# Studienarbeit Software Engineering II (IB)

Prof. Dr. Oliver Braun

Fakultät für Informatik und Mathematik

Hochschule München

Letzte Änderung: 26.04.2017 07:44

Nachdem Sie in Zweiergruppen im vergangenen Semester sehr erfolgreich eine Web-Anwendung für einen Pizzalieferdienst als Auftragsarbeit implementierten, haben Sie beschlossen Ihre Anwendung so zu verbessern, dass Sie sie auch anderen Lieferdiensten anbieten können. Dabei stehen für Sie nicht neue Features im Vordergrund, sondern Sie müssen sich nun viel mehr um Dokumentation, Tests, Code-Qualität etc. kümmern.

Um in der kurzen Zeit zweimal ein "potentially shippable product" am Ende zweier Scrum-Sprints fertig stellen zu können, schließen Sie sich neu in 4er bzw. 5er-Gruppen zusammen. Dabei können Sie die Arbeit dann, wie in einem richtigen Projekt, beliebig (aber möglichst gleichmäßig) auf die Gruppenmitglieder aufteilen.

## Rahmenbedingungen

Die Studienarbeit ist als Gruppenarbeit durchzuführen. Für die Benotung ist jedoch eine exakte personelle Zuordnung der Tätigkeiten notwendig. Der Nachweis erfolgt über Issues in Redmine.

Für jede Tätigkeit die Sie innerhalb des Projektes durchführen ist ein Ticket anzulegen. Das Ticket wird der **verantwortlichen** BearbeiterIn zugewiesen. Auch wenn Sie gemeinsam an einem Ticket arbeiten, kann immer nur eineR verantwortlich sein. Wechseln Sie sich in dem Fall mit den Verantwortlichkeiten ab.

Im jeweils ersten Praktikumstermin müssen Sie sich in max 4 Gruppen je Praktikumsteilgruppe einteilen. Jede Gruppe sollte aus 4-5 Studierenden bestehen. Die verschiedenen weiteren Termine sind dann individuell für die 8 Gruppen. Die Gruppen werden wie folgt nummeriert:

- Praktikumsgruppe 1

  - Gruppe 1A
  - Gruppe 1B
  - Gruppe 1C
  - Gruppe 1D

- Praktikumsgruppe 2
  - Gruppe 2A
  - Gruppe 2B
  - Gruppe 2C
  - Gruppe 2D

Das Projekt läuft in 3 Phasen ab: Einer Startphase und zwei Sprints. Die genauen Zeiten finden Sie unter Termine auf https://ob.cs.hm.edu/lectures/swengiiib.html.

Sie arbeiten weiter mit Scala und dem Play Framework. Die Zusammenarbeit erfolgt über ein gruppeneigenes Redmine-Projekt und ein GitHub-Repository welches ich Ihnen zur Verfügung stellen werde.

## Arbeiten mit Redmine

Redmine ist eine freie, webbasierte Projektmanagementsoftware. Sie sollen Redmine zur Zusammenarbeit innerhalb der Gruppe nutzen. Alles was Sie im Projekt machen, muss über Tickets abgedeckt sein.

Legen Sie für Ihre Tickets verschiedene Kategorien an: Startphase, Product Backlog, Sprint 1, Sprint 2, …, so dass Sie jedes Ticket genau zuordnen können. Für die Tickets die später in den Sprints bearbeitet werden sollen, ist insbesondere der geschätzte Aufwand in Stunden wichtig.

Jedes Ticket ist während der Bearbeitung und danach einer Person zugeordnet. Diese Person ist verantwortlich für das Ticket und die damit verbunden Arbeiten. Über die Tickets weiß ich am Ende wer was gemacht hat um es entsprechend zu bewerten.

Arbeiten Sie mit den Tickets, d.h. buchen Sie Aufwand darauf, schreiben Sie Kommentare dazu, buchen Sie Ihren Aufwand und weisen Sie das Ticket (temporär) jemandem zu, der etwas damit machen muss. Sofern Sie im Ticket etwas machen was im Repository erscheint, verlinken Sie in Kommentaren die entsprechenden Commits auf GitHub.

Kommunizieren Sie auch mit mir über Tickets. Wenn ich beispielsweise etwas an Git, Jenkins, … ändern soll, erzeugen Sie ein Ticket und weisen Sie es mir zu.

Remine verfügt auch über ein Scrum-Plugin mit dem Sie die Scrum-spezifischen Infos verwalten können.

## Anforderungen an das Shippable Product zum Ende der beiden Sprints

- Der Code muss üblichen Standards genügen (Variablennamen, …).
- Beim Compilieren darf keine Warning ausgegeben werden. Aus dem Grund sollten Sie aus Warnings Fehler machen. Fügen Sie dazu

  ```
  scalacOptions ++= Seq("-unchecked", "-deprecation", "-feature",
      "-Xfatal-warnings")
  ```

  zur `build.sbt` hinzu.
- Außerdem darf auch die Überprüfung durch Scalastyle keine Errors oder Warnings erzeugen. Die zu verwendete Scalastyle-Konfigurationsdatei finden Sie unter https://gist.github.com/obcode/bd2e623dfc41157d8cf5.
- Der Code muss komplett mit ScalaDoc kommentiert sein. Eine komplette Abdeckung ist erst zum Ende des zweiten Sprints notwendig.
- Der Code muss mit Specs2-Tests abgedeckt sein. Die Abdeckung muss für alle selbst geschriebenen Klassen und Objekte zum Ende des zweiten Sprints mindestens 80% betragen. Dies kann durch das Tool Scoverage überprüft werden.
- Eine Benutzerhandbuch für die Mitarbeiter beschreibt die gesamte Anwendung und ist in Markdown formatiert. Wo es sinnvoll ist unterstützen UML-Diagramme (z.B. Sequenz-Diagramme) das Verständnis.
- Zum Sprint Review wird die neueste Fassung, z.B. auf Heroku, deployed.
- Bis zum Ende von Sprint 1 haben Sie einen Travis-Job erzeugt, der Ihre Anwendung baut, die Doku baut, die Tests ausführt, etc. Ein Travis-Job ist für ein öffentliches GitHub-Repository gratis.

## Live-Anwendung und Bugs

Zum Ende der drei Phasen muss die Anwendung in der jeweiligen Release-Version deployed werden und während der gesamten Zeit bis zum Ende des Semesters zur Verfügung stehen.

Wird ein Bug gefunden, so ist im laufenden Sprint zu entscheiden wie mit dem Bug verfahren wird. Es gibt dabei drei Möglichkeiten:

- Der Bug wird sofort gefixt und eine neue Version (die nur den Fix enthält) deployed. Wichtig ist dabei, dass nicht einfach der aktuelle Stand im Sprint deployed wird, sondern nur der Bugfix.
- Der Bug wird im laufenden Sprint gefixt, aber erst mit dem nächsten Release am Ende des Sprints deployed.
- Der Bug wird erst im nächsten Sprint gefixt.

Die Entscheidung ist im Ticket zu vermerken und das Ticket mir dann zuzuweisen, damit ich informiert bin. Ich nehme es dann zur Kenntnis und weise es dem Bearbeiter wieder zu.

Bugs können nicht nur von mir oder dem Team, sondern auch von allen anderen Teilnehmern der Veranstaltung gefunden und reportet werden. **Wer Bugs in einer Anwendung einer anderen Gruppe findet und einen Bug-Report im Redmine-Projekt der anderen Gruppe erstellt, bekommt einen Zusatzpunkt, der sich positiv auf die Note der Studienarbeit auswirkt.** Das Finden und Reporten eines solchen Bugs wirkt sich **nicht negativ** auf die Bewertung der Entwickler der Anwendung aus!

## Startphase

Die Startphase dient dazu sich als Team zu organisieren (Rollen verteilen etc.), eine Codebasis auszuwählen und vorzubereiten, ein Product Backlog in Form von Tickets in Redmine anzulegen und Arbeitsbereitschaft für die folgenden Sprints herzustellen.

Am Ende der Startphase ist folgendes zu liefern:

- Das gemeinsame Projekt (Code, Dokumente) ist im neuen Repository. Alle haben Zugriff und können pushen und pullen.
- Rollen aller Teammitglieder sind festgelegt und stehen in der Datei `README`, die von GitHub angezeigt wird.
- Die Datei `README` muss, mit Markdown formatiert werden.
- In Ihrem Redmine-Projekt ist ein Product-Backlog angelegt, dass voraussichtlich für beide Sprints ausreicht.
- Deployen Sie Ihre Anwendung so, dass Sie über das Internet zugreifbar ist. Dafür bietet sich z.B. Heroku an. Die URL unter der die Anwendung bis zum Ende des Semesters läuft, muss in der `README` stehen.

Das Repository ist zum Ende der Startphase mit dem Tag “0.1-RELEASE” zu versehen.

## Sprint 1 & 2

Die Sprints bestehen aus dem Sprint Planning, einem oder mehreren exemplarischen Daily Scrums, dem Sprint Review und der Sprint Retrospektive. Diese Meetings werden zu den festgelegten Zeiten jeweils im Praktikumsraum unter meiner Anwesenheit durchgeführt. Die anderen Gruppen sind als Gäste ohne Rederecht selbstverständlich willkommen.

Was in den Meetings und im Sprint zu tun ist, ist Teil Ihrer Aufgabe.

Als Mindestanforderung für Anwendung gilt die in der Studienarbeit zu Software Engineering I definierte “Funktionalität Version 2”:

- Funktionalität Version 1
- Produkt-Verwaltung (anzeigen, hinzufügen, ändern)
- Kategorien-Verwaltung
- Löschen von Kunden und Produkten (Löschen jedoch nur, wenn es noch keine zugehörige Bestellung gibt, andernfalls nur deaktivieren)
- Login-Verfahren mit Name und Passwort
- Pro Bestellung können mehrere Produkte bestellt werden
- Status der Bestellung mitführen (z.B. bestellt, in Lieferung, ausgeliefert, storniert)
- Weitere Auswertungen über Bestellungen
- Weitere Funktionen nach eigenem Wunsch

Das Repository ist zum Ende des ersten Sprints mit dem Tag "1.0-RELEASE" und zum Ende des zweiten Sprints mit dem Tag "1.1-RELEASE" oder mit dem Tag "2.0-RELEASE" zu versehen, je nachdem wie groß das Delta zwischen dem Release 1.0 und diesem ist. Ist ein neues Major-Release gerechtfertigt oder ist es eher ein Minor-Release.