Praktikumsanmeldesoftware
UniKit (Stand 30.11.2015)

Projektbeschreibung:

UniKit ermöglicht es den Studenten sich mit einem Team bestehend aus zwei bis 10 Studenten (je nach Vorgabe der Uni) für eine Veranstaltung anmelden zu können. Dies erfolgt intern in zwei Phasen, die während der Benutzung aber gleichzeitig stattfinden. In Phase Eins melden sich die Studenten verbindlich für das Praktikum an und in Phase Zwei haben die Studenten dann die Möglichkeit sich für eine angemeldete Veranstaltung in einem Team zusammen zu finden. Sie können entweder ein Team gründen oder einem bestehenden Team beitreten. Ist Phase Zwei beendet und somit auch die Anmeldephase, springt unser Algorithmus an und versucht möglichst alle Teams konfliktfrei auf die Praktikumsgruppen zu verteilen. Ist ein Student keinem Team beigetreten, ist dies irrelevant und er wird dennoch einer Praktikumsgruppe zugeteilt. Als Ergebnis liefert UniKit dann einen Belegungsplan für alle Praktika und den dazugehörigen Praktikumsgruppen.

Agenda für das Meeting am 01.12.2015:

1. Einleitung
    - Aktueller Zustand des Projektes
    - Zeitplan
2. Qualitätsdefinition und -merkmale
3. Das System in Aktion
4. Probleme
5. Offene Fragen

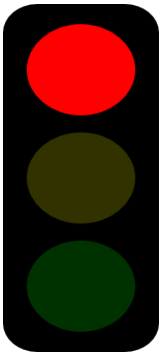
Die aktuell lauffähige Version unserer Anmeldesoftware ist unter folgender Adresse erreichbar:
http://www.uni-kit.net:9000/

Das Passwort für alle Studenten ist `test` .

Statusbericht

- **Projektname**: UniKit
- **Aktuelle Version**: 0.8 (vom 30.11.2015)
- **Nächste Version**: 1.0 (vermutlich zum nächsten Praktikumstermin)

Gesamtprojektzustand:



Lagebericht:

Aktueller Stand:

Zum einen sind wir derzeit etwas hinter dem eigentlichen Zeitplan. Zum anderen gibt es extreme Mängel in der Software, die durch das eigentliche Refactoring entstanden sind. Hier haben wir erneut den Fehler aus unserer ersten Entwicklungsphase gemacht, einen großen Bereich an getätigten Aufgaben erst kurz vor der Abnahme in den Production-Branch zu mergen. Diesmal leider noch ohne Erfolg. Daher ist der Projektstatus derzeit rot. Wir sind aber zuversichtlich, dass wir die ausstehenden Aufgaben innerhalb der nächsten 2 Wochen erledigt zu haben und zu diesem Zeitpunkt auch unsere erwarteten Qualitätsmerkmale in der bisherigen Software umgesetzt zu haben. Der zeitliche Verzug ist durch eine falsche Einschätzung des Zeitaufwands entstanden.
Weiterhin muss nach Behebung der Merge-Konflike noch das Verschicken von Emails implementiert werden. Die Planung und Spezifizierung der Tests ist abgeschlossen. Die automatisierten Tests müssen jedoch noch größtenteils implementiert werden.
Die Entwicklung des Verteilungsalgorithmus hat begonnen. Das Grundgerüst für das weitere Vorgehen wurde erarbeitet. Die Probleme die der Algorithmus aufwirft wurden konkretisiert und dokumentiert. Erste Testversuche wurden durchgeführt.

- Wir arbeiten weiterhin mit dem Ticketsystem „YouTrack“. Dort übersichtlich möglich die Aufgaben in Form von Tickets an die einzelnen Teammitglieder zu vergeben. Seitdem wir uns auf diese Weise organisieren, haben wir einen besseren Überblick über die anstehenden Aufgaben.

- Auch die neue Protokollierung der wöchentlichen Teammeetings mittels der von Ihnen vorgestellten Protokolltechnik hat entscheidend dazu beigetragen die Organisation zu verbessern.
- Die Planung der Sprints erfolgt nun langfristiger. Dies trägt zu einer besseren Verteilung des Arbeitsumfangs bei. Wie gesagt gibt es aber noch Probleme, die Zeitaufwände der Aufgaben abzuschätzen.
- Test-Verfahren wurden entwickelt.

Probleme:

Da das Refactoring von „Phase 2“ bisher mehr Zeit in Anspruch genommen hat als wir vermutet haben und ist leider immernoch nicht abgeschlossen (siehe oben). Dadurch ist die Implementierung der automatisierten Tests nach hinten gerutscht.

Qualität:

Zur Definition der Qualität haben wir Codekonventionen, wie Namenskonventionen, eingeführt. Durch einen einheitlichen Stil soll der Code übersichtlicher und leichter verständlich werden. Außerdem sollen Schnittstellenkommentare zur besseren Verständlichkeit beitragen. Somit soll die Qualität durch bessere Wartbarkeit erhöht werden. Für den Algorithmus und die Datenbank gibt es bereits Unittests, die die Funktionalität auf ihre Richtigkeit und mögliche Fehlerfälle überprüfen. Da wir mit einer Test-Datenbank arbeiten, werden Auswirkungen der Arbeit auf die anderen Daten vermieden.

Die fehlenden automatisierten Tests im Backend wurden bereits angesprochen. Da während des Refactorings von Phase 2 stark umstrukturiert wurde, wurde ihre Implementation vorerst nach hinten verschoben. Die Definition der Testfälle (der Regelfälle und Grenzfälle) ist abgeschlossen. Die Implementation soll in Kürze ergänzt werden.

Wir haben vor der Implementierung von Aufgaben, sehr detailliert dokumentiert und entworfen, was wir überhaupt bauen werden. Dies ist in Form von Klassendiagrammen, Use-Cases, einem Glossar und diversen anderen Dokumenten geschehen, die wir auch als Auslieferungsgegenstand abgeben werden.

Wir legen in unserem Projekt starken Wert auf Stabilität des Programmes. Daher haben wir viele eigene Exceptions eingeführt und auch die Mittel der statischen Typsicherheit ausgenutzt (wie Objekte für IDs, damit z.B. nicht versehentlich eine (ganzzahlige) ID für ein Team einer Methode übergeben werden kann die eine (ganzzahlige) ID für einen Kurs erwartet). Genau diese Änderungen haben aber im Rahmen des Refactoring einen extremen Aufwand darstellt und zu dem aktuellen Stand des Projektes geführt.

Weiterhin versuchen wir ein möglichst schnelles Produkt auszuliefern. Dafür nutzen wir zum Beispiel eine umfangreiche Indizierung auf Datenbankenebene und achten darauf, dass Daten aus der Datenbank nur dann geladen werden, wenn dies wirklich notwendig ist. Dafür nutzen wir dafür eine Lazy-Evaluation, die wir in unserer Datenbank-Anbindung „Hibernate“ konfiguriert haben. In der Anmeldesoftware werden u.a. durch diese Techniken gute

Ergebnisse erzielt.
Bisher ist allerdings noch nicht abschätzbar, wie performant der Verteilungsalgorithmus sein wird.