JOHANNES KEPLER
UNIVERSITY LINZ

Institut fuer Wirtschatsinformatik
Software Engineering

Praktikum Software Engineering

Antonio Garmendia

Unit 0 - Introduction & Preliminary Discussion

Agenda

- Introduction
- Grouping
- Evaluation
- Tools
- Task assigment for Workshop on 17.03

Goal of the Internship

- **Development of an application in a team**
  - Specify, plan and design a software product
  - Object-oriented programming and Testing (Unit tests & Code quality)
  - Teamwork
  - Application of SE tools
    - Version management (Repositories, GitHub)
    - Project management (GitHub Projects, Zenhub)
    - Build / Continuous Delivery (Maven + CircleCi)
  - Planning of the Sprints and Release Versions
  - Creation of System (Architecture, Code, Test cases, Documentation)

Topic: Digital Twin Application

Development of an Application for Smart Rooms

A team of three developers should implement this project in several sprints over a period of 4 months creating all the necessary artifacts, such as: Software, Tests, Documentation, etc.

- Create, Read, Update and Delete (CRUD operations)
- Database storage solution
- Visualize data + available devices of a room
- Interact with the devices in the room
- Create automation rules

Requirements

- **High-Level Requirements**
- **Programming Language: Java**
- **Technology**
    - Backend: Java
    - Frontend: Swing, JavaFX. It is also possible the development of a web-based application. This is recommended if team members are familiar with web technologies.

Organization

- Working in teams of 3 students
- Tasks should be equally distributed considering the amount of effort
- Effort: 6 ECTS (~ 150 working hours) internship and group appointments included
- LVA-leader is your Client and Advisor
- Recommendation: Completion of the Software Engineering courses (Soft1, Soft2)

Each team member must participate in the implementation of the application – Equally distributed implementation tasks!

Time Schedule

- **The Software Product is being developed in three releases**
    - *Release 1*: April 5. 2022 (12.00 o'clock)
    - *Release 2*: May 10. 2022 (12.00 o'clock)
    - *Release 3*: June 21. 2022 (12.00 o'clock)
    - *Final Product Delivery*: July 7. 2022

- **Submission per Release:** Branch in Git with all the Documentation + Code
- **Final Submission should be uploaded no later than 7. July 2022**

Appointments - Sprint Meetings

- **3 Sprint Planning Meetings**
  - Mandatory attendance of the entire team
  - 10 minutes presentation (Slide-Template)
  - Each member should participate in the presentation
  - Discussion, Status, Next Steps…

- **Three individual appointments (24.03, 28.04 and 26.05) per Team**
  - Feedback & Questions (30 Minutes)

	March				May										June						July
Date	**10/03**	**17/03**	**22/03**	**24/03**	**31/03**	**05/04**	**07/04**	**14/04**	**21/04**	**28/04**	**05/05**	**10/05**	**12/05**	**19/05**	**26/05**	**02/06**	**09/06**	**16/06**	**21/06**	**23/06**	**07/07**
ToDo:	Instructions	Req. Workshop Sprint Planning 1	Sprint Planning 1 Completed in Zenhub	Project Meeting		**R1**	Release 1 Review Sprint Planning 2	Eastern Holidays		Project Meeting		**R2**	Release 2 Review Sprint Planning 3		Project Meeting (Code Review)	Qapture: Digital Twin of the Room			**R3**	Final Sprint Planning, Final Presentation	**Final Release + Final Documenttion**

Agile Software Development

- **Iterative development (Sprints)**
  - 1 Week to max. 1 Month
- **Prioritize a set of requirements, the Team decides which ones must be implemented in each sprint**
- **Result of a Sprint = New version of the product**
- **No dedicated roles in the team**
  - Between 5 and 9 developers per Team
- **High level of self-organization**

Release 1

- **Goal: UI Prototype and OO Design**
- **Deliverables:**
  - First concept for building the application (which Features, Components,..)
  - UML Class Diagram with the most important classes (Class names, Hierarchies, Methodology, Patterns…) with a UML Tool!
  - Entity Relationship Diagram of the database structure
  - UI Prototype
  - Continuos Integration in CircleCI
  - Presentation of the Project Status 1 (for Sprint Planning Meeting)

Release 2

- **Goal: Prototype Implementation and Unit Tests**
- **Deliverables:**
  - Extended/updated UML Diagrams
  - Prototype Implementation:
    - First version of the User Interface
    - Some implemented functionality
  - Unit Tests for individual (important) classes
  - Use Case Description (see Use Case Template)
  - Code Quality Report (The team should present at maximum 2 fixes proposed by a code quality tool)
  - Presentation of the Project Status 2 (for Sprint Planning Meeting)
    - Code Quality Report

Release 3

- **Goal: Documentation**
- **Deliverables:**
  - Extended/updated UML Diagrams
  - Extended Unit Tests
  - Implementation:
    - User Interfaces
    - Implemented most of the functionalities (all Features available)
  - First version of the project documentation
  - Final Code Quality Report (What is the quality of the final code?)
  - Presentation of the Project Status 3 (for Sprint Planning Meeting)
  - Code coverage equal or higher to 85% for all non-UI test code
  - Live Demo/Screencast of the Application

Final Product

- **Deliverables:**
  - Final Project documentation
  - Executable, final version of the application (on Github main branch)
  - Github Documentation (Readme with Installation Instructions, etc.)
  - Javadoc for important classes, Interfaces and Methods

Evaluation

- **The criteria for assessment as follows:**
    - Functionality of the product
    - External Quality of the Product (Stability, Efficiency, User Interface)
    - Internal Quality of the Product (Quality of the design, Programming Quality, API-Documentation)
    - Widespread Unit Tests and Quality of the Unit Tests
    - Quality of the Documentation (Design, Test cases, Experience Report)
    - Presentations

Tools for the Course

- Github Projects, ZenHub
- Git (GitHub)
- Timescale Database
- Maven
- CircleCI
- UML Editor / UI Prototyping Tool
- Code Coverage Library (e.g., JaCoCo)
- Code Quality: Static Code Analyzer
  - Code Quality Analysis with PMD, SonaLint, etc. More info: https://github.com/jku-win-se/teaching.ss22.prse.prwiki.en/blob/master/wiki/code-quality/README.md

Project Organisation with Github Projects/Zenhub

- **Implementation details (detailed specification) in Github Projects**
  - For each release: Requirements, Tasks, Bugs, etc.
  - Assign to each task a responsible and a cost in time! – The responsible must implement the source code (Code + Unit Tests)

- **Create a Release Planning (Roadmap) in Github projects/Zenhub**
  - At the end of each release, the respective tasks, requirements, bugs, etc must be completed and closed.

Source Code Management with Git

- **GitHub to manage Code and Documentation**
  - Code must be committed in Github at least 1 per Week
  - Always enter the respective id for each commit (#TaskNr). – Each team member must write some code and make commits!

- **Quality feedback – The source code must be kept clean**
- **Document the problems that are not be fixed accordingly**

The submission for each release must be committed in a separate Github branch

Shared Wiki

- **Documentation, Tutorials, Links….**

  https://github.com/jku-win-se/teaching.ss22.prse.prwiki.en

Next steps

- **Now:**
  - Build teams of 3 Students - 1 “Team Leader” Email to antonio.garmendia@jku.at [Subject: PR_SE2022 Team] (Name, Matr.Nr, email, GitHub user)
  - Distribution of topics for the Workshop
- **For Next week (17.3.2022):**
  - Get familiar with the requirements and prepare questions for the Workshop
  - Plan the first version of the product and define the initial responsibilities for each member
  - Get familiar with GIT, Maven, Github Projects, Zenhub…
- **By 22.03.2020: Complete planning for Release 1 in Zenhub**

SE Tools Workshop: 17.3.2020

- Topic-1: Git
- Topic-2: Maven + CircleCI
- Topic-3: UML Tools / Editors
- Topic-4: UI Prototyping + Tools
- Topic-5: Timescale Database

Topic-1: Git

- **Git Functions and Markdown**
  - Create branch
  - Commit
  - Push
  - Minimal example of how to resolve conflicts
- **Tools:**
  - Git Bash, Git in Eclipse, Git Desktop, SourceTree
- **Tutorial: https://rogerdudler.github.io/git-guide/index.de.html**

Topic-2: Maven + CircleCI

- What is Maven? How add dependencies to the project?
- What is CircleCI?
- Create a Maven Project (e.g., sum calculator)
- Create at least an Unit Test (e.g., test the sum class)
- Compile and Test with CircleCI
- Execute the jar

Topic-3: UML Tools / Editors

- **Explore different UML Tools**
  - https://github.com/jku-win-se/teaching.ss22.prse.prwiki.en/blob/master/wiki/uml/README.md
- **The team should explore at least 4 tools and show minimal examples**
- **The group should show the functionalities (e.g., diagram creation, code generation, etc.) of the tools**
- **Small comparison of the tools**

Topic-4: UI Prototyping + Tools

- **Explore different UI Mockup tools**
  - https://github.com/jku-win-se/teaching.ss22.prse.prwiki.en/blob/master/wiki/uiprototype/README.md
- **The team should explore at least 4 tools and show minimal examples (SceneBuilder mandatory)**
- **The group should show the functionalities (e.g., diagram creation, code generation, etc.) of the tools**
- **Small comparison of the tools**

Topic-5: Timescale Database

- What is Timescale Database?
  - https://docs.timescale.com/
- What is the difference between Timescale and PostgreSQL? What is the benefit of a Hypertable?
- How to set up a Timescale Database?
- Create a minimal example of a Hypertable using Timescale. Perform an SQL operation.