

**Autor: Jana Wengenroth  
27.10.2015**

## **Protokollzusammenfassung (Phase1)**

Deadline des Projektes: 18.12.2015

### **Verwendete Tools (Verantwortlicher: Andi)**

- Redmine: Tickets, Wiki, Datenaustausch, Kalender  
→ *wurde verworfen*
- GitLab: zunächst nur zur Versionskontrolle, nach Verwerfung von Redmine auch für Tickets, Wiki, Datenaustausch
  - \* Wenn Issue bearbeitet wurde, Commit Link in den Kommentar
  - \* Für jeden Entwickler wird ein Branch eingerichtet (→ *verworfen, stattdessen Feature-Banches*)
- Visual Paradigm: UML, Codegenerierung
- Mumble: sonstige Kommunikation  
→ *wurde verworfen, statt dessen Stratoserver*
- Framework: Play
- IDE: Eclipse und IntelliJ → *Eclipse verworfen*
- Sprache: Java
- Hibernate für Datenbankbindung
- Apache Commons Email für Mailversand
  - Projektseite: <http://commons.apache.org/proper/commons-email/userguide.html>
  - Beispielcode: <https://www.playframework.com/documentation/1.3.x/emails>
- Wir werden zur **Anbindung der MySQL-Datenbank** die Bibliothek 'Hibernate' nutzen.
  - Projektseite: <http://hibernate.org/orm/>
  - Beispielcode: [http://www.tutorialspoint.com/hibernate/hibernate\\_examples.htm](http://www.tutorialspoint.com/hibernate/hibernate_examples.htm)
- Stisys nachbauen: Style-Sheets übernehmen, HTML selber machen
- Tests → JUnit

### **Was soll die Software leisten? (Entwurf 1)**

- Studenten melden sich in einer definierten Phase in Teams an
- Nach Ablauf der Phase sortiert das System die Teams, so dass möglichst wenig Terminüberschneidungen entstehen

**Autor: Jana Wengenroth  
27.10.2015**

- Ausgabe des Systems: Belegungspläne für alle Praktika
- Wunschtermine können vermerkt werden (Erstwahl, Zweitwahl etc.) ◦ Kein must-have, Implementierung vorerst als Dummy → *verworfen*
- Benötigte Rohdaten: ◦ Eindeutige Studierendenreferenz (z.B. Matrikelnummer) ◦ Praktikumstermine (Bezeichnung, Kalenderwochen, Uhrzeiten) ◦ Verfügbare Plätze im jeweiligen Praktikum
- Bisherige Teams werden semesterübergreifend gespeichert → *verworfen*

### **Fragen**

- Muss die Software sowohl Anmeldungen aus den Fremdsystemen, als auch eigene Anmeldung verarbeiten? → *Beides*
- Erstellen die Studenten Teams (mit Invites), oder ist die Anzahl der Teams festgelegt? → *Studenten erstellen Teams, Anzahl abhängig von Teamgröße, max. Gruppengröße und Anzahl der Gruppen*
- Sollen Studenten die nur Kurse aus einem Semester belegen bei der Sortierung bevorzugt werden? → *?*
- Woher hat das System die Rohdaten? → Fragenkatalog ◦ ICS Dateien für einige Studiengänge vorhanden (HAW) ◦ Falls möglich: Exportschnittstelle vom Fremdsystem? → *keine Schnittstelle gegeben, Dummy-Daten*

### **Prozessmodell**

keine täglichen Treffen  
regelmäßige Reviews

Wir wählen primär das interkrementelle Modell (mehrere Releases), mit einigen Aspekten aus Scrum. Wöchentliches Treffen mit 2 Stunden Zeit.

### **Vorgehen**

Ferien:

Klausurzeit keine Treffen!

Jonas: nicht mit einbezogen, wegen Urlaub

Thomas: nicht verfügbar von bis: 24.06. 28.06. 10.07.

- Einarbeitung in Frameworks

Treffen donnerstags

Aufgabenverteilung in den Ferien:

- Thomas: Team-Erzeugen
- Jana: Einzelanmeldungen für Veranstaltungen (Phase1)
- Christopher: Teilnehmer einladen → *von Andi zuende gemacht, da Chris ausgeschieden*

nach den Ferien:

- Treffen immer Dienstag 16 Uhr

Aufgabenverteilung:

- Andi: Projektleitung, Qualitätsbeauftragter, assistierender Entwickler → *nicht mehr QS, neu:Infrastrukturbeauftragter, Frontend*
- Thomas: Backend → *fachlicher Designer*

**Autor: Jana Wengenroth  
27.10.2015**

- Jana: Backend → *hauptsächlich QS*  
Christopher: Frontend (*ausgeschieden*)  
- Jonas: Algorithmus → *und Backend, technischer Designer*  
- Schnittstellen für die Einführungsprojekte (mit TestDaten) werden von Andi gestellt

## **Anforderungen**

### **1. Phase:**

- Einzelanmeldung für Praktika verbindliche Anmeldung und Einteilung als default (> AlleingängerPool)
- Bildung eines (internen) AlleingängerPools bei Abschluss von Phase 1

### **- 2.Phase:**

- Teambildung auf Grund von Ergebnissen aus Phase 1
- (optionale) Möglichkeit Wünsche zu äußern, wenn kein Team und kein Wunsch geäußert wird Gruppe random zugeordnet → *verworfen: keine Wunsch/Prioritäten erlaubt*
- Übersicht über bisherige Anmeldungen aus Phase 2 in Praktikumsübersicht
- Anfragen an bestehende Gruppen möglich (an alle Teammitglieder)
- Kommentare möglich, wenn z.B. manuelles Mergen vorgenommen werden soll: eine Gruppe mit 2 Leuten möchte einer Gruppe mit 3 Leuten beitreten entsprechende E Mail Benachrichtigungen
- Teammitglieder werden bevorzugt in gleiche Gruppen eingeteilt (1)
- Möglichst in gleiche Gruppe, aber im Notfall können auch einzelne Teammitglieder in andere Gruppen eingeordnet werden (z.B. bei Unstimmigkeiten in Phase 2, wie zu kleine Gruppe)
- “Mergen” von Teams:  
z.B. Praktikum von 5 Leuten, eine Gruppe mit 3 Leuten und eine Gruppe mit 2 Leuten
- Teamgröße sollen während Praktikusanmeldung vorgegeben werden
- Teamleiter verschickt Einladungen an potenzielle Teammitglieder
- Potenzielle Teammitglieder bestätigen Anmeldung
- Übersicht sowohl für geladene Personen (welche Einladungen habe ich erhalten)
- Alleingänger müssen sich NICHT in Phase 2 anmelden, sondern sind per default teamlos (bzw. ein Einerteam)
- Nicetohave: Vorschläge für Personen auf Grund von bisherigen Anmeldungen
- Problem: Teams-zuordnen:
  - akzeptable Lösung: Benutzer-Auflösung des Problems
  - lieber automatische Ermittlung mit Rückmeldung, dass nicht umsetzbar für einzelne Personen (Zukunft)
  - 2 Stunden Laufzeit
- Gruppengröße als Priorität? → Teamerhaltung

**Autor: Jana Wengenroth  
27.10.2015**

### **Datenbank und Implementation**

- Veranstaltungen, die ein Student belegen darf sind in Student gespeichert.
- aktueller Benutze über Matrikelnummer
- statt verfügbaren Kursen für Student, nicht verfügbare Kurse
- Zwischentabelle Studiengang\_Kurs
- Studiengang bei Student ergänzen
- max. Teilnehmerzahl für Kurse
- Student kennt nicht verfügbare Kurse, sondern unverfügbare (relevant für Importschnittstelle)
- Importinterfaces in den 'models' Ordner
  - \* Keiner verändernden Operationen
- Anpassung fachliches Modell / Datenbank: Student kennt seine Verfügbaren Kurse
- Neustrukturierung der Datenbank:
  - \* Menge der von UniKit persistierten Daten wird reduziert um Inkonsistenzen mit Fremdsystem-Datenbank zu vermeiden.
  - \* Lediglich Wahlen der Veranstaltungen, Teambelegungen und Gruppenbelegungen werden persistiert
  - \* Verantwortlicher für Modellierung: Andi
- "Student" bekommt Attribut Studiengang (Fachbegriff undefiniert)
- "Course" bekommt Attribut Studiengang
- Zwischentabelle KursZuFachrichtung (Fachbegriff undefiniert)  
Wichtig beispielsweise für WPs (studiengangsübergreifend)
- Liste von Studenten und Liste von Veranstaltungen, welcher Student darf welche Veranstaltungen belegender

### **Aufgabenverteilungen**

- Thomas:
  - UseCase für die Anmeldung → ?
  - Implementierung von Phase 1 (*erledigt*)
    - \* Vorbedingungen: Ein angemeldeter Student der seine belegbaren Veranstaltungen kennt
    - \* Anzeige der Verfügbaren Veranstaltungen, Persistierung mittels Hibernate in Datenbank
  - Auswahl der Kurse in Controller übergeben (*erledigt*)
  - Persistierung mittels Hibernate (*erledigt*)
  - Glossar → ?
- Andi:
  - MookUp Teambildung (Phase 2) (*erledigt*)
  - SQL Skripte für DB erstellt (*erledigt*)
  - Java Modells für Hibernate erstellt (*erledigt*)
  - Studentendaten erzeugt (Testdaten) (*erledigt*)
  - Aufsetzen der Datenbank (*erledigt*)
  - Datenbank erstellt und befüllt (*erledigt*)
  - Hibernate Mapping durch Annotationen in Models ersetzen (*erledigt*)
  - Fertigstellung aller Branches (*erledigt*)
- Jana: → vorläufiges Glossar (*erledigt*) → *verworfen*

**Autor: Jana Wengenroth  
27.10.2015**

- Hibernate Mappings (*erledigt*) → *verworfen*
- Komponentendiagramm überarbeiten (*erledigt*)
- Jonas:
  - Code Conventions (*erledigt*)
  - Implementierung der Importschnittstellen (*erledigt*)
  - Initialisierung der SessionFactory mittels statischem Konstruktor → ?
  - Bisher kriegt der Student Kurse aller Fachrichtungen, Logik und Datenbankschema müssen angepasst werden → ?

### **Begriffe für das Glossar**

- Leute ohne Team sind: Teamlose (*verworfen, da Englisch*)

### **Namenskonventionen**

- \* Controller: [Name]Controller
- \* Implementation: [Name]Impl
- \* Interface: [Name]

### **Algorithmus**

- Vor Sortierung: Prüfung, ob sich Praktikums-/Vorlesungstermine überschneiden
- Ersatz-Sortierungs-Algorithmus:
  - Anmelden für Praktikum
  - Team bilden
  - Anmelden für Gruppenbelegungen
    - e-mail bei Konflikt
    - alle können anmelden
    - Pop-up bei Konflikt
- Apriori angelehnt?
- Priorisierung innerhalb des Fachsemesters in dem die meisten Kurse belegt wurden, bei gleich vielen Belegungen → niedrigeres Semester

### **Präsentation SE2P1**

Aufgabe 1: Liste der Teammitglieder -> Jana

Aufgabe 2: Projektplan reviewen/anpassen-> Jana

- Was/Wann/Wer: An Meilensteinen orientiert, wird vor nächstem Meilenstein geplant
- Rollen:
  - Andi: Projektleiter, Infrastrukturbeauftragter
  - Jana: Qualitätssicherung
  - Jonas: Chefdesigner technisch
  - Thomas: Chefdesigner fachlich
- Aufgabe 3: Deployment des Prototypen -> Andi

**Praktikum zur Lehrveranstaltung  
„Software Engineering 2“  
WiSe 2015/16**

**HAW Hamburg  
Department Informatik  
Gruppe: UniKit**

**Autor: Jana Wengenroth  
27.10.2015**

- Aufgabe 4: Statusbericht des Projekts -> Jonas