Duale Hochschule
Baden-Württemberg
Karlsruhe

LASTENHEFT

Version: 0.2

Datum: 27.06.2020

DOKUMENTVERSIONEN

Versionsnr.	Datum	Autor	Änderungsgrund / Bemerkungen
0.1	25.05.2020	Dr. Marc Schanne	Ersterstellung
0.2	27.06.2020	Gennaro Izzo	Review

INHALT

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

Ziel dieses Softwareentwicklungs- und Refactoring-Projektes ist es eine Java-Implementierung des Programmierspiels Core War (Krieg der Kerne) durch ein umfangreiches Refactoring zu renovieren und für die erfolgreiche Nutzung auf dem modernen Desktop aufzubereiten.

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 außer einer mit Java, ab Version 8, lauffähigen Softwareversion dieses Lastenheft sowie weitere Dokumentation zum Core War und den dafür notwendigen Spezifikationen.

2. ANFORDERUNGSBESCHREIBUNG

Ausgangspunkt für das vorliegende Refacoring-Projekt ist die Implementierung des bekannten Programmierspiels Core War in einem Java Applet aus den Neunziger-Jahren des letzten Jahrtausends.
Core War ist ein Programmierspiel das A. K. Dewdney im Mai 1984 als Zerstreuung für Programmierer erfunden hat. Hintergrund war der Aufstieg von Computer-Viren und der Kampf gegen diese unerwünschten „Programme“ auf modernen Peronal Computern mit Anti-Viren-Programmen.
Dazu hat Dewdney einen virtuellen Computer mit einem eindimensionalen Ringspeicher definiert, in dem zwei oder mehr Programme, die in einer assemblerartigen Sprache (Redcode) geschrieben sind, als Krieger im gemeinsamen Speicher (MARS, Memory Array Redcode Simulator) ausgeführt werden.
Die International Core War Society (ICWS) hat 1984 und 1988 Revisionen von Redcode veröffentlicht und ein Update-Vorschlag von 1994 wurde zwar als Draft veröffentlicht, aber nicht final verabschiedet.
Als Referenzimplementierung für MARS mit einem Redcode-Interpreter (in Teilen bereits für die Version 1994), der die Ausführung der Kämpfer simuliert, wurden über die Jahre auf sourceforge.net in C als portable Memory Array Recode Simulator (pMARS) entwickelt.
Diverse Neuimplementierung (z.B. nMARS) aber auch eine Online-Version unter https://crypto.stanford.edu/~blynn/play/redcode.html sind mittlerweile (noch) verfügbar.
Ausgangspunkt dieses Refactoring-Projekts ist eine Implementierung in Java (jMARS), die als Java-Applet und SWT-Applikation verfügbar war. Für die Nutzung mit modernem Java ist der Code in einem Github Repository als einfache AWT-Applikation restauriert und mit einer Projektstruktur in Apache Maven als Apache NetBeans Projekt aufbereitet worden.

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	

2.2.1 Beschreibung

Der vorliegende Programmcode von jMARS (Github) kann seinen Ursprung als Applet nicht verleugnen. Ziel des Refactoring-Projektes ist es die Änderung hin zu einer reinen Desktop-Applikation abzuschließen und den Code für die funktionale Weiterentwicklung (105) vorzubereiten. Die Verwendung moderner OO-Programmierrichtlinien und eine Softwareentwicklung nach bekannten Best Practices muss das Ziel sein.
Wünschenswerte Änderungen sind betreffen insbesondere:
Corewars.jmars.jMARS – das Hauptprogramm: Wie kann die Weitergabe der Konfiguration von der Kommandozeile hier nicht über Instanzvariablen in Methoden ohne Argumente erfolgen? Wie kann außerdem die Trennung von Logik für den Verarbeitungsthread des MARS und den Start des Programms (mit oder ohne GUI) umgesetzt werden?
Die Trennung von Verantwortlichkeiten (für Klassen bzw. Methoden) sollte das Ziel sein.
Die Statistik-Ausgabe ist ebenfalls fest in den MARS-Thread integriert, wie könnte das entzerrt werden, eine lose Kopplung zwischen MARS-Ausführung, der Anzeige und der Ergebnisausgabe soll das Ziel sein.

2.2.2 Wechselwirkungen

Anders als bei der Umsetzung des letzten Refactoring-Projektes am Beispiel von Martin Fowler gibt es für das „Cleanup“ von jMARS keine detaillierte Vorgabe der notwendigen Code-Verbesserungen.
Mit Wissen über übliche Code-Smells muss der AN die bestehenden Unzulänglichkeiten des Projektcodes beseitigen.
Sinnvolle Refactorings finden sich bei der Weitergabe von Parametern und Ergebnissen über Zustände, die Kapselung von Code und die Struktur des Kontrollflusses allgemein.

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	

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	

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	Erweiterung des Simulators		
Quelle		Verweise		Priorität	kann

2.5.1 Beschreibung

Das „Cleanup“ des Refactoring-Projekt steht im Fokus der Entwicklung. Durch die Umgestaltung des Codes (Anf. 102) soll insbesondere die bestehende Logik (d.h., Ausführung der Kämpfer) nicht verändert werden.

Mögliche Erweiterungen des Simulators, die mit dieser Anforderung gewünscht werden beziehen sich auf die folgenden Punkte:

- Interaktionsmöglichkeit mit der Applikation (z.B. Laden und Starten der Kämpfer)
- Graphische Darstellung und Ausführung der Kämpfer (evtl. zum Debugging)
- Ausgabe der Ergebnisse (ausführlichere Statistik)

2.5.2 Wechselwirkungen

Die geplanten Erweiterungen der jMARS-Applikation müssen als Teil des Pflichtenhefts beschrieben werden.

2.5.3 Risiken

Änderungen und Erweiterungen am Code dürfen auf keinen Fall die bestehende Logik für den Kampf zweier Programme und die statistische Auswertung zerstören.
Ein aktueller Test des Projekts führt zwei Kämpfer in der Arena mit einem eindeutigen Sieger (Next Dwarf) aus.
Sonstige automatisierte Tests gibt es leider nicht, bevor man Änderungen an interner Logik durchführt sollte man diese ergänzen und bei Änderungen der statistisch ausgabe ist der bestehende test anzupassen.

3. GENEHMIGUNG

Die Genehmigung erfolgt...

Datum:	
Unterschrift Auftraggeber:	
Unterschrift Projektleiter:	
Weitere Unterschriften:	