Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Software Engineering 2 (SWT2)

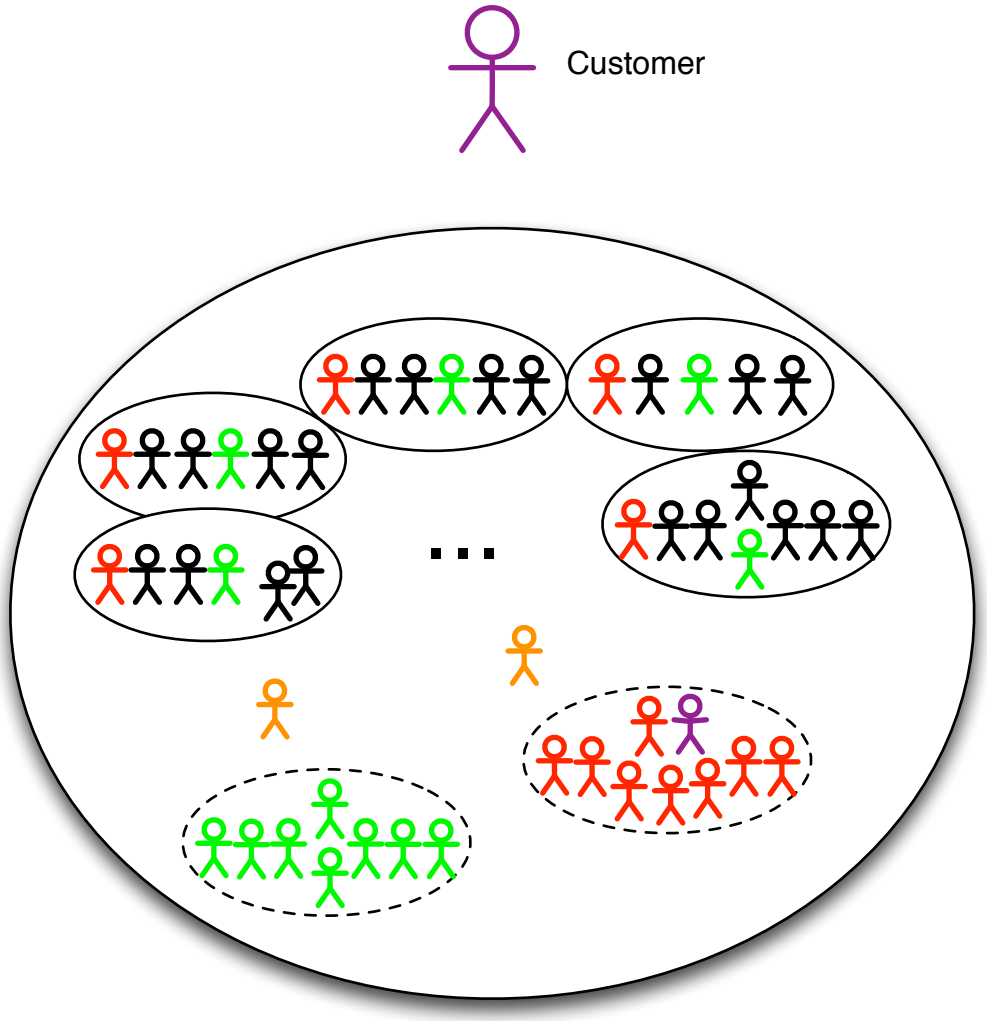
Project Kickoff:

Development Process & Collaboration Infrastructure

Agenda: Process & Infrastructure

- Development Process for the project (Scaling Scrum)
- Collaboration Infrastructure
  - Communication & Coordination (Email, Calendar)
  - Application Lifecycle Management System (Redmine)
  - Continuous Integration (Hudson)
  - Version Control (GIT)

Recap: High-level Overview of SWT2



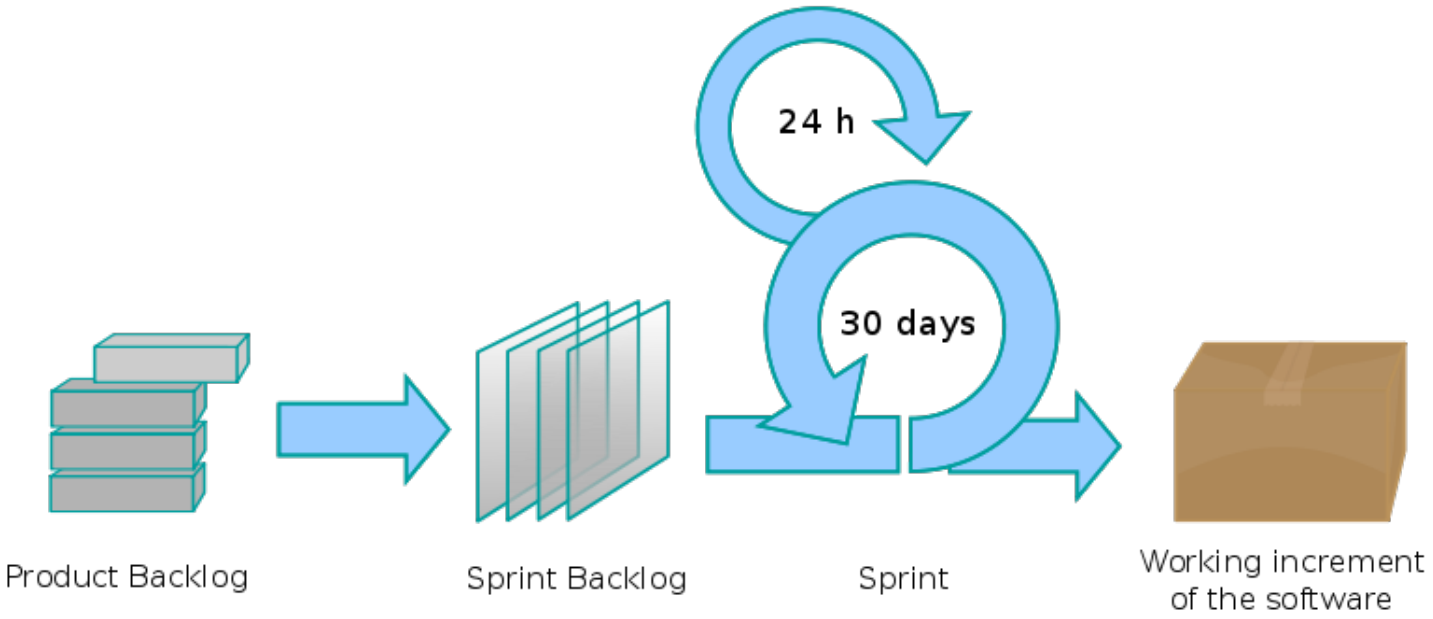
Implications of the Setup

What's needed in such an environment?

- A development process with clearly defined responsibilities
- Communication on multiple levels
- An Infrastructure for collaboration

Recap: Scrum

Foundation: Scrum

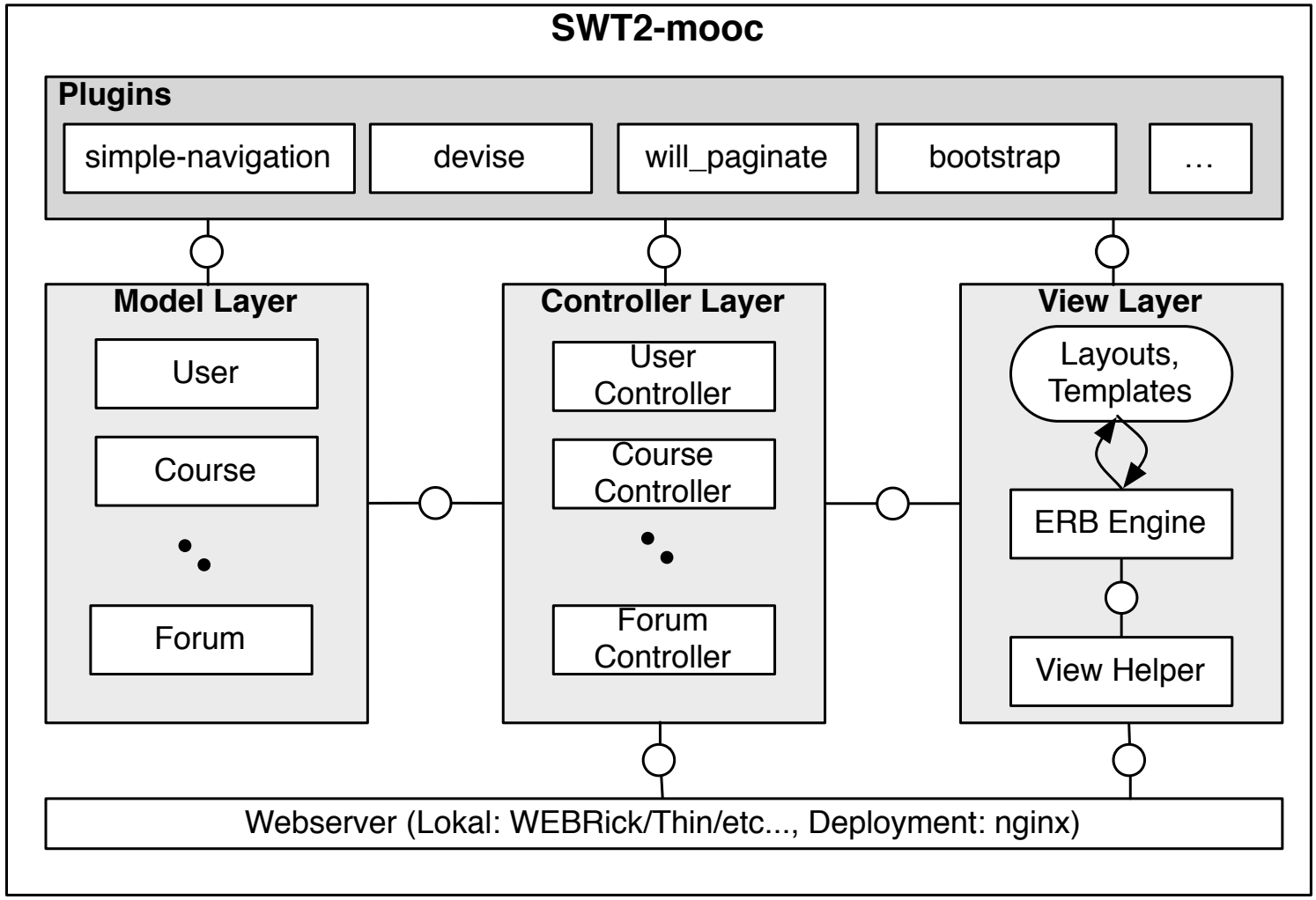


Question: How to scale this to multiple teams?

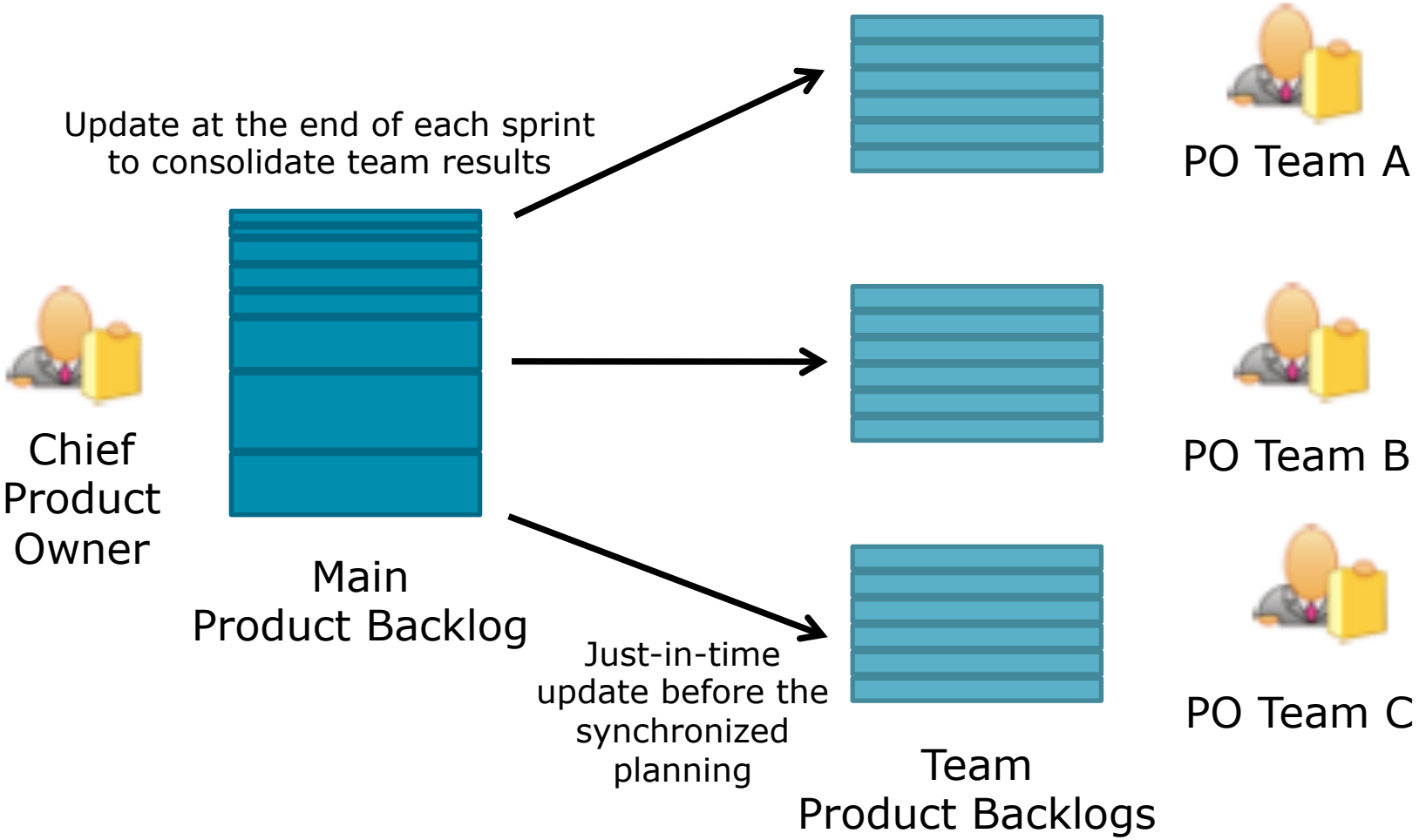
Scaling Scrum: Project Start

- Start small and grow organically
  - Single Scrum team for preparation
  - Work out foundation for the first sprints
  - Scale when it becomes necessary

Architecture Overview



Scaling Scrum: Product Owner Hierarchy



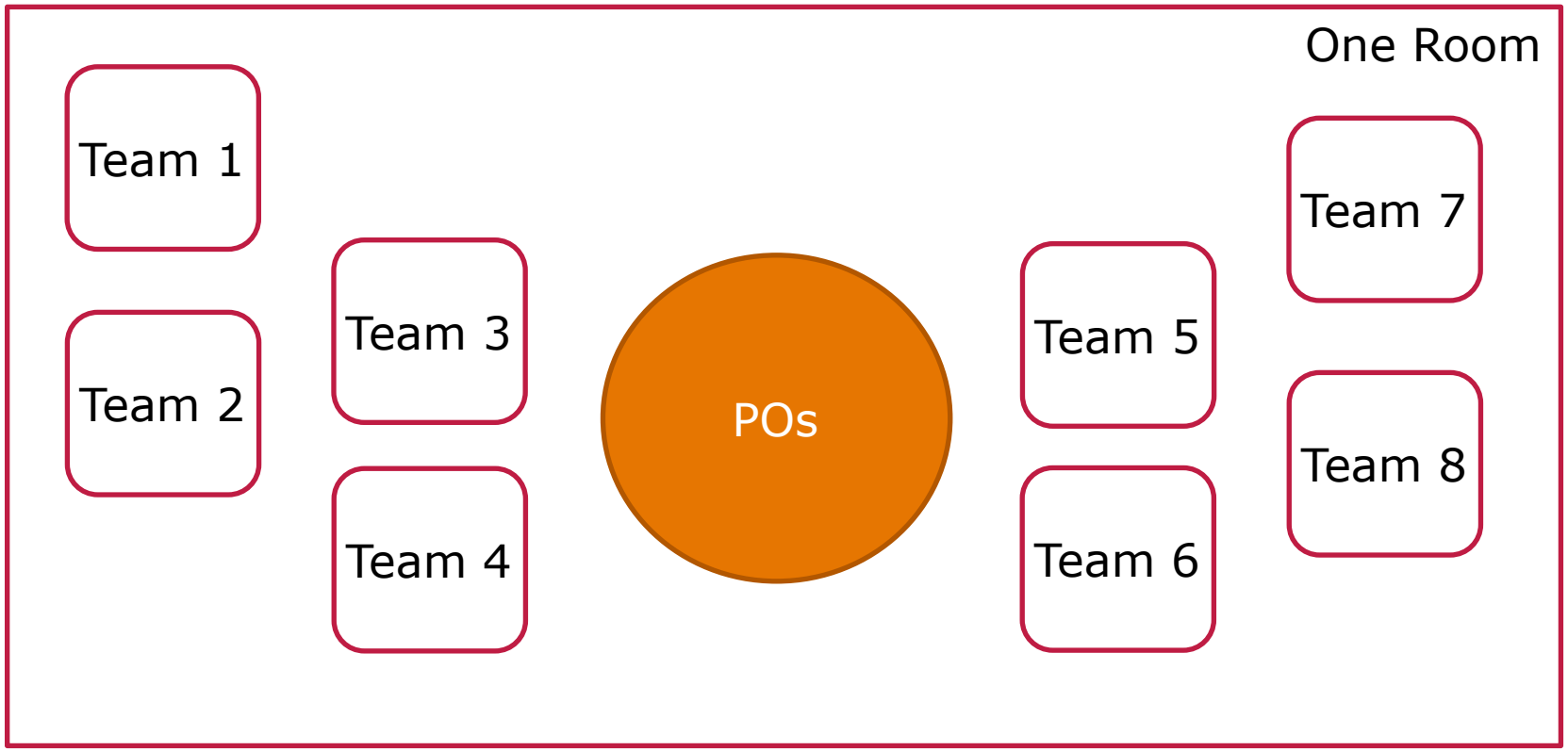
[Christoph Mathis, Scrum Center]

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

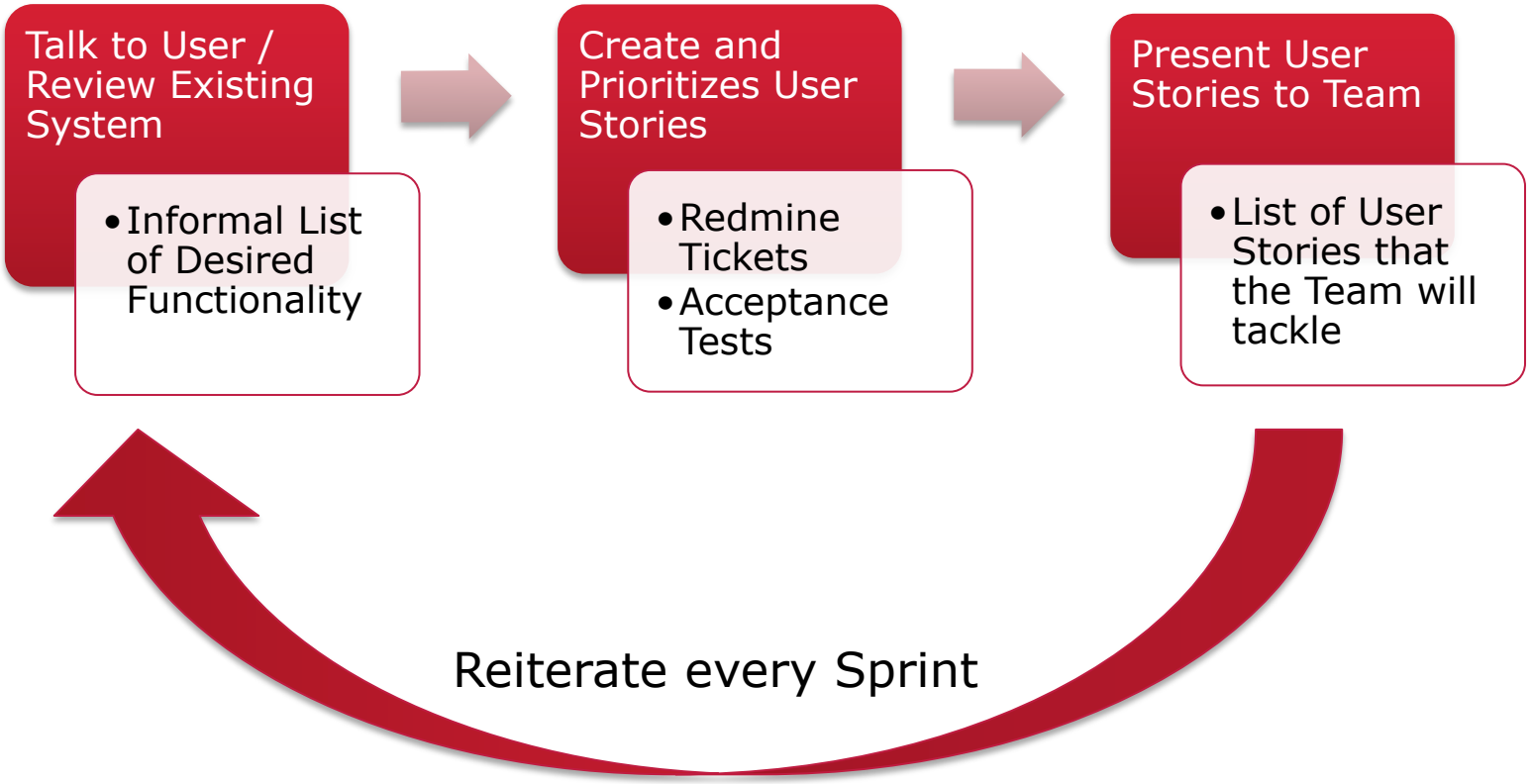
- Another Option: Co-located planning (we'll try for final sprint!)



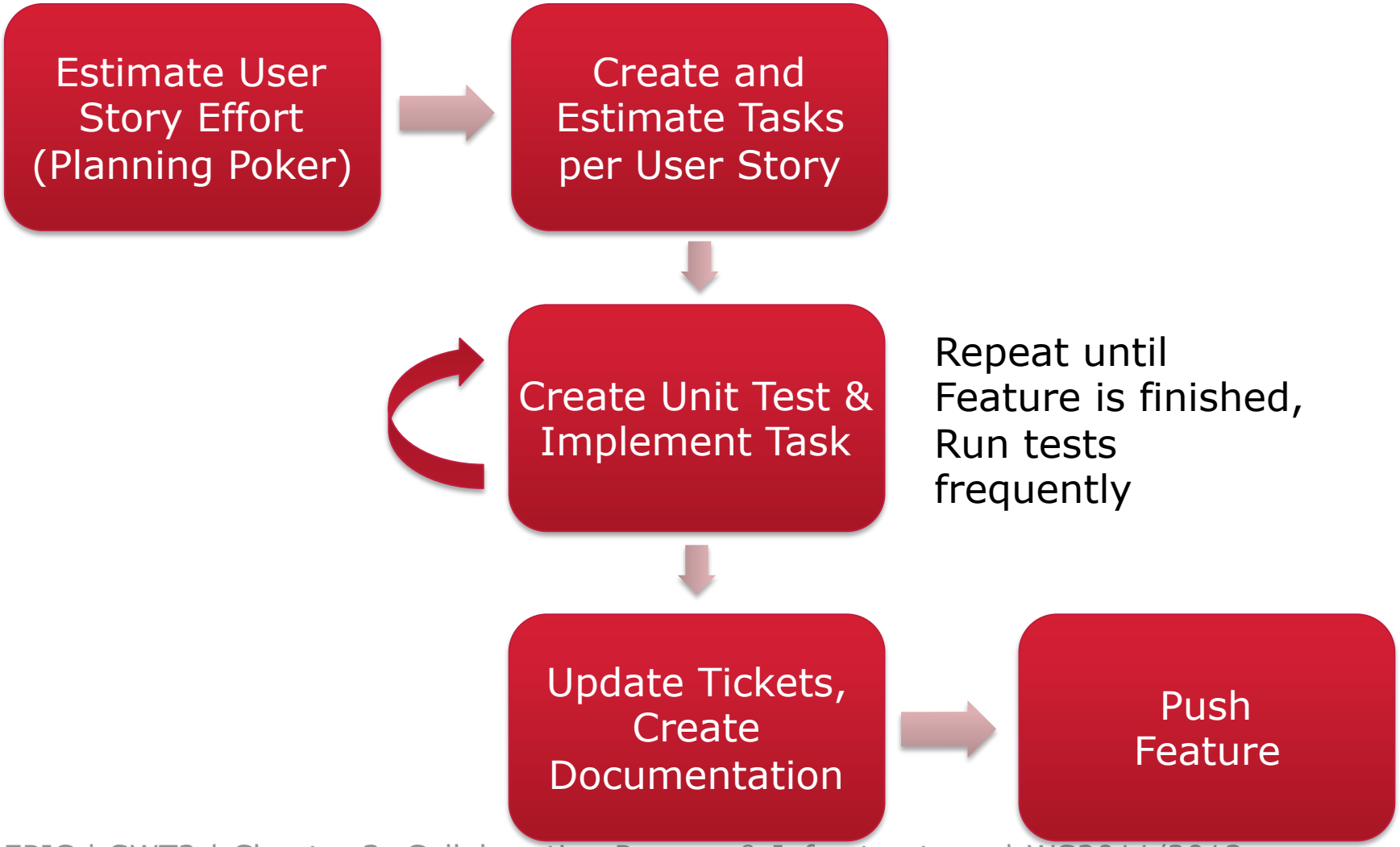
Scaling Scrum: Scrum of Scrums

- Synchronization within the Sprints
- Ideally after each Daily Scrum (weekly in our project)
- **Participants:** Whoever is best suited for current topics, not necessarily the ScrumMaster
- Scaling the Daily Scrum questions to team level
- Additional question: What actions might affect other teams?
- Keep notes!

Project Workflow - PO



Project Workflow - Developer



Communication Infrastructure

- Email lists
  - Separate lists for each team
  - Keep your teammates in the loop
  - Rules and filters help organizing your inbox
- Wiki for lean and globally accessible documentation
- Ticket system for overview and feedback about current tasks and progress
- Telephone, Skype, IRC, … oh, and personal contact for direct communication
- … be creative! (but let us know, we are interested in learning what might be useful in the future)

Time Management

Google Calendar

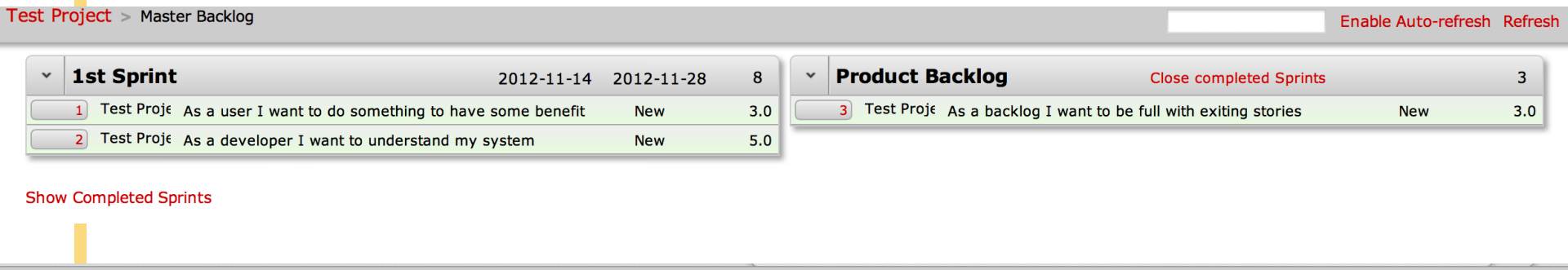
- Advantages:
  - Available Everywhere
  - Easy Integration with Outlook & iCal (see “Useful Links”)
- Overview of team appointments
- Access granted by our tutors

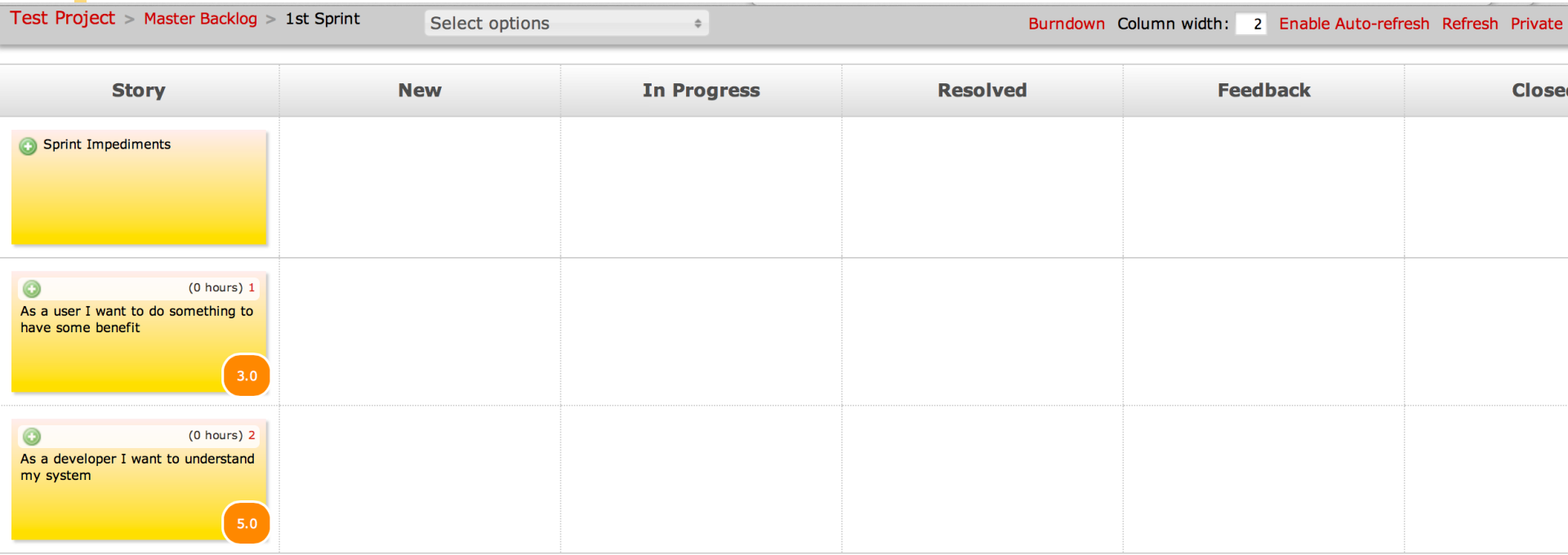
Application Lifecycle Management

The Swiss Army knive for software development

- Integrating tools for most common activities in one place
- Wiki, Bug Tracking, Time Management, Project Analytics, …
- Some examples: MS Team Foundation Server, Codebeamer, Trac, Plan.io (SaaS)
- Our tool: Redmine (http://www.redmine.org/)

SCRUM support in Redmine - Backlogs

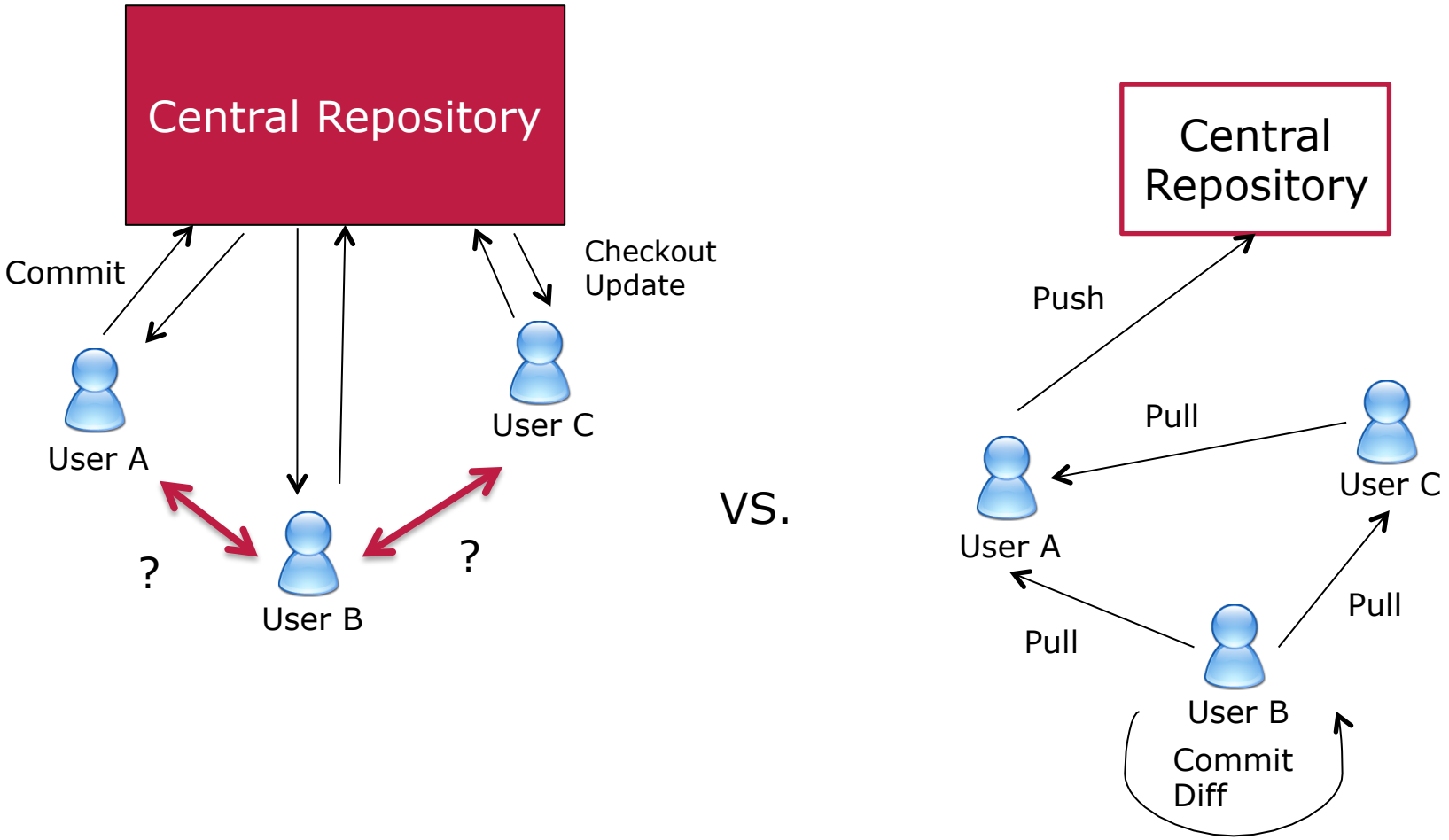




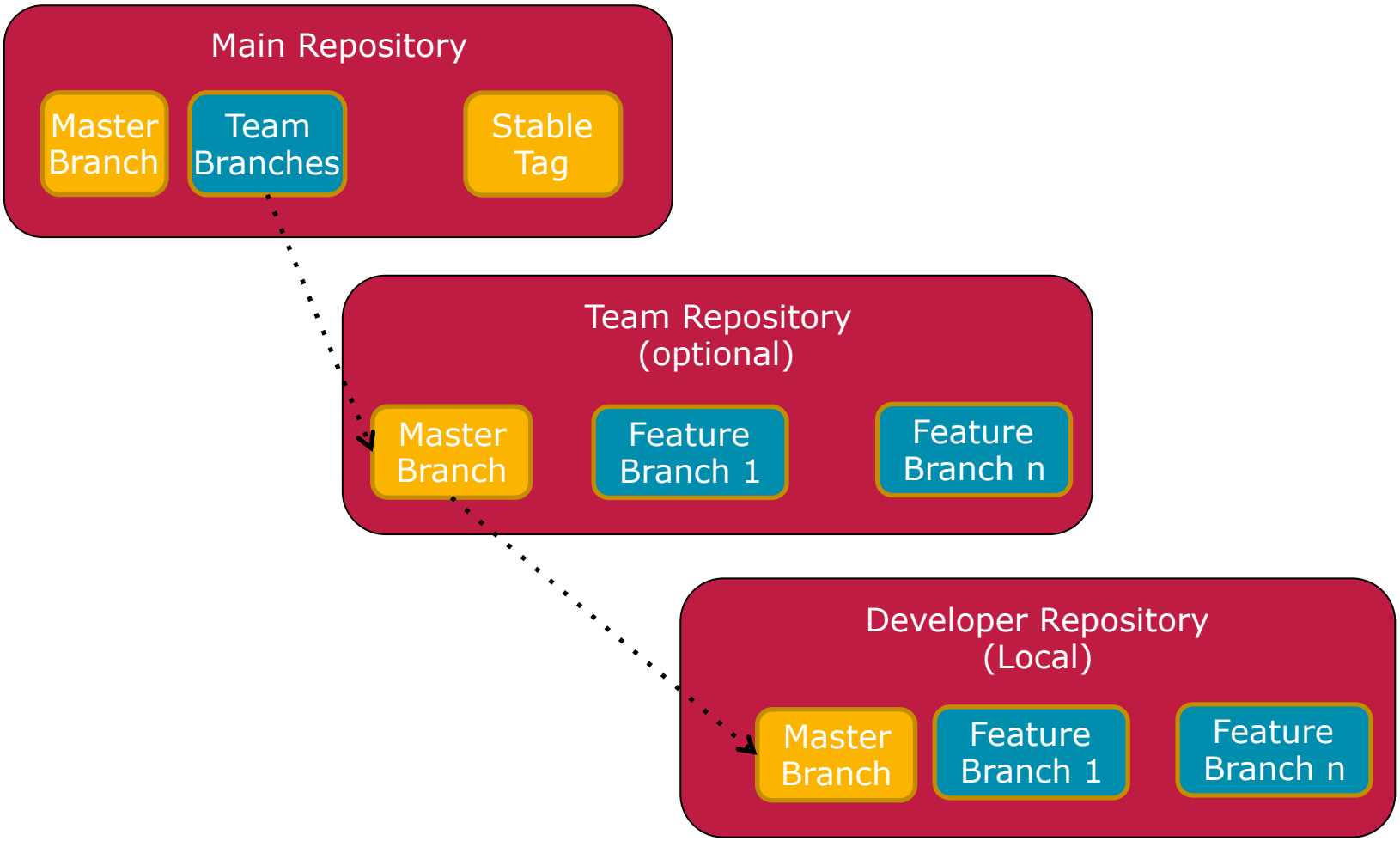
Version Control Systems

- Repository to store the configuration items
- Versioning
- Dealing with variants: branches
- Access control
  - Authentication, authorization
  - Locking
  - Concurrent development
- Reporting
  - How many versions, variants, changes, persons
  - History of changes

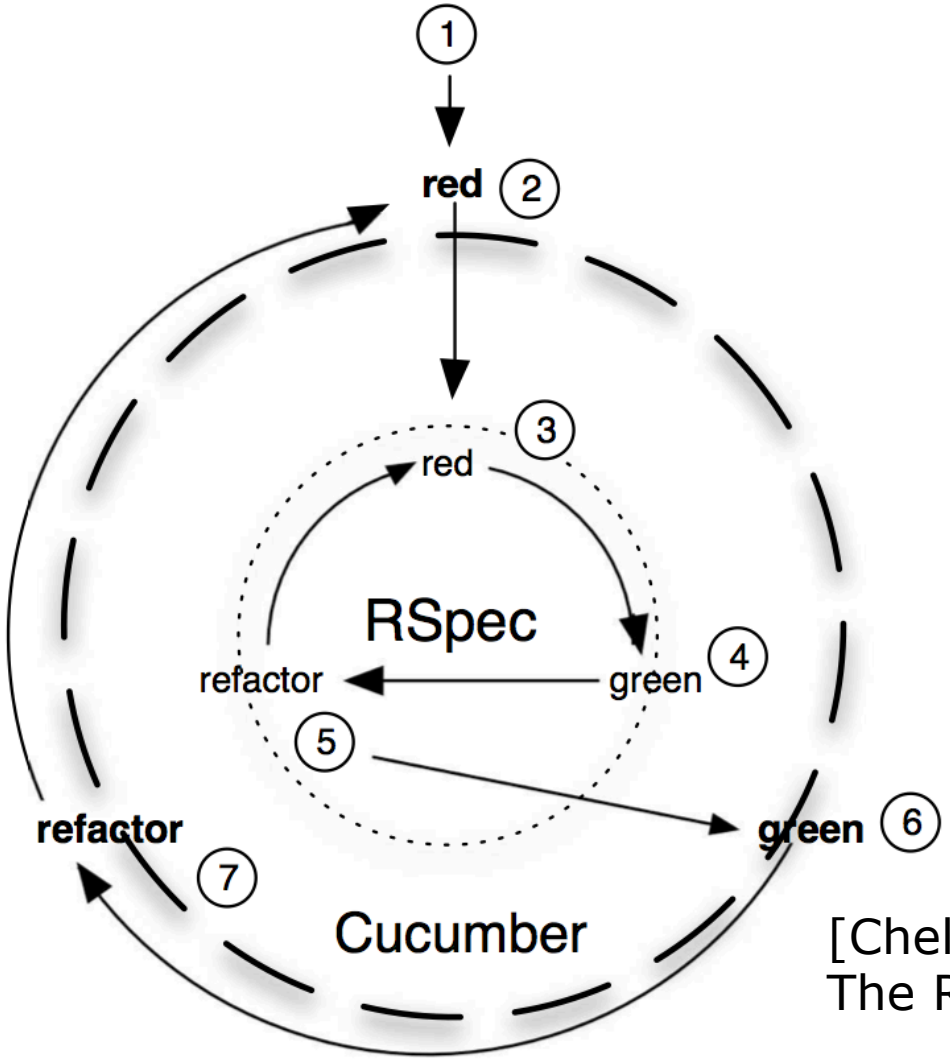
Centralized vs. Distributed VCS



Project Repository Setup



BDD/TDD Cycle



[Chelimsky et al.:
The Rspec Book, 2010]

Continuous Integration

- Problem: How to check continously that your software works?
- Solution: Continuous Integration Server
  - Connected to version control
  - Customizable run scripts
  - Ideally covering all development branches
  - Checkout -> prepare environment -> run tests -> run statistics
  - Examples: CruiseControl, Anthill
  - Our system: Hudson-CI

Project Timeline

- KW 46 (12.-16.11.): Planning Sprint #1
- KW 48 (26.-30.11.): Review Sprint #1 / Planning Sprint #2
- 14.12. Intermediate Presentation
- KW 51 (17. – 21.12.): Review Sprint #2 / Planning Sprint #3
- KW 3 (14. – 18.01.): Review Sprint #3 / Planning Sprint #4
- KW 6 (4. – 8.2.): Review Sprint #4

Let's get started!

- POs
  - Meet with the customer
  - Extract Requirements + create user stories (Redmine)
  - Prepare Sprint planning
- Teams
  - Prepare working environment
    - Clone repository, try to get application working, understand architecture, ideally: play around
  - Find a regular timeslot for your meetings
    - Within the team + tutors
    - Enter into Google Calendar until **Nov 13, 12pm CET**
- **1st Sprint Planning -> Next week**
- **No lecture next week**

Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Software Engineering 2 (SWT2)

Sprint Planning Sessions

Effort, Schedule, and Cost Estimation

- Depends on software engineering process
- Highly uncertain, must be negotiated and revised with stakeholders
- Effort estimation (non-agile software development models)
  - Methods: calibrated estimation model based on historical size (Function Points, LOC, …); expert judgment; …
  - Output: X man-months
- Effort estimation (agile software development models)
  - Iterative methods, shorter planning horizon
  - Output: functionality to be implemented in the next iteration

Requirements in Scrum

- In Scrum, requirements are defined as user stories
- INVEST properties
  - I – independent
  - N – negotiable
  - V – valuable
  - E – estimatable
  - S – small
  - T – testable

[Pichler, 2008, p. 44]

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

Effort estimation in Scrum with "Planning Poker"

- Activities
  - 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
  - A new vote is conducted
  - Moderator asks if the most occurring value or the average value is acceptable
  - If not, another round is played
  - The moderator decides the size of the backlog item and notes it in the product backlog
  - The game ends if all backlog items are sized or time is over

Post-Planning-Poker-Activities

- Log Estimations
- Assign Stories to Developer(s)
- Break Down Stories into Tasks and fill your Scrum Board
- Implement the stories task by task

- Done and sprint ain't over?
  - Help your Teammates
  - Refactor, Write Tests, Document
  - Ask the Product Owner for more work

Now let’s get started!

- Assignment:
  - Come up with a proper name for our project
  - Document it within the wiki
  - We’ll vote at the end of the project