Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Prof. Dr. Andreas Hackelöer

2. Mai 2024

Übungen zur Vorlesung Software Engineering I
Sommersemester 2024

Übungsblatt Nr. 4

(Abgabe in Teams von max. 3 Personen bis: Mittwoch, den 15. Mai 2024, **9:00 Uhr**)

Wichtige Anmerkung:

Wie in der ersten Vorlesung bereits angekündigt, finden in der kommenden Woche (KW 19) **keine Vorlesung und keine Übungen** statt, weil der 9.5. ein Feiertag (Christi Himmelfahrt) ist. Die Übung am 8.5. fällt aus organisatorischen Gründen ebenfalls aus!

Sie haben deshalb für die Bearbeitung dieser Übung eine Woche länger Zeit, die Abgabefrist endet erst am **15.5.2024** (siehe oben).

Aufgabe 1 (Anforderungen und User Stories, 8 Punkte):

1.1:
Bitte geben Sie jeweils ein Beispiel für eine funktionale, eine nicht-funktionale und eine technische Anforderung an. Das sollen natürlich andere Beispiele als die aus der Vorlesung sein.

1.2:
Bewerten Sie die folgenden User Stories nach den INVEST-Kriterien. Geben Sie dabei zu jedem „Buchstaben“ von INVEST bei jeder User Story an, ob das betreffende Kriterium eher erfüllt wird oder eher nicht. Geben Sie das Ergebnis Ihrer Einschätzung in tabellarischer Form an (Spalten: I N V E S T, Zeilen: U1 bis U5, Inhalt einer Zelle: „+“ oder „-“). Sie brauchen Ihre Einschätzung hier nicht zu begründen.

U1: „Als Kunde der Buchungsplattform möchte ich die Hintergrundfarbe von weiß auf grün umstellen können.“

U2: „Als Stammkunde des Webshops möchte ich ein Benutzerprofil anlegen können, um meine Adresse nicht bei jeder Bestellung erneut eingeben zu müssen.“

U3: „Als Datenanalyst möchte ich das Data Warehouse mindestens dreimal schneller als bisher abfragen können, um meine Analysen schneller durchführen zu können.“

U4: „Als CEO des Unternehmens möchte ich jederzeit auf jede in meinem Unternehmen erstellte Auswertung direkt und sofort zugreifen können, um bei Bedarf zu jeder Entscheidung schnell nachforschen zu können.“

U5: „Als Autofahrer möchte ich während der Fahrt eine Liste mit lohnenden Ausflugszielen in der Nähe im Navigationssystem angezeigt bekommen, um bei Interesse zu einem dieser Ziele fahren zu können."

1.3:
Geben Sie zu jeder dieser User Stories an, ob Sie sie in dieser Form für sinnvoll formuliert halten oder nicht und begründen Sie dies. Sie müssen dabei nicht auf jedes INVEST-Kriterium eingehen, sollten es aber herausstellen, wenn es offensichtliche Gründe gibt, warum eine User Story nicht oder nur extrem schwierig umsetzbar ist.

Aufgabe 3 (Implementierung, 22 Punkte):

Das Unternehmen NullPointer Software GmbH beauftragt Sie mit der Entwicklung eines Programms zur Verwaltung von User Stories. Dazu erhalten Sie folgende User Stories (US):

US1: „Als Entwickler möchte ich die notwendigen Daten einer User Story nacheinander über eine Kommandozeile eingeben können."

Erfassen Sie dazu die folgenden Daten:
- Eine eindeutige ID der User Story
- Eine schriftliche Beschreibung der User Story
- Eine Priorisierung der User Story nach dem MoSCoW-Prinzip
- Eine Menge von Tasks, die der User Story zugeordnet werden; ein Task hat ebenfalls eine eindeutige ID sowie eine schriftliche Beschreibung; Task-IDs brauchen nicht notwendigerweise disjunkt zu User-Story-IDs zu sein, müssen aber untereinander eindeutig sein

US2: „Als Entwickler möchte ich eine Übersicht über alle eingegebenen User Stories bekommen. Dabei sollen zu jeder User Story im System alle Tasks ausgegeben werden."

US3: „Als Entwickler möchte ich, dass alle Eingaben in einer Datei abgespeichert werden können, so dass ich sie nach Programmneustart wieder laden kann."

Weitere Vorgaben zu den User Stories:

- Alle Befehle sollen über eine Kommandozeile (keine GUI!) eingegeben werden können, nach dem Muster:
  `> befehl <parameter>`

- Als Befehle sind die folgenden vorgesehen:
  - `story`: Eingabe einer User Story **ohne** die Eingabe der zugeordneten Tasks

- `task`: Eingabe eines Tasks **ohne** Zuordnung des Tasks zu einer User Story
- `assign`: Zuordnung einer (weiteren) Task-ID zu einer User-Story-ID
- `stories`: Ausgabe aller gespeicherten User Stories, wobei zu jeder User Story alle zugeordneten Tasks ausgegeben werden
- `tasks`: Ausgabe aller gespeicherten Tasks, die zugeordnete User Story (sofern vorhanden) braucht dabei nicht mit ausgegeben zu werden
- `load`: Laden der gespeicherten User Stories aus einer Datei
- `save`: Speichern der eingegebenen User Stories in einer Datei

Beispiel-Interaktion mit dem System:

```
> story 1 "Erste User Story" should-have
> task 10 "Task für erste User Story"
> task 20 "Noch ein Task für erste User Story"
> stories
Die folgenden User Stories sind im System gespeichert:
ID: 1, Beschreibung: Erste User Story, Priorität: Should Have
Zugeordnete Tasks: keine
> tasks
Die folgenden Tasks sind im System gespeichert:
ID: 10, Beschreibung: Task für erste User Story
ID: 20, Beschreibung: Noch ein Task für erste User Story
> assign 1 10
Task mit ID 10 wurde erfolgreich User Story mit ID 1 zugeordnet.
> assign 1 20
Task mit ID 20 wurde erfolgreich User Story mit ID 1 zugeordnet.
> stories
Die folgenden User Stories sind im System gespeichert:
ID: 1, Beschreibung: Erste User Story, Priorität: Should Have
Zugeordnete Tasks:
Task ID: 10, Beschreibung: Task für erste User Story
Task ID: 20, Beschreibung: Noch ein Task für erste User Story
> save
User Stories und Tasks wurden erfolgreich gespeichert.
> load
User Stories und Tasks wurden erfolgreich geladen.
```

Die Beispiel-Interaktion dient nur als Richtlinie; Sie können das System auch mit einer anderen Syntax und anderen Formulierungen ausgestalten.

Test:
Entwickeln Sie JUnit-Tests für die einzelnen Klassen bzw. Methoden mit hinreichender Testabdeckung. Testen Sie auch die Ende-zu-Ende-Funktionalität Ihrer Anwendung (Test der Anwendung „als Ganzes“). Eine Test-Automatisierung ist nicht erforderlich.