
JOHANNES KEPLER
UNIVERSITY LINZ

Institut fuer Wirtschatsinformatik
Software Engineering

Praktikum Software Engineering

Michael Vierhauser

Einheit 0 - Einführung & Vorbesprechung

JOHANNES KEPLER
UNIVERSITY LINZ

Agenda

- Vorbesprechung
- Gruppeneinteilung
- Github Zugang
- Redmine Zugang
- Aufgabenverteilung für Workshop am 12.03

Ziel des Praktikums

- **Selbständiges Entwickeln eines Softwareproduktes im Team**
  - Spezifizieren, Planen, und Entwerfen eines Softwareproduktes
  - Objektorientierte Programmierung und Testen (Unit Tests & Code Qualität)
  - Arbeiten im Team
  - Arbeiten mit SE Tools
    - Version management (Repositories, GitHub)
    - Projektmanagement (Redmine)
    - Buildmanagement (Maven + CircleCi)
  - Planen von Sprints und Release Versionen
  - Erstellen einer Systemdokumentation (Architektur, Dokumentation von Testfällen)

Thema: Travel Management

Entwicklung eines neuen Reisemanagement Portals unter Verwendung von Java mit geeigneten Bibliotheken entweder als Desktop-Applikation oder als Webapplikation.

Ein Team aus drei Entwickler soll das Projekt in mehreren Sprints in einem Zeitraum von 4 Monaten realisieren und alle notwendigen Artefakte, Software,Tests, Dokumentation etc. erstellen.

- Erstellung und Verwaltung von Reisen
- Speichern und Backup
- Filter, Sortieren und Kategorien
- Reisekostenabrechnung

Vorgaben

- **High-Level Requirements**
- **Programmiersprache: Java**
- **Technologie**
  - Backend: Java
  - Frontend: Swing, JavaFX oder bei ausreichenden Kenntnissen der Teammitglieder auch web-basiertes Framework möglich

Organisation

- Arbeiten in Teams zu 3 Studierenden
- Aufwand und Aufgaben sollen innerhalb des Teams gleich verteilt werden
- Üblicher Aufwand: 6 ECTS (~ 150 Arbeitsstunden)

  Praktikumstermine und Gruppentermine mit eingerechnet
- LVA-Leiter fungiert als Auftraggeber und Mentor
- Empfehlung: Absolvierung der LVAs Soft1, Soft2, (Software Engineering)

<u>Jedes Teammitglied</u> hat Implementierungsaufgaben zu übernehmen – auf annähernd gleiche Verteilung achten!!

Zeitplan

- **Das Softwareprodukt wird in drei Releases entwickelt.**
  - *Release 1*: 30. Maerz 2020 (12.00 Uhr)
  - *Release 2*: 11. Mai 2020 (12.00 Uhr)
  - *Release 3*: 22. Juni 2020 (12.00 Uhr)
  - *Finale Abgabe*: 16. Juli 2020

- **Abgabe pro Release:** Branch in Git mit allen Dokumenten + Code
- **Finale Abgabe des entwickelten Softwareprodukts und der notwendigen Dokumentation erfolgt bis spätestens 16. Juli 2020**

Präsenztermine - Sprint Meetings

- **3 Sprint Planning Meetings**
  - Anwesenheitspflicht des gesamten Teams
  - 10-15 Minten Praesentation (Slide-Template)
  - jedes Teammitglied präsentiert
  - Diskussion, Status, Next Steps…
- **2 Individuelle Termine (23.04 und 04.06) pro Team**
  - Feedback & Fragen (30 - 60 Minuten)

Maerz					April					Mai					Juni					Juli
5.03	12.03	19.03	26.03	30	2.04	9.04	16.04	23.04	30.04	7.05	11	14.05	21.05	28.05	4.06	11.06	18.06	22.06	25.06	16.07
	T078					O	O						F			F				
Vorbesprechung	Req. Workshop Sprint Planning 1			R1	Sprint Planning 2			Project Meeting			R2	Sprint Planning 3			Project Meeting (Code Review)			R3	Final Sprint Planning	**Final Release + Final Documenttion**

Agile Software Entwicklung

- **Entwicklung in definierten Iterationen (Sprints)**
  - 1 Woche bis max. 1 Monat
- **Priorisierung der Anforderungen, Team entscheidet welche im jeweiligen Sprint umgesetzt werden**
- **Ergebnis eines Sprints = Neue Version des Produktes**
- **Keine dedizierten Rollen im Team**
  - Zwischen 5 und 9 Entwickler pro Team
- **Hohes Maß an Selbstorganisation**

Release 1

- **Fokus: UI Prototyp und OO Design**
- **Deliverables:**
  - Erstes Konzept zum Aufbau der Applikation (welche Features, Komponenten,..)
  - UML Klassendiagramm mit den wichtigsten Klassen (Klassennamen, Hierarchien, Methoden…) mit UML Tool!
  - UI Prototype
  - Lauffähige Builds in CircleCI
  - Status Präsentation 1 (für Sprint Planning Meeting)

Release 2

- **Fokus: Prototyp Implementierung und Unit Tests**
- **Deliverables:**
  - Erweiterte/aktualisierte UML Diagramme
  - Prototyp Implementierung:
    - Erste lauffähige Version des User Interfaces
    - Teile der Features funktionsfähig
  - Unit Tests fuer einzelne (wichtige) Klassen
  - Use Case Beschreibungen (siehe Use Case Template)
  - Status Präsentation 2 (Für Sprint Planning Meeting)

Release 3

- **Fokus: Dokumentation & Code Qualität**
- **Deliverables:**
  - Erweiterte/aktualisierte UML Diagramme
  - Erweiterte Unit Tests
  - Implementierung:
    - User Interfaces
    - Großteils lauffähige Version (alle Features vorhanden)
  - Analyse der Codequalität mit PMD, Findbugs, etc.
  - Erste Version der Projektdokumentation
  - Status Präsentation 3 (Für Sprint Planning Meeting)
  - Live Demo/Screencast der Applikation

Finales Produkt

- **Deliverables:**
  - Finale Projektdokumentation
  - Lauffähige, finale Version der Applikation
  - Github Dokumentation (Readme mit Installationsanleitung, etc.)
  - Javadoc für wichtige Klassen, Interfaces und Methoden

Beurteilung

- **Folgende Kriterien werden für die Beurteilung herangezogen**
  - Funktionsumfang des Produktes
  - Externe Qualität des Produktes (Stabilität, Effizienz, Benutzeroberfläche)
  - Interne Qualität des Produktes (Qualität des Entwurfs, Qualität der Programmierung, API-Dokumentation)
  - Umfang und Qualität der Unit Tests
  - Qualität der Dokumentation (Entwurf, Testfälle, Erfahrungsbericht)
  - Präsentationen

Tools fuer die LVA

- Redmine
- Git (GitHub)
- Maven
- CircleCI
- UML Editor / UI Prototyping Tool
- Code Quality: Static Code Analyzer

Projektorganisation mit Redmine

- **Umsetzungsdetails (Detailspezifikation) im Redmine verwalten**
  - Für jedes Release: Requirements, Tasks, Bugs, etc. anlegen
  - Verantwortungen und Aufwände eintragen! - Verantwortlichkeit definiert, wer den Source-Code schreibt (Code + Unit Tests)

- **Release Planung (Roadmap) in Redmine erstellen**
  - Am Ende jedes Sprints müssen die jeweiligen Tasks, Requirements, Bugs, etc. geschlossen werden und den Releases zugeordnet sein.

Sourcecodeverwaltung mit Git

- **GitHub für die Code und Dokumentations Verwaltung**
  - Es muss mindestens 1x pro Woche code in GitHub committed werden.
  - Bei jedem commit immer die jeweilige id eintragen (#TaskNr). - Jedes Teammitglied muss code schreiben und comitten!
- **Qualitätsfeedback - Der Source Code ist sauber zu halten.**
- **Problemstellen die nicht gefixt werden sollen, entsprechend dokumentieren.**

Die Abgabe für jedes Release erfolgt in einem eigenen Branch in GitHub

Shared Wiki

- **Dokumentation, Tutorials, Links….**

  https://github.com/jku-win-se/teaching.ss20.prse.prwiki

Nächste Schritte

- **Jetzt:**
  - Teambildung: 3 Personen - 1 “Teamleiter” Email an michael.vierhauser@jku.at [Betreff: PR_SE2020 Team] (Name, Matr.Nr, email, GitHub user)
  - Themenverteilung für Workshop
- **Bis nächste Woche (12.3.2020):**
  - Mit den Req. vertraut machen und Fragen für den Workshop vorbereiten
  - Erste Releaseplanung und erste Verantwortlichkeiten für die Umsetzung der Anforderungen definieren
  - Mit GIT, Maven, Redmine... vertraut machen
- **Bis 19.03.2020: Planung für Release 1 in Redmine fertigstellen**

SE Tools Workshop: 12.3.2020

- **Topic-1: Git**
  - Git Funktionen und Markdown
  - Git in Eclipse
  - Tutorial: https://rogerdudler.github.io/git-guide/index.de.html
- **Topic-2: Redmine**
- **Topic-3: Maven**
- **Topic-4: UML Tools / Editoren**
- **Topic-5: UI Prototyping + Tools**

Projektverwaltung

- **Redmine**
  - Für jedes Release: Requirements, Tasks, Bugs, etc. anlegen
  - Verantwortungen und Aufwände eintragen!
  - Am Ende jedes Sprints müssen die jeweiligen Tasks, Requirements, Bugs, etc. geschlossen werden und den Releases zugeordnet sein.
- **GitHub**
  - Es muss mindestens 1x pro Woche Code in github committed werden.
  - Bei jedem commit immer die jeweilige id eintragen (#TaskNr). - Jedes Teammitglied muss code schreiben und comitten!

Die Abgabe für jedes Release erfolgt in einem eigenen Branch in GitHub