Anforderungsanalyse

Anforderungsanalyse der Gruppe 29 (Christin Krause, Jan Rehfeld, Sven Wolff)

Übersicht

Im nachfolgenden Dokument werden die Anforderungen für das Projekt entsprechend der im Modul *Praktikum Softwareengineering* im Sommersemester 2020 festgelegten Spezifikationen vom 08.04.2020 festgehalten. Bezüge zum Spezifikationsdokument sind im Nachfolgendem durch Querverweise gekennzeichnet. Das Ziel ist die Entwicklung eines plattformunabhängigen Schachspiels auf Basis von Java 11.

Änderungen der Spezifikationen vom 08. Juni 2020 für die dritte Iteration sind in Rot ergänzt worden.

Vorgehen und nicht-funktionale Anforderungen

Wir arbeiten prototyporientiert in drei Iterationen, an dessen Ende ein lauffähiger Prototyp entstehen soll:

1) Festlegen der Anforderungen, Textbasierte Konsolenschnittstelle, Mensch-gegen-Mensch-Spiele und Nichtzulassen ungültiger Züge (2.1)
2) 2D-GUI, Spiel gegen den Computer (2.2)
3) Optionale Features, die erste im Lauf der Entwicklung festgelegt werden (z.B. Speichern und Laden, Schachrätsel, etc.), sowie ein vorerst unbekanntes Feature (2.3). Das optionale Feature ist das Wechseln der Sprache und eine ‚schlaue' Schach-KI, die ihre Züge mittels min/max Suche mit Alpha/Beta-Pruning in einer Tiefe von mindestens 3 berechnet (5.6.1).

Individuelle Entwicklungsmeilensteine werden zudem separat innerhalb der Bearbeitergruppe festgelegt. Der Zeitplan ist vom 08.04.2020 (Kick-off) bis zum 05.07.2020 (Endabnahme) ausgelegt.

Jede Iteration wird durch eine Zwischenabnahme durch den Kunden, im Nachfolgenden als „Prüfer" bezeichnet, abgeschlossen. Dazu wird dem Prüfer ein lauffähiger und mittels Unit-Tests verifizierter Prototyp vorgelegt. Zum Projektabschluss wird dem Prüfer das Endprodukt im Rahmen einer Präsentation übergeben. Die abzugebenden Dateien sind dazu im Git-Repository mittels der Tags „it1", „it2" oder „it3" gekennzeichnet (4.2).

Zum Projektmanagement wird eine Versionsverwaltung genutzt (Git) mit zentralem Repository im GitLab-Server des ISP (3.3). Ebenso wird für die generelle Verwaltung ein Projekt im Redmine-Server des ISP genutzt (3.3). Wartung und Sicherung des Servers fallen unter die Kompetenzen des ISP.

Eine hohe Qualität der Software wird durch entsprechendes Testen mit einer Testabdeckung von mindestens 90 % (ohne GUI Klassen in View und Controller) mittels JUnit sichergestellt (2.4.2). Zudem werden Style-Conventions eingehalten (2.4.1) oder mit einer guten Begründung abgeändert. Die Lesbarkeit des Quellcodes durch die Überprüfung festgelegter Metriken und die Testabdeckung wird durch Continious Integration sichergestellt (2.4.1; 3.3). Die Dokumentation wird durch nötige Kommentare des Quellcodes (2.4.1), sowie JavaDoc (2.4.3) und einer Bedienungsanleitung mit Spielregeln im Git bereitgestellt (2.4.3).

Ein langfristiger Betrieb durch Kommerzialisierung ist nicht vorgesehen, daher wird es keine weitere Wartung oder Addition der Software um neue Features nach der Endabnahme geben.

Systemumfang und funktionale Anforderungen

Das fertige Programm umfasst ein komplettes Schachspiel, das im GUI-Modus durch ein Startfenster initialisiert wird. Optional zu implementierenden Features sind entsprechend gekennzeichnet. Das Starten des Programms erfolgt durch einen Konsolenaufruf. Dabei kann über die Eingabe --no-gui das Spiel als reine Konsolenanwendung gestartet werden (5.1). Im (~~optional 3D; 5.5.2~~) GUI-Modus wird man auf ein Spielfeld weitergeleitet (Abbildung 1, 5.3.2), ~~wobei sich das Fenster auf eine beliebige Größe ziehen lässt (optional; 5.5.9).~~

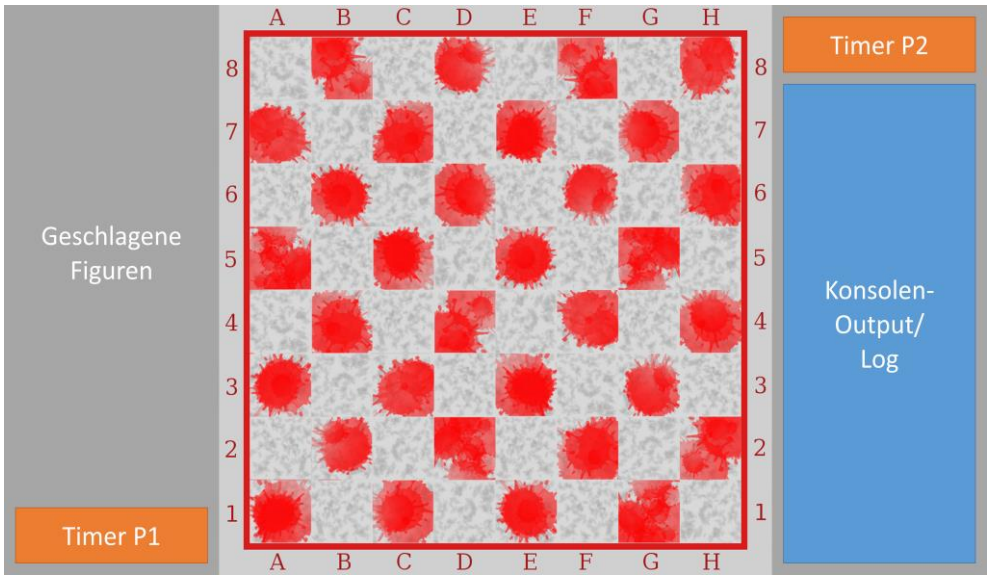

Abbildung 1: Big Picture, GUI-Konzept

Die Funktionalität ist im Anwendungsfalldiagramm (Abbildung 2) dargestellt. Zunächst startet der menschliche Spieler ein neues Spiel ~~oder lädt einen bereits vorhandenen Spielstand (optional; 5.5.4)~~. ~~Neben einem herkömmlichen Schachspiel oder Blitzschach gegen andere Menschen oder eine (optional sehr kluge, 5.5.1) Schach-KI (5.2.1; 5.3.1) können Schachrätsel gespielt werden (optional; 5.5.3)~~. Der Spieler kann entscheiden, ob er gegen einen anderen Menschen spielen will, oder gegen eine einfache oder kluge Schach-KI (5.2.1; 5.3.1). ~~Gegen andere Menschen kann man entweder lokal oder über eine Netzwerkverbindung spielen (optional; 5.5.7).~~ Als Zusatzfeature lässt sich die Sprache jederzeit von Deutsch auf Englisch ändern (optional; 5.6.8).

Der Computergegner soll in der Grundausstattung alle aktuellen Züge erfassen und draus mittels einer Stellungsbewertung den besten Zug auswählen (5.4). In der fortgeschrittenen Implementierung ~~können~~ wurde dazu ein komplexerer Algorithmus, eine Min/Max-Suche mit Alpha/Beta-Pruning des Suchbaumes, benutzt ~~genutzt werden (~~5.6.1~~)~~.

Spieler 1 (weiß) beginnt die Partien mit seinem Zug. ~~Im Netzwerkspiel ist dies der Host, im Spiel gegen die Schach-KI kann der menschliche Spieler seine Farbe auswählen (5.3.1).~~ Ein Zug kann das Bewegen einer Spielfigur auf ein leeres Feld oder ein Feld mit einer gegnerischen Schachfigur entsprechend den *Regeln des deutschen Schachbundes*[1] sein. Anschließend zieht Spieler 2 (schwarz).

Wird das Spiel im --no-gui-Modus gestartet, erfolgt die Eingabe durch den Spieler textuell (5.2.2). Fehleingaben werden mit entsprechenden Meldungen quittiert (5.2.3), ebenso werden gültige Spielzüge bestätigt (5.2.5). Das Spielfeld wird durch Unicode-Symbole auf der Konsole ausgegeben (5.2.7). Es ist ebenso möglich, sich durch die Eingabe „beaten“ alle bereits geschlagene Figuren

[1] https://www.schachbund.de/files/dsb/srk/2019/FIDE-Regeln-2018-Final-DEU.pdf

anzeigen zu lassen (5.2.8), außerdem werden die Züge des Computergegners ebenso angezeigt (5.2.6).

Im GUI-Modus kann man sich durch das Anklicken einer Spielfigur alle in dieser Position möglichen Züge optional hervorheben lassen (5.3.8). Das Wechseln der aktiven Figur ist ebenso noch möglich (5.3.9). Durch Klicken auf ein gültig erreichbares Feld, bewegt sich die Figur (5.3.4), ungültige Züge werden hier ignoriert (5.3.5). Optional ist es möglich, das Schachbrett zum aktiven Spieler zu drehen (5.3.3). Im Gegensatz zur Konsolenversion werden geschlagene Figuren stets links vom Spielfeld angezeigt (5.3.6). Dennoch lässt sich eine Historie anlegen, die einer Spielpartie über der Konsole entspricht (5.3.7).

Hierzu kommt, dass von jeder Stelle des Spiels beliebig viele Züge rückgängig gemacht werden können (5.5). Dies erfolgt in der Konsolenversion durch Eingabe von „undo“, bzw. in der GUI-Version durch Drücken eines Buttons oder Klicken auf einen Eintrag in der Historie. Auch ein „redo“ ist auf diese Weise möglich, solange kein nächster Zug dem „undo“ gefolgt ist.

Befindet sich der König eines Spielers im Schach, wird ihm eine Meldung angezeigt (5.2.9, 5.3.10).

Wenn ~~der Timer eines Spielers (optional; 5.5.5) abgelaufen ist oder sein~~ der König eines Spielers schachmatt gesetzt wurde, hat er verloren und das Spiel ist beendet. Es wird zudem die Möglichkeit eines Unentschiedens abgedeckt. Dies wird dem Spieler entsprechend mittgeteilt (5.2.10, 5.3.11). In allen Fällen sind keine Züge mehr möglich (5.3.11) und in der Konsolenversion wird das Spiel unverzüglich beendet (5.2.10). Es kann jederzeit zum Auswahlmenü zurückgekehrt werden (5.3.12).

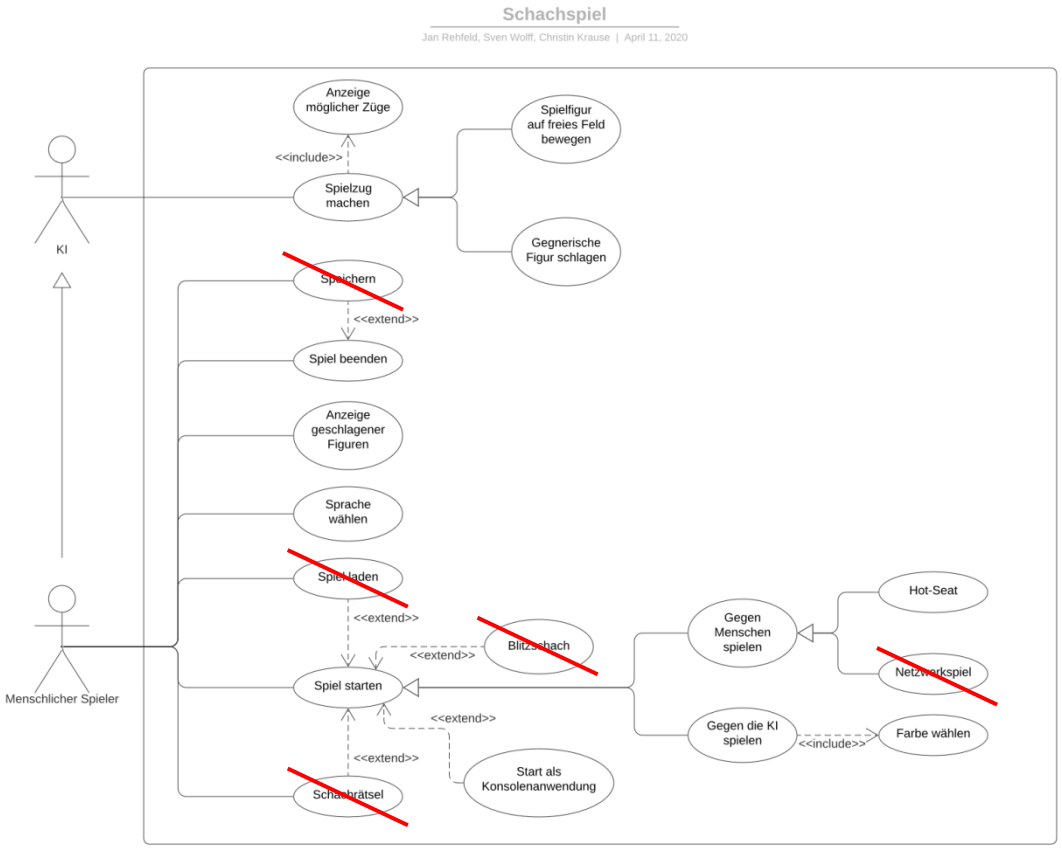

Abbildung 2: Anwendungsfalldiagramm für das Schachspiel