Project Kickoff & Project Infrastructure

Software Engineering II
WS 2020/21

Enterprise Platform and Integration Concepts
Let’s Get Started

Project
- **Sprint 1** (2 weeks, 27.11.20—11.12.20)
  - Start with **Sprint Planning** meeting
  - Sprint 1 **Review & Retrospective Meetings** at the end
- **Weekly Stand-ups** in weeks with no other Scrum meeting
  - Short, timeboxed. 2 minutes per person.

Meetings
- Exact meeting dates **negotiated** with tutors
- Sprint Review & Planning for next sprint can be collocated
- **On demand:**
  - User research with customer. Clarify questions!
  - Coordination with other SMs, POs, devs, teams
Let’s Get Started

POs (reminders)
- Extract requirements + create user stories (GitHub tickets)
- Get an idea of the interaction workflows in the system (mockups?)
- Prepare Sprint Planning, inform yourselves on what the team is doing
- PO should roughly know what the team has done before the review

Developers
- Clone repository, get application working locally, read & run tests
- Understand architecture and dependencies
  - If you were lead architect, how would you construct this?
  - Greatest challenges in the problem domain? What libraries could help?
- Play around and try things out.
  - Where does the system have problems? What makes no sense to you?
Let’s Get Started

SMs
- Part of your job is research and retrospection
  - Observe the **meta-level** of a meeting
    - Equal participation? How is the team communicating?
    - What is working well in the team? What isn't?
  - How can team meetings be structured? Prepare agenda
- This is a hard job, **focus on it**
- Every team is different. **Experiment!**

Meeting spaces
- Regular meeting + work timeslot
- We can reserve spaces in the Villa at the EPIC chair, if this is needed
Communication Infrastructure

- Large choice of options
  - Probably best to focus on a few key ones
  - *Keep information together*, build a knowledge base

- We've setup some course communication infrastructure
  - See the website for the links
  - Telephone and personal contact for direct communication?

- ... be **creative**!
  (but let us know, we're interested in learning what might be useful in the future)
The Swiss army knife of software development

- Integrating tools for most common activities in one place
- Wiki, bug tracking, time management, project analytics, discussions, ...

Examples:
- Microsoft Team Foundation Server
- Jira
- Redmine
- Gitlab
- GitHub
Version Control System

Repository to store and organize development artifacts

Features
■ Versioning
■ Dealing with variants: branches
  □ main & dev
  □ Pull Requests
■ Access control
  □ Authentication, authorization
  □ Concurrent development
■ Reporting and communication
  □ How many versions, variants, changes, persons
  □ History of changes
How do you make sure your software *always works*?

**Continuous Integration**
- **Connected** to version control
  - Run tests when code changes
  - Ideally covering all development branches
- **Display errors & notify**
- Customizable run scripts
- **Examples:**
  - GitHub Actions
  - Jenkins/Hudson
  - Travis CI
Example: GitHub Actions
Example: GitHub Actions

```yaml
name: CI/CD

on:
push:
  branches: [ main, dev ]
pull_request:
  branches: [ main, dev ]

jobs:
  # Label of the runner job
  CI:
    # You must use a Linux environment when using service containers or container jobs
    runs-on: ubuntu-latest

  # https://docs.github.com/en/free-pro-team@latest/actions/guides/creating-postgresql-service-containers
  # Service containers to run with `CI`
  services:
    postgres:
      # Docker Hub image
      image: postgres
```
How can you always have a running version of the application available? (why would you want to?)

Deploy the application
- Simple: test deployment on local machine
- Deployment on separate machine:
  - Dedicated Servers
  - Hosted and managed by a (paid) third party
- Continuous Deployment:
  - Deployment automatically triggered by successful CI build
  - Deployment config is part of the project
  - No extra effort
How can you ensure that the software adheres to certain quality standards (complexity, test coverage, etc.)

Check for compliance

■ Review your own code (diff), code reviews by others

■ **Automatic** checks
  □ Hosted tools: e.g. CodeClimate, Codefactor, Codebeat
  □ Local code coverage: SimpleCov (http://www.simplecov.org/)
    – Can run automatically during each test run
    – coverage/index.html in your application folder
  □ Local code smells: RuboCop (https://www.rubocop.org)
Example: CodeFactor
Example: RuboCop

```
spec/views/notes/edit.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "renders the edit note form" do

  before(:each) do

spec/views/notes/index.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [2/1].
  it "renders a list of notes" do

spec/views/notes/index.html.erb_spec.rb:10:5: C: Style/For: Prefer each over for.
  for note in @notes do ...

  end

  before(:each) do

spec/views/notes/new.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "renders new note form" do

  before(:each) do

spec/views/notes/show.html.erb_spec.rb:8:3: C: RSpec/MultipleExpectations: Example has too many expectations [3/1].
  it "shows the details of a note" do

52 files inspected, 48 offenses detected, 38 offenses auto-correctable

connections-portal git:(dev)
```
Example: SimpleCov

```
Models (100.0% covered at 1.0 hits/line)

3 files in total.
10 relevant lines, 10 lines covered and 0 lines missed. (100.0%)

<table>
<thead>
<tr>
<th>File</th>
<th>% covered</th>
<th>Lines</th>
<th>Relevant Lines</th>
<th>Lines covered</th>
<th>Lines missed</th>
<th>Avg. Hits / Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>app/models/application_record.rb</td>
<td>100.00%</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>app/models/note.rb</td>
<td>100.00%</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>1.00</td>
</tr>
<tr>
<td>app/models/user.rb</td>
<td>100.00%</td>
<td>10</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1.00</td>
</tr>
</tbody>
</table>
```

Generated by simplecov v0.19.1 and simplecov-html v0.12.3 using RSpec

November 26, 2020
More than likely someone else has already solved a specific (web-dev) problem.

Libraries (Ruby gems) & external dependencies

- Most likely more mature and bug-free than your custom solution
- Someone else has checked the code in your application
- You are maximizing development time
- **But:** dependencies introduce complexity
  - Mostly very powerful, generic solutions
  - Require extensive configuration
  - Need to be learned by every developer (effort multiplied by X)
  - Consensus among developers on usage?
Your Project

Communication infrastructure
Continuous Integration
Continuous Deployment
Code Quality
Dependencies

**Any other tools you might want to use!**

*Something you have had good experiences with in the past?*  
*Your favorite development tool?*  
But, your team is not the only one using it, communicate.