# SOFTWARE REQUIREMENTS SPECIFICATION

# for

# MELT Chess

**Version 0.1**

**23. April 2021**

# Inhaltsverzeichnis

# 1 Einführung

## 1.1 Projektanforderungen

Entwickelt werden soll ein Schach Programm, welches es ermöglicht gegeneinander als auch gegen eine künstliche Intelligenz Schach zu spielen. Das Spiel soll dafür sowohl über eine graphische als auch über eine konsolenbasierte Benutzerschnittstelle verfügen. Das Spiel soll in englischer Sprache umgesetzt werden. Die Entwicklung soll sich dabei in drei Iterationen gliedern. In diesem Dokument sind die Anforderungen an die erste Iteration dargelegt.

## 1.2 Zielgruppe des Dokuments und weitere Resourcen

Dieses Anforderungsdokument richtet sich zum einen an die beteiligten Entwickler und dient zur Orientierung ob die Funktionalität des Projekt gemäß den Anforderungen umgesetzt wird, und zum anderen an die Kontaktperson(en) des Moduls um die Planung des Projekts zu überprüfen.

Einen weiteren Überblick bieten die Storycards, welche im Gitlab des Projekts zu finden sind und in Form von Issues umgesetzt werden.

# 2 Umfang des Projekts

## 2.1 Funktionale Anforderungen

Zum Umfang gehört in der 1. Iteration des Projekts die Bereitstellung einer Textbasierten Konsolen-Schnittstelle in der es möglich ist Mensch-gegen-Mensch Spiele zu spielen. Dazu ist die Umsetzung des Pakets model nötig, in dem der Zustand des Bretts sowie die Schachregeln[1] implementiert werden.

In der 2. Iteration wird die geschaffene Basis durch eine 2D-GUI unter Verwendung des JavaFX Moduls erweitert und eine rudimentäre Schach KI im Modul engine erstellt.

In der 3. Iteration soll die Anwendung um Netzwerkfähigkeit mit der Anwendung der Gruppe 2 erweitert werden. Zusätzlich soll eine Auswahl von möglichen Erweiterungen umgesetzt werden, welche in Summe einen Aufwand von mindestens 10 Einheiten aufweisen müssen. Die Auswahl mit möglicher Punktzahl lauten:

- Verbesserte KI mithilfe Min-/Max-Suche mit $\alpha/\beta$-Pruning (5)
- 3D-GUI (5)
- Eindimensionales Schach (5)
- Rückgängig-Machen von Zügen (3)
- Speichern/Laden von Spielen (3)
- Schachuhren (2)
- Zweisprachigkeit (2)
- Resizeable GUI (1)

## 2.2 Nicht-funktionale Anforderungen

### 2.2.1 Codingstyle, Metriken, Testabdeckung

Das Template verwendet die Plugins PMD und JaCoCo zur Generierung von Reports über Metriken, Codestyle und Testabdeckung. Es wird eine Testabdeckung des gesamten Codes von 90% Instruction Coverage erwartet, ausgenommen der GUI-Klassen. Im Template wird ein PMD-Regelsatz eingebunden. Diese Regeln sind für den Code und die Testfälle einzuhalten.

[1] Nach den Regeln des Weltschachverbands (FIDE) in der deutschen Übersetzung von 2018

### 2.2.2 Überprüfungen

Es soll mindestens alle zwei Wochen ein Treffen mit dem zugewiesenen Tutor stattfinden. Neben den Deadlines zu den drei Iterationen gibt es außerdem noch folgende Termine:

- 23.04.2021: Abgabe Anforderungsanalyse, Vorgehensplan, Prüfung erfolgreiche Einrichtung der Infrastruktur
- 19.05.2021: Prüfung, ob auslieferbare Version vorliegt, die Anforderungen der ersten Iteration genügt mit automatisierten Tests und Theorie-Abfrage
- 16.06.2021: Prüfung, ob auslieferbare Version vorliegt, die Anforderungen der zweiten Iteration genügt mit Zwischenpräsentation
- 14.07.2021: Endabgabe mit Abschlusspräsentation

### 2.2.3 Abgabeformat

An den Schlusstagen der Iterationen, inklusive Endabgabe, muss der zu überprüfende/bewertende Stand mittels Tags (it1,it2,...) im git-Repository gekennzeichnet werden.

# 3 Anhang

## 3.1 Tabellarische Anforderungsanalyse

| Klassifikation | Priorität | Kategorie | Kriterium | Iteration |
|---|---|---|---|---|
| funktional | must | Hauptfunktion | gegen eine KI spielen | 2 |
| funktional | must | Hauptfunktion | gegen einander spielen | 1 |
| funktional | must | Schnittstelle | graphisch | |
| funktional | must | Schnittstelle | konsolenbasiert | 1 |
| funktional | must | Sprache | Englisch | |
| funktional | must | Züge | nicht zulassen ungültiger Züge | 1 |
| funktional | must | GUI | 2D | 2 |
| funktional | should | Features | Verbesserte KI mithilfe Min-/Max-Suche mit α/β\alpha/\betaαα/β-Pruning (5) | 3 |
| funktional | should | Features | 3D-GUI (5) | 3 |
| funktional | should | Features | Eindimensionales Schach (5) | 3 |
| funktional | should | Features | Rückgängig-Machen von Zügen (3) | 3 |
| funktional | should | Features | Speichern/Laden von Spielen (3) | 3 |
| funktional | should | Features | Schachuhren (2) | 3 |
| funktional | should | Features | Zweisprachigkeit (2) | 3 |
| funktional | should | Features | Resizeable GUI (1) | 3 |
| funktional | must | Hauptfunktion | Netzwerkspiel | 3 |
| nicht-funktional | must | Codequalität | Einhaltung der Metriken | |
| nicht-funktional | must | Codequalität | Einhaltung Style-Convention | |
| nicht-funktional | should | Codequalität | Kommentierung des Codes | |
| Nebenbedingung | must | Testabdeckung | Abdeckung von 90 % des Codes (ohne GUI Klassen) durch JUnit-Tests | |
| nicht-funktional | must | Dokumentation | Stets aktualisierte Anforderungsdokumentation via Story-Cards | |
| nicht-funktional | must | Dokumentation | Stets aktualisierte Dokumentation der Architektur unter Zuhilfenahme von UML-Diagrammen (Klassen-, Objekt-, Sequenz-, Zustand-) | |
| nicht-funktional | must | Dokumentation | Stets aktualisierte Javadoc-Dokumentation des Programms | |
| nicht-funktional | must | Dokumentation | Stets aktualisierte Dokumentation der Programmverwendung (Bedienungsanleitung) | |
| Nebenbedingung | must | Sprache | Java, Version 11 | |
| Nebenbedingung | must | Bibliothek | JavaFX | |
| Nebenbedingung | could | Bibliothek | externe Bibliotheken für die Umsetzung 3D-GUI | |
| Nebenbedingung | must | Version | Versionsverwaltung in Git | |
| Nebenbedingung | must | Build-Managment | Maven | |