Duale Hochschule
Baden-Württemberg
Karlsruhe

LASTENHEFT

Version: 0.2

Datum: 20.06.2020

DOKUMENTVERSIONEN

Versionsnr.	Datum	Autor	Änderungsgrund / Bemerkungen
0.1	25.05.2020	Dr. Marc Schanne	Ersterstellung
0.2	20.06.2020	Gennaro Izzo	Review

INHALT

DOKUMENTVERSIONEN 1
INHALT 2
1. Einleitung 3
1.1 Allgemeines 3
1.1.1 Ziel und Zweck dieses Dokuments 3
1.1.2 Projektbezug 3
1.1.3 Abkürzungen 3
1.1.4 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten 3
1.2 Verteiler und Freigabe 3
1.2.1 Verteiler für dieses Lastenheft 3
1.3 Reviewvermerke und Meeting-Protokolle 4
1.3.1 Erstes bis n-tes Review ..... **Fehler! Textmarke nicht definiert.**
2. Konzept und Rahmenbedingungen ..... **Fehler! Textmarke nicht definiert.**
2.1 Benutzer / Zielgruppe ..... **Fehler! Textmarke nicht definiert.**
2.2 Ziele des Anbieters ..... **Fehler! Textmarke nicht definiert.**
2.3 Ziele und Nutzen des Anwenders ..... **Fehler! Textmarke nicht definiert.**
2.4 Systemvoraussetzungen ..... **Fehler! Textmarke nicht definiert.**
2.5 Ressourcen ..... **Fehler! Textmarke nicht definiert.**
3. Anforderungsbeschreibung 5
3.1 1. Anforderung 5
3.1.1 Beschreibung 5
3.1.2 Wechselwirkungen 5
3.1.3 Risiken 5
3.1.4 Vergleich mit bestehenden Lösungen ..... **Fehler! Textmarke nicht definiert.**
3.1.5 Schätzung des Aufwands 5
3.2 2. Anforderung 5
3.2.1 Beschreibung 5
3.2.2 Wechselwirkungen 6
3.2.3 Risiken 6
3.2.4 Vergleich mit bestehenden Lösungen 6
3.2.5 Schätzung des Aufwands ..... **Fehler! Textmarke nicht definiert.**
4. Genehmigung 8
5. Anhang 9

1. EINLEITUNG

Dieses Lastenheft ist Teil der Prüfung des Moduls Software Engineering II im Jahrgang WWI17 an der DHBW Karlsruhe. Die Prüfungsleistung als Portfolio enthält zwei Workshops zum Nachweis der Lerninhalte aus den Semestern 5 und 6, sowie Grundlagen aus dem Modul Software Engineering I.

1.1 Allgemeines

In diesem Dokument werden konkrete Anforderungen für die Umsetzung in einem Phasen- bzw. Dokument-getriebenen Softwareenwicklungsprozess (Wasserfall) vorgegeben.

1.1.1 Ziel und Zweck dieses Dokuments

Ziel dieser Spezifikation ist es Anforderungen des Auftraggebers (AG) an das vorliegende Refactoring-Projekt festzuhalten, damit der Auftragnehmer (AN) auf dieser Grundlage eine Lösung in einem Pflichtenheft spezifizieren kann.
Das Pflichtenheft ist dann Basis aller weiteren vertraglichen Vereinbarungen, insbesondere für die Definition der Abgabe-Artefakte mit entsprechenden Meilensteinen.

1.1.2 Projektbezug

Das Refactoring-Projekt entspricht dem Code-Beispiel aus dem Buch von Martin Fowler. Refactoring: Improving the Design of Existing Code, Addison-Wesley, 1999, 1. Auflage.

1.1.3 Abkürzungen

TBC …

1.1.4 Ablage, Gültigkeit und Bezüge zu anderen Dokumenten

Dieses Lastenheft ist die Basis für ein vom AN zu erstellenden Pflichtenhefts, das in klarer Form beschreibt, wie er die Anforderungen des Lastenhefts zu lösen gedenkt.
Eine weitere Dokumentation der notwendigen Refatorings findet sich in Martin Fowlers Vortrag auf der JavaOne 2000.
Ausgangspunkt für die Refactorings ist ein Code-Repository auf Github. Hier sind neben dem Code auch dieses Lastenhefts, Vorlagen für ein Pflichtenheft und die Präsentation von Martin Fowler abgelegt.

1.2 Verteiler und Freigabe

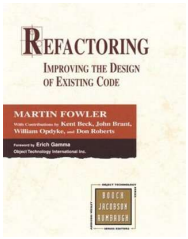
1.2.1 Verteiler für dieses Lastenheft

Rolle	Name	Telefon	E-Mail	Bemerkungen
Projektleiter	Dr. Marc Schanne	015 771 750 265	marc@schanne.org	
Kunde, AG	Gennaro Izzo			
Projektteam, AN	???			

1.3 Reviewvermerke und Meeting-Protokolle

Der im Github Repository hinterlegte Code enthält alle von Martin Fowler für das Refactoring-Beispiel seines Buches spezifizierte Klassen in Java, ab Version 8.
Die verwendete Projektstruktur nutzt Apache Maven als Build-System und NetBeans 11 als Entwicklungsumgebung.

2. ANFORDERUNGSBESCHREIBUNG



Die Anforderungen dieses Refactoring-Projekts betreffen ein Filmverleih-Projekt wie es Martin Fowler in seinem Buch Refactoring: Improving the Design of existing Code aus dem Jahr 1999 als Beispiel einführt.
Die Anforderungen sind aus Kundensicht und für das anstehende Refactoring miss der AN diese sinnvoll verfeinern und in einem Pflichtenheft die gesamte Entwicklung mit Meilensteinden planen.

2.1 1. Anforderung

Nr. / ID	101	Nichttechnischer Titel	Projektplanung mit Meilensteinen und Definition von Artefakten		
Quelle		Verweise		Priorität	muss

2.1.1 Beschreibung

Vor Umsetzung des geforderten Refactorings (Anforderung 102) und notwendiger Erweiterungen (Anforderungen 105) ist die vollständige Planung des Projekts (Anforderungen 101, 103, 104) erforderlich.

2.1.2 Wechselwirkungen

Artefakte des vorliegenden Refactoring-Projekts sind neben einer vollständigen und kleinschrittigen Projekt-Historie in einer Software-Versionsverwaltung (z.B. Github) zu archivieren. Dies umfasst insbesondere eine saubere Projektplanung mit Pflichtenheft und Meilensteinplan.

2.1.3 Risiken

Die sinnvolle Dokumentation des Refactoring-Fortschritts durch Commit-Messages ist obligatorisch. Die dadurch entstehende Projekt-Historie muss dem AG eine Abnahme des Projekts in den definierten Meilensteinen ermöglichen.

2.1.4 Schätzung des Aufwands

Für das gesamte Projekt steht dem AN nach Übergabe dieses Lastenhefts die Entwicklungszeit im Rahmen eines Workshoptages mit 4 Zeitstunden zur Verfügung. Die sinnvolle Strukturierung der notwendigen Arbeiten in 4 Meilensteinen (Zeitaufwand je 1 Stunde) ist somit angeraten.

2.2 2. Anforderung

Nr. / ID	102	Nichttechnischer Titel	Refactoring anhand von Code-Smells		
Quelle		Verweise		Priorität	muss

2.2.1 Beschreibung

Funktionale Hauptanforderung des Refactoring-Projekts ist das „Cleanup“ des bestehenden Beispiel-Projekts. Die Verwendung moderner OO-Programmierrichtlinien und eine Software-entwicklung nach bekannten Best Practices muss das Ziel sein, damit der Code auch für zukünftige funktionale Erweiterungen und Anpassungen vorbereitet ist.

Refactorings analog der Präsentation von Martin Fowler sind:

- Extract Method
- Move Method
- Replace Temp with Query
- Replace Type Code with State/Strategy
- Replace Switch with Polymorphism
- Form Template Method

2.2.2 Wechselwirkungen

Die von Martin Fowler in seinem Buch, bzw. in der Präsentation auf der JavaOne 2000 vorgestellten Refactorings gelten als Richtlinie für die notwendigen Code-Verbesserungen durch den AN.
Mit Wissen über übliche Code-Smells muss der AN die bestehenen Unzulänglichkeiten des Beispielcodes beseitigen.

2.2.3 Risiken

Mittels Definition geeigneter Tests muss für das Refactoring eine geeignete Basis geschaffen werden, damit der bestehende Code ohne Sorge um funktionale Veränderungen überarbeitet werden kann.

2.2.4 Vergleich mit bestehenden Lösungen

Die Nutzung bekannter Entwurfsmuster in der objektorientierten Programmierung sind obligatorisch.

2.3 3. Anforderung

Nr. / ID	103	Nichttechnischer Titel	Softwareentwicklung nach Wasserfall		
Quelle		Verweise		Priorität	muss

2.3.1 Beschreibung

Die Softwareentwicklung mit einem Entwicklungsprozess nach Wasserfall-Modell ist obligatorisch und der AN muss notwendigen Dokumente für den Übergang der einzelnen Phasen im Entwicklungsprozess explizit nutzen.

2.3.2 Wechselwirkungen

Genutze Dokumente muss der AN in einer Software-Versionsverwaltung archivieren.

2.3.3 Vergleich mit bestehenden Lösungen

Auch wenn der klassische Prozess des Softwareentwicklung nach Wasserfall die sequentielle Entwicklung mit Weitergabe von Informationen innerhalb des (großen) Entwicklungsteams über Dokumente vorsieht ist für die Programmierung im kleinen Team auch die Nutzung von modernen Prozessmerkmalen denkbar.

Ansätze wie Par Programming (zwei Entwickler vor einem Rechner, ein Driver, ein Navigator, im Wechsel), Test Driven Development (TDD) oder Continuous Integration (z.B. mit Jenkins) sind hier erlaubt.

2.4 4. Anforderung

Nr. / ID	104	Nichttechnischer Titel	Werkzeugumgebung		
Quelle		Verweise		Priorität	soll

2.4.1 Beschreibung

Moderne Softwareentwicklung verlangt die Nutzung geeigneter Werkzeugunterstützung für Entwicklung (IDE), Build-Management, Software-Versionsverwaltung und Tests.

2.5 5. Anforderung

Nr. / ID	105	Nichttechnischer Titel	Funktionale Erweiterungen		
Quelle		Verweise		Priorität	kann

2.5.1 Beschreibung

Ausgangspunkt des Refactoring-Projekts ist eine schlecht strukturiertes Programm. Nach Refactoring (Anf. 102) ist es möglich die Funktionalität der Customer-Komponente zu erweitern und leicht neue Typen für die Komponente Movie mit speziellem Verhalten zu ergänzen ohne die bestehende Logik verändern zu müssen.
Bei diesen Erweiterungen ist das Open Closed Principle (OCP) – offen für Erweiterungen, geschlossen für Veränderungen – zu beachten.

2.5.2 Wechselwirkungen

Diese Erweiterungen sind ebenfalls im Rahmen des Vortrags von Martin Fowler beschrieben und müssen für die Spezifikation des Pflichtenhefts genutzt werden.

3. GENEHMIGUNG

Die Genehmigung erfolgt...

Datum:	
Unterschrift Auftraggeber:	
Unterschrift Projektleiter:	
Weitere Unterschriften:	

4. ANHANG