HPI Hasso Plattner Institut

IT Systems Engineering | Universität Potsdam

Software Engineering 2 (SWT2)

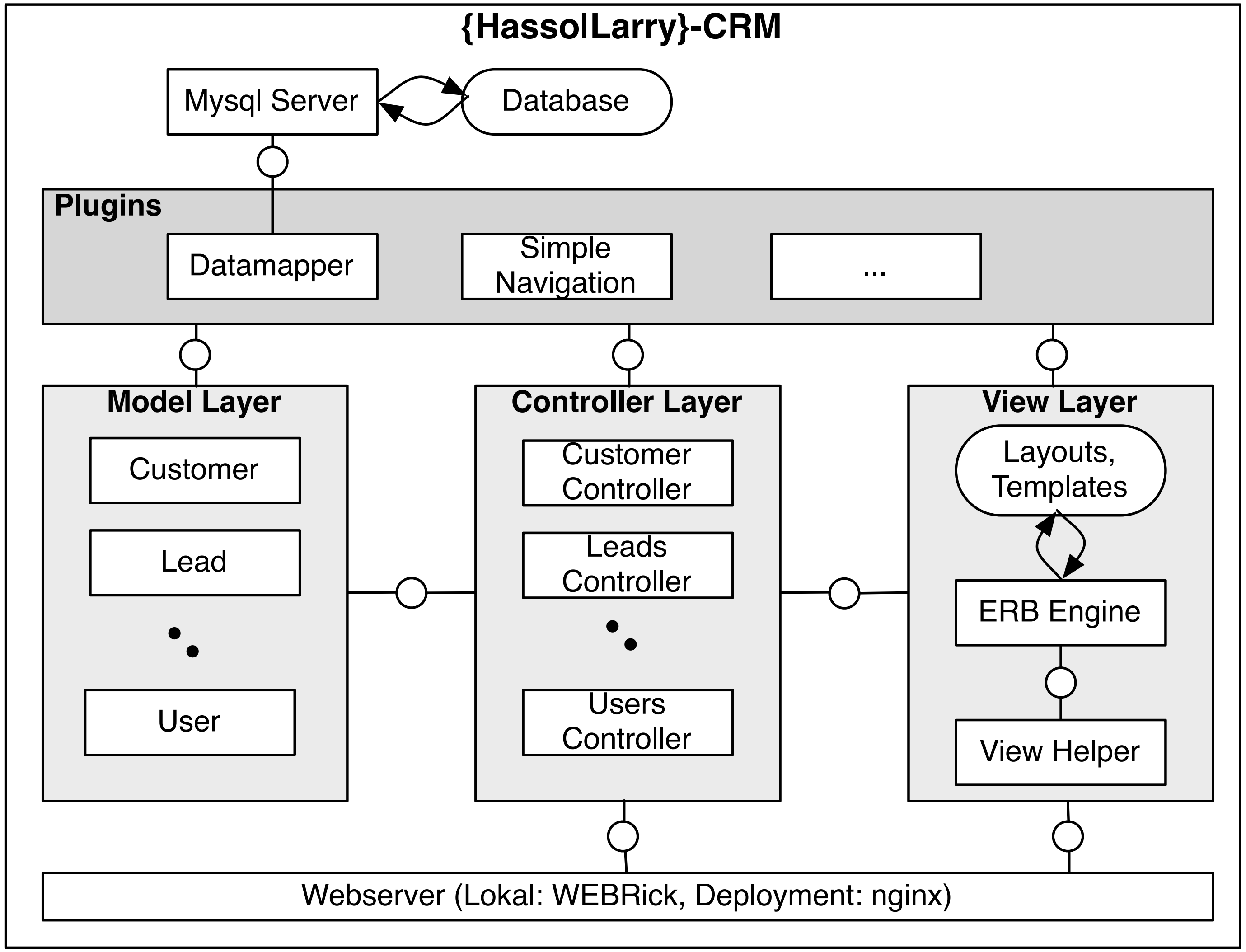
Chapter 4:

Development Process & Collaboration Infrastructure

Agenda: Process & Infrastructure

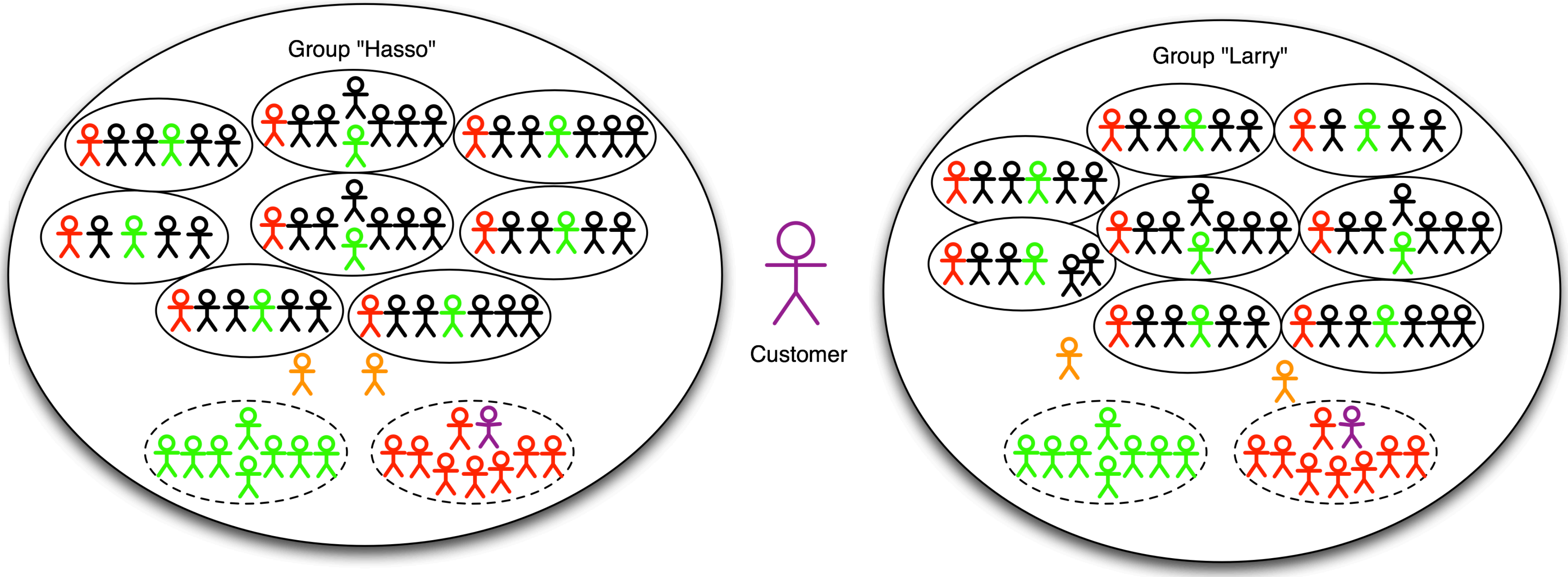
- Architecture Overview
- Development Process for the project (Scaling SCRUM)
- Collaboration Infrastructure
  - Communication & Coordination (Email, Calendar)
  - Application Lifecycle Management System (Agilo)
  - Continuous Integration (Hudson)
- Version Control
  - Central vs. Distributed Version Control Systems
  - A GIT Workflow

Architecture Overview



Recap: High-level Overview of SWT2

Software development in the large



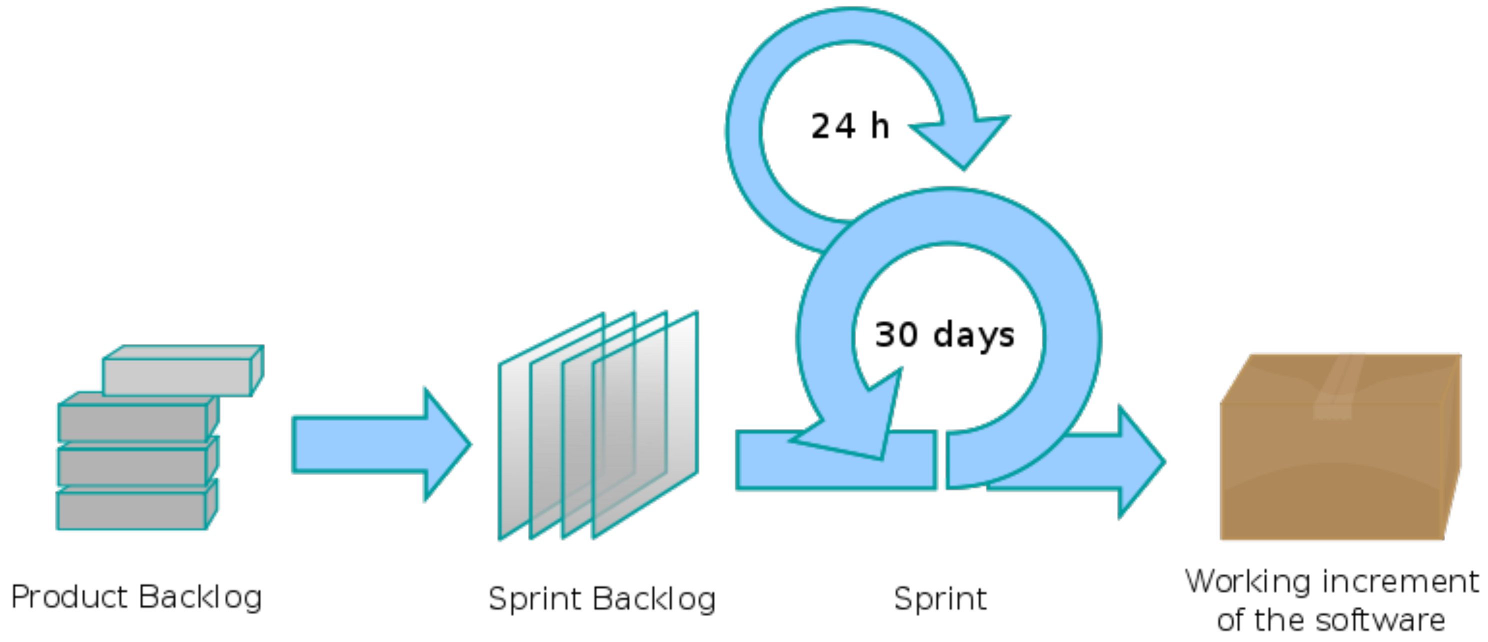
Implications of the Setup

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

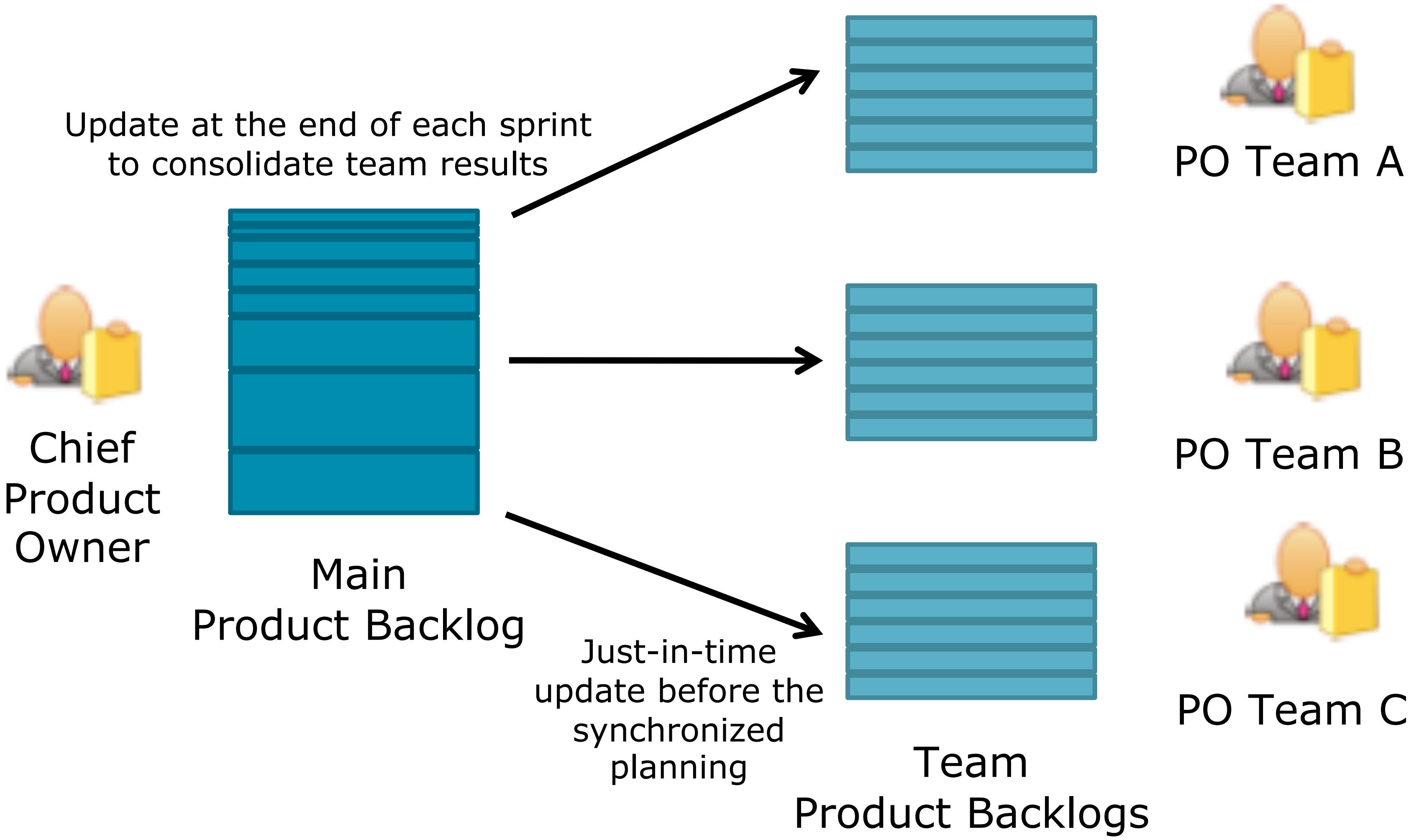


Question: How to scale this to multiple teams?

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

Scaling Scrum: Product Owner Hierarchy


Update at the end of each sprint to consolidate team results
Chief Product Owner
Main Product Backlog
Just-in-time update before the synchronized planning
Team Product Backlogs
PO Team A
PO Team B
PO Team C

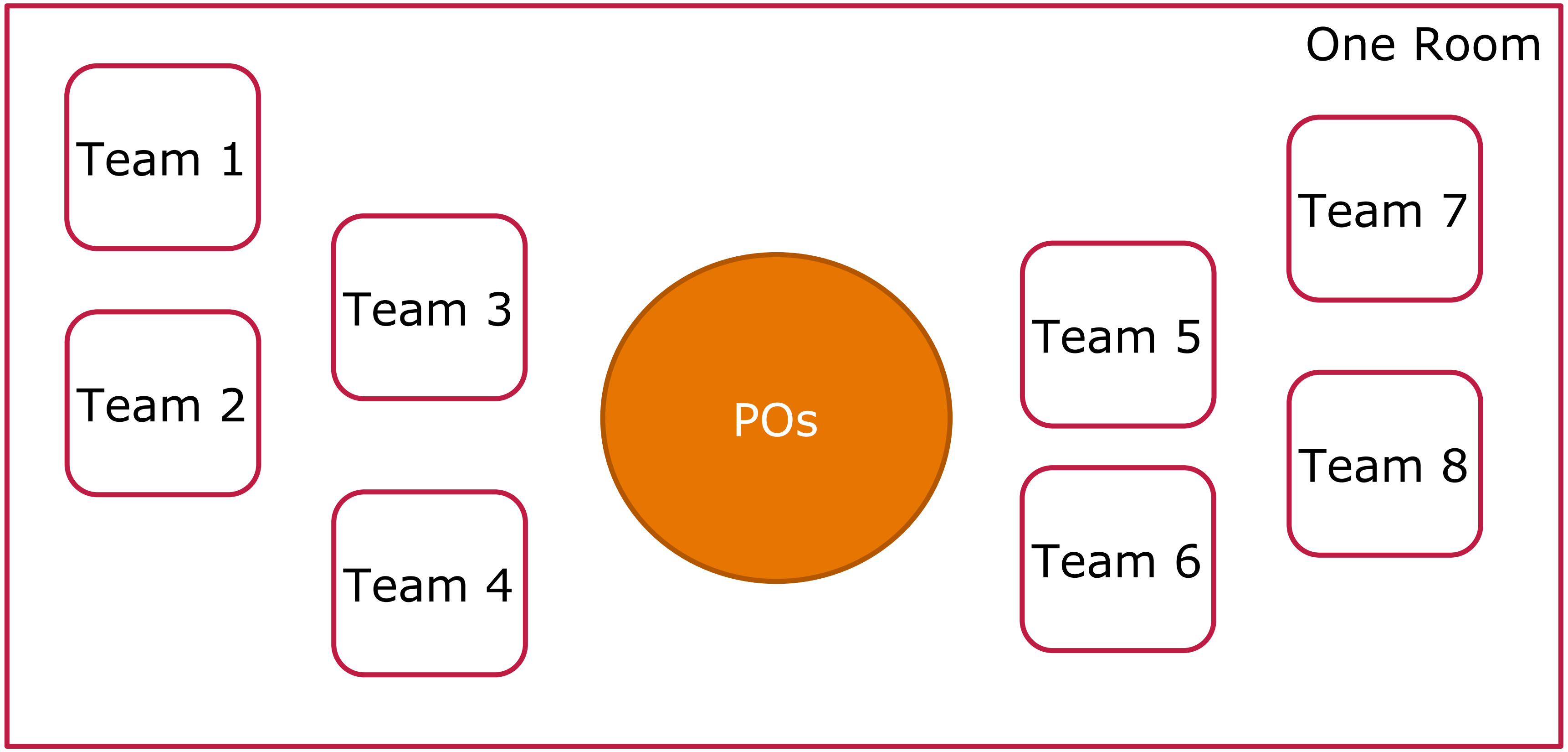
[Christoph Mathis, Scrum Center]

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 deliverred)
- 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



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!

Scaling Scrum: Implications

Multi-Team setups require thorough planning, structured processes, and a working infrastructure for collaboration

Communication Infrastructure

- Email lists
  - Separate lists for each group and team
  - Keep your teammates in the loop
  - Rules and filters are your friends here
  - Anonymous address
- Trac Wiki
  - Integrated within the ALM solution
  - Lean documentation
- Facebook page
- … be creative! (but let us know)

Collaboration Infrastructure

- Tracking
  - Responsibilities
  - Bugs
  - Effort
  - Appointments
- Testing
  - Functionality
  - Build process
  - Code quality
- Sharing
  - Code
  - Documents

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, Plan.io
  - Our tool: Agilo (http://www.agile42.com)

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

Code Metrics

- Measured code complexity with Flog
- http://ruby.sadi.st/Flog.html

"Flog shows you the most torturous code you wrote. The more painful the code, the higher the score."

- Example input class and report

```
class Test
  def blah
    a = eval "1+1"
    if a == 2 then
      puts "yay"
    end
  end
end
```

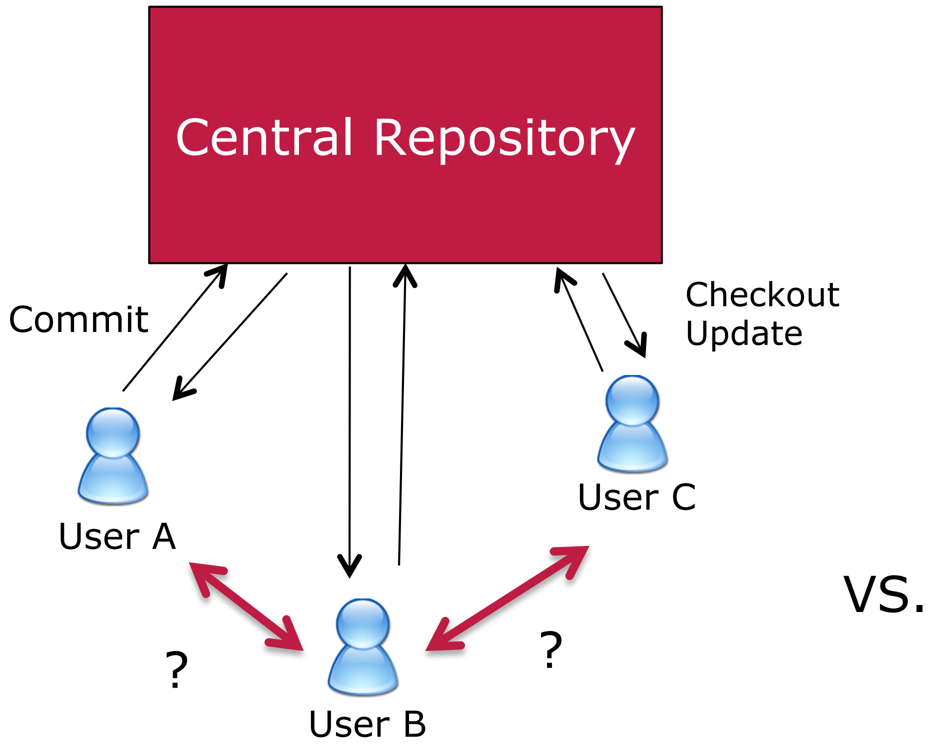
```
Test#blah: (11.2)
  6.0: eval
  1.2: branch
  1.2: ==
  1.2: puts
  1.2: assignment
  0.4: lit_fixnum
```

- Other Ruby complexity tools: Roodi, Saikuro
- Basic Information also available (LoC, No. of classes, etc.)

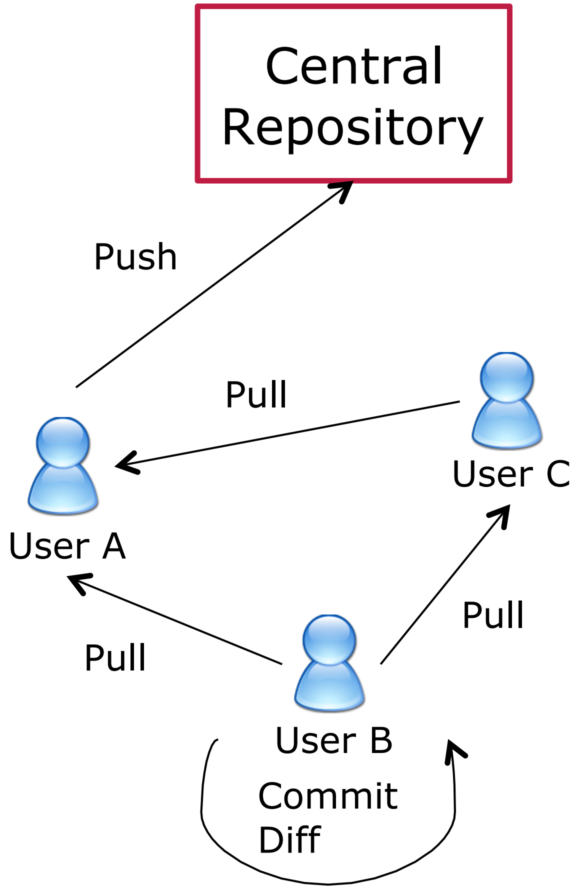
Version Control Systems

- 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



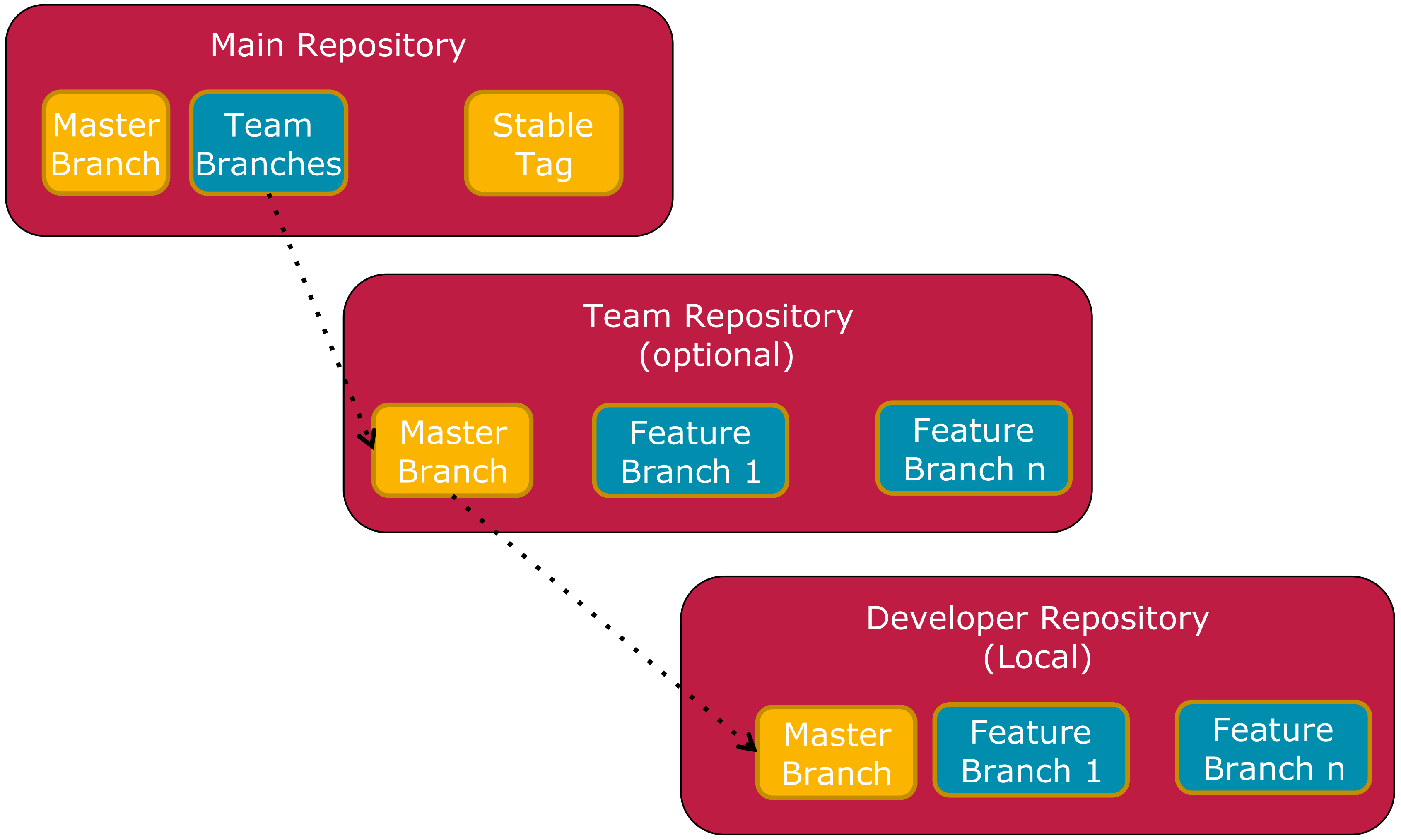
VS.



GIT

- Developed in 2005 by Linus Torvalds for managing the source code of the linux kernel
- Non-linear development
- No central server required
- Cryptographic security of project history
- Foundation for various useful tools

Project Repository Setup



A Git Workflow for Feature Development

1. Pull from the master repository to update your local master

   $> git pull origin master

2. Checkout a branch for the new feature

   $> git checkout –b **12**-add-authors

3. Work on the branch with frequent commits

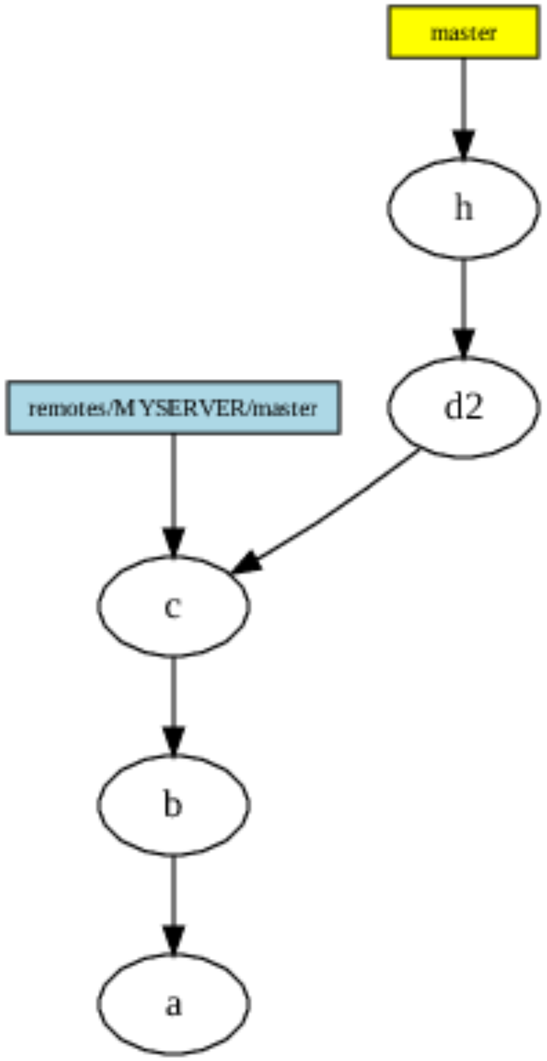
Git Workflow: continued

4. Rebase against the master branch
```
$> git fetch origin master
$> git rebase origin/master
```

5. Clean up your branch history
```
$> git rebase –i origin/master
```

6. Merge your changes to the master
```
$> git checkout master
$> git merge 12-add-authors
```

7. Push your changes upstream
   4. `$> git push origin master`



Git Workflow: A Shortcut

Git extension for working with such branching models

- Installation from source, through macports, or using an installer
- Creates new commands for git
- Resulting Workflow:
  1. $> git flow init (only required once)
  2. $> git flow feature start 12-add-authors
  3. Do your work
  4. $> git flow feature finish 12-add-authors

Literature

- General literature
  - Swicegood, T.: Pragmatic Guide to Git, 2010 (→ Git)

Useful Links

- http://blog.docx.org/2009/08/19/google-kalender-in-outlook-einbinden/
- http://www.google.com/support/calendar/bin/answer.py?hl=en&answer=99358#ical
- http://www.agile42.com
- http://hudson-ci.org/
- http://eagain.net/articles/git-for-computer-scientists/
- http://reinh.com/blog/2009/03/02/a-git-workflow-for-agile-teams.html
- http://tbaggery.com/2008/04/19/a-note-about-git-commit-messages.html
- http://jeffkreeftmeijer.com/2010/why-arent-you-using-git-flow/
- http://github.com/nvie/gitflow

Next Week: Scrum Test Drive

- Due til **Tuesday 3pm:** Select your ScrumMaster

- Lego Exercise ☺

- 11:00 – 12:30: Group Hasso
- 13:30 – 15:00: Group Larry
- The other group meanwhile gets to talk about the architecture and is provided with a GIT introduction

Thank you for
your time!