Software Engineering Practical Training

MINGJIE LI,   TSINGHUA UNIVERSITY

DIRECTOR：DR. XIAOYING BAI

Outlines

Course Design

Automation Evolution

We Measure

Course Design

Organization

Last term, there were

- 11 projects, each with a TA and a customer representative
- 33 teams
- 141 students

Projects come from

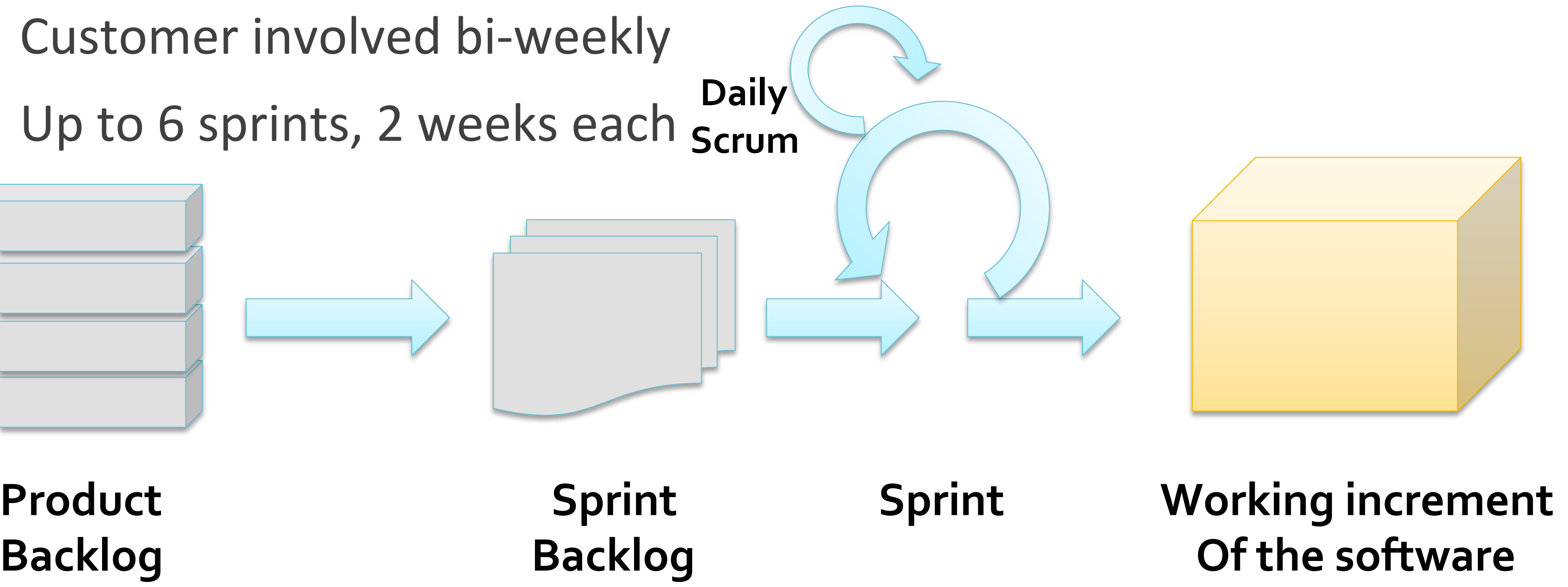
- Other courses, a 32-bit CPU with FPGA
- Campus customers, such as an online office system
- Student associations, such as a platform for AI competitions
- Industrial customers, such as a game using face recognition

Agile Process

Team meeting weekly

Customer involved bi-weekly

Up to 6 sprints, 2 weeks each



Automation Evolution

Two years ago

Two platform

- Team project
- Test

homework for test

- Design test data only for a given scenario
- Implement test cases for a given function
- https://github.com/xin-xinhanggao/railgun

However,

- Test is used to ensure the software quality
- We should monitor the test result

Continuous Integration

Here is the pipeline

- git commit -> git push -> Jenkins build -> SonarQube

In Jenkins build

- Call a build script with unit tests, which is written by students
- Create coverage reports under given folder
- Collect reports

Continuous Deployment

It's tricky to make a system run……

- Version, dependencies, configurations

How to make your work run on TA's computer?

- Submit a virtual machine?

Docker comes

- Build the running environment with Dockerfile
- List all requirements at the same time

Deploy in Kubernetes. To be applied in the next semester

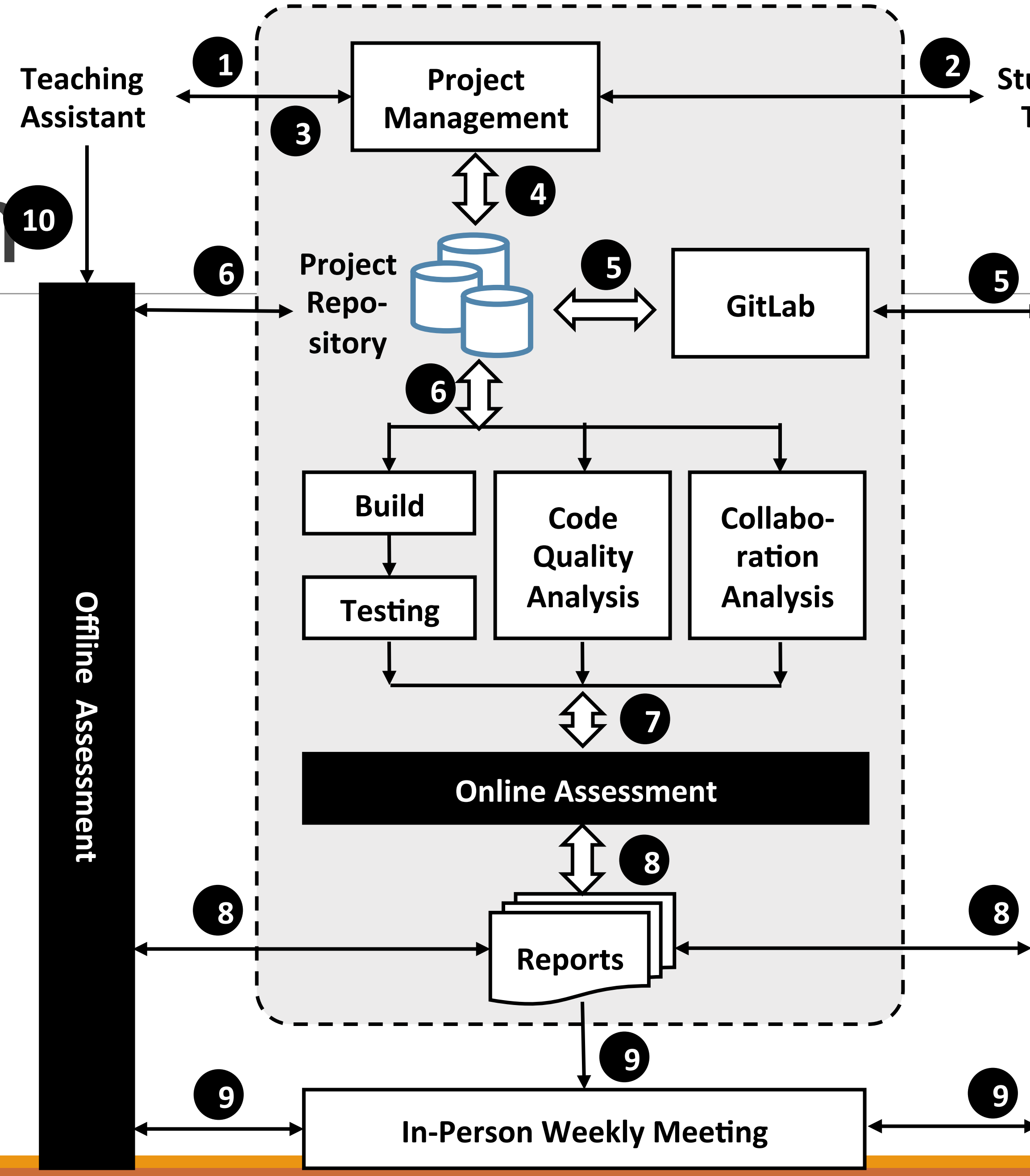
DevOps Platform

Version Control
- GitLab

Source Analysis
- SonarQube

Continuous Integration
- GitLab CI
- Jenkins last year

Issue Tracker
- GitLab



We Measure

FOR MONITOR, FOR FEEDBACK

Task

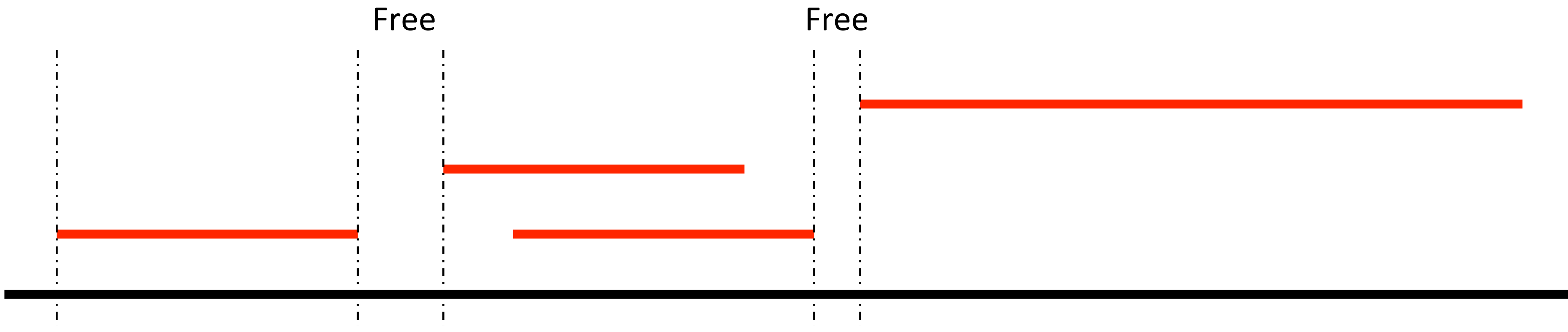
We regard GitLab Issue as an online task recorder

- Task assignment
- Effort estimation and record

$$PROG = max\{\frac{nT - \sum_{i=1}^{n}(SLACK_i + DELAY_i}{nT}$$

Count invalid interval

- Free time over 3 days, issue not closed over 2 weeks



Contribution

Take commit as one's contribution

- Based on the observation of previous repos, we judge one commit by
  - Modification grain
  - Message length
  - Frequency
- Is there free riding?
  - Standard deviation
  - Divided by the mean to make it comparable

$$COMMIT_i = \sum_{j=1}^{c_i} mod_{i,j} \times msg_{i,j} \times freq_{i,j}$$

Cooperation

Branch Pattern

- Appleton .et .al concluded some patterns in 1998

Cooperation

Merge Often

- Branches shall be merged each sprint

$$\begin{tikzcd}
 & & C_1 \arrow[r, dashed] & C_2 \arrow[rrr, dashed] \arrow[dr] & & & C_3 \\
B_1 \arrow[r] & B_2 \arrow[ur] \arrow[r] \arrow[dr] & B_3 \arrow[r, dashed] & B_4 \arrow[r] & B_5 \arrow[r] & B_6 \arrow[r, dashed] & B_7 \\
 & & D_1 \arrow[r, dashed] & D_2 \arrow[rrr, dashed] \arrow[urr] & & & D_3
\end{tikzcd}$$

Cooperation

Merge Often

Early Branching

- Encourage branch
- Group commits in one branch by their relativities

$$\begin{tikzcd}
C_1 \arrow[d] & & & & & & & & \\
C_2 \arrow[d] & & C_1 \arrow[d, -] & & & C_3 \arrow[dl, -] \arrow[dr, -] & & & \\
\vdots \arrow[d] & \Longrightarrow & C_2 & & C_4 \arrow[rr, -] \arrow[dl, -] & & C_5 & \cdots & C_n \\
C_n & & & C_6 & & & & &
\end{tikzcd}$$

Cooperation

Merge Often

Early Branching

Merge Your Own Code

- Limit the author of merge commits to be one of those of parent commits

References

1. Lukas Alperowitz, Dora Dzvonyar, and Bernd Bruegge. Metrics in agile project courses. In Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16. ACM Press, 2016.
2. Brad Appleton, Stephen P. Berczuk, Ralph Cabrera, and Robert Orenstein. Streamed lines: Branching patterns for parallel software development. PLoP, 1998.

Thank you for listening

- Continuous Delivery of Personalized Assessment and Feedback in Agile Software Engineering Projects. ICSE-SEET'18: 40th International Conference on Software Engineering: Software Engineering Education and Training Track. https://doi.org/10.1145/3183377.3183387