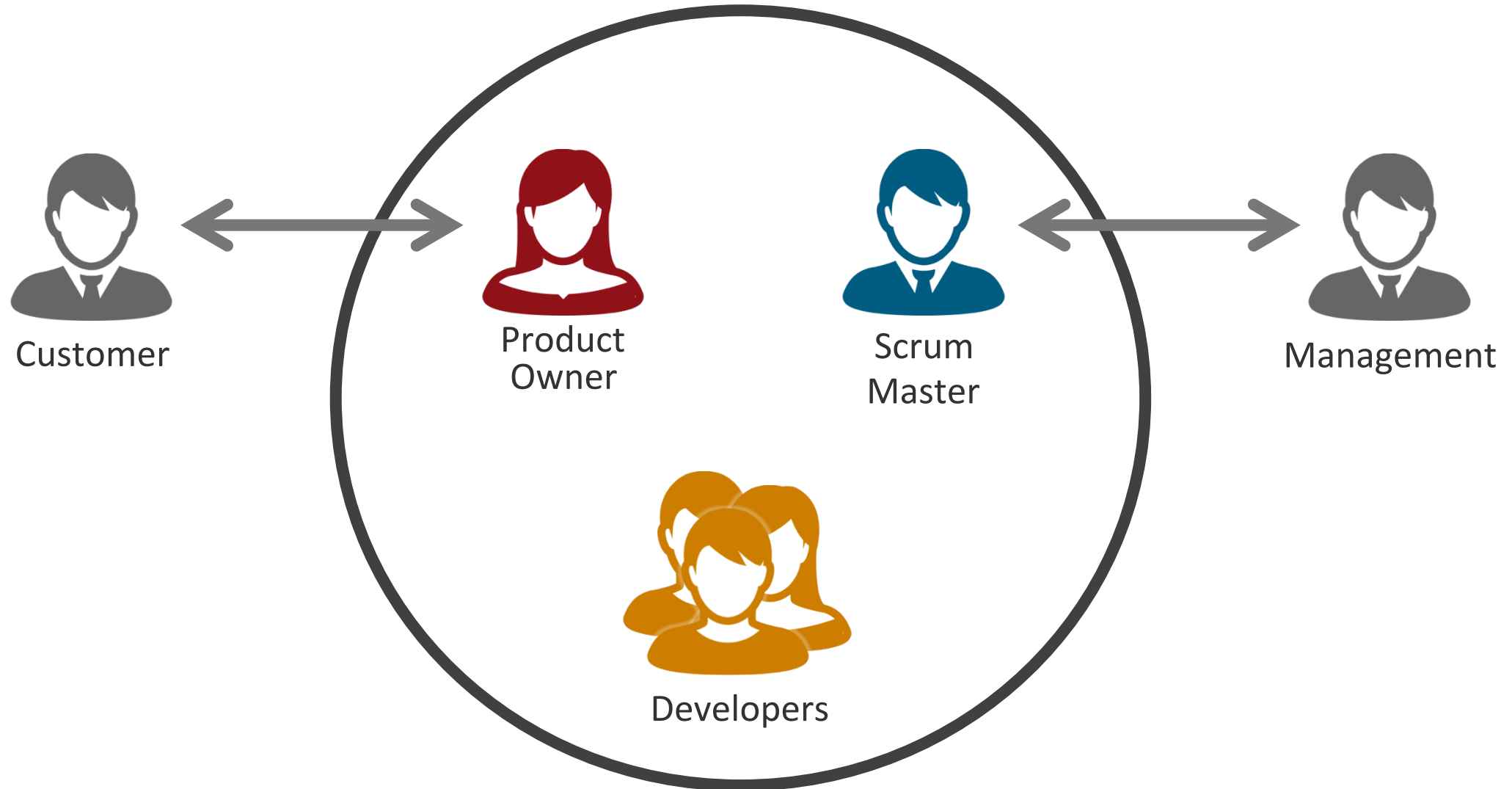


1. The Case for Agile
- 2. The Scrum Process**
3. Scaling Scrum

Scrum



The Team



Product Owner



Responsibilities

- **Customer** communication
 - Contact person for team
- Product Backlog
 - **User Stories**
 - Priorities
- Acceptance Criteria & Tests



Scrum Master



Responsibilities

- **Process** manager
 - Moderator in meetings
- Management communication
 - Remove **impediments**
- Enabler, not boss



Developers



Responsibilities

- Communication
 - **Critically** discuss all inputs
 - Honestly share important information
 - Represent team as expert
- Sprint Backlog
- Developing ;-)

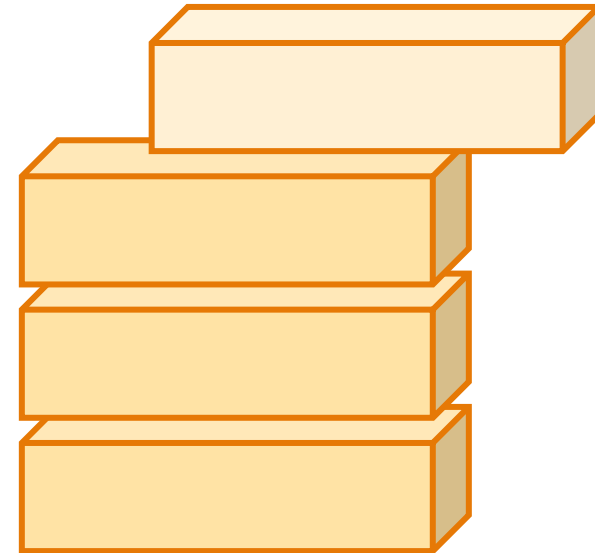


Product Backlog



List of work items

- Requirements (modification **requests**)
 - Features
 - Bug fixes
- Ordered/**prioritized**



Requirements



In Scrum, requirements are **often** defined as user stories:

“As <role>, I want <feature> to <reason>”

Requirements need to fulfill **INVEST** properties:

- I – Independent
- N – Negotiable
- V – Valuable
- E – Estimable
- S – Small
- T – Testable

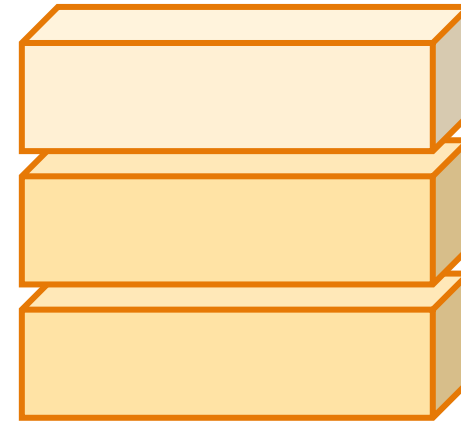
<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

Product Backlog Refinement Meeting



Ensure the backlog is ready for the next sprint

- Aka “Backlog Grooming”
- Team and PO discuss top Product Backlog items
 - Clarify questions
 - E.g. “Who can access this?”
“what happens if...?”
- Make sure Product Backlog Items conform to team’s “Definition of Ready”
- Ready is when the team says: “Ah, we get it” – Jeff Sutherland



Planning Meeting



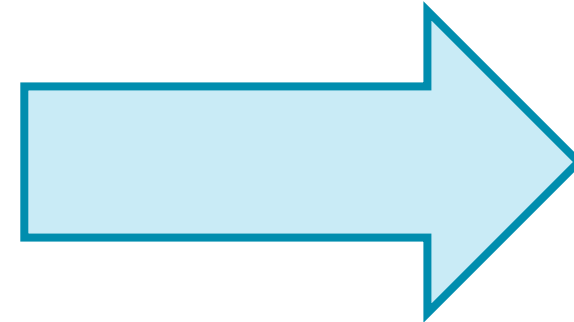
Filling the sprint

- Estimate Backlog items
- Move items from Product to **Sprint Backlog**

Defining the work

- **Break down** Backlog items into tasks
- PO not required

Total time: 2 hours per week of sprint



Tasks



For better planning, stories are broken down into tasks

Tasks should be **SMART**:

- S – Specific
- M – Measurable
- A – Achievable
- R – Relevant
- T – Time-boxed

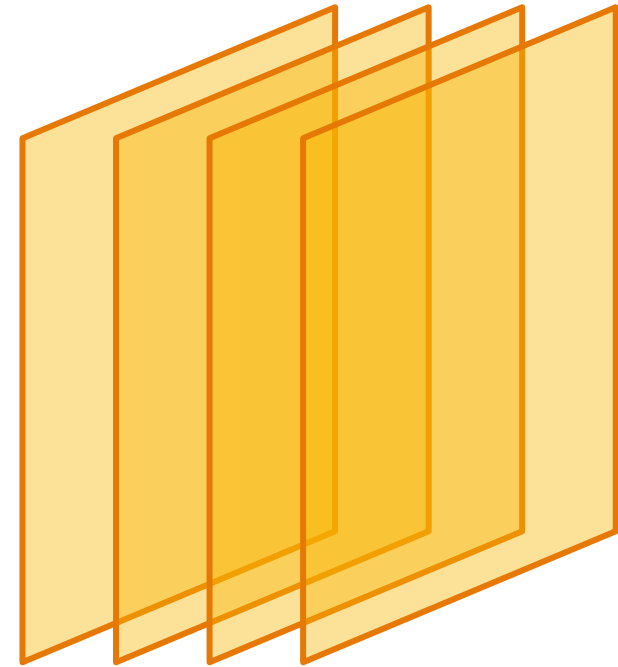
<http://xp123.com/articles/invest-in-good-stories-and-smart-tasks/>

Sprint Backlog



List of tasks for a sprint

- Tasks are **signed-up** for, not assigned
- During the sprint
 - **No new features**
 - Team may change/add tasks



Daily Scrum Meeting



Status update

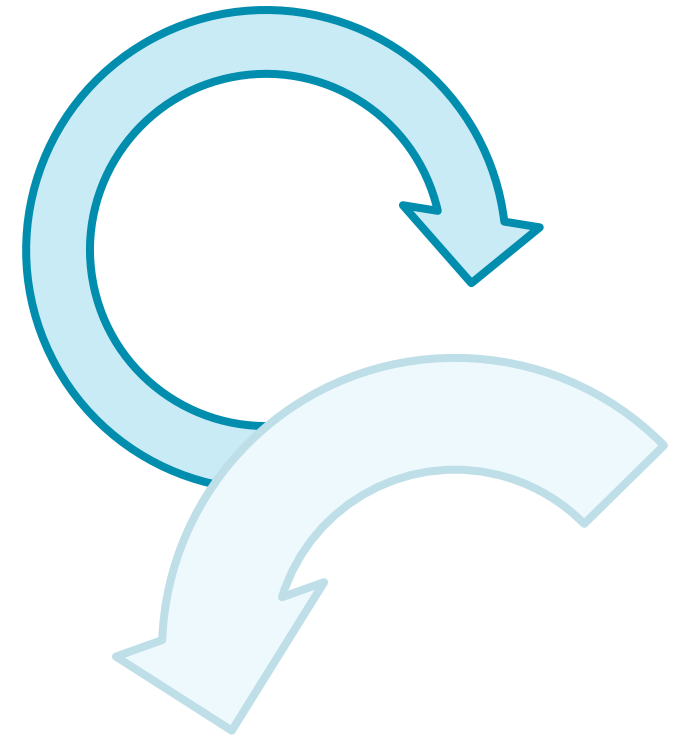
- Last achievements
- Next steps
- Problems

Max. **2 min** per person

Stand-up meeting?

Discussions?

- Schedule **subsequent** expert's meeting

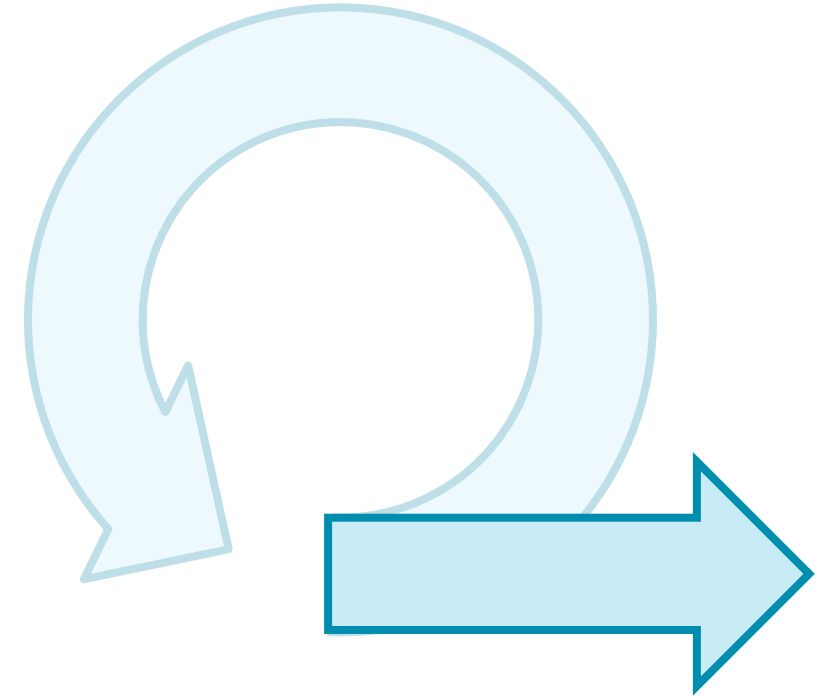


Review Meeting



Acceptance of Features

- Demo to PO
 - PO should be prepared
 - Optional: invite other stakeholders
- Comments by developers

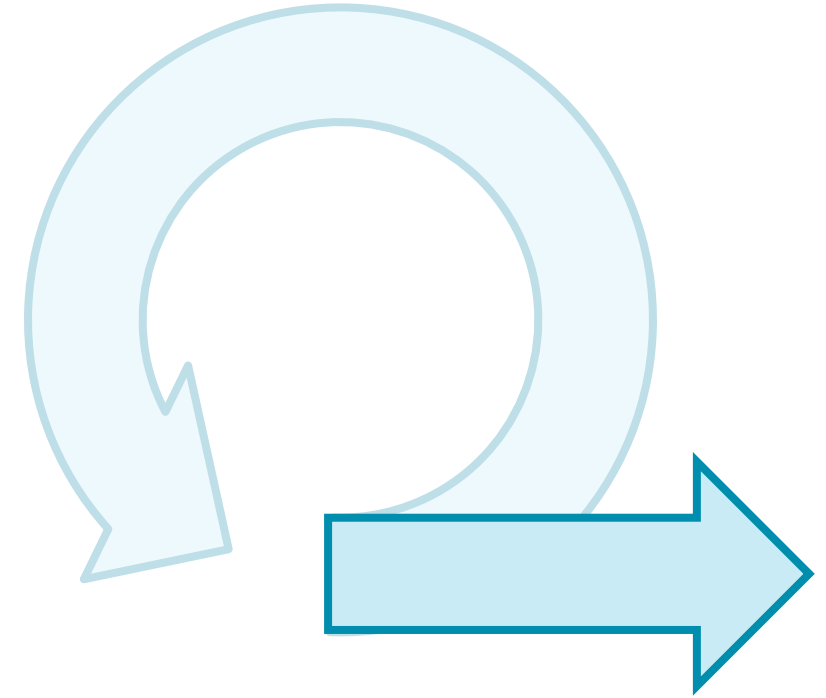


Retrospective Meeting



Internal team evaluation

- PO not required
- Discuss process and problems
- **Measure** improvements

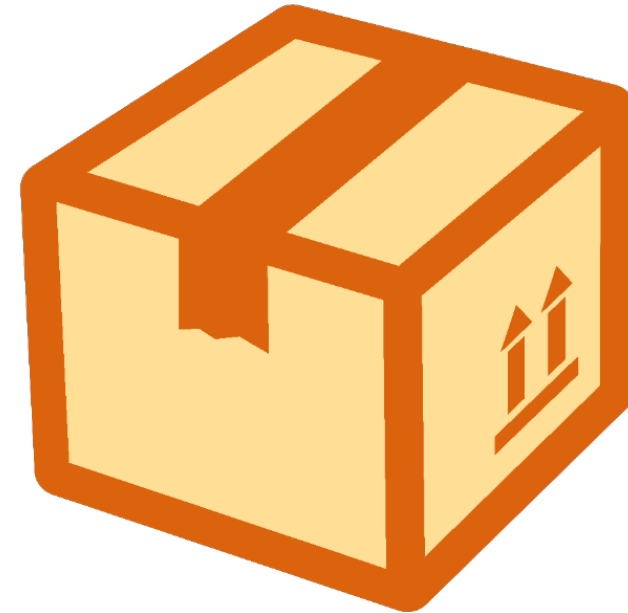


Product Increment



Potentially shippable increment

- Complete according to **Definition of Done**
 - Even if not actually released
- **No regrets** if project ended now



Scrum



Team

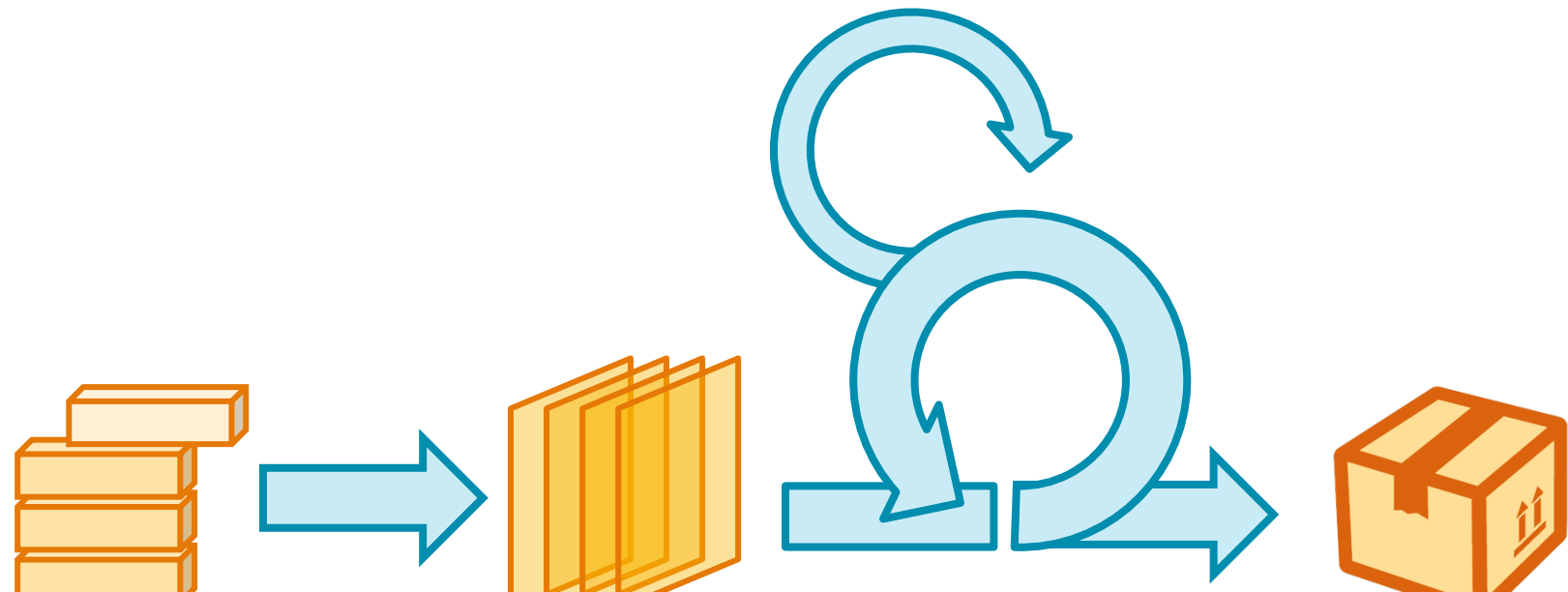
- Product Owner
- Scrum Master
- Developers

Meetings

- Planning
- Daily Scrum
- Review
- Retrospective

Artifacts

- Product Backlog
- Sprint Backlog
- User Stories
- Software Increment



Effort, Schedule, and Cost Estimation



- Depends on software engineering process
- Highly **uncertain**, must be negotiated and revised with stakeholders

- Waterfall effort estimation
 - Methods: calibrated estimation model based on historical size (Function Points, LOC, ...); expert judgment; ...
 - Output: X man-months
- Agile effort estimation
 - **Iterative** methods, **shorter** planning horizon
 - Output: functionality to be implemented in the **next iteration**

Effort Estimation in Scrum with “Planning Poker”



Participants

- **Everyone** operationally involved in creating the software product
- Product Owner (and Scrum Master) are not playing

Preconditions

- Product backlog is complete and **prioritized**
- Backlog items are known by the team
- The effort for a small backlog item was determined as a **reference**
- Every participant has a set with sizing cards

Planning Poker 1/2



- Product owner explains a backlog item
- Product owner **answers questions** of team members
- Every participant evaluates the complexity of the backlog item and chooses a card (**hidden**)
- All cards are shown simultaneously
- Participants with highest and lowest number **explain choices**
- The arguments are **discussed** in the group

Planning Poker 2/2



- A new vote is conducted
- **Team agrees** on item size
 - Most occurring or average value is acceptable
 - If not, another round is played
- The moderator notes size of backlog item in the product backlog
- The game ends if all backlog items are sized or **time is over**

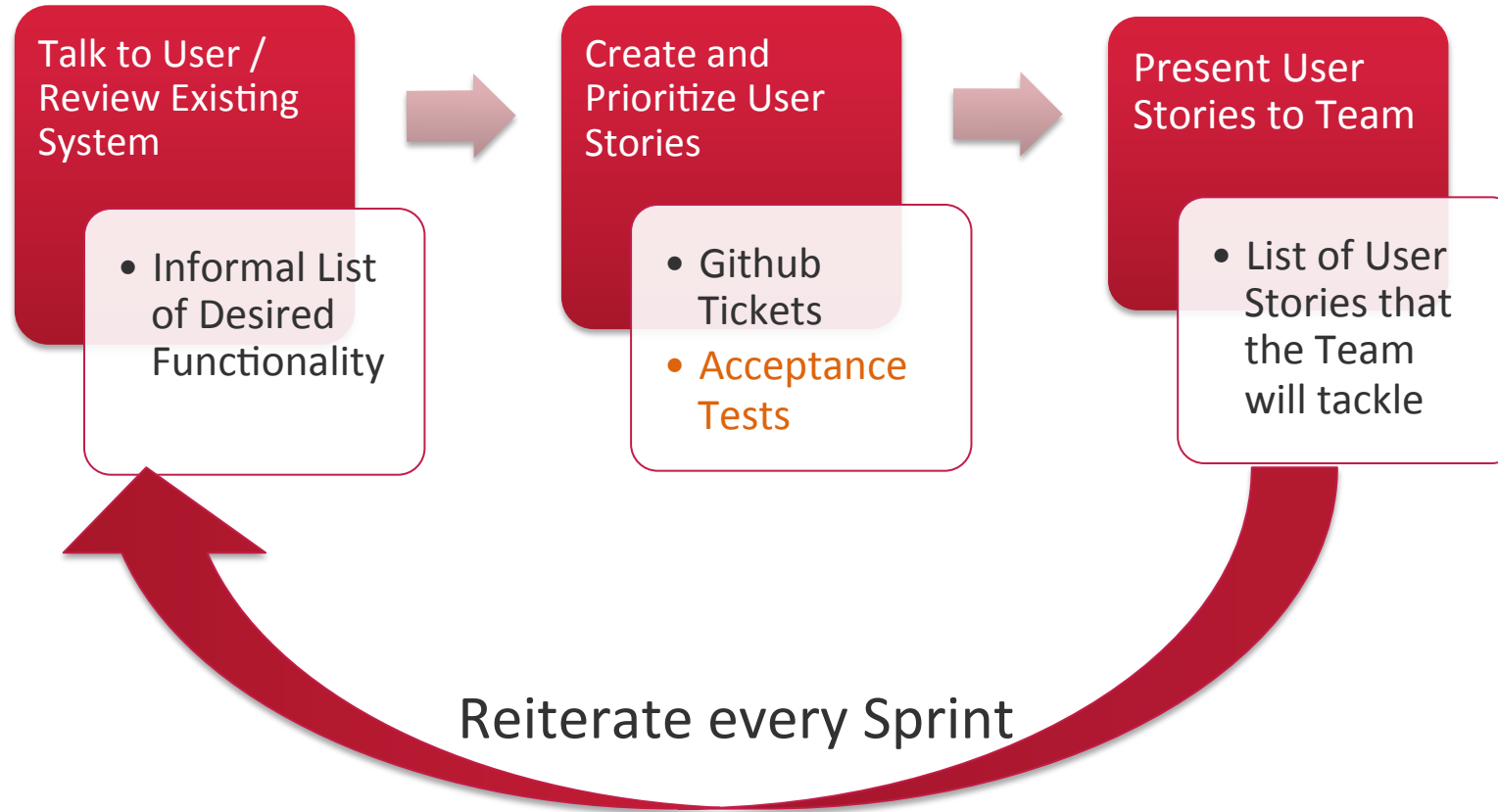
After Planning Poker



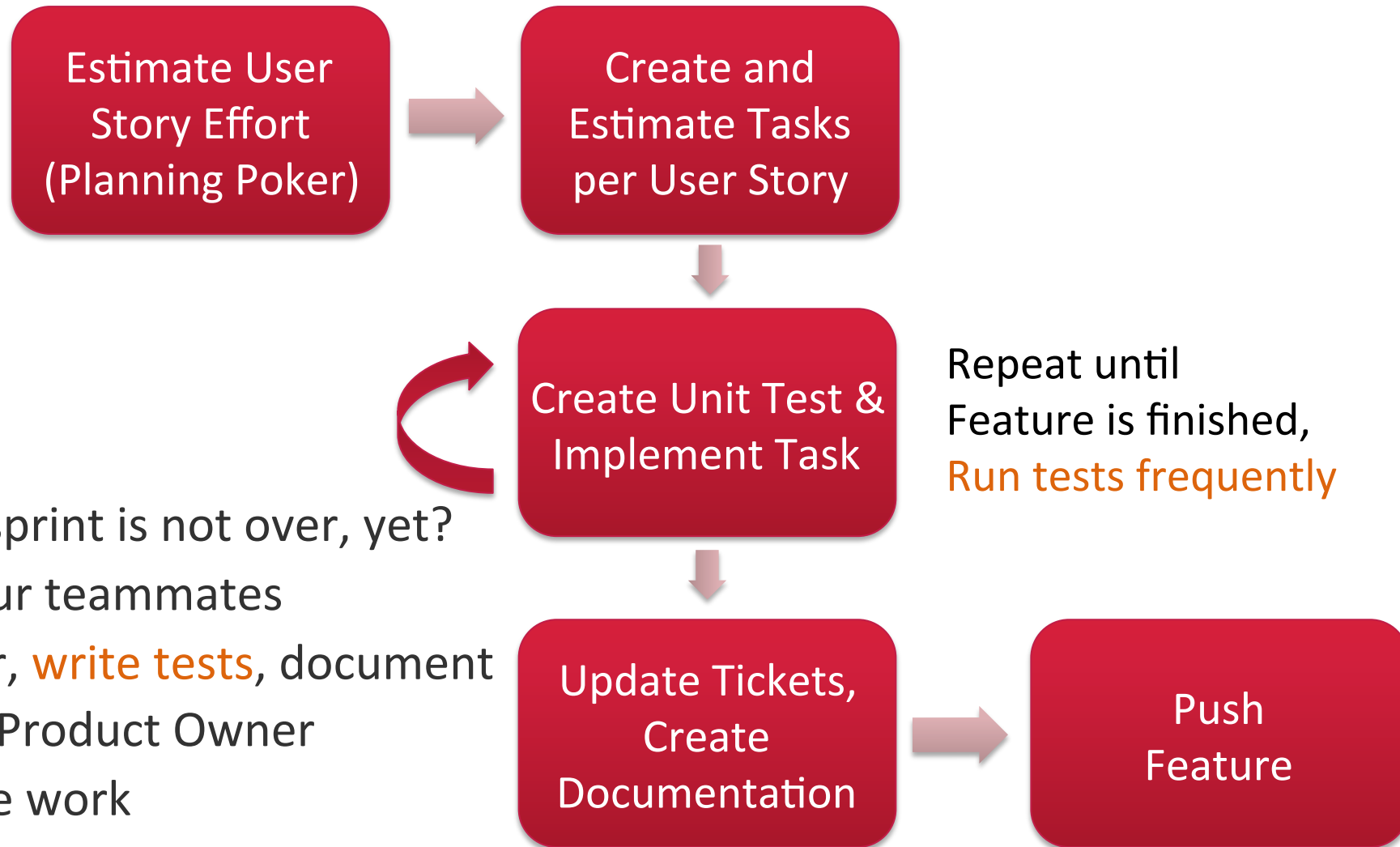
Begin the sprint

- Select stories until sprint is full
- Break down stories into tasks and fill your **Scrum Board**
(e.g. ToDo – In Progress – Verify - Done)
- Assign stories to developer(s)
- **Implement** the stories task by task

Projekt Workflow: Product Owner



Project Workflow: Developers



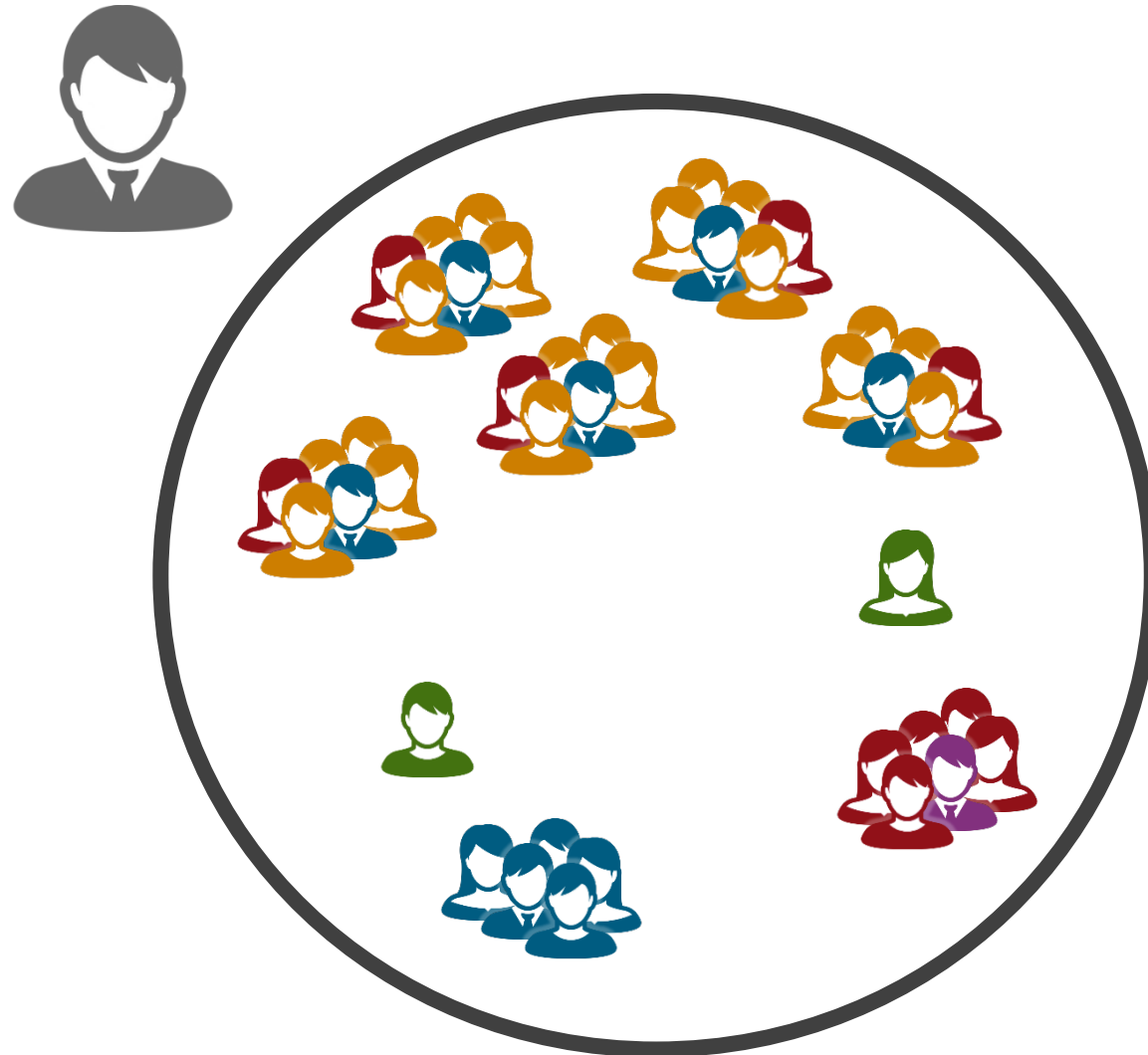
Done and sprint is not over, yet?

- Help your teammates
- Refactor, write tests, document
- Ask the Product Owner for more work



Scaling Scrum

Recap: High-level Overview of SWT2



Implications of the Setup



What's needed in such an environment?

- Development **process**
- **Communication** on multiple levels
- Infrastructure for **collaboration**

Scaling Scrum: Project Start



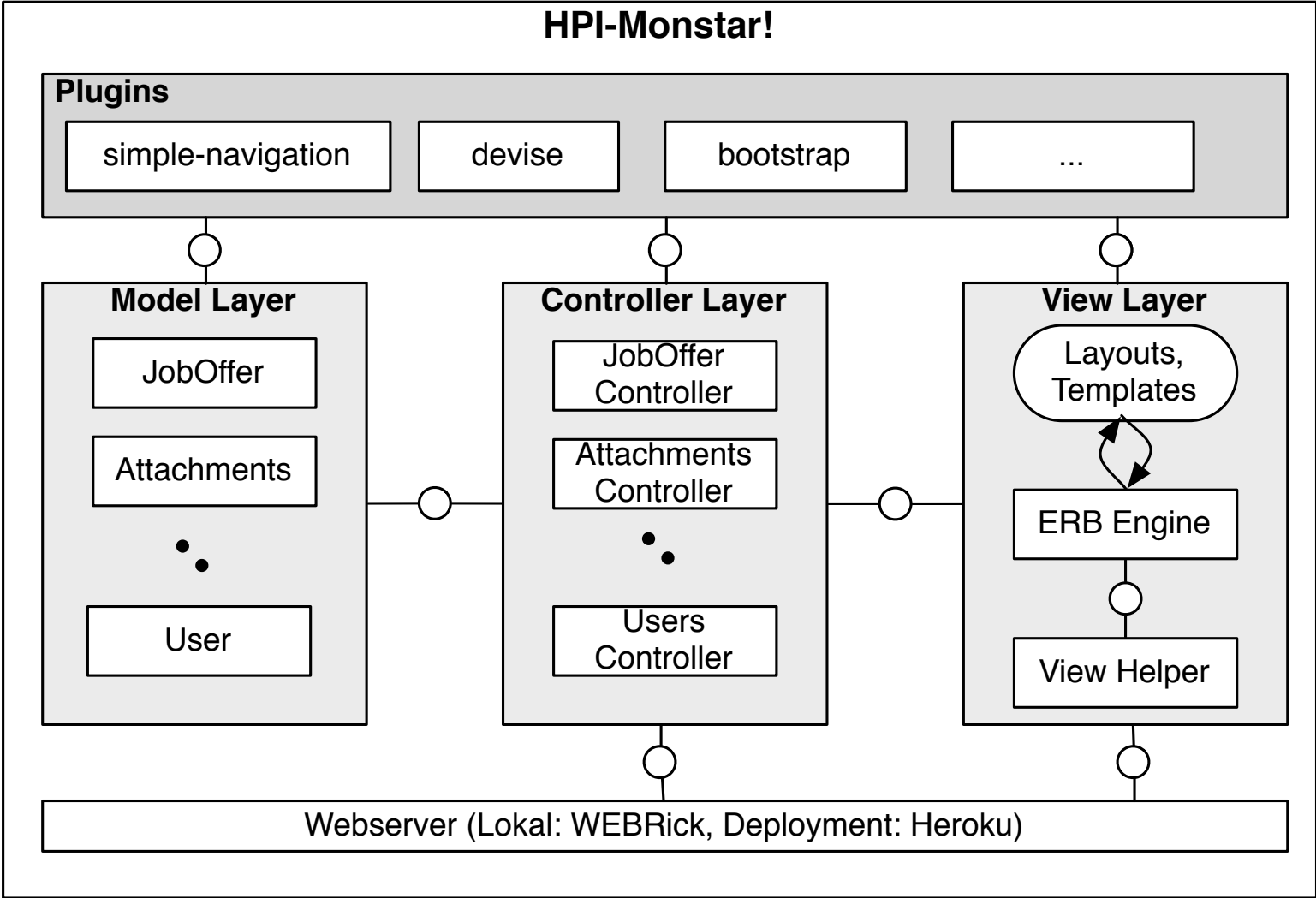
Start small and grow organically

- Single Scrum team for preparation
- Work out foundation for the first sprints
- Scale when it becomes necessary

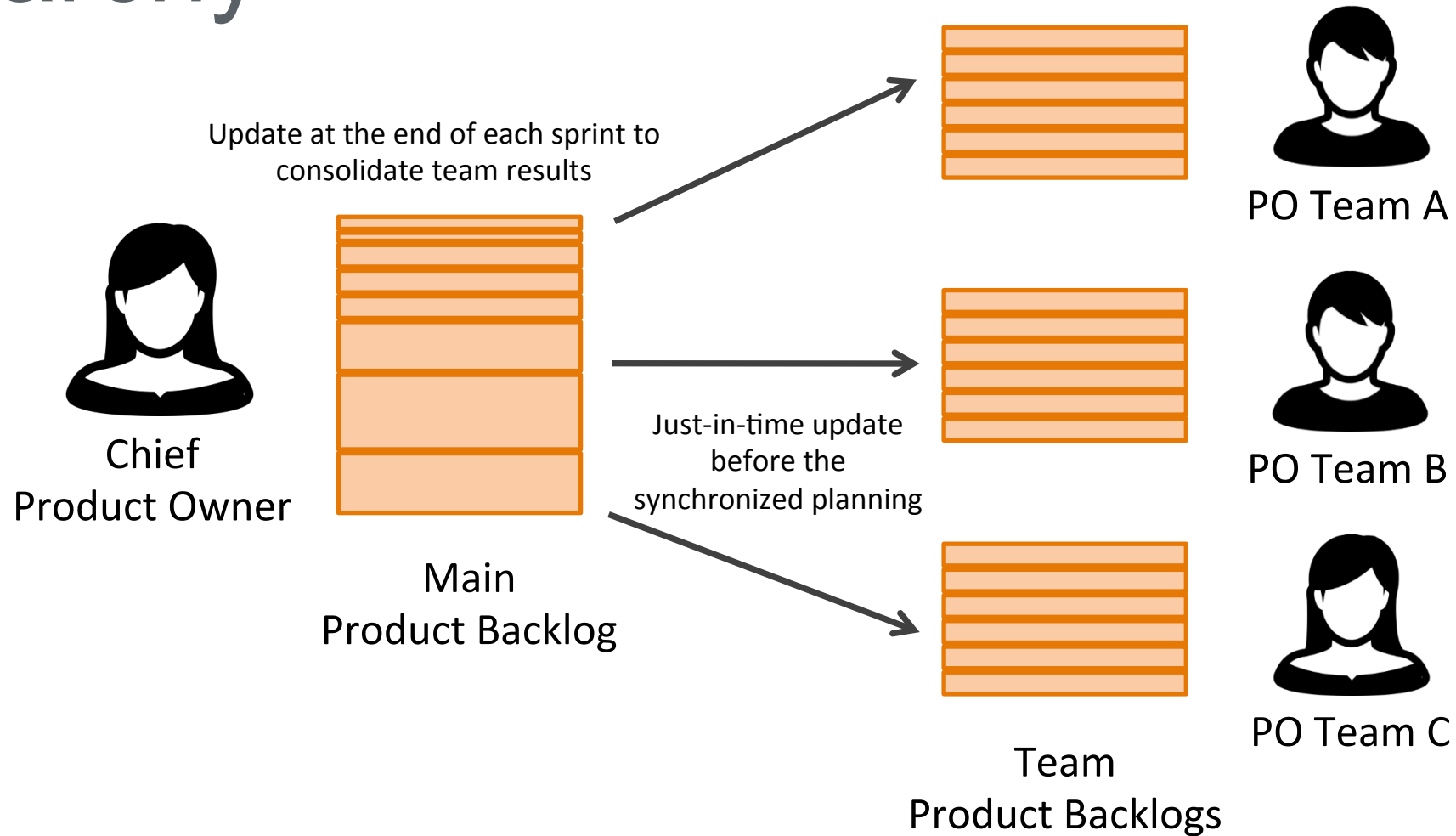
We are now at the first scaling point!

- Rudimentary architecture is present
- Infrastructure is prepared and **ready to go**

Architecture Overview



Product Owner / Backlog Hierarchy



[Christoph Mathis, Scrum Center]

Scaling Scrum: Sprint Planning

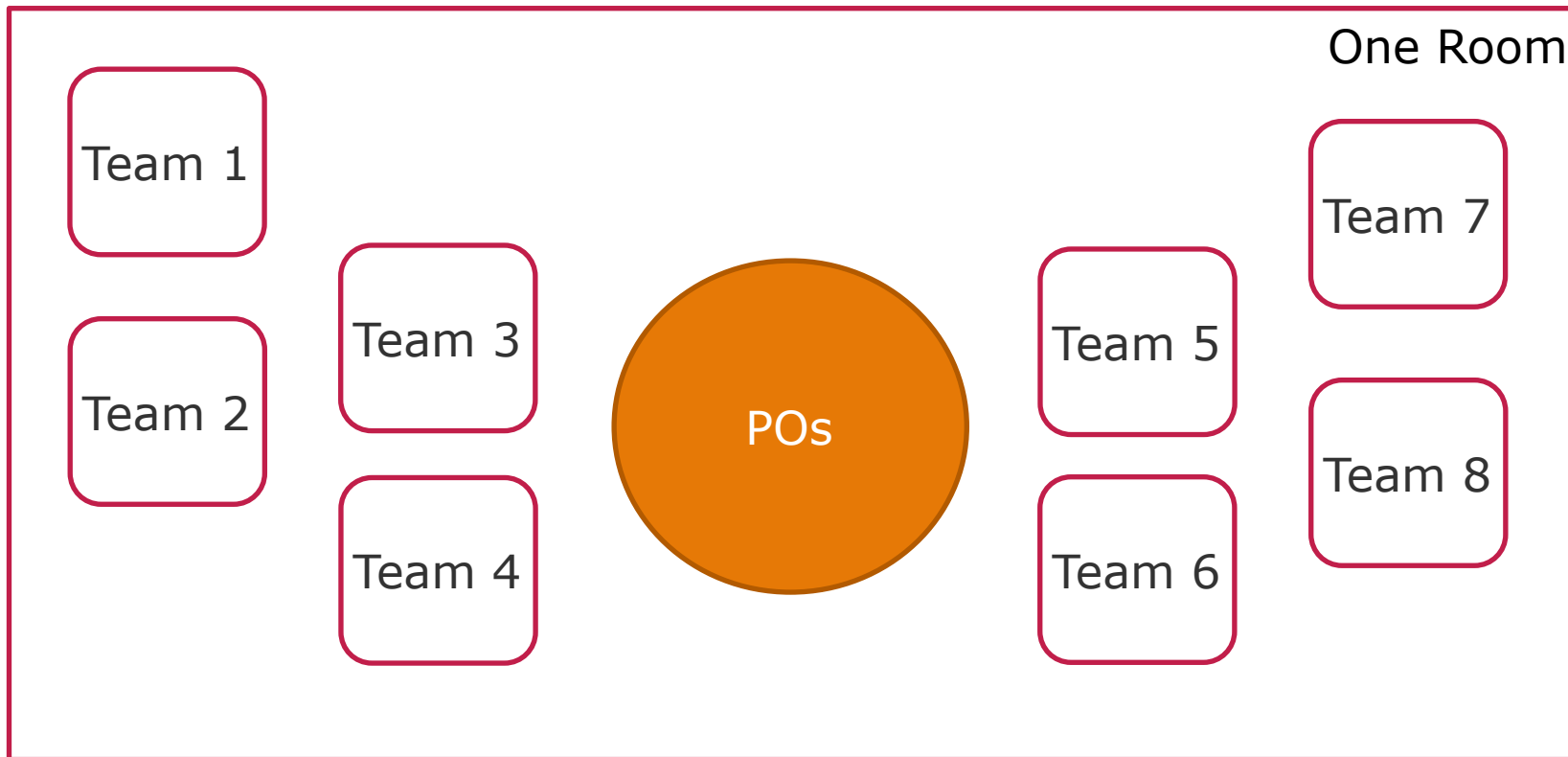


- Preparation
 - Individual review and retrospection meetings
 - Meeting of all teams with 1-2 members each:
 - Review of the last sprint
 - Input dependencies (What is needed)
 - Output dependencies (What needs to be delivered)
- Execution
 - Individual plannings (strict timeboxing)
 - Discussion of identified additional input or output dependencies
 - Final sprint planning
- Problem: Time consuming & high degree of coordination needed!

Scaling Scrum: Sprint Planning



Another Option: Co-located planning



Scrum of Scrums



Goal: Synchronize team effort with minimal coordination overhead

- Regular meeting of all Scrum masters.
 - Developers join if necessary (**ambassador principle**)
- Scrum masters
 - Share their learnings
 - Report completions & next steps
 - Coordinate inter-team dependencies
 - Negotiate responsibility
- Developers discuss technical interfaces across teams
- Distribute information back into the teams

Scrum

1. The Case for Agile
2. The Scrum Process
3. Scaling Scrum

Questions?

Image Credits



- "ST vs Gloucester - Match - 23" by PierreSelim - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:ST_vs_Gloucester_-_Match_-_23.JPG#mediaviewer/File:ST_vs_Gloucester_-_Match_-_23.JPG
- "Scrum process" by Lakeworks - Own work. Licensed under Creative Commons Attribution-Share Alike 3.0-2.5-2.0-1.0 via Wikimedia Commons - http://commons.wikimedia.org/wiki/File:Scrum_process.svg#mediaviewer/File:Scrum_process.svg
- „Wien - Seestadt, SW-Areal 2013 (2)“ von Bwag - Eigenes Werk. Lizenziert unter Creative Commons Attribution-Share Alike 3.0-at über Wikimedia Commons - [http://commons.wikimedia.org/wiki/File:Wien_-_Seestadt,_SW-Areal_2013_\(2\).JPG#mediaviewer/File:Wien_-_Seestadt,_SW-Areal_2013_\(2\).JPG](http://commons.wikimedia.org/wiki/File:Wien_-_Seestadt,_SW-Areal_2013_(2).JPG#mediaviewer/File:Wien_-_Seestadt,_SW-Areal_2013_(2).JPG)