Agile Softwareentwicklung mit SCRUM Refactoring 1 with M. Fowler

Version 0.4

01.04.2022

# 

# Dokumentenversionen

|  |  |  |  |
| --- | --- | --- | --- |
| Versionsnr. | Datum | Autor | Änderungsgrund / Bemerkungen |
| 0.1 | 02.06.2020 | Dr. Marc Schanne | Ersterstellung |
| 0.2 | 20.06.2020 | Gennaro Izzo | Review |
| 0.3 | 13.02.2021 | Dr. Marc Schanne | Portfolio-Prüfung SE II |
| 0.4 | 01.04.2022 | Dr. Marc Schanne | Update Epics, User Stories |

Inhalt

[Dokumentenversionen 1](#_Toc64108250)

[Einleitung 3](#_Toc64108251)

[Allgemeines 3](#_Toc64108252)

[Epic 1: Projektplanung mit Meilensteinen und Definition von Artefakten 3](#_Toc64108253)

[User-Story 1: 3](#_Toc64108254)

[Epic 2: Refactorings entsprechend der Vorgaben von Martin Fowler 5](#_Toc64108255)

[User-Story 2: 5](#_Toc64108256)

[User-Story 3: 5](#_Toc64108257)

[User-Story 4: 5](#_Toc64108258)

[User-Story 5: 5](#_Toc64108259)

[User-Story 6: 5](#_Toc64108260)

# Einleitung

Diese Projektbeschreibung ist Teil der Portfolio-Prüfung des Moduls Software Engineering II an der DHBW Karlsruhe. Die Prüfungsleistung als Portfolio enthält zwei Workshops zum Nachweis der Lerninhalte aus den Semestern 3 und 4, sowie Grundlagen aus dem Modul Software Engineering I.

### Allgemeines

In diesem Dokument werden Anforderungen für eine agile Umsetzung des Refactoring-Projekts in Epics und User-Stories mit SCRUM vorgegeben.

# Epic 1: Projektplanung mit Meilensteinen und Definition von Artefakten

Projektumsetzung mithilfe agiler Methoden und zugehöriger Artefakte

* Backlog muss vorliegen
  + User-Stories sollten in Sprints verteilt werden (Sprint Planning)
  + User-Stories sollten in entsprechende Tasks eingeteilt werden
  + Tasks sollten nachweislich umgesetzt werden (siehe Pushes)
* Durchführung eines Backlog-Groomings pro Sprint
  + Hierzu gehört auch die Bestimmung der Story-Points (ggf. Story Point Poker?)
* Durchführung eines Sprint Plannings vor Beginn eines jeden Sprints
* Daily zum Abgleich der Arbeit in einem Sprint können spontan eingebaut werden, Ziel ist es die eigene Arbeit und besonders die Abhängigkeit zu anderen Tasks im Team bekannt zu machen.
* Sprint-Review sollte durchgeführt werden
* Github o.ä. um Code-Iterationen (Pushes) zu erkennen (sollten zu User-Stories, bzw. den daraus abgeleiteten Tasks passen)

Die hierfür erforderliche Arbeit eines Product-Owners wurde schon teilweise abgenommen, dies beinhaltet:

* Erstellen relevanter User-Stories aus dem entsprechenden Haupt-Epic
* Priorisieren der User-Stories nach diversen Faktoren (Bearbeitungs-, bzw. Relevanz-Priorisierung wird durch die Nummerierung, bzw. Reihenfolge der Refactoring-User-Stories vorgegeben)

Folgende Punkte sind für die vorhandenen User-Stories noch zu erledigen (Definition of Ready):

* Story-Point-vergabe je User-Story
* Verständnis im Team
* DoD geklärt?

Aufbau einer User-Story:

Als [Nutzer / Rolle] möchte ich [was?], dass / weil / um [warum? / Nutzen / Wert].

### User-Story 1:

Als „Software Quality Engineer“ möchte ich, dass sich das Entwicklungsteam im zu bearbeitenden Coding auskennt, um eine hohe Entwicklungsqualität und -geschwindigkeit zu gewährleisten.

* Beispielstask: Zusammensetzen und Code (IST-Stand) gemeinsam analysieren.
* Definition of Ready:
  + Definition of Done geklärt (Projektübergreifend)
  + Story-Points vergeben (Grooming)
  + jedem Teammitglied ist klar, dass es darum geht, den Code zu verstehen
  + In Task(s) unterteilt -> z.B. Einrichten der Entwicklungsumgebung + erstes öffnen des Projekts (jeder im Team) + gemeinsame Analyse / selbstständige Analyse etc.

# Epic 2: Refactorings entsprechend der Vorgaben von Martin Fowler

Die in den Folien von Martin-Fowler beschriebenen Arbeitsschritte werden im Folgenden in Form von User-Stories abgebildet (siehe Anforderung 1).

## User Story 2:

Als Kunde erwarten wir, dass der Bestandscode für zukünftigen Erweiterungen vorbereitet wird. Die von Martin Fowler beschriebenen Refactorings müssen dafür identifiziert und umgesetzt werden.

Tasks:

* Das Entwicklungsteam muss ein Verständnis für das bestehende design und die geplanten Erweiterungen aufbauen.
* Die von Martin Fowler empfohlen Refactorings sind auf den Code abzubilden und in konkreten User Stories für die kommenden Sprints darzustellen.

### User-Story 3:

Als „Software Quality Engineer“ möchte ich, dass die switch-Anweisung zur Preisberechnung in eine eigene Methode ausgelagert wird, sodass hierdurch die Übersichtlichkeit und Wartbarkeit des Codes gesteigert wird.

Tasks:

* Amount-Calculation in eigene Methode auslagern (entsprechende Benennung)
* Aufruf der neuen Methode in altem Code
* Interne Variablen der extrahierten Methode umbenennen

### User-Story 4:

Als „Software Quality Engineer“ möchte ich, dass sich die Methode zur Preisberechnung in der hierzu geeigneten Klasse befindet, sodass die Kohäsion des Codes erhöht wird.

Tasks:

* Methode in Rental-Klasse verschieben
* Methodenaufruf anpassen

### User-Story 5:

Als „Software Quality Engineer“ möchte ich, dass temporäre Variablen vermieden und die Methode zur Preisberechnung soweit möglich als Query aufgerufen wird, sodass die Code-Menge reduziert wird.

### User-Story 6:

Als „Software Quality Engineer“ möchte ich, dass die Berechnung der frequentRenterPoints in die relevante Klasse ausgelagert (Kombination aus Extract und Move Method), um ???

### User-Story 7:

Als „Software Quality Engineer“ möchte ich, dass die temp-Variablen „totalAmount“ und „frequentRenterPoints“ aus dem Code entfernt werden, um ???

**Ab hier sollten die Studenten die User-Stories selbstständig beschreiben.**