

# Scrum and AMOS

---

Dirk Riehle, Univ. Erlangen

**AMOS B02**

Licensed under [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/)

# Reminder on Commits to Repository

Don't forget to sign-off and declare your co-authors, if any [1]

```
dirk@host$ git commit -a -m "Fixed problem  
> Co-authored-by: Stefan Buchner <stefan.buchner@fau.de>"  
> --signoff
```

[1] For more details, please see the slide deck AMOS B01 on Team and Tools

# Agenda

---

1. The AMOS process
2. The team meeting
  - Meeting preparation
  - Sprint review
  - Sprint release
  - Sprint retrospective
  - Sprint planning
3. Bill of materials
4. Software architecture

# **1. The AMOS Process**

---

# Scrum in Student Projects

---

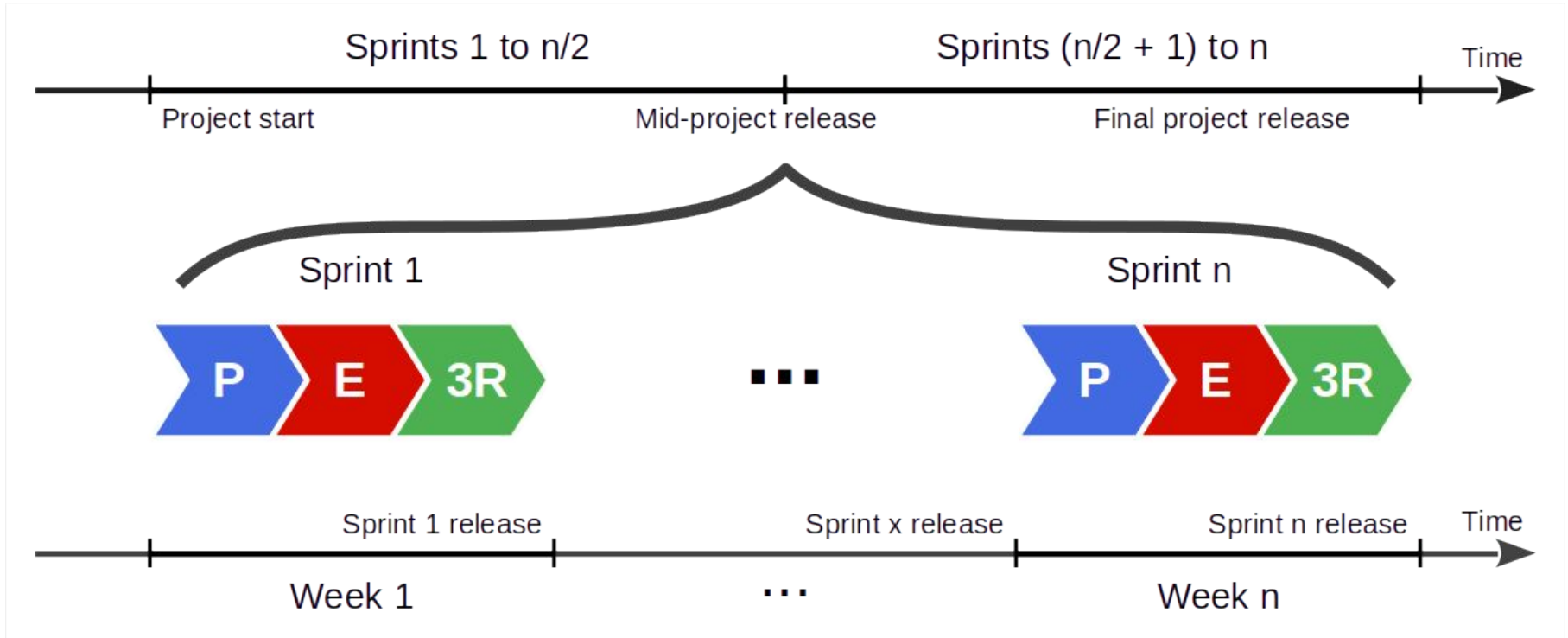
## Challenges

- Widely differing abilities and experiences
- Not 100% on project, but in multiple courses
- Transient rather than persistent teams
- Not available at same place, not at same time
- Sometimes extrinsically motivated (grades)

## Solutions

- Context-aware instantiation of framework
- Supporting teaching team, coaching

# Overall AMOS Project Timeline



# Time-boxed Sequence of Releases

---

**A release is**

- A named identifiable, consistent, and useful snapshot of the product

**A sprint release is**

- A release used to gather feedback from the industry partner to steer the project

**A project release is**

- A release that is deployed to production where it is supposed to perform its job

In the AMOS Project there are two releases (mid-project and final release)

# Project Schedule

---

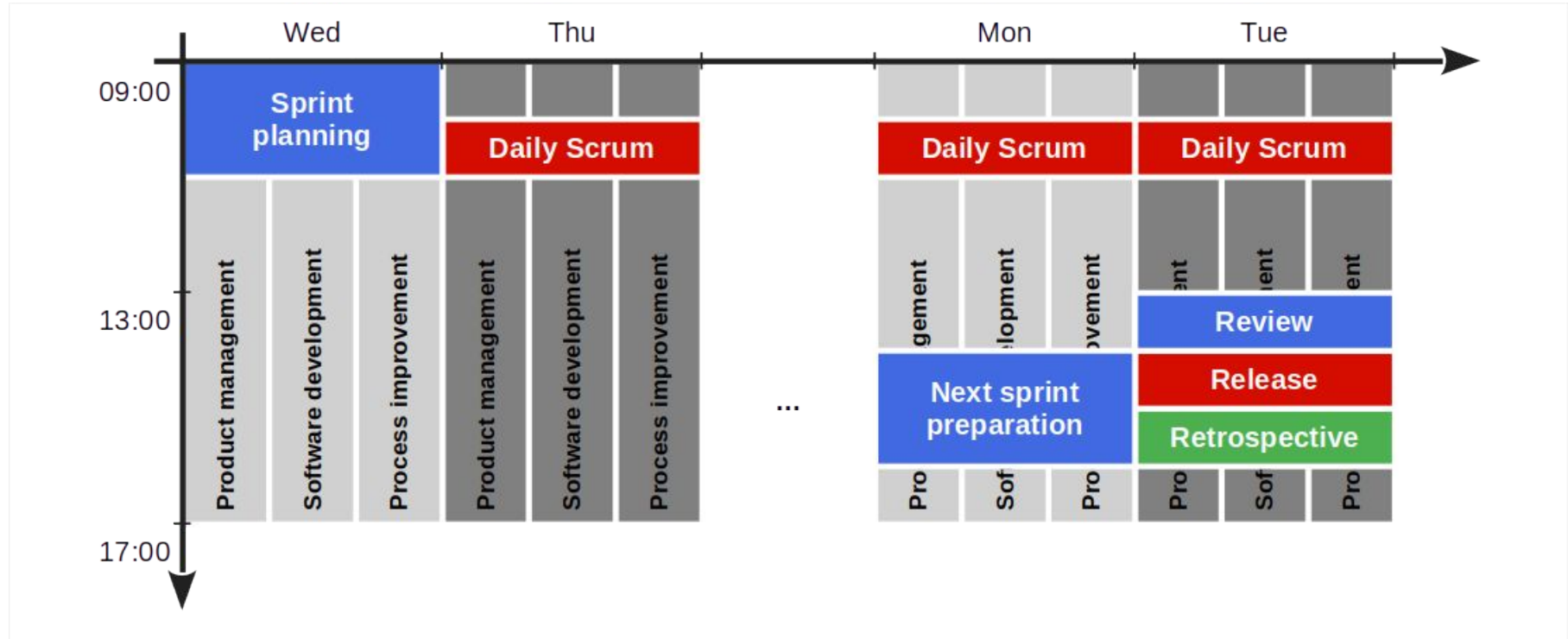
Please see the **Schedule** tab of the **Course Organization** doc



## **2. The Team Meeting**

---

# Logical Structure of a Scrum Sprint



# The AMOS Team Meeting



# 1a. Meeting Preparation

## Product owner

- Performs backlog grooming in a **next sprint preparation** meeting
  - Should include at least one developer (may want to plan this out)
- Ensures that the product backlog is ready for sprint planning
  - There are enough high-quality entries at least for the upcoming sprint
    - High-quality = meets INVEST criteria, explained later
  - Product backlog entries may be
    - New features, bug fixes, and refactorings

# 1b. Meeting Preparation

---

## Release manager

- Ensures that a working demo system will be available
- Tags release candidate with **sprint-xx-release-candidate**
  - Where xx is your sprint number (see deliverables through project schedule)

# Tagging Release Candidates and Releases

## Release candidate

Releases

Tags

Choose a tag

Target: main

Choose an existing tag, or create a new tag when you publish this release.

sprint-07-release-candidate

Write

Preview

H

B

I

↵

<>

🔗

☰

☷

☰

🌐

🔗

↶

Generate release notes

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

↓ Attach binaries by dropping them here or selecting them.

☒ Set as a pre-release

This release will be labeled as non-production ready

☐ Set as the latest release

This release will be labeled as the latest for this repository.

Publish release

Save draft

## Release [1]

Releases

Tags

Choose a tag

Target: main

Choose an existing tag, or create a new tag when you publish this release.

sprint-07-release

Write

Preview

H

B

I

↵

<>

🔗

☰

☷

☰

🌐

🔗

↶

Generate release notes

Describe this release

Attach files by dragging & dropping, selecting or pasting them.

↓ Attach binaries by dropping them here or selecting them.

☐ Set as a pre-release

This release will be labeled as non-production ready

☒ Set as the latest release

This release will be labeled as the latest for this repository.

Publish release

Save draft

[1] Do not rename release candidate

## 2a. Sprint Review

---

### Release manager

- Checks-out fresh code base using release candidate tag
- Compiles, builds, and runs tests for release candidate
- Deploys release candidate to test environment

The release manager does not run the review

## 2b. Sprint Review

### Product owner

- Walks through “Awaiting review” column item by item
  - Asks developer to demo item under review
    - Insists that developer shows, not just tells
  - Checks fulfillment of acceptance criteria
  - Checks fulfillment of definition of done, if required
  - Checks other criteria incl. logging output for problems
  - If successfully implemented
    - Moves item to feature archive
    - Asks developers about real size, add it to the item
  - If not successfully implemented
    - Moves item back to product backlog



## 2c. Sprint Review

---

### Software developer (individually)

- Is called upon by product owner for backlog item
  - Demos backlog item as requested by product owner
  - Answers questions about item design and implementation
  - Provides real size as determined after implementation

Only talking, not showing, is not acceptable

- The product manager needs to insist on showing not just talking
- If a developer only talks, product owner and developer failed

# 3a. Sprint Release

---

## Product owner

- Decides whether release candidate should be released
  - Only in case of significant regression should you not release
  - Later in the course you will use a definition of done
  - Specifics depend on type of release
- Consults with software developers if necessary

## 3b. Sprint Release

---

### Release manager

- If the release candidate is to be released
  - Tags release with **sprint-xx-release** where xx is your sprint number
  - Deploys sprint release to operations environment
- If there is a change log (optional)
  - Updates change log with release information

# 4a. Sprint Retrospective

## Scrum Master

- Reviews this sprint's impediments and improvements
  - Reports on progress
  - Reviews remaining problems
- Performs roll call, asks everyone individually:
  - What has gone well?
  - What hasn't gone well?
  - What can we do better?
- Puts new impediments and improvements into imp-squared backlog

## 4b. Sprint Retrospective

---

### Everyone

- Answers to happiness index

# 5a. Sprint Planning

---

## Product owner

- Reprioritizes product backlog items, if necessary, on-the-fly
- Works through top-prioritized backlog items one-by-one until finished
  - For each product backlog item, explains it, asks developers to estimate and commit
  - You are finished, if the team does not want to take on more backlog items

## 5b. Sprint Planning

---

### Software developers (as team)

- Estimate size of each backlog item using planning poker
- After planning, commit to backlog items in sprint backlog

# Story Points

## Story points

- Is an arbitrary numeric measure of size of a given backlog item

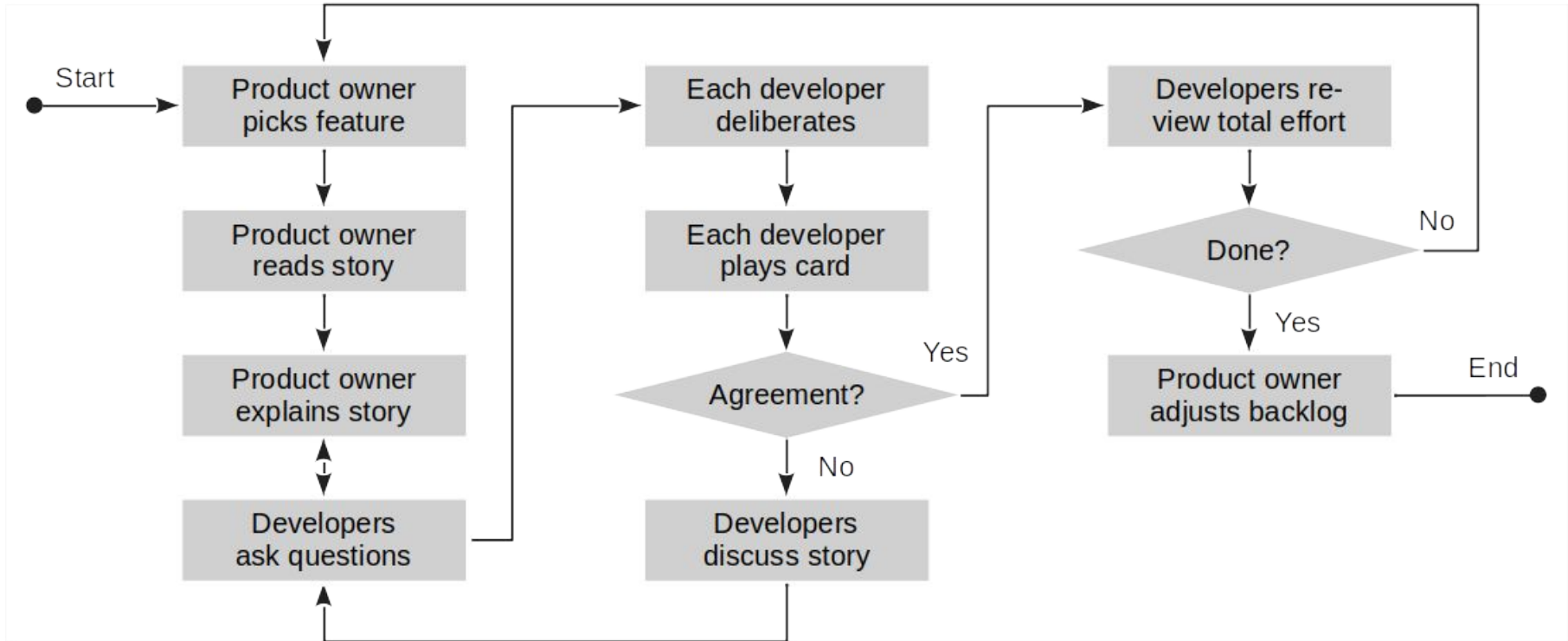
## Properties

- Is a measure of size, not of effort or duration
- Measured in non-linear increments, forcing choice
- Is socially agreed upon, depends on team estimation history
- Is independent of a particular person (and their skills)
- Is mapped to time using the team's velocity (development speed)

Points	Meaning
0	No size
1	Trivial size
2	Small size
3	Medium size
5	Large size
8	Very large size
13	Too large (size)



# Sprint Planning with Planning Poker [1]



# 6a. Meeting After-work

---

## Product owner

- Updates planning document to consistent state
  - Cleans up product and sprint backlog
  - Ensures feature archive is current

## 6b. Meeting After-work

---

### Software developer (as team)

- Plan programming tasks (1 feature = 1+ tasks)
  - Agree on which developer(s) work(s) on which tasks
  - If pair programming, ensure you document the pair

## 6c. Meeting After-work

---

### Scrum Master

- Works on impediments and improvements during sprint
- Documents resolutions in imp-squared backlog

# An Efficient Team Meeting Takes 90 Min. (or Less)

#	Section	Duration
1	Meeting preparation	-
2	Sprint review	~35%
3	Sprint release	~5%
4	Sprint retrospective	~20%
5	Sprint planning	~40%
6	Meeting after-work	-

### **3. Bill of Materials**

---

# Bill of Materials [1]

---

A **bill of materials** (of some artifact) is

- A linear list of materials (the parts) constituting the artifact (the whole)

A bill of materials can contain any kind of material

- Not just software

If purely software, the bill of materials is also called the

- Software bill of materials (SBOM)

# Software Bill of Materials (SBOM)

For each dependency, provide this (recommended, not required) information

Field	Name	Example
1	Context	com.google.code.gson
2	Name	gson
3	Version	2.3.1
4	License	Apache-2.0
5	Comment (optional)	Pulled from Maven Central



# Regular Deliverable: Software Bill of Materials

---

Please initialize your software bill of materials and keep it up-to-date

- You can limit this to your first-level dependencies

You can use a tool, e.g. a build tool plugin to generate the SBOM

Please update every time you change your dependencies

## **4. Software Architecture**

---

# Agile Architecture?!

Agile methods eschew detailed planning

- The proof of the software is in the feedback of the customer

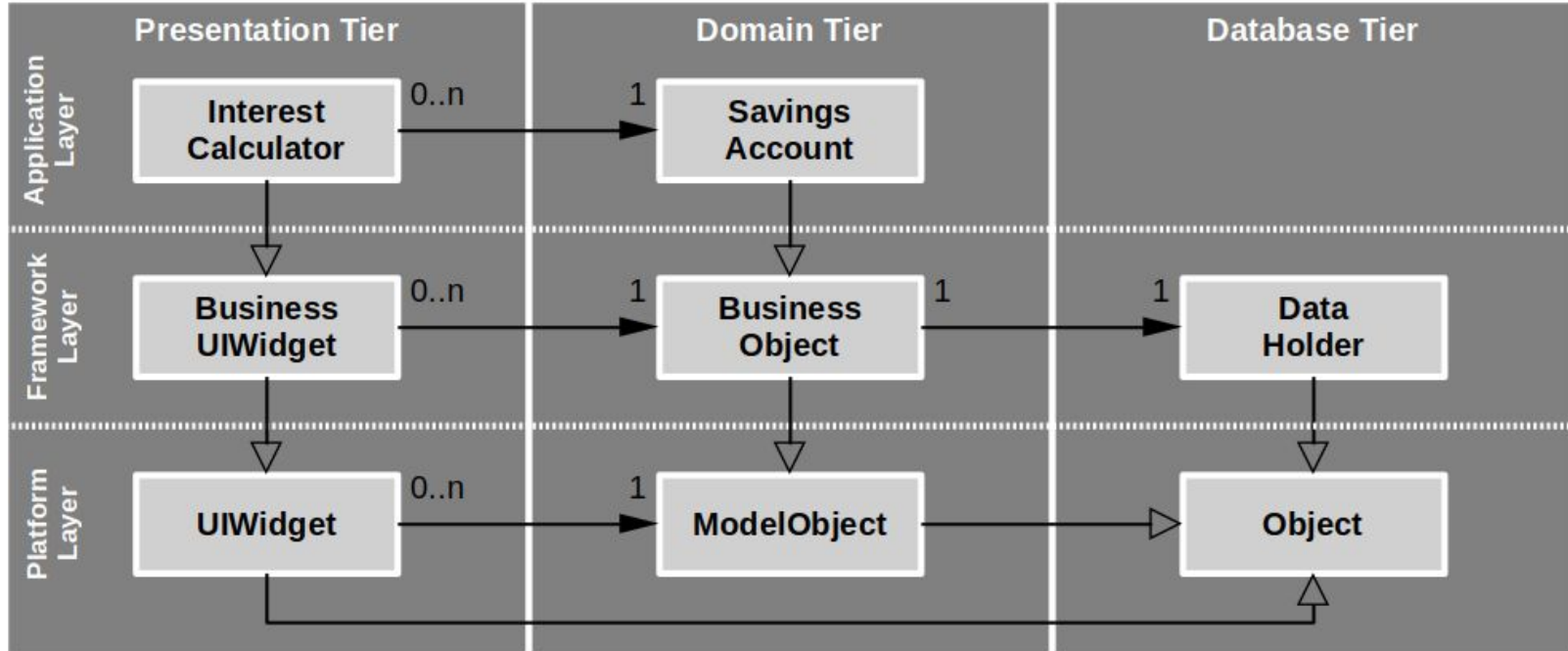
Software architecture is the overall design of a system

- Including static (structural) and dynamic aspects
- Covering everything of wide impact to the system
- Ignoring everything with limited (localized) impact

Agile software architecture is software architecture that

- Emerges from risk-adjusted planning / visibility
- Delays architectural investment to the last minute

# Runtime Objects / Tiers vs. Code / Layered Architecture



# One-time Deliverable: Architecture Description

---

Provide a description of the initial planned architecture including (at a minimum)

1. The runtime architecture
2. The code (static) architecture
3. The tech stack you are building on

Feel free at the end of the project to review planning with reality

# Summary

---

1. The AMOS process
2. The team meeting
  - Meeting preparation
  - Sprint review
  - Sprint release
  - Sprint retrospective
  - Sprint planning
3. Bill of materials
4. Software architecture

# Thank you! Any questions?

---

[dirk.riehle@fau.de](mailto:dirk.riehle@fau.de) – <https://oss.cs.fau.de>

[dirk@riehle.org](mailto:dirk@riehle.org) – <https://dirkriehle.com> – [@dirkriehle](#)

# Legal Notices

---

## License

- Licensed under the [CC BY 4.0 International](https://creativecommons.org/licenses/by/4.0/) license

## Copyright

- © Copyright 2009, 2024 Dirk Riehle, some rights reserved