JOHANNES KEPLER
UNIVERSITY LINZ

Praktikum Software Engineering

Johannes Bräuer

Einheit 0 - Einführung, Vorbesprechung

JOHANNES KEPLER
UNIVERSITY LINZ

Agenda

- Organisatorisches
- Gruppeneinteilung
- GitHub Zugang
- Aufgabenverteilung für Workshop am 14.03.2025

© 2025 Business Informatics - Software Engineering

Inhalt des Praktikums

- **Agile Entwicklung eines Softwareproduktes im Team**
  - ➜ Spezifizieren, Planen und Entwerfen eines Produktes
- **Implementierung, Testen und Qualitätssicherung**
- **Arbeiten mit Software Engineering Werkzeugen**
  - ➜ Versionsmanagement (Git)
  - ➜ Projektmanagement (GitHub, Clockify)
  - ➜ Buildmanagement (Maven, GitHub Actions)
- **Planen von Sprints und Releases**
- **Erstellen einer Systemdokumentation**

© 2025 Business Informatics - Software Engineering

Thema: Lunchify

- **Entwicklung einer Applikation zur Refundierung von Essensausgaben**
  - ➜ Einfache Benutzerverwaltung
  - ➜ Upload von Rechnungen
  - ➜ Filtern und Sortieren
  - ➜ Analyse und Statistiken

- **Technologien:** Java mit geeigneten Bibliotheken, entweder als Desktop-Applikation, Webapplikation oder mobile Applikation*

* **Alle** Teammitglieder müssen mit der ausgewählten Technologie vertraut sein!

© 2025 Business Informatics - Software Engineering

Organisation

- **Arbeiten in Teams zu 3 Studierenden**
  - ➜ Grundlegende Requirements als Vorgabe
- **Aufwand und Aufgaben müssen gleich verteilt werden**
  - ➜ Aufwand: 6 ECTS (~ 150 Arbeitsstunden)
- **Dringende Empfehlung**
  - ➜ Absolvierung der LVAs Soft1, Soft2, SE

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

© 2025 Business Informatics - Software Engineering

Agile Software Entwicklung

- **Entwicklung in definierten Iterationen (Sprints)**
  - ➔ Sprint von 3 Wochen
- **Priorisierung der Requirements**
  - ➔ Team entscheidet welche Requirements im jeweiligen Sprint umgesetzt werden
- **Ergebnis eines Sprints = Neue Version (Release) des Produktes**
- **Hohes Maß an Selbstorganisation**

© 2025 Business Informatics - Software Engineering

Zeitplan

- **Die Applikation wird in drei Sprints entwickelt**
  - ➜ Release 1: Di, 8. April (12.00 Uhr) - **release-v0.1.0**
    - Präsentation: Fr, 11. April
  - ➜ Release 2: Di, 29. April (12.00 Uhr) - **release-v0.2.0**
    - Präsentation: Fr, 2. Mai
  - ➜ Release 3: Di, 20. Mai (12.00 Uhr) - **release-v0.3.0**
    - Präsentation: Fr, 23. Mai
  - ➜ Abschlusspräsentation: Fr, 13. Juni (12.00 Uhr)
  - ➜ Finale Abgabe: Fr, 27. Juni (12.00 Uhr) - **release-v1.0.0**

- **Abgabe <u>pro Sprint</u>:**
  - ➜ GitHub Release mit allen Dokumenten und Code

© 2025 Business Informatics - Software Engineering

Präsenztermine - Sprint Planning Meetings

- **Anforderungsworkshop, Teampräsentationen (14. März)**
- **Präsentation Prototype, JavaFX (21. März)**
- **3 Sprint Planning Meetings (Release 1-3)**
  - ➔ Anwesenheitspflicht des gesamten Teams
  - ➔ 12 Minuten Präsentation (Folien-Template)
  - ➔ Jedes Teammitglied präsentiert
  - ➔ Diskussion, Status, Next Steps, Einsatz von KI, …
- **Abschlusspräsentation (13. Juni)**

- **(optional) 1 individueller Termine pro Team**
  - ➔ Feedback & Fragen (30 Minuten)
  - ➔ Anwesenheitspflicht des gesamten Teams

© 2025 Business Informatics - Software Engineering

Präsentation Prototype - 21. März

- **Fokus: UI Prototyp**
- **Deliverables**
  - ➜ UI Prototype
  - ➜ *Lauffähige Builds in GitHub Actions*
  - ➜ Grobkonzept zum Aufbau der Applikation
  - ➜ Planung Release 1
  - ➜ Live Demo des UI Prototyps

© 2025 Business Informatics - Software Engineering

Sprint 1 – 11. April

- **Fokus: Prototyp Implementierung und Unit Tests**
- **Deliverables**
  - ➜ UML Klassendiagramm mit den wichtigsten Klassen (Klassennamen, Hierarchien, Methoden…) mit UML Tool!
  - ➜ Prototyp Implementierung
    - Erste lauffähige Version des UIs
    - Ca. 1/3 der User-Stories umgesetzt
  - ➜ Unit Tests für einzelne Klassen
  - ➜ Planung Release 2
  - ➜ Live Demo der Applikation

© 2025 Business Informatics - Software Engineering

Sprint 2 - 2. Mai

- **Fokus: Dokumentation & Code Qualität**
- **Deliverables**
  - ➜ Erweiterte/aktualisierte UML Diagramme
  - ➜ Erweiterte Unit Tests
  - ➜ Implementierung
    - Ca. 2/3 der User-Stories umgesetzt
    - Analyse der Codequalität mit SonarQube, PMD, Findbugs, etc.
  - ➜ Erste Version der Projektdokumentation
  - ➜ Planung Release 3
  - ➜ Live Demo der Applikation

© 2025 Business Informatics - Software Engineering

Sprint 3 - 23. Mai

- **Fokus: Stabilität und Projektfinalisierung**
- **Deliverables**
  - ➔ Implementierung
    - Alle User-Stories umgesetzt
  - ➔ Testplan für Sonderfallbehandlungen
  - ➔ GitHub Dokumentation (Installationsanleitung, usw.)
  - ➔ Javadoc für wichtige Klassen, Interfaces und Methoden
  - ➔ Live Demo der Applikation

© 2025 Business Informatics - Software Engineering

Abschlusspräsentation - 13. Juni

- **Deliverables**
  - ➜ Finale Projektdokumentation
  - ➜ Lauffähige, finale Version der Applikation
  - ➜ GitHub Dokumentation (Installationsanleitung, usw.)
  - ➜ Individueller Erfahrungsbericht

Projektabgabe: 27. Juni

© 2025 Business Informatics - Software Engineering

Beurteilung

- **Kriterien für die Beurteilung**
  - ➜ Funktionsumfang des Produktes
  - ➜ Externe Qualität des Produktes
    - Stabilität, Performanz, User Interface
  - ➜ Interne Qualität des Produktes
    - Qualität des Entwurfs, Implementierung, Dokumentation
  - ➜ Umfang und Qualität der Unit Tests
  - ➜ Präsentationen

© 2025 Business Informatics - Software Engineering

Pause

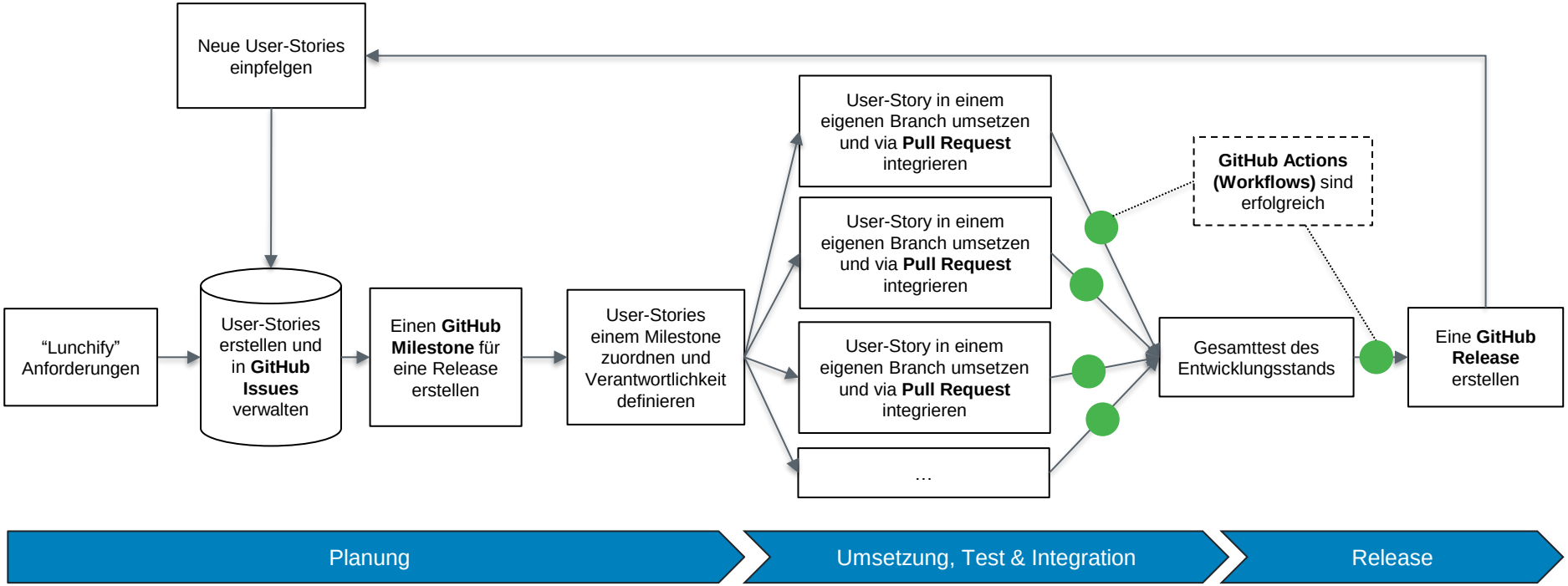
Teameinteilung: 3 Personen / Team

Werkzeuge

- **GitHub / GitHub Actions**
  - ➜ Sourcecode Verwaltung
  - ➜ Verwaltung von User-Stories
  - ➜ Build-Management (Continuous Integration)
- **Clockify**
  - ➜ Zeitaufzeichnung
- **Maven**
  - ➜ Build-Management
- **UML Editor / UI Prototyping Tool**
- **Code Qualität: Statische Code Analyse Tools**

© 2025 Business Informatics - Software Engineering

Projektmanagement mit GitHub (1/2)



© 2025 Business Informatics - Software Engineering

Projektmanagement mit GitHub (2/2)

- Verwenden sie **GitHub Issues**, um den Backlog an User-Stories zu verwalten.
- Erstellen sie pro Release einen **GitHub Milestone**, um die User-Stories zu bestimmen, die für die Release geplant sind.
- (optional) Verwenden sie **GitHub Projects***,* um den groben Zustand der Umsetzung darzustellen und Verantwortlichkeiten zu verwalten.
- Die Verwendung von **Pull Requests** pro User Story ist verpflichtend. Details diesbezüglich folgen in der LVA.
- Eine User Story darf nur dann geschlossen werden / Ein Pull Request darf nur "*gemerged* werden, wenn die **Definition of Done** erfüllt ist (s.o.)
- Eine User Story darf nur dann geschlossen werden / Ein Pull Request darf nur "*gemerged* werden, wenn die **Workflows erfolgreich** sind.
- Stellen Sie dabei auch sicher (Gesamttest), dass die Akzeptanzkriterien aller umgesetzten User Stories erfüllt sind.
- Nach dem Abschluss aller User-Stories die dem Milestone zugeordnet sind, ist eine **GitHub Release** mit einem **Tag** zu erstellen (z.B. v0.0.1), damit es möglich ist, sich jederzeit diesen Release-Stand anzuschauen. Details diesbezüglich folgen in der LVA.

© 2025 Business Informatics - Software Engineering

Sourcecodeverwaltung mit Git

- **GitHub für die Code- und Dokumentationsverwaltung**
  - ➜ Commit mindestens 1x pro Woche (von jedem Teammitglied)
  - ➜ Bei jedem commit immer die jeweilige Issue-ID eintragen
- **Auf Codequalität achten**
  - ➜ Probleme die nicht gefixt werden, entsprechend dokumentieren
- **Abgabe jedes Releases als eigene GitHub Release**
  - ➜ `release-v0.1.0`
  - ➜ `release-v0.2.0`
  - ➜ `release-v0.3.0`
  - ➜ `release-v1.0.0`

© 2025 Business Informatics - Software Engineering

Shared Wiki

- **Dokumentation, Zeitplan, Tutorials, Links**

  https://github.com/jku-win-se/teaching.ss25.prse.prwiki.braeuer

© 2025 Business Informatics - Software Engineering

AI Tools

- **Experimentieren mit AI Tools (GitHub Copilot, ChatGPT, …) ist erlaubt und auch erwünscht**
  - ➔ Für Programmierung als auch Projektmanagement
- **Klare Kennzeichnung von Code der generiert wurde**
  - ➔ Wird sonst als Plagiat gewertet
- **Erfahrungsbericht im Rahmen der Releasepräsentationen**

© 2025 Business Informatics - Software Engineering

Nächste Schritte

- **Jetzt:**
  - ➔ Teameinteilung: 3 Personen

    Ein E-Mail pro Team an johannes.braeuer@gmx.at
    Betreff: PR_SE2024 Team
    Inhalt: Name, Matr.Nr, E-Mail, **GitHub User, Clockify User**

  - ➔ Themenverteilung für Workshop
- **Bis Sonntag 18:00 ist GitHub Repository verfügbar**
- **Bis nächste Woche (14. März)**
  - ➔ Mit den Requirements vertraut machen und Fragen für den Workshop vorbereiten
  - ➔ Je nach Themenwahl mit Git, GitHub & Clockify, Maven & GitHub Actions, UML-Tools und UI Tools vertraut machen und **README** in GitHub Repository erstellen
    - 15min Präsentation des README und Live-Demo
    - Wissensbasis für die anderen Teams (Shared Wiki)

© 2025 Business Informatics - Software Engineering

Themen (1)

- **Topic-1: Git (mit GitHub)**
  - ➜ Git Funktionalitäten
  - ➜ Git: Desktop-Anwendungen und Integration in Programmierumgebung
- **Topic-2: GitHub & Clockify**
  - ➜ Issues, Projects, Milestones, Releases
  - ➜ Verantwortlichkeiten, Zeiterfassung mittels Clockify
- **Topic-3: Maven & GitHub Actions**
  - ➜ Maven Funktionalität
  - ➜ Projekt mit GitHub Actions bauen

© 2025 Business Informatics - Software Engineering

Themen (2)

- **Topic-4: UML Tools / Editoren**
  - ➜ Open Source oder Demo Version bis Juli
  - ➜ Mindestens 3 Tools (bevorzugt SaaS Lösungen)

- **Topic-5: UI Prototyping + Tools**
  - ➜ Open Source oder Demo Version bis Juli
  - ➜ Mindestens 3 Tools (bevorzugt SaaS Lösungen)

- **Topic-6: GitFlow**
  - ➜ Wie entwickelt man ein neues Feature?
  - ➜ How-to Guide und Live-Übung mit allen Studierenden

© 2025 Business Informatics - Software Engineering

JOHANNES KEPLER
UNIVERSITY LINZ