ulm university universität
uulm

Fakultät für Ingenieurwissenschaften, Informatik und Psychologie
Institut für Softwaretechnik und Programmiersprachen

Softwaregrundprojekt Meilenstein 4

Softwaregrundprojekt an der Universität Ulm

Vorgelegt von:
Gruppe 10

Dozent:
Florian Ege

Betreuer:
Stefanos Mytilineos

2019

Inhaltsverzeichnis

Teil I
Überblick

1 Einleitung

Bei diesem Dokument handelt es sich um das Pflichtenheft zum Projekt *Fantastic Feasts*. Es dient dazu, das zu entwickelnde System vollständig zu spezifizieren und alle bisher erarbeiteten Entwurfsdokumente strukturiert zusammenzufassen. Es baut auf dem Lastenheft auf, in dem bereits einige Anforderungen und Spefizikationen des Auftraggebers formuliert wurden, ist jedoch umfangreicher und detaillierter, was die Anforderungen und Spezifikationen des Systems betrifft. Damit stellt es den Vertrag zwischen Entwicklern und Auftraggebern dar. Alle hier aufgelisteten Leistungen müssen vom Entwicklerteam erbracht werden. Insgesamt richtet sich das Dokument sowohl an die Entwickler, die es als Referenz in der Implementierungsphase verwenden, als auch an den Kunden, der eine umfangreiche und genaue Beschreibung der zu erwartenden Leistungen zur Hand hat.

Bei dem Projekt handelt es sich um die Konzeption und Implementierung eines online Multiplayer-Spiels aus der Welt von Harry Potter – genauer: *Fantastic Feasts*. Es ist eine rundenbasierte Form des bekannten Spiels Quidditch.

Im Mittelpunkt des Projekts steht das Erlernen von Fähigkeiten im Umgang mit einem größeren Softwareprojekt. Es werden keine kommerziellen Ziele verfolgt.

Auftraggeber des Projektes ist die Servicegruppe Informatik der Universität Ulm, die das Modul Softwaregrundprojekt veranstaltet. Die Umsetzung erfolgt durch sechs Studenten der Informationssystemtechnik, die das Entwicklerteam bilden.

Das Dokument gliedert sich in vier Teile:

- Teil I: Überblick
- Teil II: Anforderungsanalyse
- Teil III: Softwarespezifikation
- Teil IV: Randbedingungen

Teil I beschreibt den Kontext und alle Rahmenbedingungen des Projekts. Teil II formuliert Fachwissen der Systemdomäne, den Anwendungskontext sowie funktionalen Anforderungen des Systems. Teil III spezifiziert das Softwaresystem aus Sicht der Entwickler. Benutzerschnittstellen, Nutzungskonzept, Datenmodell und die einzelnen Funktionen des Systems werden hier beschrieben, wobei auf die Anforderungen Bezug genommen wird. In Teil IV werden Qualitätsanforderungen an den System formuliert und Wege beschrieben, wie ihre Einhaltung überprüft werden kann.

2 Motivation

Die Motivation für das Projekt lässt sich in die des Auftraggebers und die der Auftragnehmer aufteilen.

Der Auftraggeber – im weitesten Sinne die Universität Ulm – verfolgt das Ziel, den Studenten Fähigkeiten zu vermitteln und sie anschließend nach genau definierten Maßstäben zu bewerten:

Zunächst einmal stehen Planen, Formulieren von Anforderungen und Modellierung von Software an. Es folgt die Auseinandersetzung mit verschiedenen Plattformen und Technologien auf die für die Implementierung zurückgegriffen werden soll. Gleichzeitig wird das Ziel verfolgt, übergeordnete Fähigkeiten zur Qualitätssicherung, zur Versionenverwaltung oder zu agilen Entwicklungsprozessen im Team zu erwerben. Erst dann kommen praktische Programmierfähigkeiten zum tragen. Auch hier ist es das Ziel, diese auszubauen.

Darüber hinaus liegt es im Interesse der Universität Ulm aus Prestigegründen gute Absolventen ihres Bachelor-Programms auszubilden.

Die Auftragnehmer – in diesem Fall sechs Studenten der Informationssystemtechnik – verfolgen das Ziel, das Projekt nach den Anforderungen im Lastenheft erfolgreich umzusetzen und die Abnahmeprüfung zu bestehen. Mit dem Erwerb und Ausbau der oben genannten Fähigkeiten leisten sie einen wichtigen Beitrag zu einer erfolgreichen Ausbildung in ihrem Fach. Zusätzlich soll ein Spiel entwickelt werden, das funktioniert und Spaß macht.

3 Vision

Das fertige System folgt einer Client-Server-Architektur. Darin kommunizieren ein oder mehrere Clients mit dem Server, auf dem die Spiellogik läuft. Die Spieler haben clientseitig eine ansprechende GUI, über die sie *Fantastic Feasts* spielen, eine Partie als Gast verfolgen, Charaktere und Ausrüstung zu Teams mitsamt Farben und Logo zusammen stellen können und die Möglichkeit haben, Partien zu konfigurieren. Begleitet wird die visuelle Darstellung von Soundeffekten und einer thematisch ansprechenden Spielmusik.

Im durch und durch taktischen Spiel mit zwei sich genüberstehenden Quidditch-Teams können die Spieler Runde für Runde Spielfiguren auf Besen über das Spielfeld jagen lassen, Punkte erzielen, den Gegner sabotieren und Publikumseffekte zu ihrem Vorteil einsetzen. Doch selbst bei noch so guter Taktik kann ihnen der Zufall einen Strich durch die Rechnung machen, da nicht immer alles so eintritt, wie es sich die einzelnen Spieler vielleicht erhofft haben.

Was die Spielmodi betrifft, ist das Kern-Szenario das Multiplayer-Spiel. Hier entsteht durch den Wettstreit zweier Spieler die größte Spannung. Doch um das Spiel auch alleine spielbar zu machen, existiert ein Singleplayer-Modus. Eine ausgefeilte KI mit voraussichtlich mehreren Schwierigkeitsstufen stellt für Einzelspieler eine spannende Herausforderung dar. Wer sich noch auf den großen Wettkampf vorbereitet, die Taktik andere Spieler erlernen will oder einfach Spaß am Zuschauen hat, kann sich im Gast-Modus in andere Multiplayer-Partien einklinken. Damit bleibt einer breiten Zielgruppe an Spielern kaum etwas zu wünschen übrig.

4 Projektkontext

Auftraggeber des Projektes ist die Servicegruppe Informatik der Universität Ulm, die das Modul Softwaregrundprojekt veranstaltet. Der Tutor Stefanos Mytilineos vertritt den Auftraggeber während der Projektlaufzeit, in höherer Instanz ist Florian Ege verantwortlich. Als weiterer Stakeholder tritt das Team auf, das das Projekt letztendlich entwickelt. Es besteht aus sechs Studenten der Informationssystemtechnik: Tarik Enderes, Tim Luchterhand, Jonas Merkle, Paul Nykiel, Björn Petersen und Michael von Hohnhorst. Diese bearbeiten das Projekt nicht in Vollzeit, da sie parallel den weiteren Verlauf ihres Studiums verfolgen.

Indirekt am Entwicklungsprozess beteiligt ist das Standardisierungskomitee, in das auch aus diesem Team ein Vertreter geschickt wird. Dort werden alle nötigen Protokolle und Schnittstellen definiert, die bei der Entwicklung von *Fantastic Feasts* benötigt werden. Weitere Stakeholder, die jedoch erst später in Erscheinung treten, sind andere Teams, die gegebenenfalls auf einen Komponenten dieses Projektes angewiesen sind. Sie werden auf einer bevorstehenden Messe zu potenziellen Kunden. In ihrem Interesse liegt eine saubere Implementierung bei gleichzeitig guter Dokumentation des Komponenten. Zu guter letzt muss dieses Projekt sowie die Einzelleistung eines jeden Team-Mitgliedes die Prüfer in der Abnahmeprüfung überzeugen – sie stellen so gesehen die wichtigsten Kunden dar.

Da bei der Implementierung auf den agilen Entwicklungsprozess Scrum zurückgegriffen wird, sollen auch hier die Rollen kurz benannt werden. In dieser Phase übernimmt der oben genannte Tutor die Rolle des Product Owners. Der Scrum-Master wird innerhalb des oben genannten Teams ernannt.

Folgeprojekte von *Fantastic Feasts* sind derzeit nicht vorgesehen. Denkbar wäre jedoch ein Publishing des Spiels mit ständig laufendem Server und beliebig vielen Server-Instanzen – so könnte *Fantastic Feasts* weltweit von zahlreichen Spielern gespielt werden.

Teil II
Anforderungsanalyse

1 Fachwissen

Im Folgenden werden grundlegende Begriffe für den Austausch über das Projekt definiert.

Begriff	**Nutzer**
Beschreibung	Ein Mensch, der einen Rechner bedient und entweder den Client zum Spielen des Spiels oder zur Beobachtung einer Partie benutzt, oder den Team-Editor bedient. Jeder Nutzer hat einen Nutzernamen, mittels dem er von anderen Nutzern erkannt werden kann.
Ist-ein	-
Kann-sein	Spieler, Gast
Aspekt	Zur Beschreibung des Programmverlaufs
Bemerkung	-
Beispiel	Der Nutzer „JägerMaister69“ ist der Partie beigetreten.

Begriff	**Spieler**
Beschreibung	Ein Nutzer, der in dem Computerspiel „Fantastic Feasts“ eine Partie gegen einen anderen Nutzer bestreitet. Er meldet sich beim Server als Spieler an und führt die Partie mittels Aktionen. Pro Partie kann es höchstens zwei Spieler geben.
Ist-ein	Nutzer
Kann-sein	-
Aspekt	Zur Beschreibung des Programmverlaufs
Bemerkung	-
Beispiel	Der Spieler „KäptnCola“ hat die Partie verlassen.

Begriff	**Gast**
Beschreibung	Ein Nutzer, der eine laufende Partie beobachtet. Er kann den derzeitigen Stand der Partie und des Spielfeldes sehen, sie aber nicht beeinflussen.
Ist-ein	Nutzer
Kann-sein	-
Aspekt	Zur Beschreibung des Programmverlaufs
Bemerkung	-
Beispiel	Der Nutzer „TequilaLove“ ist der Partie als Gast beigetreten.

Begriff	**Client**
Beschreibung	Das Computerprogramm, das mit einer grafischen Oberfläche ausgestattet ist und einem Nutzer erlaubt, eine Verbindung mit einem Server herzustellen und damit zu kommunizieren. Der Client erlaubt es einem Nutzer, einer Partie als Spieler mit einer ausgewählten Team-Konfiguration oder als Gast beizutreten. Dies wird durch ein Hauptmenü organisiert. In der Partie dient die grafische Oberfläche dazu, das Spielgeschehen zu visualisieren und die Steuerung der Spielfiguren zu ermöglichen.
Ist-ein	-
Kann-sein	-
Aspekt	Zum Spielen des Spiels „Fantastic Feasts“
Bemerkung	Der Begriff bezieht sich nicht auf den Menschen, der das Programm bedient. Der Client ist eine eigenständige Komponente.
Beispiel	-

Begriff	**Server**
Beschreibung	Die zentrale Komponente, in der die Spiellogik implementiert ist. Mit dem Server können sich Clients verbinden, um eine Partie zu spielen oder zu beobachten. Die Kommunikation erfolgt über Nachrichten im JSON-Format. Es handelt sich dabei um eine Kommandozeilenanwendung. Ein Administrator startet den Server auf einem Port eines Rechners mit einer ausgewählten Partie-Konfiguration.
Ist-ein	-
Kann-sein	-
Aspekt	Ist für die Kommunikation von Clients, für das Verwalten des Spielgeschehens, Ressourcenverwaltung und die Spiellogik verantwortlich
Bemerkung	Der Server wird mit mittels Kommandozeile gestartet.
Beispiel	-

Begriff	**Konfigurator**
Beschreibung	Ermöglicht einem Nutzer mit einer grafischen Oberfläche, Team- und Partie-Konfigurationen zu erstellen, zu editieren, zu speichern und zu laden. Die Einstellungen werden danach als JSON-Datei gespeichert.
Ist-ein	-
Kann-sein	-
Aspekt	Zur Erstellung von nutzereigenen Teams
Bemerkung	Der Konfigurator hat eine grafische Oberfläche.
Beispiel	-

Begriff	**KI-Client**
Beschreibung	Der KI-Client meldet sich beim Server wie ein normaler Client an und simuliert mit einer KI einen menschlichen Spieler. Er hat keine grafische Oberfläche.
Ist-ein	-
Kann-sein	-
Aspekt	Zum Spielen gegen einen Computergegner
Bemerkung	Meldet sich mit dem Nutzernamen „KI“ an.
Beispiel	-

Begriff	**KI**
Beschreibung	Definiert die Regeln, nach denen der KI-Client auf die durch den Server vermittelten Geschehen im Spiel reagiert. Es soll möglich sein, Schwierigkeitsgrade auszuwählen.
Ist-ein	-
Kann-sein	-
Aspekt	Zum Spielen gegen einen Computergegner
Bemerkung	Die KI ist die Logik, nach der der Computer das Spiel spielt und kein Programm.
Beispiel	-

Begriff	**Spielfeld**
Beschreibung	Ein grafisch darstellbares Raster, auf dem sich alle Spielobjekte bewegen.
Ist-ein	-
Kann-sein	-
Aspekt	Dient als virtuelles Spielbrett mit klar definierten Abgrenzungen
Bemerkung	Wird nicht Spielumgebung genannt um Verwechslung mit dem Client zu vermeiden.
Beispiel	Die Spielfiguren werden auf dem Spielfeld platziert.

Begriff	**Zelle**
Beschreibung	Die kleinste Einheit des Spielfeldes, also ein Quadrat davon.
Ist-ein	-
Kann-sein	Zentrumszelle, Torring, Hüterzonenzelle
Aspekt	Mögliche Standorte der Spielobjekte
Bemerkung	Wird nicht Feld genannt, um Verwechselungen mit dem Spielfeld zu vermeiden.
Beispiel	Ein Spieler wählt die Zelle, auf die er seine Spielfigur bewegen möchte.

Begriff	**Zentrum**
Beschreibung	Der 3x3 Zellen große Abschnitt in der Mitte des Spielfeldes.
Ist-ein	-
Kann-sein	-
Aspekt	Summe aller Zentrumszellen
Bemerkung	Bezieht sich auf das Mittelfeld im Lastenheft. Die Bälle befinden sich zu Spielbeginn im Zentrum.
Beispiel	-

Begriff	**Hüterzone**
Beschreibung	Die Bereiche am linken und rechten Rand des Spielfeldes, in denen sich die Torringe befinden.
Ist-ein	-
Kann-sein	-
Aspekt	Summe aller Hüterzonenzellen und Torringe
Bemerkung	Nur eine gegnerische Spielfigur darf sich gleichzeitig in der eigenen Hüterzone befinden.
Beispiel	-

Begriff	Torring
Beschreibung	Die Zellen, in die beide Teams den Quaffel bewegen wollen. Es wird zwischen eigenen und gegnerischen Torringen unterschieden.
Ist-ein	Zelle
Kann-sein	Eigener Torring, Gegnerischer Torring
Aspekt	Hauptquelle von Punkten
Bemerkung	Die Torringe befinden sich innerhalb der Hüterzone.
Beispiel	-

Begriff	Zentrumszelle
Beschreibung	Eine Zelle im Zentrum des Spielfeldes (siehe Zentrum).
Ist-ein	Zelle
Kann-sein	-
Aspekt	Startpunkt für Quaffel und Klatscher
Bemerkung	-
Beispiel	-

Begriff	Hüterzonenzelle
Beschreibung	Eine Zelle in einer Hüterzone.
Ist-ein	Zelle
Kann-sein	-
Aspekt	Limitierendes Element für das Abliefern des Quaffel
Bemerkung	Betritt ein Jäger eine Hüterzonenezelle, wird geprüft, ob sich bereits ein weiterer Jäger desselben Teams auf einer anderen Hüterzonenzelle befindet.
Beispiel	-

Begriff	Spielobjekt
Beschreibung	Jedes Objekt, das sich auf dem Spielfeld befindet und darauf bewegt werden kann.
Ist-ein	-
Kann-sein	Ball, Spielfigur
Aspekt	-
Bemerkung	Nicht zu verwechseln mit Spielfigur.
Beispiel	Die Spielobjekte werden vom Client visualisiert.

Begriff	**Ball**
Beschreibung	Ein Spielobjekt, das nicht direkt, nur indirekt von einem Spieler beeinflusst werden kann. Bälle erfüllen bestimmte Funktionen und dienen als zentrale Elemente, um die herum die Spieler ihre Aktionen richten.
Ist-ein	Spielobjekt
Kann-sein	Quaffel, Klatscher, Schnatz
Aspekt	Festpunkte zur Steuerung des Spielverlaufs
Bemerkung	-
Beispiel	Der Quaffel ist ein Ball.

Begriff	**Spielfigur**
Beschreibung	Ein Spielobjekt, das von einem Spieler direkt gesteuert wird. Jede Spielfigur hat einen Namen, einen Besenrang, ein Geschlecht, ein Team und eine Rolle. Man unterscheidet außerdem zwischen eigenen und gegnerischen Spielfiguren.
Ist-ein	Spielobjekt
Kann-sein	Hüter, Sucher, Jäger, Treiber
Aspekt	Mitglieder eines Teams
Bemerkung	Spieler im Lastenheft
Beispiel	Zu Spielbeginn platzieren die Spieler ihre Spielfiguren auf dem Spielfeld.

Begriff	**Quaffel**
Beschreibung	Passives Spielobjekt, mit dem Jäger und Hüter interagieren können und von ihnen nach Möglichkeit in einen gegnerischen Torring befördert werden soll.
Ist-ein	Ball
Kann-sein	-
Aspekt	Zentrales Spielobjekt
Bemerkung	Der Quaffel bietet den Spielern die Möglichkeit, Punkte zu sammeln.
Beispiel	-

Begriff	**Klatscher**
Beschreibung	Ball, der sich von selbst auf Spielfiguren zubewegt, die keine Treiber sind und diese betäuben kann. Kann von Treibern geschlagen und dadurch beeinflusst werden.
Ist-ein	Ball
Kann-sein	-
Aspekt	Zusätzliches taktisches Spielelement
Bemerkung	-
Beispiel	-

Begriff	**Schnatz**
Beschreibung	Beschreibung: Ball, der sich selbstständig bewegt und von den Suchern gejagt wird. Wird er von einem Sucher gefangen, bekommt dessen Team 30 Punkte und die Partie ist zu Ende.
Ist-ein	Ball
Kann-sein	-
Aspekt	Definiert Spielende
Bemerkung	Der Schnatz erscheint erst im Laufe der Partie und ist nicht dierkt von Anfang an vorhanden.
Beispiel	-

Begriff	**Partie**
Beschreibung	Ein einzelnes Spiel. Beginnt beim Platzieren der Figuren und endet mit dem Bestimmen des Gewinners.
Ist-ein	-
Kann-sein	-
Aspekt	Beschreibung des Spielablaufs
Bemerkung	-
Beispiel	Spieler „VodkaVodka98“ spielt eine Partie gegen Spieler „LongEiländ“

Begriff	**Hüter**
Beschreibung	Spielfigur, deren Aufgabe es ist, den Quaffel von den eigenen Torringen fernzuhalten.
Ist-ein	Spielfigur
Kann-sein	Eigener Hüter, Gegnerischer Hüter
Aspekt	Letzte Verteidigungslinie
Bemerkung	Jedes Team hat genau einen Hüter.
Beispiel	Nach einem erfolgreichen Torschuss bekommt der gegnerische Hüter den Quaffel.

Begriff	**Sucher**
Beschreibung	Spielfigur, die durch Einfangen des Schnatzes eine Partie beenden kann.
Ist-ein	Spielfigur
Kann-sein	Eigener Sucher, Gegnerischer Sucher
Aspekt	Beendet die Partie
Bemerkung	Jedes Team hat genau einen Sucher.
Beispiel	Der Sucher „Darth Vader“ fängt den Schnatz und beendet damit die Partie.

Begriff	**Jäger**
Beschreibung	Spielfigur, die den Quaffel in ein einen gegnerischen Torring befördern soll.
Ist-ein	Spielfigur
Kann-sein	Eigener Jäger, Gegnerischer Jäger
Aspekt	Holt Punkte für das eigene Team
Bemerkung	Jedes Team hat genau 3 Jäger.
Beispiel	Der Jäger „Han Solo“ erzielt ein Tor.

Begriff	**Treiber**
Beschreibung	Spielfigur, mit der der Spieler eigene Spielfiguren vor Klatschern schützt und gegnerische damit abschießen kann.
Ist-ein	Spielfigur
Kann-sein	Eigener Treiber, Gegnerischer Treiber
Aspekt	Interagiert mit Klatschern
Bemerkung	Jedes Team hat genau 2 Treiber.
Beispiel	Der Treiber „Boba Fett“ schlägt den Klatscher auf die Zelle 3:1.

Begriff	**Geschlecht**
Beschreibung	Ein Parameter, der bei der Team-Konfiguration für jede Spielfigur entweder auf „männlich“ oder „weiblich“ gesetzt werden muss.
Ist-ein	-
Kann-sein	-
Aspekt	Team-Editierung
Bemerkung	-
Beispiel	-

Begriff	**Team**
Beschreibung	Die Menge aller Spielfiguren auf dem Spielfeld, die von einem einzigen Spieler kontrolliert wird. Ein Team hat einen Namen, ein Motto, ein Logo, eine Hauptfarbe und eine Ersatzfarbe. Ein Nutzer hat die Möglichkeit, seine Team-Konfiguration mit dem Client zu ändern. Die Team-Konfigurationen können mit dem Konfigurator verändert und erstellt werden.
Ist-ein	eigenes Team, gegnerisches Team
Kann-sein	-
Aspekt	Beschreibung einer Partie
Bemerkung	Ein Team besteht aus 7 Spielfiguren: Ein Sucher, ein Hüter, drei Jäger und zwei Treiber.
Beispiel	Ein Nutzer wählt sein zuvor im Team-Editor konfiguriertes Team im Client aus.

Begriff	**Punkte**
Beschreibung	Der Spieler mit mehr Punkten am Ende der Partie gewinnt. Ein Team kann Punkte erzielen, indem es den Quaffel durch einen gegnerischen Torring wirft oder den Schnatz fängt.
Ist-ein	-
Kann-sein	-
Aspekt	Bestimmung des Gewinners
Bemerkung	Ein erfolgreicher Torschuss ist 10 Punkte wert, das Fangen des Schnatzes 30.
Beispiel	-

Begriff	**Besetzen**
Beschreibung	Eine Spielfigur besetzt die Zelle, auf der sie sich befindet.
Ist-ein	-
Kann-sein	-
Aspekt	Beschreibung des Spielgeschehens
Bemerkung	Zwei Spielfiguren können nicht ein und die selbe Zelle besetzen.
Beispiel	Die Spielfigur „Chewbacca“ besetzt die Zelle 5:3.

Begriff	**Besenrang**
Beschreibung	Jede Spielfigur hat einen Besenrang von 1 bis 5, der die Wahrscheinlichkeit bestimmt, dass sie nach dem Ziehen erneut um ein Feld ziehen darf. Der Besenrang 1 ist der beste.
Ist-ein	-
Kann-sein	-
Aspekt	Unterscheidet Qualität der Spielfiguren.
Bemerkung	Ersetzt die verschiedenen „Besen“ aus dem Lastenheft mit einer Skala von 1 bis 5 zur besseren Übersicht.
Beispiel	Die Spielfigur „Yoda“ hat den Besenrang 1.

Begriff	**Aktion**
Beschreibung	Jede durch einen Spieler hervorgerufene Änderung der Spielsituation. Ein Spieler führt Aktionen mittels des Clients durch.
Ist-ein	-
Kann-sein	Ziehen, Schießen, Schlagen, Einmischung, Übernahme, Befördern
Aspekt	Weiterführung der Partie
Bemerkung	-
Beispiel	Spieler steuern das Spielgeschehen durch Aktionen.

Begriff	**Ziehen**
Beschreibung	Die Bewegung einer Spielfigur von einer Zelle auf eine andere durch direkten Befehl des Spielers.
Ist-ein	Aktion
Kann-sein	-
Aspekt	Beschreibung des Spielverlaufs
Bemerkung	Bezieht sich nicht auf erzwungene Bewegungen einer Spielfigur.
Beispiel	Die Spielfigur „Obiwan Kenobi“ zieht von Zelle 8:7 auf Zelle 9:7.

Begriff	**Befördern**
Beschreibung	Bewegen des Quaffel mittels einer Spielfigur.
Ist-ein	-
Kann-sein	Aktion
Aspekt	Bewegen des Quaffel, allgemeiner Begriff
Bemerkung	Nicht zu verwechseln mit Schießen. Eine Spielfigur kann den Quaffel aktiv befördern (also eine Aktion der Spielfigur), wenn die Spielfigur den Quaffel hält und mit ihm auf eine andere Zelle zieht. Der Quaffel kann aber auch durch ein anderes Ereignis auf eine Zelle befördert werden.
Beispiel	Ein Hüter zieht, während er den Quaffel hält, auf eine andere Zelle und befördert ihn damit aktiv (Aktion des Hüters). Wenn ein Treiber auf Zelle zieht, auf der der Quaffel liegt, wird der Quaffel indirekt auf eine andere freie Zelle befördert.

Begriff	**Schießen**
Beschreibung	Die Bewegung des Quaffel durch einen Hüter oder Jäger auf eine andere, entfernte Zelle ohne Bewegung der Spielfigur.
Ist-ein	Aktion
Kann-sein	-
Aspekt	Bewegung des Quaffel um mehrere Felder
Bemerkung	„Werfen“ im Lastenheft. Analog zum Schussvektor benannt.
Beispiel	Der Jäger „Mace Windu“ schießt den Quaffel auf Zelle 10:4.

Begriff	**Schlagen**
Beschreibung	Die erzwungene Bewegung eines Klatschers durch einen Treiber.
Ist-ein	Aktion
Kann-sein	-
Aspekt	Interaktion mit Klatschern
Bemerkung	„Kloppen“ im Lastenheft
Beispiel	Der Treiber „R2-D2“ schlägt einen Klatscher auf Zelle 5:10.

Begriff	Einmischung
Beschreibung	Hilfsfähigkeiten, die nicht von Spielobjekten ausgehen. Werden von einem Spieler gesteuert. Bei jeder Benutzung besteht eine Chance, dass die verwendete Einmischung bis zum Ende der Partie für den jeweiligen Spieler vom Schiedsrichter deaktiviert werden.
Ist-ein	Aktion
Kann-sein	Teleportation, Fernangriff, Impuls, Schnatzjagd
Aspekt	Zusätzliche taktische Element
Bemerkung	Ersetzt die „Fans“ aus dem Lastenheft.
Beispiel	Der Schiedsrichter erkennt eine Einmischung mit einer in der Partie-Konfiguration festgelegten Wahrscheinlichkeit.

Begriff	Teleportation
Beschreibung	Einmischung, die eine Spielfigur auf eine zufällige Zelle bewegt.
Ist-ein	Einmischung
Kann-sein	-
Aspekt	-
Bemerkung	Ersetzt „Elfen“ aus Lastenheft. Die teleportierte Spielfigur kann aus einem beliebigen Team sein.
Beispiel	Der Sucher „Lando Calrissian“ wird auf Zelle 6:6 teleportiert.

Begriff	Fernangriff
Beschreibung	Trifft eine gegnerische Spielfigur. Das Ziel verliert gegebenenfalls den Quaffel und wird auf eine zufällige benachbarte, freie Zelle bewegt.
Ist-ein	Einmischung
Kann-sein	-
Aspekt	-
Bemerkung	Statt „Kobolde“ im Lastenheft
Beispiel	Der Treiber „Jango Fett“ wird von einem Fernangriff auf Zelle 5:6 gestoßen.

Begriff	**Impuls**
Beschreibung	Wenn der Quaffel von einer Spielfigur gehalten wird, wird er von dieser verloren.
Ist-ein	Einmischung
Kann-sein	-
Aspekt	-
Bemerkung	Statt „Trolle“ im Lastenheft
Beispiel	Der Hüter „C-3PO“ verliert wegen eines Impulses den Quaffel.

Begriff	**Schnatzstoß**
Beschreibung	Bewegt den Schnatz in eine zufällige Richtung um eine Zelle.
Ist-ein	Einmischung
Kann-sein	-
Aspekt	-
Bemerkung	„Schnatzschnappen“ im Lastenheft
Beispiel	Ein Schnatzstoß treibt den Schnatz auf Zelle 4:12.

Begriff	**Entfernung**
Beschreibung	Eine Entfernung zwischen zwei Zellen ist, wie oft eine Spielfigur mindestens ziehen muss, um von der Einen auf die Andere zu gelangen.
Ist-ein	-
Kann-sein	-
Aspekt	Spielfeldgeometrie
Bemerkung	-
Beispiel	Die Entfernung zwischen dem Jäger und dem Torring beträgt drei Zellen.

Begriff	**Schussvektor**
Beschreibung	Pfeil vom Mittelpunkt einer Zelle zum Mittelpunkt einer anderen.
Ist-ein	-
Kann-sein	Torschussvektor
Aspekt	Spielfeldgeometrie
Bemerkung	-
Beispiel	Der Client visualisiert den Schussvektor für einen Schuss vom jeweiligen Jäger oder Hüter auf die vom Spieler ausgewählte Zelle.

Begriff	Torschussvektor
Beschreibung	Schussvektor zu einem Schuss, der möglicherweise in einem Torschuss resultiert. (Ein Schussvektor, der die linke oder rechte Seite eines Torrings schneidet.)
Ist-ein	Schussvektor
Kann-sein	-
Aspekt	Punkte sammeln
Bemerkung	Auch wenn ein Schussvektor durch einen Torring verläuft (und damit ein Torschussvektor ist), kann der Schuss den Torring verfehlen.
Beispiel	Ein Schussvektor, der von einer Zentrumszelle zu einer anderen zeigt, ist kein Torschussvektor. Ein Schussvektor, der von der Zelle links von einem Torring auf die Zelle rechts davon zeigt, ist ein Torschussvektor.

Begriff	Torschuss
Beschreibung	Ein Jäger schießt den Quaffel in einen Torring und holt damit Punkte für sei Team.
Ist-ein	-
Kann-sein	-
Aspekt	Punkte sammeln
Bemerkung	Nur erfolgreiche Schüsse auf das Tor werden als Torschüsse bezeichnet. Nach einem Torschuss bekommt der Hüter des verteidigenden Team den Quaffel.
Beispiel	Der Jäger „Darth Sidious“ führt einen erfolgreichen Torschuss aus und erzielt 10 Punkte für sein Team.

Begriff	Runde
Beschreibung	Der Zeitraum, in dem jedes Spielobjekt auf dem Spielfeld einmal in einer Rundenphase gesteuert wird und jede verfügbare Einmischung einmal ausgeführt werden kann.
Ist-ein	-
Kann-sein	-
Aspekt	Zeiteinteilung
Bemerkung	Nicht die Rundenphasephase einer Spielfigur
Beispiel	Die Überlängenbehandlung tritt nach einer in der Partie-Konfiguration festgelegten Anzahl an Runden in Kraft.

Begriff	Spielerphase
Beschreibung	Ein Zeitraum, in dem eine einzelne Spielfigur entweder von einem Spieler oder automatisch vom KI-Client gesteuert werden kann. Eine Runde enthält mehrere Spielerphasen.
Ist-ein	-
Kann-sein	-
Aspekt	Zeiteinteilung
Bemerkung	Die Rundenphase einer Spielfigur beinhaltet das Ziehen der Spielfigur (wenn gewünscht) und gegebenenfalls eine weitere Aktion.
Beispiel	Eine Spielfigur zieht um eine Zelle und schießt danach den Quaffel, der in ihrem Besitz war.

Begriff	Endphase
Beschreibung	Letzter Teil einer Runde. Der Spieler kann darin Einmischungen vornehmen.
Ist-ein	-
Kann-sein	-
Aspekt	Zeiteinteilung
Bemerkung	-
Beispiel	Die KI setzt während ihrer Endphase Teleportation ein, um den gegnerischen Jäger von den eigenen Torringen fernzuhalten.

Begriff	Ballphase
Beschreibung	Der Teil einer Runde, in der die Klatscher und gegebenenfalls der Schnatz automatisch bewegt werden.
Ist-ein	-
Kann-sein	-
Aspekt	Zeiteinteilung
Bemerkung	-
Beispiel	Ein Klatscher betäubt während der Ballphase den gegnerischen Sucher.

Begriff	Verlieren
Beschreibung	Der Quaffel wird auf eine zufällige freie Zelle bewegt, die an die Zelle angrenzt, auf der sich die Spielfigur, die bis jetzt in Ballbesitz war, befindet.
Ist-ein	-
Kann-sein	-
Aspekt	Spielablauf
Bemerkung	„Vertändeln“ im Lastenheft
Beispiel	Der Jäger „Jar Jar“ verliert den Quaffel, der sich dadurch um eine Zelle nach links bewegt.

Begriff	Halten
Beschreibung	Ein Jäger oder Hüter kann den Quaffel halten. Ist das der Fall, bewegt sich der Quaffel auf die Zelle, auf die die Spielfigur zieht.
Ist-ein	-
Kann-sein	-
Aspekt	Beschreibung des Spielgeschehens
Bemerkung	-
Beispiel	-

Begriff	Übernahme
Beschreibung	Ein Jäger neben einer gegnerischen Spielfigur, die den Quaffel hält, kann diesen mit einer bestimmten Wahrscheinlichkeit übernehmen und hält ihn anschließend selbst.
Ist-ein	Aktion
Kann-sein	-
Aspekt	Aggressives Spielmanöver
Bemerkung	-
Beispiel	Der Jäger „Darth Vader“ übernimmt den Quaffel vom Hüter „Anakin Skywalker“.

Begriff	Betäubt
Beschreibung	Eine betäubte Spielfigur kann in ihrer nächsten Rundenphase keine Aktion durchführen.
Ist-ein	-
Kann-sein	-
Aspekt	Wirkung der Klatscher
Bemerkung	„Ausgeknockt“ im Lastenheft
Beispiel	-

Begriff	**Foul**
Beschreibung	Handlung, wegen der eine Spielfigur vorübergehend vom Spielfeld entfernt werden kann.
Ist-ein	-
Kann-sein	Torring Blockieren, Stürmen, Großoffensive, Rammen, Schnatz Blockieren
Aspekt	Taktische Elemente
Bemerkung	Fouls sind Aktionen, die von der Spielmechanik erlaubt sind, jedoch trotzdem bestraft werden können.
Beispiel	Der Hüter „Qui-Gon Jinn“ wird wegen eines Fouls durch den Schiedsrichter vom Spielfeld entfernt.

Begriff	**Torring Blockieren**
Beschreibung	Eine eigene Spielfigur besetzt einen eigenen Torring, was einen Torschuss unmöglich macht.
Ist-ein	Foul
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Flackern“ im Pflichtenheft
Beispiel	Die KI weist einen ihrer Jäger an, den Torring zu blockieren, um einen gegnerischen Torschuss zu verhindern.

Begriff	**Stürmen**
Beschreibung	Ein Jäger, der den Quaffel hält, zieht auf einen gegnerischen Torring, was das Abliefern garantiert.
Ist-ein	Foul
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Nachtarocken“ im Lastenheft
Beispiel	Der Jäger „Han Solo“ stürmt den mittleren gegnerischen Torring und erzielt dadurch ein Tor.

Begriff	Großoffensive
Beschreibung	Ein eigener Jäger betritt eine gegnerische Hüterzonenzelle während ein anderer eigener Jäger sich ebenfalls in der gegnerischen Hüterzone befindet.
Ist-ein	Foul
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Stutschen“ im Lastenheft
Beispiel	Der Jäger „Lando Calrissia“ schließt sich dem Jäger „Chewbacca“ in einer Großoffensive an.

Begriff	Rammen
Beschreibung	Eine eigene Spielfigur zieht auf eine Zelle, die von einer gegnerischen Spielfigur besetzt wird. Dadurch wird die gegnerische Spielfigur auf eine benachbarte Zelle bewegt und verliert ggf. den Quaffel
Ist-ein	Foul
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Keilen“ im Lastenheft
Beispiel	„Boba Fett“ rammt „Jar Jar“.

Begriff	Schnatz blockieren
Beschreibung	Eine Spielfigur, die kein Sucher ist, besetzt die Zelle, auf der sich der Schnatz befindet.
Ist-ein	Foul
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Schnatzeln“ im Lastenheft
Beispiel	„Darth Maul“ blockiert den Schnatz.

Begriff	Schiedsrichter
Beschreibung	Entfernt mit einer bestimmten Wahrscheinlichkeit eine Spielfigur, die ein Foul ausführt, vom Spielfeld bis ein Torschuss erfolgt. Ebenfalls ahndet er mit einer bestimmten Wahrscheinlichkeit eine Einmischung und deaktiviert diese dadurch permanent für den Rest der Partie (für das Team, das die Einmischung ausgeführt hatte).
Ist-ein	-
Kann-sein	-
Aspekt	Taktik
Bemerkung	„Schiedsrichter“ im Lastenheft
Beispiel	„Sheev Palpatine“ wurde vom Schiedsrichter vom Spielfeld entfernt.

Begriff	Disqualifikation
Beschreibung	Tritt ein wenn drei oder mehr Spielfiguren eines Spielers in der selben Runde durch den Schiedsrichter aus dem Spiel entfernt wurden. Führt zur Niederlage des Spielers.
Ist-ein	-
Kann-sein	-
Aspekt	Erhöhtes Risiko
Bemerkung	-
Beispiel	Der Spieler „CubaLibre“ gewinnt die Partie durch Disqualifikation des Gegners.

2 Anwendungskontext

2.1 Akteure

ID:	AKT1
Titel:	Nutzer
Beschreibung:	Menschlicher Nutzer, der eine Anwendungen bedient.
Rolle:	Ein Mensch, der entweder als Spieler aktiv an einem Spiel teilnimmt, als Gast passiv einem Spiel zusieht oder den Quidditchteam-Konfigurator benutzt.

ID:	AKT2
Titel:	Spieler
Beschreibung:	Spielt das Spiel „Fantastic Feasts“.
Rolle:	Nimmt aktiv Einfluss auf das Spielgeschehen. Ist entweder Nutzer oder KI.

ID:	AKT3
Titel:	Gast
Beschreibung:	Nutzer, der mit der Client-Anwendung ein laufendes Spiel beobachtet.
Rolle:	Beobachtet eine Partie als Außenstehender, hat jedoch keinen Einfluss auf das Spielgeschehen.

ID:	AKT4
Titel:	Systemadministrator
Beschreibung:	Person, die die Möglichkeit hat, die Serveranwendung des Projektes zu verwalten.
Rolle:	Der Systemadministrator ist dafür verantwortlich, eine Instanz der Serveranwendung zu starten und zu betreuen. Zudem hat er Zugriff auf die Partie-Konfiguration und kann diese bei Bedarf verändern.

ID:	AKT5
Titel:	Entwickler
Beschreibung:	Person, die an der Entwicklung der Anwendung beteiligt ist.
Rolle:	Der Entwickler implementiert die gesamte Anwendung.

ID:	AKT6
Titel:	KI
Beschreibung:	Vom Computer gesteuerter Spieler.
Rolle:	Spieler, dessen Entscheidungen und Züge von einem Computerprogramm, dem KI-Client, getroffen werden. Es wird somit ein menschlichen Spieler imitiert.

ID:	AKT7
Titel:	Kunde
Beschreibung:	Der Kunde gibt das Projekt in Auftrag.
Rolle:	Stellt Anforderungen und Wünsche an das Entwicklerteam und nimmt das Projekt ab.

ID:	AKT8
Titel:	Client
Beschreibung:	Programm, das einem Nutzer eine grafische Oberfläche, zum Spielen oder Beobachten des Spiels, zur Verfügung stellt.
Rolle:	Der Client stellt eine Verbindung zum Server her, visualisiert die empfangenen Daten und sendet seinerseits die Eingaben des Nutzers.

ID:	AKT9
Titel:	KI-Client
Beschreibung:	Simuliert einen menschlichen Gegner.
Rolle:	Der KI-Client kommuniziert wie der normale Client mit dem Server. Allerdings werden die Entscheidungen von der KI getroffen und nicht von einem Nutzer. Er stellt keine grafische Oberfläche zur Verfügung.

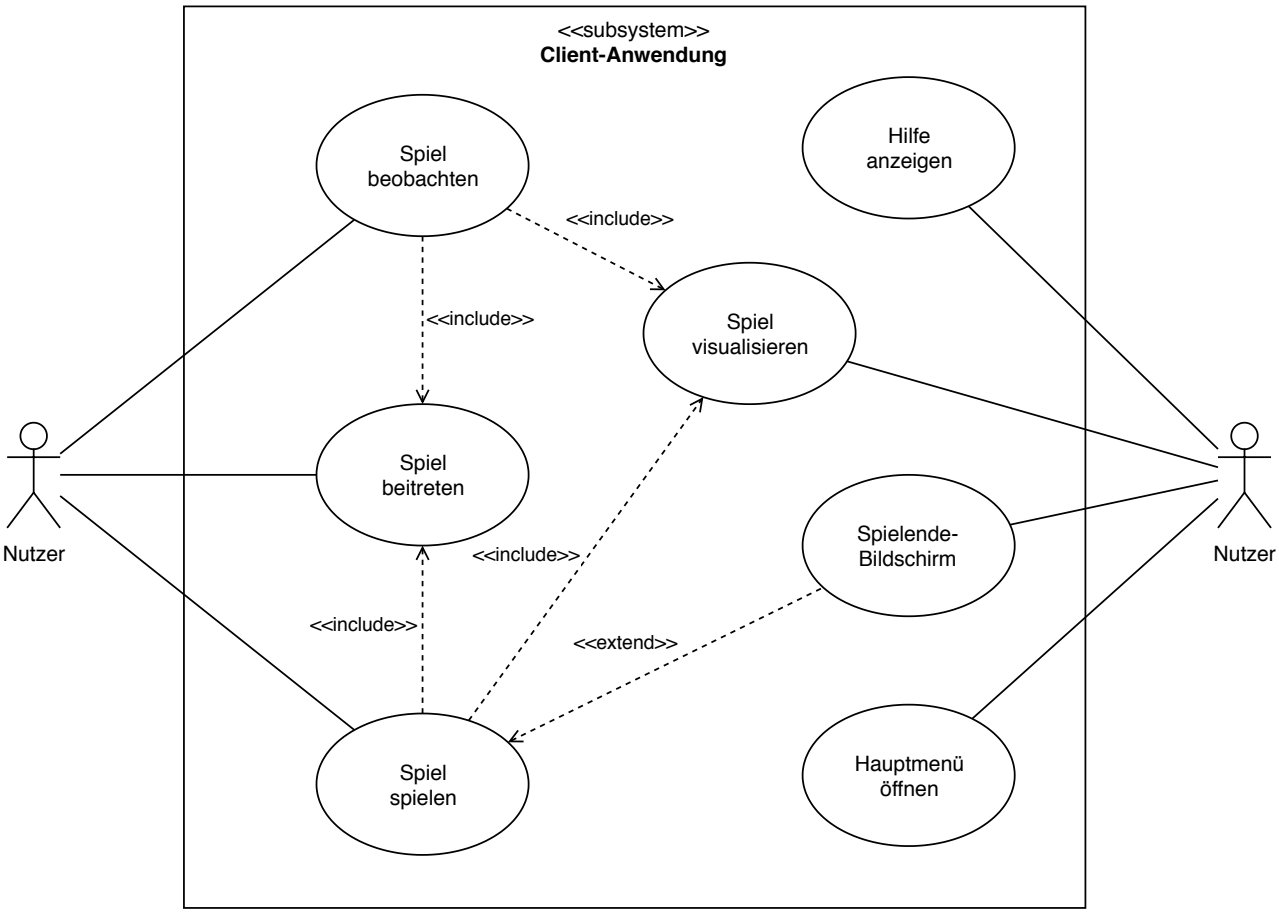
ID:	AKT10
Titel:	Server
Beschreibung:	Zentrale Komponente des Projekts, die alle anderen Komponenten vernetzt.
Rolle:	Auf dem Server läuft die eigentliche Spiellogik. Er fungiert dabei als Bindeglied zwischen den am Spiel beteiligten Clients und stellt für diese alle benötigten Informationen, wie etwa die Spielfeldkonfiguration oder die Züge des Gegners, bereit.

ID:	AKT11
Titel:	Quidditchteam-Konfigurator
Beschreibung:	Computerprogramm, mit dem Team-Konfigurationen erstellt und bearbeitet werden können.
Rolle:	Der Quidditchteam-Konfigurator erstellt Teamkonfigurationsdateien, die später vom Client geladen werden, um ein Spiel zu starten.

2.2 Anwendungsfälle

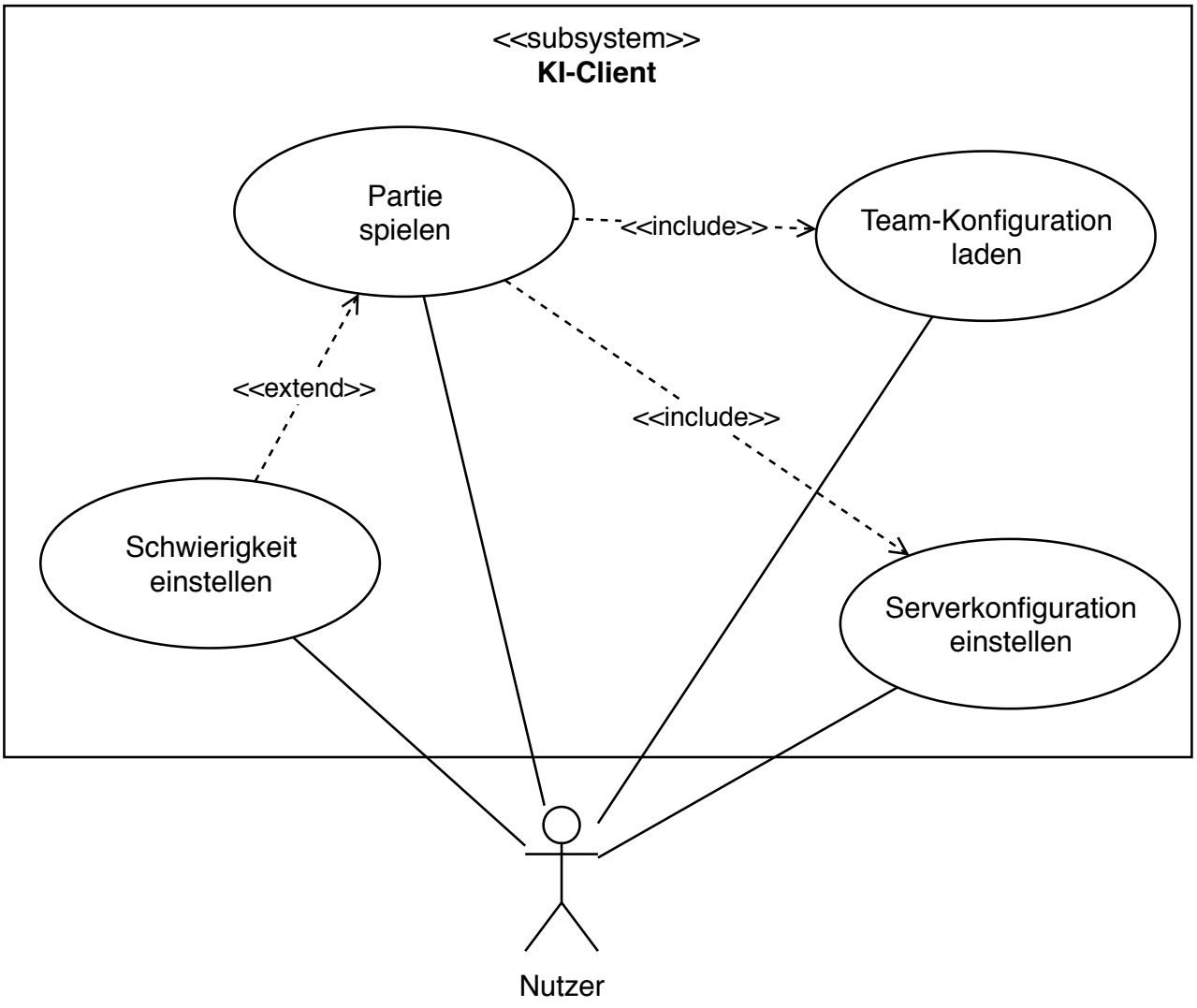
2.2.1 Client

Anwendungsfälle der Client-Anwendung. Der Anwendungsfall „Spiel spielen“ wird unter 2.2.4 konkretisiert. Hinweis: Der zweite Akteur wurde nur aus Übersichtlichkeitsgründen hinzugefügt und stellt keinen zweiten Nutzer dar.



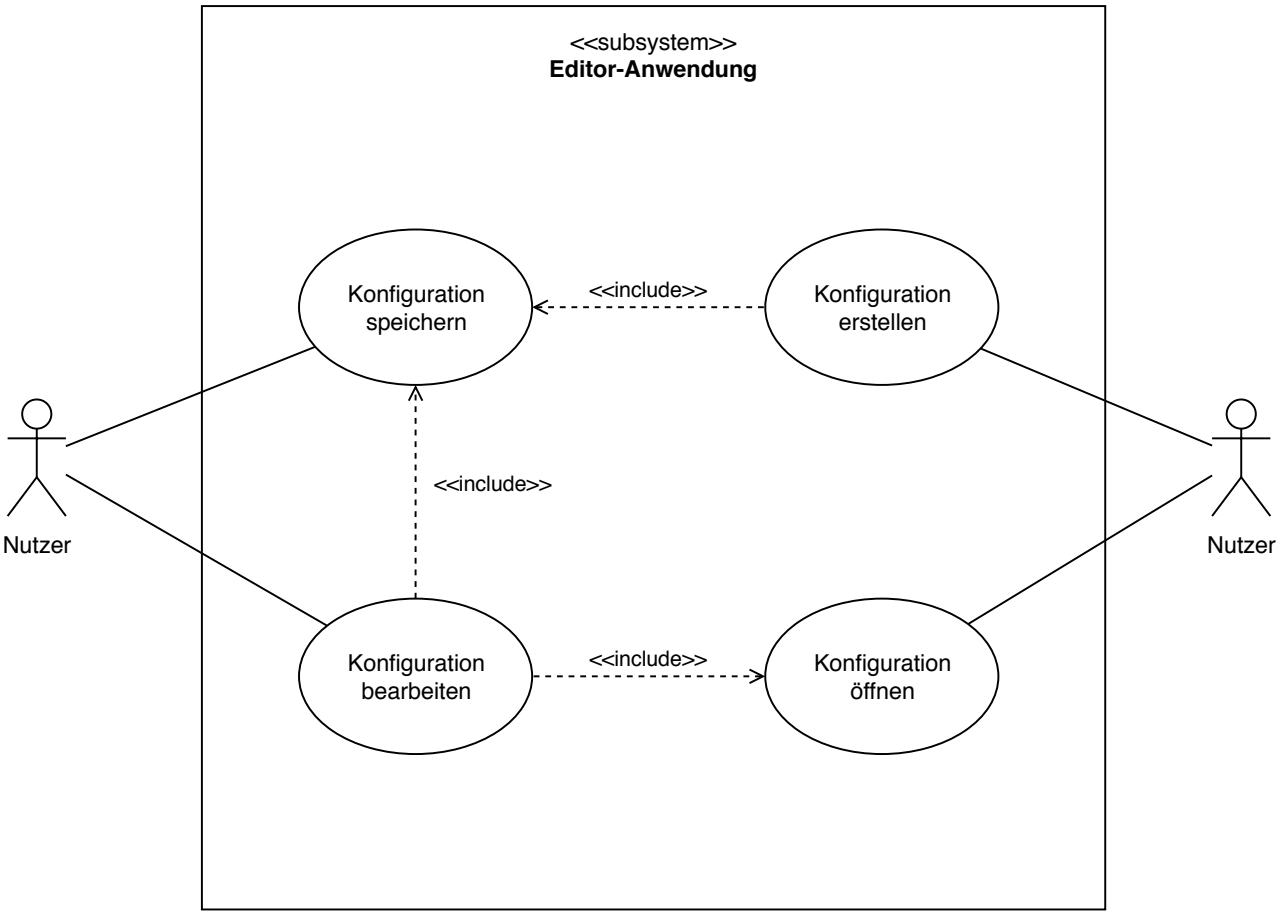
2.2.2 KI-Client

Anwendungsfälle des KI-Clients. Diese Anwendung verbindet sich mit einem Server und simuliert mithilfe einer zuvor konfigurierten KI einen menschlichen Gegenspieler. Der Anwendungsfall „Spiel spielen“ wird unter 2.2.4 konkretisiert.



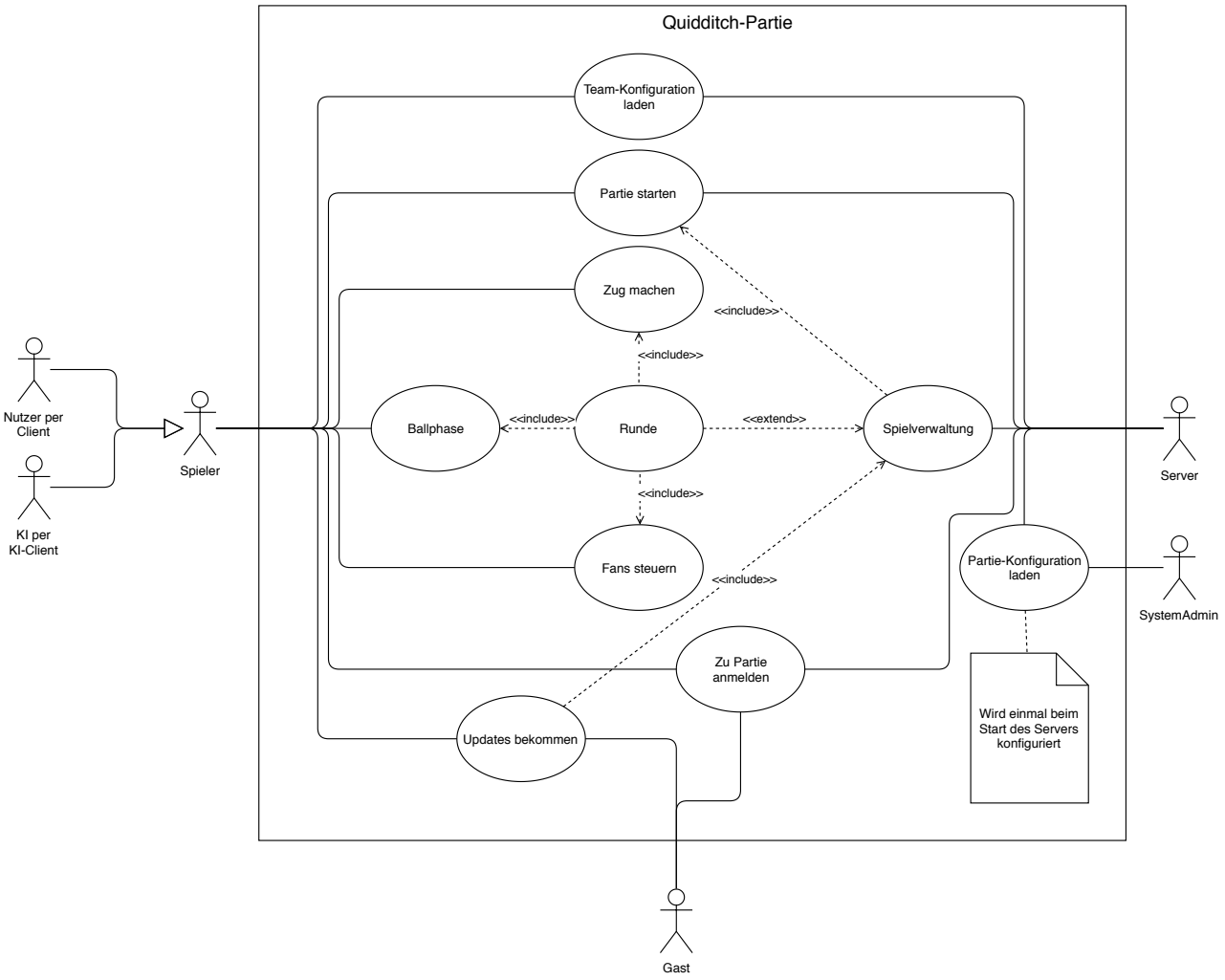
2.2.3 Editor

Anwendungsfälle der Editor-Anwendung. Der zweite Akteur wurde nur aus Übersichtlichkeitsgründen hinzugefügt und stellt keinen zweiten Nutzer dar.



2.2.4 Partie

Mögliche Anwendungsfälle, die während einer Partie auftreten. Die Runde stellt einen abstrakten Anwendungsfall dar, der sich aus den einzelenen Phasen einer Runde zusammensetzt.



2.3 Zuordnung der Anforderungen zu Anwendungsfällen

Hier wird aufgelistet, welche funktionalen Anforderungen (siehe 3) von den jeweiligen Anwendungsfällen abgedeckt werden.

2.3.1 Hauptmenü öffnen

FA60

2.3.2 Hilfe anzeigen

FA65

2.3.3 Spielende-Bildschirm

FA49, FA62

2.3.4 Spiel visualisieren

FA1, FA2, FA3, FA4, FA5, FA6, FA64

2.3.5 Spiel beobachten

FA55, FA66

2.3.6 Spiel beitreten

FA55, FA61

2.3.7 Spiel spielen

FA14, FA15, FA16, FA17, FA18, FA19, FA20, FA55, FA67, FA68, FA69, FA73

2.3.8 Serverkonfiguration einstellen

FA74

2.3.9 Team-Konfiguration laden

FA75

2.3.10 Konfiguration erstellen

FA14, FA15, FA53, FA54, FA71

2.3.11 Konfiguration speichern

FA14, FA15, FA53, FA54, FA71

2.3.12 Konfiguration bearbeiten

FA14, FA15, FA53, FA54, FA70, FA72

2.3.13 Konfiguration öffnen

FA14, FA15, FA53, FA54, FA63

2.3.14 Team-Konfiguration laden

FA14, FA15, FA54, FA55, FA63

2.3.15 Partie starten

FA50, FA51, FA55

2.3.16 Zug machen

FA8, FA21, FA22, FA24, FA26, FA27, FA28, FA29, FA30, FA37, FA38, FA39, FA40, FA41, FA42, FA46, FA55

2.3.17 Runde

FA10, FA11, FA12, FA13, FA25, FA44, FA55

2.3.18 Ballphase

FA10, FA11, FA12, FA13, FA29, FA45, FA55

2.3.19 Fans steuern

FA31, FA32, FA33, FA34, FA35, FA47, FA55

2.3.20 Spielverwaltung

FA1, FA2, FA3, FA4, FA5, FA6, FA7, FA9, FA23, FA30, FA36, FA43, FA48, FA49, FA52, FA53, FA54, FA56, FA58, FA59

2.3.21 Updates bekommen

FA55

2.3.22 Zu Partie anmelden

FA55

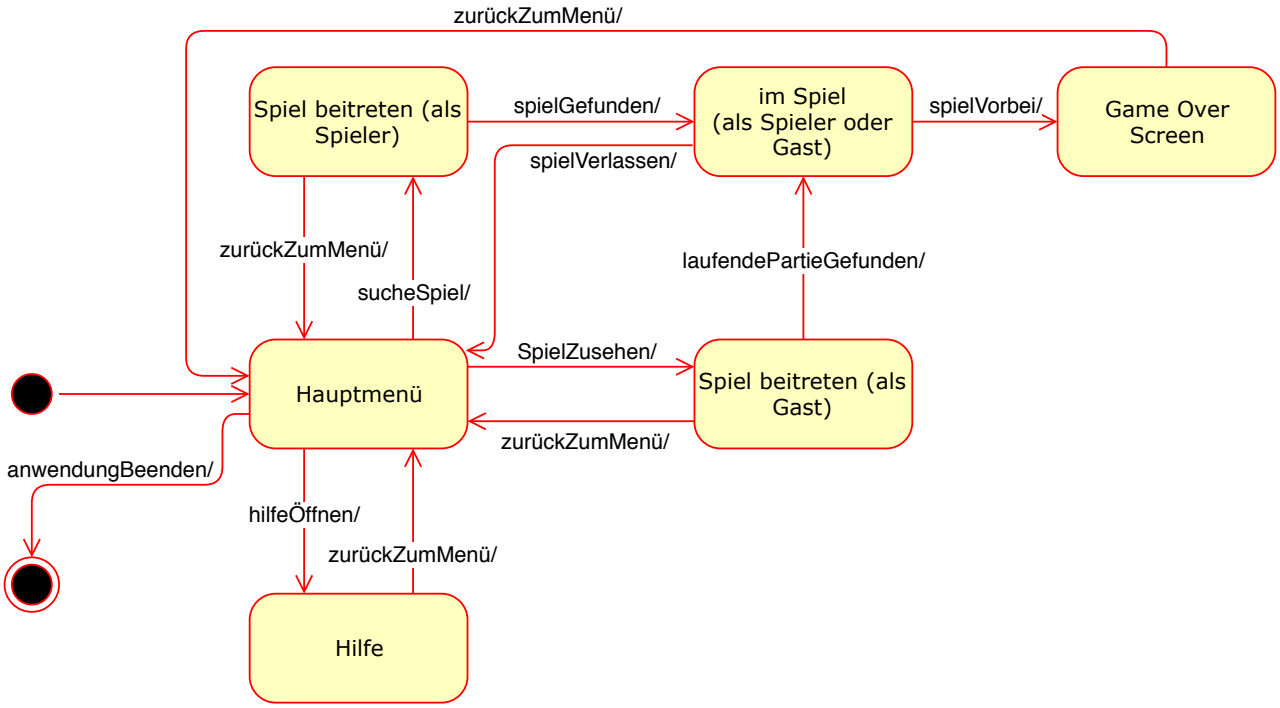
2.3.23 Partie-Konfiguration laden

FA53, FA57

2.4 Abläufe im System

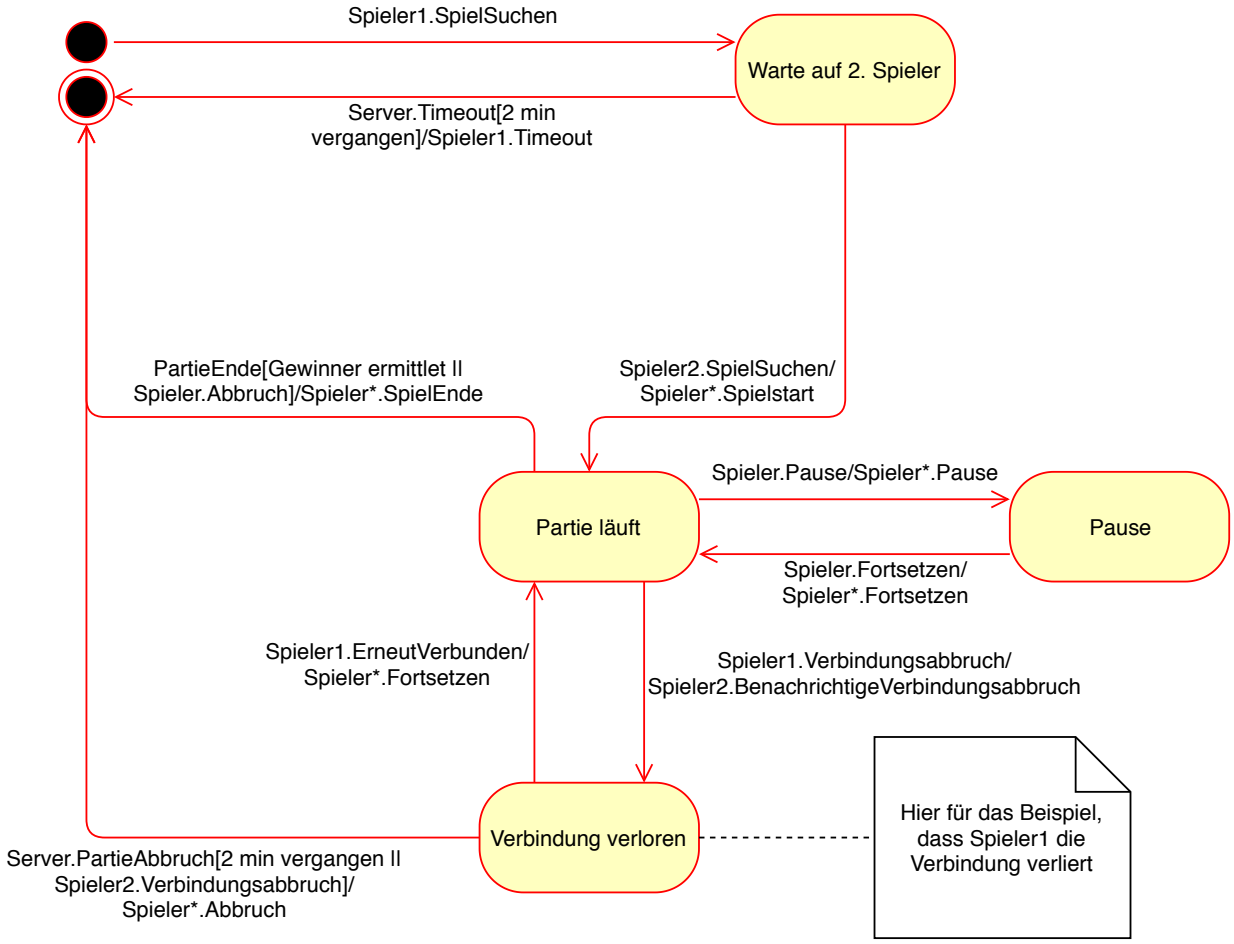
2.4.1 State-Machine Clientanwendung

Zustandsdiagramm für die verschiedenen Ansichten der Client-Anwendung.



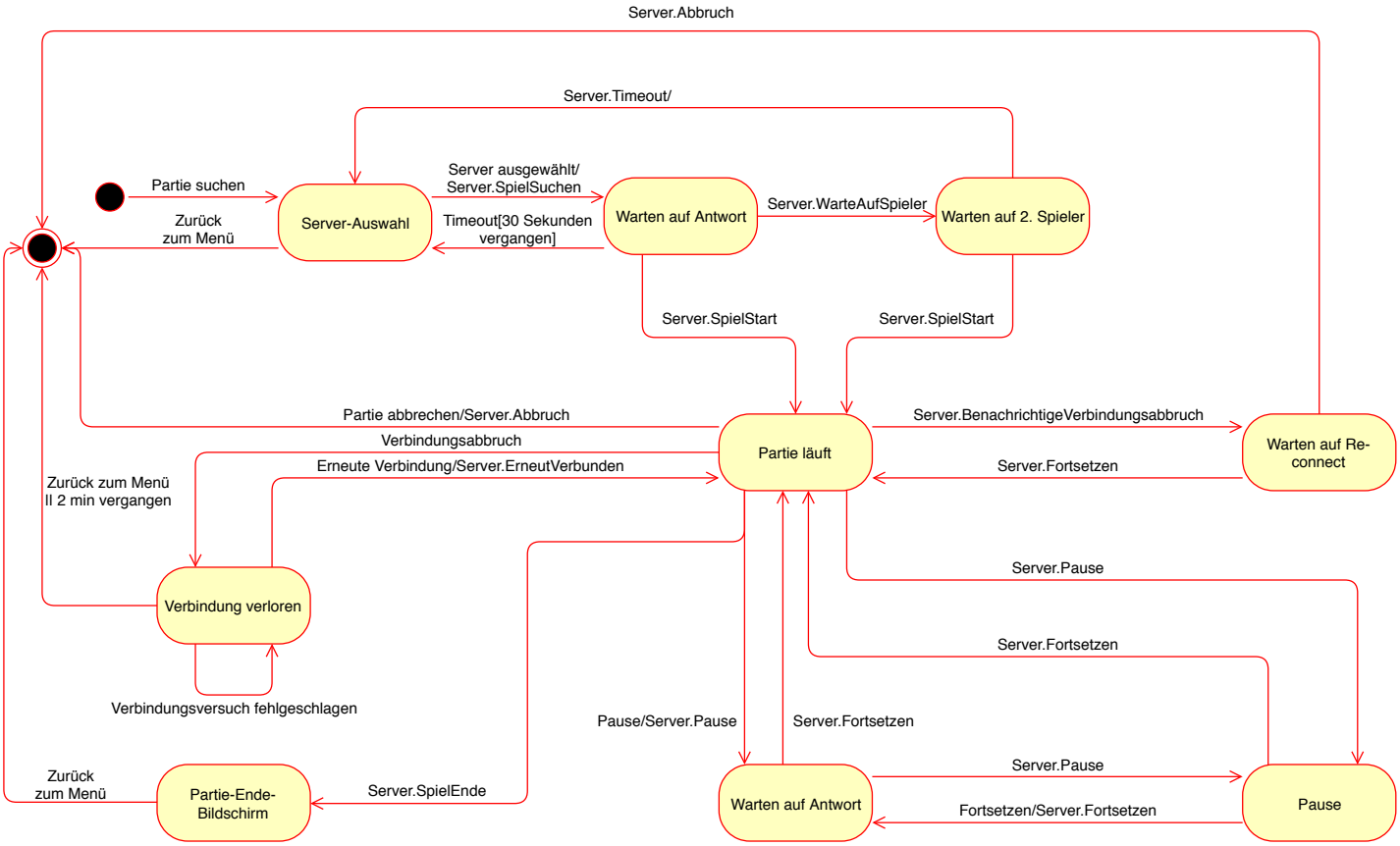
2.4.2 State-Machine Partie Server

Zustandsdiagramm des Servers für eine komplette Partie, inklusive Anmeldung der Spieler und eventuelle Verbindungsabbrüche.



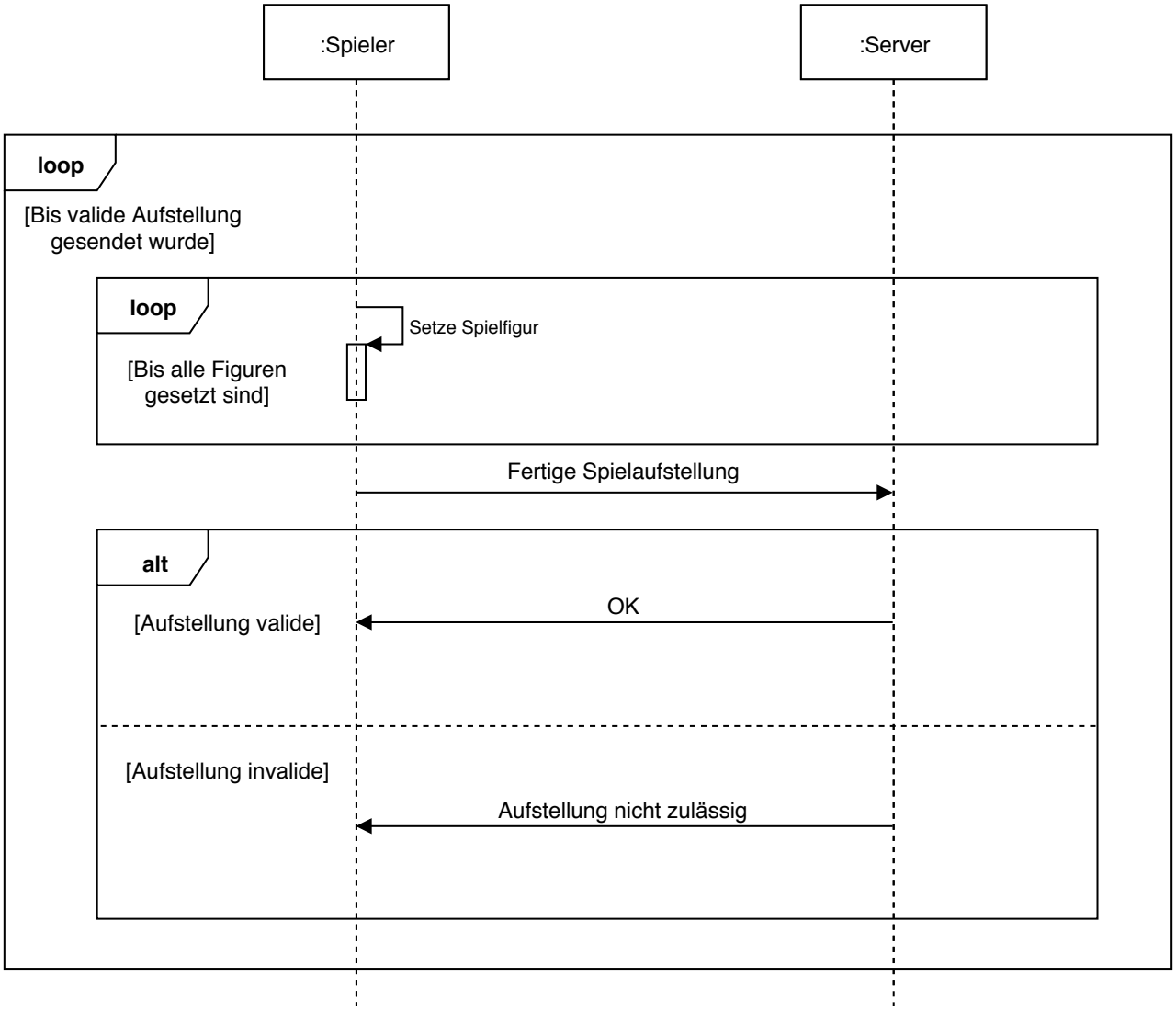
2.4.3 State-Machine Partie Client

Zustandsdiagramm der Client-Anwendung für eine komplette Partie.



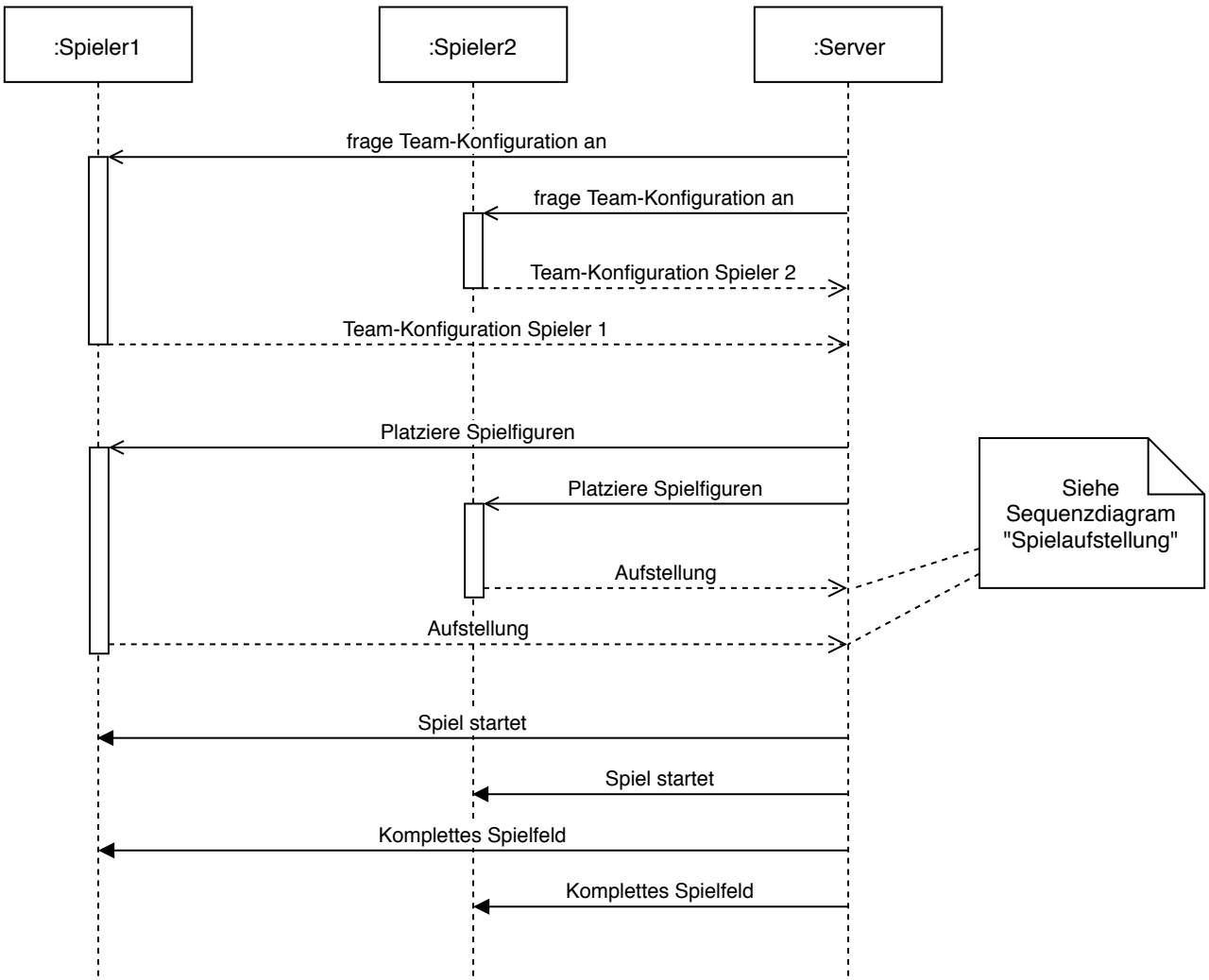
2.4.4 Sequenzdiagramm Spielaufstellung

Kommunikation zwischen Spieler und Server während der Spielaufstellungsphase.



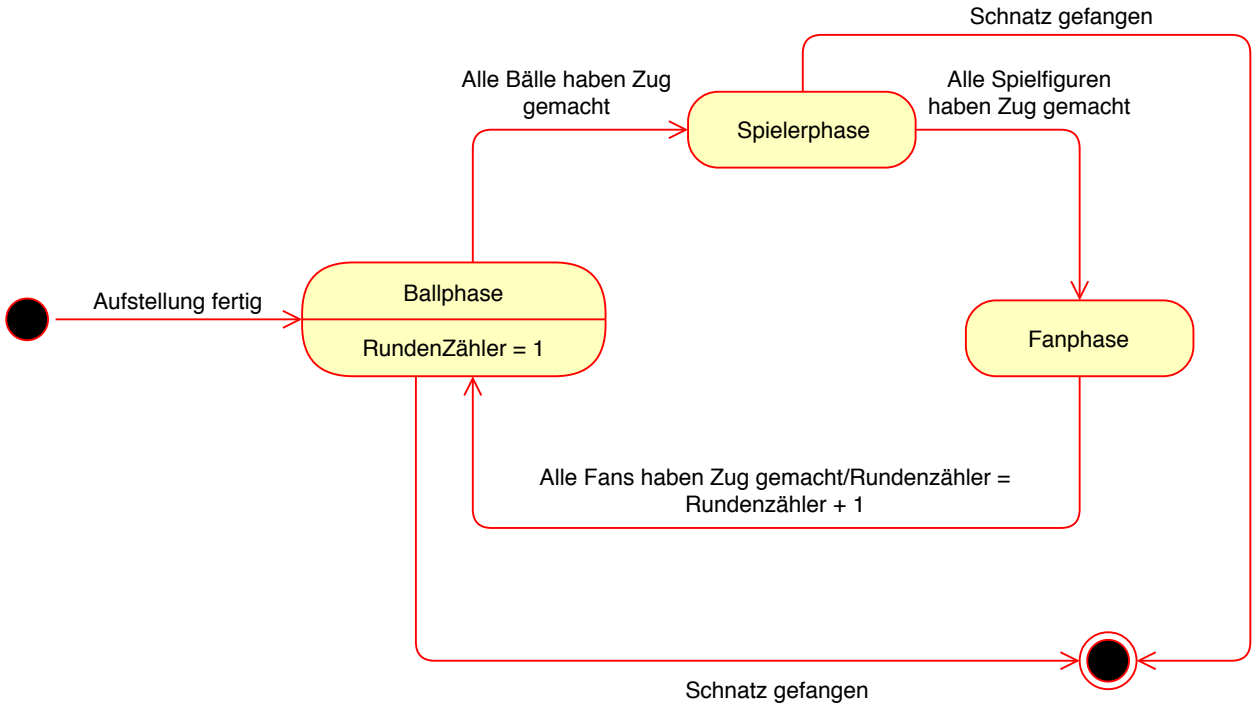
2.4.5 Sequenzdiagramm Spielvorbereitung

Nachrichtenaustausch zwischen Server und den Spielern während der Vorbereitung auf die Partie.

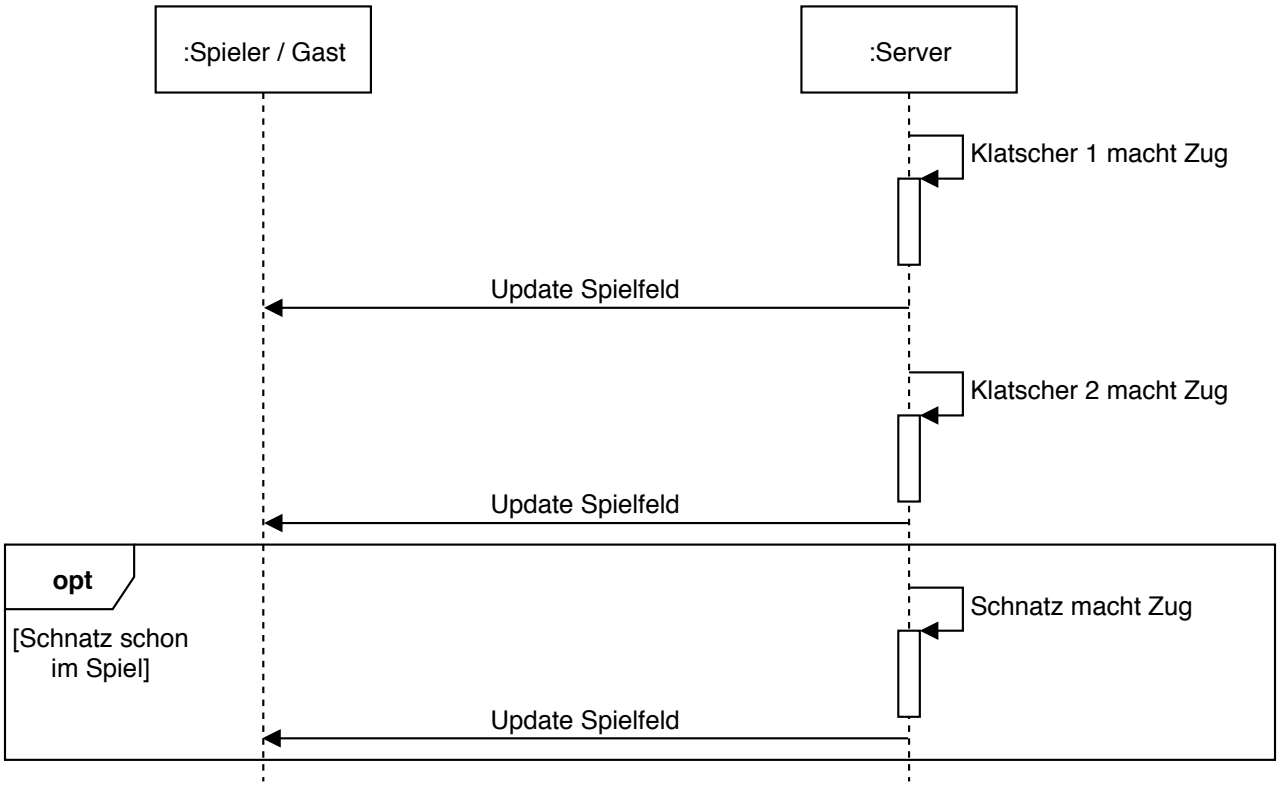


2.4.6 State-Machine Rundenablauf

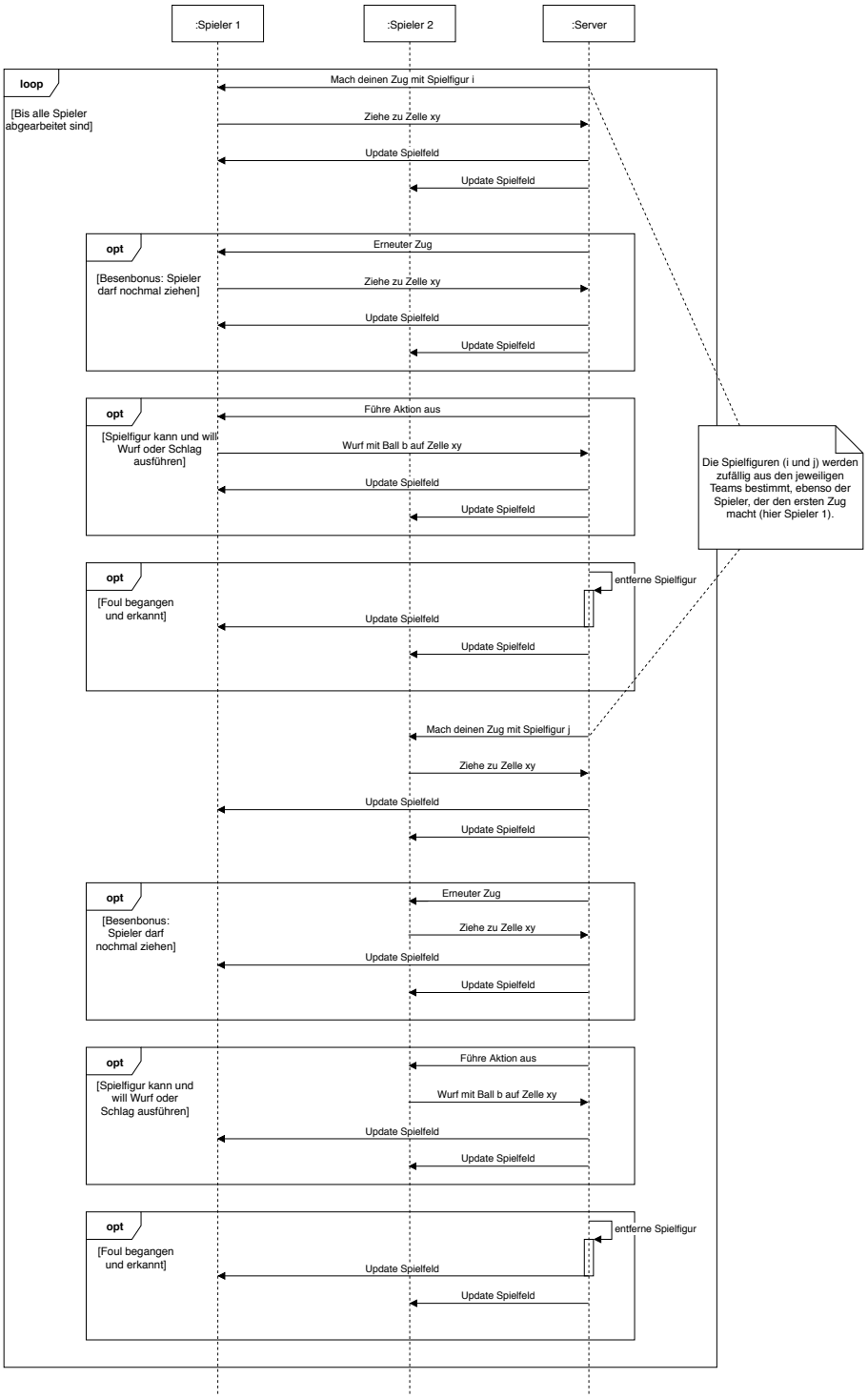
Ablauf der Spielphasen im Spiel



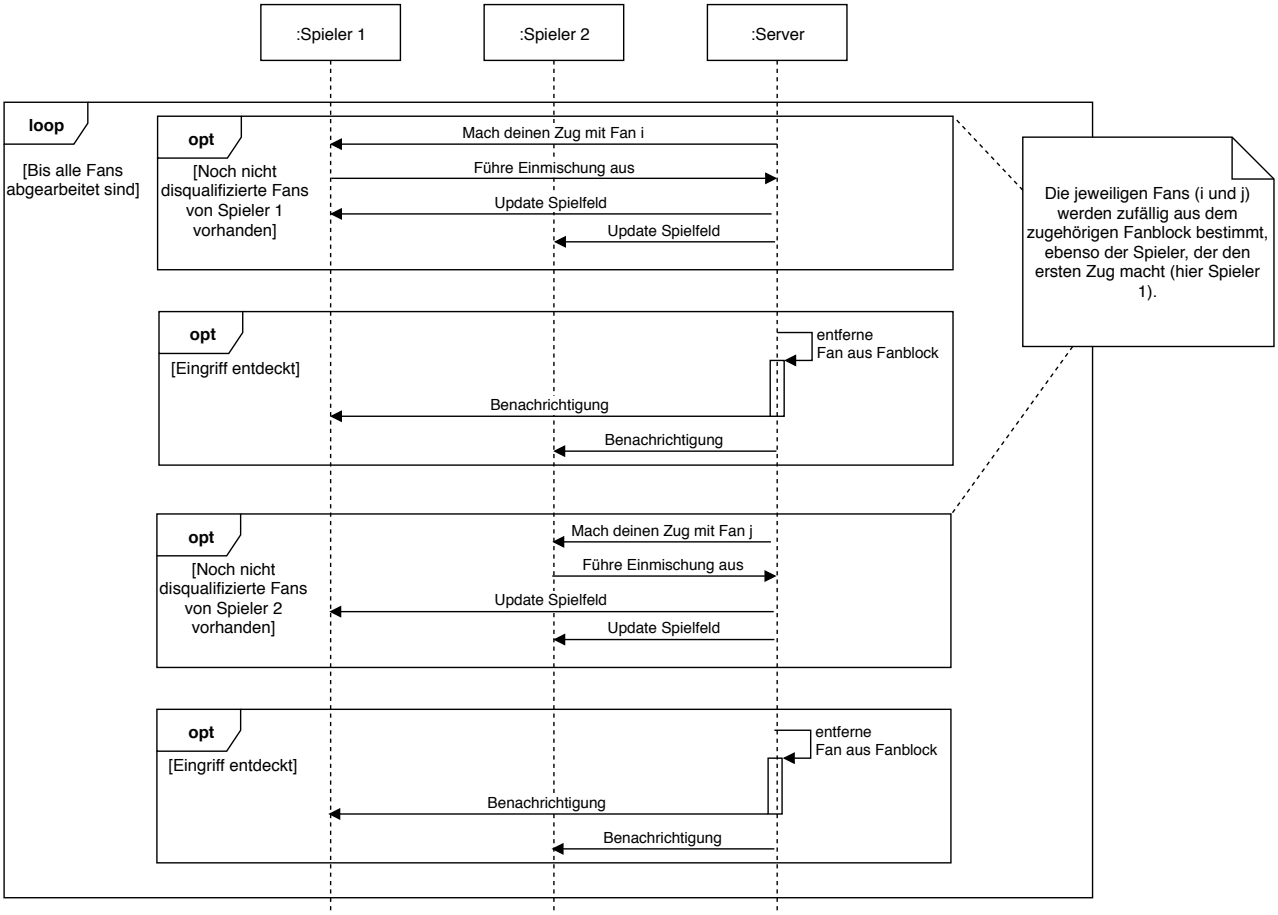
2.4.7 Sequenzdiagramm Ballphase



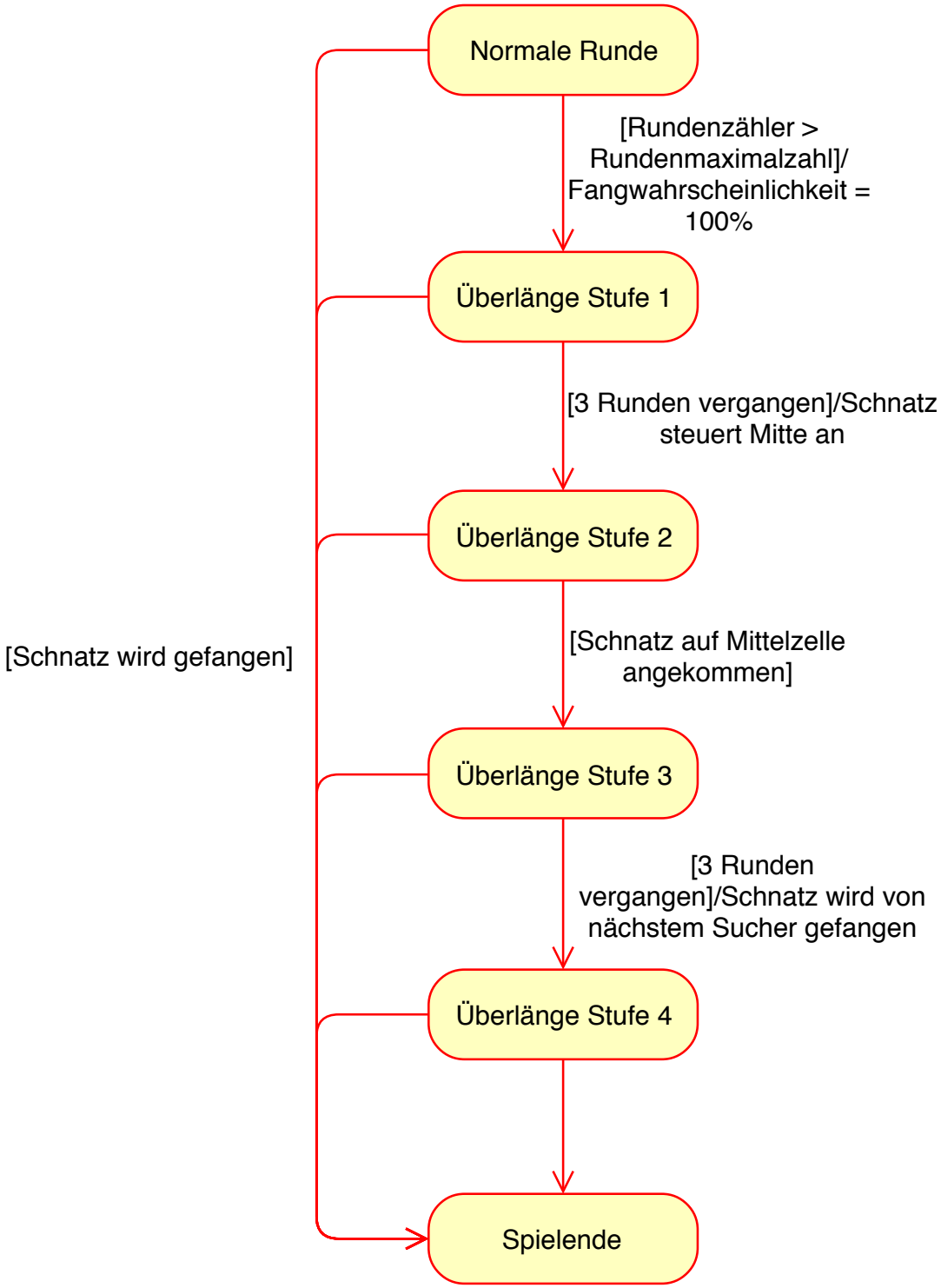
2.4.8 Sequenzdiagramm Spielerphase



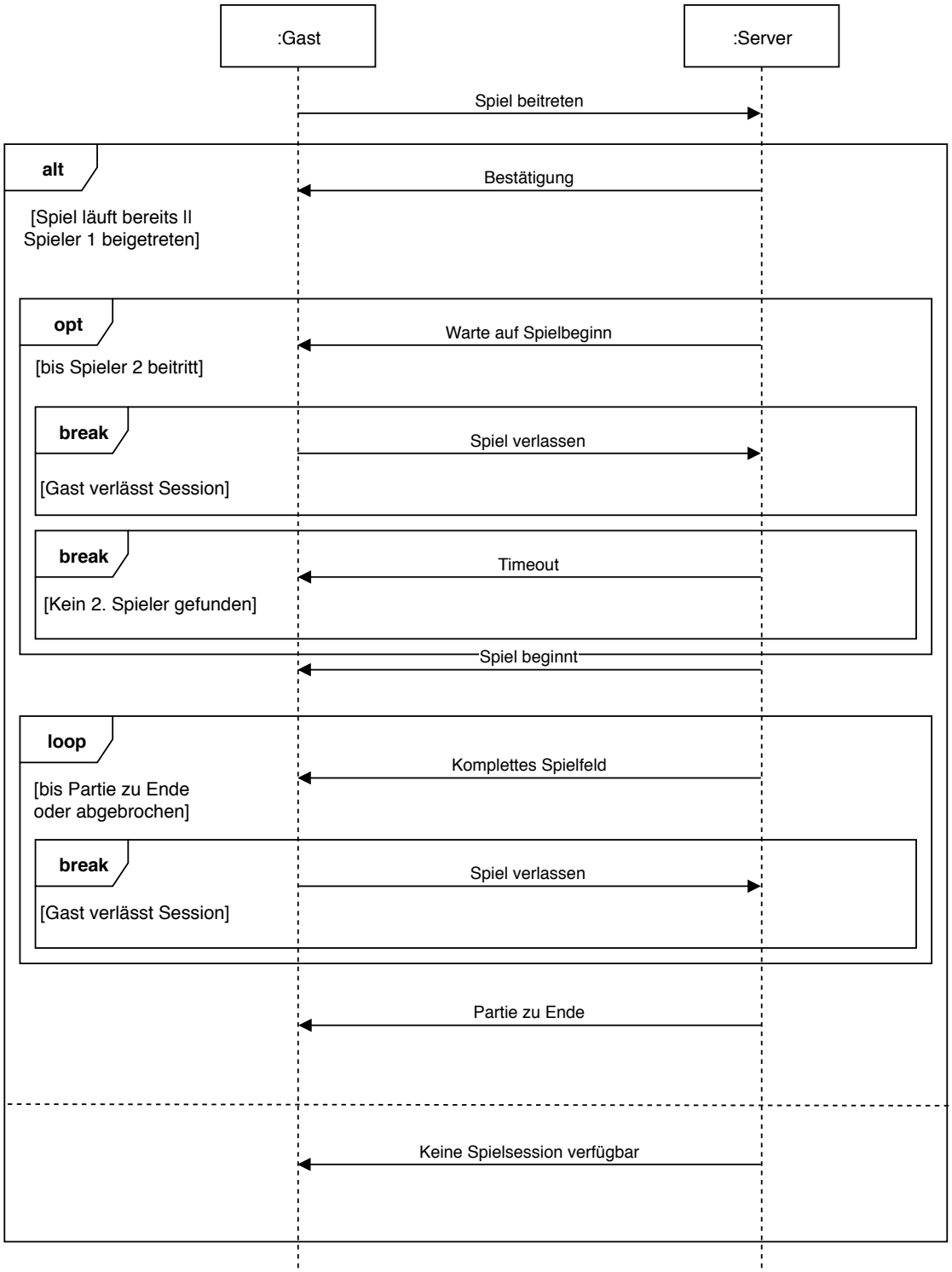
2.4.9 Sequenzdiagramm Fanphase



2.4.10 State-Machine Überlängenbehandlung



2.4.11 Sequenzdiagramm Gast



3 Anforderungsdefinition

Im Folgenden sind alle funktionalen Anforderungen aufgeführt. Dies sind Anforderungen, die am Ende realisiert werden müssen, um alle geforderten Funktionen zu erfüllen. Dabei besitzt jede Anforderung eine eindeutige ID, über die sie im gesamten Dokument referenziert ist.

Bemerkungen zu den Abhängigkeiten der Anforderungen:
Abhängigkeiten werden aus Gründen der Übersichtlichkeit vererbt. Beispielsweise besitzt die Zentrumszelle implizit alle Abhängigkeiten des Zentrums.

Bemerkungen zu den Prioritäten:

- \- Optionale Komponente
- 0 Geforderte Komponente
- \+ Geforderte, wichtige aber nicht zeitkritisch Komponente
- ++ Geforderte, wichtige und zeitkritische Komponente

3.1 Funktionale Anforderungen: Spielregeln

Die im Folgenden aufgeführten funktionalen Anforderungen beschäftigen sich primär mit Funktionen, die entweder direkt aus den Spielregeln hervorgehen oder Teil der Umsetzung der Spielmechanik sind.

ID:	**FA1**
Titel:	Quidditch-Spielfeld
Beschreibung:	Das Quidditch-Spielfeld hat eine Ovale Form, die in ein Raster aus 17x13 quadratischen Zellen eingepasst ist. Auf diesem Feld finden alle Spielhandlungen statt, die während des Spiels getätigt werden können.
Begründung:	Das Spielfeld ist die zentrale Komponente des Spiels, da sich hier während einer Partie sämtliche Abläufe abspielen.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA2**
Titel:	Zentrum
Beschreibung:	Das Zentrum ist ein Bereich auf dem Quidditch-Spielfeld, der in der Mitte angeordnet ist und aus 3x3 quadratischen Zellen besteht. Zu Beginn dürfen sich hier keine Spielfiguren befinden.
Begründung:	Das Zentrum markiert den Bereich um die Zentrumszelle des Spielfeldes, in der das Spiel gestartet wird.
Abhängigkeiten:	Spielfeld
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA3**
Titel:	Zentrumszelle
Beschreibung:	Die Zentrumszelle stellt den mittleren Punkt des Zentrums dar.
Begründung:	Die Zentrumszelle ist der Startpunkt für die Bälle beim Spielstart.
Abhängigkeiten:	Zentrum
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA4**
Titel:	Hüterzonen
Beschreibung:	Die Hüterzonen sind an den jeweils gegenüberliegenden Seiten des Quidditch-Spielfeldes platziert. Die Hüterzonen sind ovalförmig, bestehen aus 11x5 Zellen und beinhalten jeweils drei Torringe.
Begründung:	In den Hüterzonen können die Teams Punkte erzielen.
Abhängigkeiten:	Spielfeld
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA5**
Titel:	Zelle
Beschreibung:	Die Zelle ist die kleinste Einheit des Spielfeldes, auf ihr darf sich immer nur eine Spielfigur gleichzeitig befinden. Bälle können sich jedoch eine Zelle mit einem anderen Ball und / oder einer Spielfigur teilen.
Begründung:	Das gesamte Spielfeld ist aus Zellen aufgebaut. Sie bestimmen, wie sich Spielobjekte bewegen können.
Abhängigkeiten:	Spielfeld
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA6**
Titel:	Torring
Beschreibung:	Die Teams können Punkte erzielen, indem sie den Quaffel durch einen gegnerischen Torring schießen.
Begründung:	Die Torringe dienen den Teams als Hauptquelle von Punkten.
Abhängigkeiten:	Hüterzone
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA7**
Titel:	Schussvektorberechnung
Beschreibung:	Ein Schussvektor zeigt vom Mittelpunkt der Startzelle auf die Zielzelle des Schusses. Alle Zellen, die von diesem Vektor geschnitten werden, sind sogenannte überstrichene Zellen.
Begründung:	Ein Schussvektor beschreibt, wie eine Spielfigur einen Ball über das Spielfeld bewegen kann.
Abhängigkeiten:	Zelle
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA8
Titel:	Punkte erzielen
Beschreibung:	Es gibt zwei Möglichkeiten, Punkte zu erzielen: Torschüsse (entsprechen 10 Punkten) oder den Goldenen Schnatz fangen (entspricht 30 Punkten).
Begründung:	Die Punktezahl zeigt an, welcher Spieler sich im Moment besser schlägt und dient zur Bestimmung des Gewinners am Ende der Partie.
Abhängigkeiten:	Schnatz fangen, Quaffel schießen
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA9
Titel:	Entfernungsberechnung
Beschreibung:	Die Entfernung zwischen zwei Zellen ist die kleinstmögliche Anzahl an Zügen, die man braucht, um von Zelle A zu Zelle B zu kommen. Dabei darf man sich in alle Richtungen bewegen, also vertikal, horizontal und Diagonal.
Begründung:	Die Entfernung ist maßgeblich für den Erfolg von verschiedenen Aktionen, wie z.B. dem Schießen des Quaffels.
Abhängigkeiten:	Zelle
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA10
Titel:	Bälle
Beschreibung:	Es gibt 3 verschiedene Arten von Bällen: Den Quaffel, die Klatscher und den Schnatz.
Begründung:	Die Bälle sind zentraler Bestandteil des Spiels.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA11
Titel:	Quaffel [Ball]
Beschreibung:	Der Quaffel ist ein roter Lederball, mit dem die Teams Punkte erzielen können.
Begründung:	Der Quaffel ist die zentrale Punktequelle.
Abhängigkeiten:	Bälle
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA12
Titel:	Klatscher [Ball]
Beschreibung:	Die Klatscher sind kleine schwarze Bälle, die sich von alleine auf Spieler zubewegen (eine Zelle pro Runde), die keine Treiber sind.
Begründung:	Die Klatscher verleihen dem Spiel zusätzliche taktische Tiefe, da sie Spielfiguren betäuben können.
Abhängigkeiten:	Bälle
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA13
Titel:	Schnatz [Ball]
Beschreibung:	Der Schnatz ist eine kleiner, goldener Ball, der sich von alleine von Suchern wegbewegt (eine Zelle pro Runde). Das bedeutet, er achtet auf den nächsten Sucher und wählt unter allen möglichen freien Zellen, die eine größere Entfernung zu diesem haben, als seine gegenwärtige, eine zufällige aus, und bewegt sich auf diese Zelle. Falls es keine solchen Zellen gibt, bewegt sich der Schnatz auf eine zufällige freie Nachbarzelle. Der Schnatz erscheint zu Beginn der dreizehnten Runde auf einer zufällig gewählten freien Zelle, die möglichst gleich weit von beiden Suchern entfernt ist.
Begründung:	Der Schnatz dient zum Punkteerzielen und führt, wenn er gefangen wird, zum Ende des Partie.
Abhängigkeiten:	Bälle
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA14
Titel:	Besen
Beschreibung:	Jede Spielfigur besitzt einen Besen, der einen der folgenden Typen haben: Zauberfauch, Sauberwisch 11, Komet 2-60, Nimbus 2001 oder Feuerblitz. Der Rang des Besens bestimmt die Wahrscheinlichkeit, mit der eine Spielfigur nach einer Bewegung um eine Zelle eine weitere Bewegung ausführen darf. Diese Wahrscheinlichkeit wird in der Partiekonfiguration festgelegt, wobei die Besen in der genannten Reihenfolge aufsteigende Wahrscheinlichkeiten besitzen.
Begründung:	Die Besen geben den Spielfiguren eine unterschiedliche Qualität.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA15
Titel:	Teams
Beschreibung:	Ein Team besteht aus sieben Spielfiguren und sieben Einmischungen. Außerdem hat jedes Team einen Namen, ein Motto, eine Hauptteamfarbe und eine Ersatzteamfarbe. Die sieben Spielfiguren teilen sich wie folgt auf: ein Hüter, zwei Treiber, drei Jäger und ein Sucher. Bei den Spielfiguren darf jedes Geschlecht bis zu vier mal vertreten sein. Zudem muss jeder Besentyp einmal vertreten sein. Bei den sieben Einmischungen muss jeder Typ mindestens einmal vertreten sein.
Begründung:	Quidditch ist ein Teamspiel, weshalb Teams benötigt werden.
Abhängigkeiten:	Spielfigur, Einmischungen, Besen
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA16
Titel:	Spielfiguren
Beschreibung:	Es gibt 4 Arten von Spielfiguren: Jäger, Sucher, Hüter und Treiber. Jede Spielfigur hat dabei einen Namen und ein Geschlecht.
Begründung:	Die unterschiedlichen Typen der Spielfiguren geben dem Spiel taktische Tiefe.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA17
Titel:	Jäger [Spielfigur]
Beschreibung:	Jäger können den Quaffel aufnehmen und schießen und damit Punkte für ihr Team erzielen.
Begründung:	Jäger können Punkte für ihr Team erzielen.
Abhängigkeiten:	Spielfigur
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA18
Titel:	Treiber [Spielfigur]
Beschreibung:	Treiber können den Klatscher schlagen und somit zum Gegner hin und / oder von Teammitgliedern weg befördern.
Begründung:	Treiber dienen zum Schutz des eigenen Teams vor den Klatschern. Gleichzeitig können sie den Gegner aktiv sabotieren, in dem sie ihm den Klatscher zuspielen.
Abhängigkeiten:	Spielfigur
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA19
Titel:	Hüter [Spielfigur]
Beschreibung:	Hüter können den Quaffel aufnehmen und versuchen, den Gegner daran zu hindern, ein Tor zu erzielen. Landet der Quaffel auf einem Torring so geht er am Ende der Rundenphase in den Besitz des Hüters über, wenn er sich selbst in der Hüterzone befindet. Ein Hüter kann selbst keine Tore erzielen.
Begründung:	Hüter stellen die Verteidigung seines Teams dar.
Abhängigkeiten:	Spielfigur, Hüterzone
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA20
Titel:	Sucher [Spielfigur]
Beschreibung:	Sucher versuchen, den Schnatz zu finden, um Punkte zu erzielen und das Spiel zu beenden.
Begründung:	Der Sucher beendet das Spiel.
Abhängigkeiten:	Spielfigur
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA21
Titel:	Quaffel Schießen
Beschreibung:	Hüter und Jäger können den Quaffel schießen. Der Schuss wird über ein Schussvektor angegeben. Jede gegnerische Spielfigur, die sich auf einer überstrichenen Zelle des Schussvektors befindet, kann den Quaffel abfangen. Wird der Ball von keiner Spielfigur abgefangen, so ist der Schuss mit der Wahrscheinlichkeit P^d erfolgreich, wobei P eine elementare Wurfwahrscheinlichkeit und d die Entfernung zur Zielzelle ist. War der Schuss erfolgreich, so landet der Quaffel auf der Zielzelle. Wenn nicht wird der Quaffel auf einer zufälligen freien Zelle in einem $n \times n$ Quadrat um die Zielzelle platziert, wobei $n = \lceil \frac{d}{7} \rceil$ ist.
Begründung:	Das Schießen des Quaffels ermöglicht Passspiel und das erzielen von Punkten.
Abhängigkeiten:	Schussvektorberechnung, Entfernungsberechnung, Jäger, Hüter, Quaffel
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA22
Titel:	Quaffel Abfangen
Beschreibung:	Jede gegnerische Spielfigur auf einer überstrichenen Zelle des Schussvektors eines Schusses hat eine gewisse Wahrscheinlichkeit, den Quaffel abzufangen. Diese Wahrscheinlichkeit ist in der Partiekonfiguration vermerkt. Gelingt das Abfangen, so landet der Quaffel auf der Zelle der abfangenden Spielfigur.
Begründung:	Das Abfangen bietet die Möglichkeit, Schüsse des Gegners zu unterbinden.
Abhängigkeiten:	Quaffel-Schießen, Spielfiguren
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA23**
Titel:	Quaffel Abprallen
Beschreibung:	Landet der Quaffel auf einer Zelle, auf der sich eine Spielfigur befindet, die weder Hüter noch Jäger ist, so wird der Quaffel auf eine zufällige freie Nachbarzelle gesetzt.
Begründung:	Nur Jäger und Hüter können direkt mit dem Quaffel interagieren.
Abhängigkeiten:	Sucher, Treiber, Quaffel
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA24**
Titel:	Quaffel Halten
Beschreibung:	Ein Jäger oder Hüter kann den Quaffel halten, wodurch sich der Quaffel immer auf der selben Zelle befindet wie besagter Jäger bzw. Hüter.
Begründung:	Jäger und Hüter sollen den Quaffel, ohne zu passen, über das Spielfeld transportieren können.
Abhängigkeiten:	Jäger, Hüter, Quaffel
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA25**
Titel:	Quaffel Verlieren
Beschreibung:	Ein Jäger oder Hüter kann den Quaffel verlieren, was bedeutet, dass der Quaffel auf eine freie Nachbarzelle zufällig bewegt wird.
Begründung:	Durch das Verlieren des Quaffels werden Tore im Alleingang erschwert.
Abhängigkeiten:	Jäger, Hüter, Quaffel, Fouls, Klatscher
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA26
Titel:	Quaffel Übernehmen
Beschreibung:	Ein Jäger darf von einem anderen Jäger den Quaffel übernehmen, sofern sich dieser auf einer seiner Nachbarzellen befindet. Dies gelingt allerdings nur mit einer bestimmten Wahrscheinlichkeit. Jäger dürfen auch von einem gegnerischen Hüter den Quaffel übernehmen. Das gelingt jedoch nur, wenn sich der Hüter nicht in seiner eigenen Hüterzone befindet.
Begründung:	Bietet zusätzliche Möglichkeiten, den Ball zu erobern.
Abhängigkeiten:	Jäger, Hüter, Quaffel
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA27
Titel:	Tor Erzielen
Beschreibung:	Ein Jäger kann ein Tor erzielen, indem er den Quaffel erfolgreich auf eine Torringzelle schießt. Dabei muss der Schussvektor des Wurfes durch die linke oder rechte Seite der Torringzelle gehen.
Begründung:	Stellt für einen Jäger die Möglichkeit dar, Punkte zu erzielen.
Abhängigkeiten:	Torring, Quaffel Schießen, Jäger
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA28
Titel:	Klatscher Schlagen
Beschreibung:	Ein Treiber kann einen Klatscher schlagen, wenn er sich auf derselben Zelle wie der Klatscher befindet. Der Treiber wählt, um den Klatscher zu schlagen, eine Zielzelle aus, die eine maximale Entfernung von drei zu ihm hat. Zusätzlich müssen auch alle überstrichenen Zellen frei sein. Ist dies beides der Fall, so wird das Schlagen des Klatschers wie ein normaler Quaffel-Schuss behandelt, allerdings mit einer Wahrscheinlichkeit von 100%.
Begründung:	Stellt für den Treiber die Möglichkeit dar, den Katscher zu bewegen.
Abhängigkeiten:	Spielfeld, Treiber, Klatscher
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA29
Titel:	Spieler Betäuben
Beschreibung:	Bewegt sich der Klatscher auf deine Zelle, auf der sich eine Spielfigur befindet, die kein Treiber ist, wird diese mit einer gewissen Wahrscheinlichkeit betäubt. Das hat zur Folge, dass diese Spielfigur gegebenenfalls den Quaffel verliert, keinen Ball fangen und für eine Runde keine Aktion ausführen kann. Der Klatscher, der den Spieler betäubt hat, wird auf eine zufällige freie Zelle gesetzt.
Begründung:	Stellt ein zusätzliches taktisches Spielelement dar.
Abhängigkeiten:	Klatscher, Jäger, Hüter, Sucher
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA30
Titel:	Schnatz Fangen
Beschreibung:	Befinden sich ein Sucher und der Schnatz auf derselben Zelle, so fängt der Sucher den Schnatz mit einer gewissen Wahrscheinlichkeit. Dadurch erhält sein Team 30 Punkte und das Spiel ist zu Ende.
Begründung:	Das Schnatzfangen führt das Ende der Partie herbei.
Abhängigkeiten:	Schnatz, Sucher, Punkte
Priorität:	++
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA31
Titel:	Einmischung
Beschreibung:	Eine Einmischung ist eine Aktion, die das eigene Team unterstützt und / oder das gegnerische Team schwächt. Diese Einmischungen werden mit einer gewissen Wahrscheinlichkeit vom Schiedsrichter geahndet und werden vom jeweiligen Spieler in der Endphase ausgelöst. Es gibt die folgenden Typen von Einmischungen: Teleportation, Fernangiff, Impus und Schnatzstoß.
Begründung:	Sorgen für Abwechslung, Witz, und Überraschungen im Spiel.
Abhängigkeiten:	-
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA32
Titel:	Teleportation [Einmischungstyp]
Beschreibung:	Bei der Teleportation wird ein Spieler von der eigenen oder auch von der gegnerischen Mannschaft auf eine zufällige freie Zelle teleportiert.
Begründung:	Siehe Einmischung
Abhängigkeiten:	Einmischung
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA33
Titel:	Fernangriff [Einmischungstyp]
Beschreibung:	Der Fernangriff bewirkt, dass der getroffene Spieler den Quaffel verliert, sofern er diesen hat und anschließend auf eine zufällige freie Nachbarzelle gestoßen wird.
Begründung:	Siehe Einmischung
Abhängigkeiten:	Einmischung
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA34
Titel:	Impuls [Einmischungstyp]
Beschreibung:	Der Impuls sorgt dafür, dass der Spieler, der den Quaffel hält, diesen verliert.
Begründung:	Siehe Einmischung
Abhängigkeiten:	Einmischung
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA35
Titel:	Schnatzstoß [Einmischungstyp]
Beschreibung:	Bei einem Schnatzstoß macht der Schnatz eine Ausweichbewegung auf eine freie Nachbarzelle.
Begründung:	Siehe Einmischung
Abhängigkeiten:	Einmischung
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA36
Titel:	Schiedsrichter
Beschreibung:	Der Schiedsrichter ahndet mit einer gewissen Wahrscheinlichkeit Fouls von Spielern oder Einmischungen. Ahndet der Schiedsrichter eine Aktion, so wird die verursachende Spielfigur bis zum nächsten Tor blockiert, bzw. genau diese Einmischung ist für die restliche Partie gebannt (nur für das ausführende Team).
Begründung:	Der Schiedsrichter ist die rechtschaffende Instanz und sorgt dafür, dass unfaire Aktionen nicht bedenkenlos eingesetzt werden können.
Abhängigkeiten:	Spielfigur, Einmischung
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA37
Titel:	Foul
Beschreibung:	Fouls sind Spielzüge, die grundsätzlich möglich, jedoch laut Regelwerk nicht zulässig sind. Es gibt die folgenden Arten von Fouls: Stürmen, Großoffensive, Rammen, Torringe blockieren und Schnatz blockieren.
Begründung:	Fouls stellen eine zusätzliche taktische Spielkomponente dar.
Abhängigkeiten:	Spielfigur, Schiedsrichter, Spielfeld
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA38
Titel:	Torring blockieren [Foul]
Beschreibung:	Eine Spielfigur darf sich nicht direkt auf einen Torring stellen, da es sonst unmöglich wäre, durch diesen Torring ein Tor zu erzielen.
Begründung:	Ermöglicht einem Team zu verhindern, dass der Gegner Punkte erzielen kann.
Abhängigkeiten:	Foul
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA39
Titel:	Stürmen [Foul]
Beschreibung:	Diese Aktion kann nur von Jägern ausgeübt werden. Führt ein Jäger diese Aktion aus, so erzielt er zu 100% ein Tor, indem er den Quaffel hält und damit auf einen Torring zieht.
Begründung:	Stellt eine unfaire Möglichkeit dar, Punkte zu erzielen.
Abhängigkeiten:	Foul
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA40
Titel:	Großoffensive [Foul]
Beschreibung:	Diese Aktion kann nur von Jägern ausgeübt werden. Es bedeutet, dass sich zwei oder mehr Jäger vom selben Team in der gegnerischen Hüterzone befinden.
Begründung:	Gibt den Angreifern einen unfairen Vorteil.
Abhängigkeiten:	Foul
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA41
Titel:	Rammen [Foul]
Beschreibung:	Dieses Foul können alle Spielfiguren ausführen. Hierbei zieht eine Spielfigur auf das selbe Spielfeld wie eine gegnerische Spielfigur, woraufhin diese den Quaffel ggf. verliert und anschließend auf eine zufällige freie Nachbarzelle verdrängt wird.
Begründung:	Dieses Foul bietet die Möglichkeit, dem Gegner den Ball zu entwenden.
Abhängigkeiten:	Foul
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA42
Titel:	Schnatz blockieren [Foul]
Beschreibung:	Dieses Foul können alle Spielfiguren ausführen, mit Ausnahme der Sucher. Dazu bewegt sich eine Spielfigur auf die Zelle des Schnatzes, obwohl sie kein Sucher ist, wodurch sie verhindert, dass Sucher den Schnatz fangen können, da sich nie zwei oder mehr Spielfiguren auf ein und der selben Zelle befinden dürfen.
Begründung:	Diese Anforderung soll es dem Sucher schwerer machen, den Schnatz zu fangen.
Abhängigkeiten:	Foul
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA43
Titel:	Setzen auf freies Nachbarfeld
Beschreibung:	Soll eine Spielfigur oder Ball auf eine zufällige freie Nachbarzelle gesetzt werden, obwohl alle acht umliegenden Zellen bereits von Spielfiguren besetzt oder aus anderen Gründen nicht zulässig sind, so wird rekursiv von einer zufällig besetzten Nachbarzelle weiter gesucht, bis sich schließlich eine freie Zelle findet. Das bedeutet, dass die Spielfigur oder Ball nicht unbedingt auf einer Nachbarzelle landet, sondern auch weiter entfernt positioniert werden kann.
Begründung:	Es muss der Fall abgedeckt werden, dass keine freie Nachbarzelle mehr frei ist.
Abhängigkeiten:	Zelle, Bälle, Spielfigur
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA44
Titel:	Runde
Beschreibung:	Das Spiel läuft in Runden ab. Jede Runde ist dabei in Phasen unterteilt: Ballphase, Spielerphase und Endphase.
Begründung:	Bei „Fantastic Feasts“ handelt es sich, laut den Spielregeln, um ein Rundenbasiertes Spiel.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA45
Titel:	Ballphase
Beschreibung:	In dieser Phase eine Spiels bewegen sich die Bälle über das Spielfeld. Dabei machen die beiden Klatscher ihre Bewegungen in zufälliger Reihenfolge.
Begründung:	sieh Runde
Abhängigkeiten:	Runde
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA46
Titel:	Spielerphase
Beschreibung:	Jeder Spieler macht seine Aktion, wobei sich die Teams abwechseln. In jeder Runde wird neu zufällig bestimmt, welches Team dabei beginnt. Die Reihenfolge der Spielfiguren innerhalb eines Teams wird zufällig bestimmt.
Begründung:	Siehe Runde.
Abhängigkeiten:	Runde
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA47
Titel:	Endphase
Beschreibung:	Die Teams lösen abwechselnd die gewünschten Einmischungen aus. In jeder Runde wird neu zufällig bestimmt, welches Team dabei beginnt. Die Reihenfolge der Einmischungen innerhalb eines Teams wird zufällig bestimmt.
Begründung:	Siehe Runde.
Abhängigkeiten:	Runde
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA48**
Titel:	Disqualifikation
Beschreibung:	Werden in derselben Runde mehr als 3 Spielfiguren eines Teams vom Schiedsrichter aus dem Spiel entfernt, gilt das Team als disqualifiziert.
Begründung:	Unfaires Spielen muss bestraft werden.
Abhängigkeiten:	Fouls, Schiedsrichter
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA49**
Titel:	Spielende
Beschreibung:	Die Partie endet, wenn ein Sucher den Schnatz fängt oder ein Team disqualifiziert wird. Das Team mit den meisten Punkten gewinnt, sofern es nicht disqualifiziert ist.
Begründung:	Damit das Spiel endet.
Abhängigkeiten:	Schnatz, Sucher, Disqualifikation
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	**FA50**
Titel:	Spielfigur platzieren
Beschreibung:	Ein Spieler darf seine Spielfiguren in seiner Hälfte des Spilefeldes beliebig platzieren. Es darf jedoch nie mehr als eine Figur auf der selben Zelle platziert sein. Außerdem muss das Zentrum unbesetzt sein.
Begründung:	Zu Beginn des Spiels wählt jeder Spieler eine Aufstellung für sein Team.
Abhängigkeiten:	Zentrum, Spielbeginn
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA51
Titel:	Spielbeginn
Beschreibung:	Zu Beginn werden alle Spielfiguren von den Spielern auf dem Spielfeld platziert. Die Bälle, mit Ausnahme des Schnatzes, werden auf der Zentrumszelle platziert.
Begründung:	Nötige Aufstellung zu Spielbeginn.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

ID:	FA52
Titel:	Überlängenbehandlung
Beschreibung:	Maßnahmen, die ergriffen werden, falls eine Partie zu lange läuft. Zieht sich ein Spiel über mehr Runden hin, als in der Partie-Konfiguration über einen Höchstwert festgelegt wurden, ohne dass ein Sieger ermittelt wurde, so wird das Verhalten des Schnatz angepasst. Die Wahrscheinlichkeit, dass ein Sucher den Schnatz fängt, wird auf 100% gesetzt. Falls dann nach drei Runden das Spiel immer noch läuft, bewegt sich der Schnatz, ohne Suchern auszuweichen, in die Mitte des Spielfelds und verharrt dort. Wird er dort nach weiteren drei Runden immer noch nicht gefangen, so bewegt er sich in der nächsten Runde auf die nächste Zelle, auf der sich ein Sucher befindet, wodurch das Spiel automatisch beendet wird.
Begründung:	Dadurch wird sichergestellt, dass ein Spiel spannend bleibt und die Spieler nicht die Lust verlieren.
Abhängigkeiten:	Runde, Schnatz, Sucher
Priorität:	0
Akteure:	Spieler, KI, Gast, Client, KI-Client, Server

3.2 Funktionale Anforderungen: Allgemein

Bei den nachfolgenden Anforderungen handelt es sich um die funktionalen Anforderungen, die für alle Komponenten des Projektes relevant sind.

ID:	FA53
Titel:	Partie-Konfiguration
Beschreibung:	Die Partie-Konfiguration spezifiziert sämtliche Wahrscheinlichkeiten für zufällige Ereignisse im Spiel, sowie die maximale Rundenanzahl, bevor die Überlängenbehandlung eintritt.
Begründung:	Bietet dem Nutzer die Möglichkeit, eine Partie nach persönlichen Präferenzen zu gestalten.
Abhängigkeiten:	Konfigurator
Priorität:	+
Akteure:	Server, Konfigurator, Client, KI-Client, Nutzer

ID:	FA54
Titel:	Team-Konfiguration
Beschreibung:	Definiert alle Attribute eines Teams.
Begründung:	Nutzer sollen sich ihre Teams individuell zusammenstellen können.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Server, Konfigurator, Client, KI-Client, Nutzer

ID:	FA55
Titel:	Netzwerkschnittstelle
Beschreibung:	Die Clients und Server kommunizieren über eine Netzwerkschnittstelle. Die Clients kommunizieren ausschließlich mit dem Server und nicht untereinander.
Begründung:	Bei „Fantastic Feasts“ handelt sich es um ein Online Multiplayer Spiel. Es ist also notwendig, dass einzelne Komponenten miteinander Kommunizieren können.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Systemadministrator, Entwickler, Client, KI-Client, Server

ID:	FA56
Titel:	Log-Datei
Beschreibung:	Datei zum Speichern bestimmter Ereignisse. Diese Datei wird lokal auf dem Endgerät hinterlegt.
Begründung:	Um die während der Nutzung der Software aufgetretenen Aktionen im Nachhinein nachvollziehen zu können und daraus Informationen für Statistiken und Wartung zu ziehen.
Abhängigkeiten:	-
Priorität:	-
Akteure:	Client, KI-Client, Server

3.3 Funktionale Anforderungen: Server

Die folgenden funktionalen Anforderungen betreffen nur die Server Anwendung.

ID:	FA57
Titel:	Partie-Konfiguration laden
Beschreibung:	Der Server muss eine vorgefertigte Partie-Konfiguration laden können.
Begründung:	Die Partie-Konfiguration definiert maßgeblich den Spielverlauf.
Abhängigkeiten:	Partie-Konfiguration
Priorität:	0
Akteure:	Server, Systemadministrator

ID:	FA58
Titel:	Zufallsgenerator
Beschreibung:	Pseudozufallszahlengenerator, der bestimmt, ob ein Ereignis eintritt.
Begründung:	Viele Ereignisse im Spiel sind zufällig.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Server

ID:	FA59
Titel:	Spielmechanik
Beschreibung:	Der Server interpretiert die Nachrichten der Clients nach den oben definierten Spielregeln und versendet wiederum Updates des Spielzustandes.
Begründung:	Der Server ist die Zentrale Systemkomponente, die die Partie verwaltet.
Abhängigkeiten:	Oben definierten Spielregeln.
Priorität:	++
Akteure:	Server

3.4 Funktionale Anforderungen: Client

Die folgenden funktionalen Anforderungen betreffen nur die Client Anwendung.

ID:	FA60
Titel:	Hauptmenü [Ansicht]
Beschreibung:	Erste grafische Oberfläche die dem Nutzer angezeigt wird, wenn die Anwendung gestartet wurde.
Begründung:	Das Hauptmenü soll den Zentralen Punkt darstellen, von dem aus alle Funktionen der Software zu erreichen sind. Es soll also unter anderem ein Spiel gestartet werden, die Hilfe aufgerufen werden, die Einstellungen der Anwendung angepasst und eventuell vorhandene Statistiken aufgerufen werden können.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Nutzer, Client

ID:	FA61
Titel:	Spiel beitreten [Ansicht]
Beschreibung:	Grafische Oberfläche, die erscheint, wenn man versucht, sich mit einem Server zu verbinden, um eine neue Partie zu starten. Dabei soll man außerdem die Möglichkeit haben, seine Team-Konfiguration anzugeben, die man für die Partie verwenden möchte.
Begründung:	Der Nutzer muss die Möglichkeit haben, sich komfortabel mit einem Server zu verbinden.
Abhängigkeiten:	Hauptmenü [Ansicht]
Priorität:	+
Akteure:	Nutzer, Client

ID:	FA62
Titel:	Spielende [Ansicht]
Beschreibung:	Grafische Oberfläche, die die Spieler sehen, nachdem eine Partie zu Ende ist. Der Nutzer sollte hier auch die Möglichkeit haben die Anwendung zu verlassen oder wieder ins Hauptmenü zurückzukehren. Optional ist hier auch Platz für etwaige Statistiken über den Spielverlauf.
Begründung:	Nach dem Ende einer Partie muss dem Nutzer mitgeteilt werden ob er gewonnen hat oder nicht und wie es von da an weiter geht.
Abhängigkeiten:	Spiel [Ansicht]
Priorität:	0
Akteure:	Nutzer, Client

ID:	FA63
Titel:	Team-Konfiguration importieren [Ansicht]
Beschreibung:	Grafische Oberfläche zum Importieren einer Team-Konfiguration für ein Spiel.
Begründung:	Es muss für den Benutzer einen einfachen Weg geben, eine Team Konfiguration im Dateisystem zu suchen und an die Anwendung zu übergeben.
Abhängigkeiten:	Hauptmenü [Ansicht]
Priorität:	0
Akteure:	Spieler, Client

ID:	FA64
Titel:	Spiel [Ansicht]
Beschreibung:	Grafische Oberfläche, die das Spielgeschehen visualisiert.
Begründung:	Um dem Nutzer das Spiel zu visualisieren.
Abhängigkeiten:	Hauptmenü [Ansicht]
Priorität:	++
Akteure:	Nutzer, Client

ID:	FA65
Titel:	Hilfe [Ansicht]
Beschreibung:	Grafische Oberfläche, in der zum einen das Spielprinzip erklärt wird und zum anderen gezeigt wird, wie genau man die Client-Software bedient.
Begründung:	Um unerfahren Benutzer die Bedienung der Software zu erleichtern.
Abhängigkeiten:	Hauptmenü [Ansicht], Spiel [Ansicht], Beobachter [Ansicht]
Priorität:	0
Akteure:	Nutzer, Client

ID:	FA66
Titel:	Beobachter [Ansicht]
Beschreibung:	Wie die Spiel-Ansicht, nur ohne die Möglichkeit, in das Spiel einzugreifen.
Begründung:	Damit Gäste eine Partie mitverfolgen können.
Abhängigkeiten:	Hauptmenü [Ansicht]
Priorität:	0
Akteure:	Gast, Client

ID:	FA67
Titel:	Eingabeverarbeitung
Beschreibung:	Diese Einheit ist für die Verarbeitung von Benutzereingaben verantwortlich.
Begründung:	Jede Benutzereingabe muss ausgewertet werden. Valide Eingaben werden zum Steuern der Anwendung genutzt.
Abhängigkeiten:	Spiellogik
Priorität:	++
Akteure:	Spieler, KI, Client, KI-Client

ID:	FA68
Titel:	Hotkeys
Beschreibung:	Oft benötigte Funktionen werden auf bestimmte (besondere) Tasten (-Kombinationen) abgebildet.
Begründung:	Hotkeys sind optionale Features, die im Lastenheft aufgeführt sind und zu einer einfacheren Spielsteuerung und höherem Spielkomfort beitragen können.
Abhängigkeiten:	Spiel [Ansicht], Hilfemenü [Ansicht]
Priorität:	-
Akteure:	Nutzer, Client

ID:	FA69
Titel:	Pausieren
Beschreibung:	Das aktuelle Spiel pausieren.
Begründung:	Pausieren ist ein optionales Feature, das im Lastenheft aufgeführt ist und einem menschliche Spieler im Client zur Verfügung stehen sollte, um den Spielkomfort zu erhöhen.
Abhängigkeiten:	Spiel [Ansicht], Hotkeys
Priorität:	-
Akteure:	Spieler, Client

3.5 Funktionale Anforderungen: Konfigurator

Die folgenden funktionalen Anforderungen betreffen nur die Konfigurator-Anwendung.

ID:	FA70
Titel:	Team- / Partie-Konfiguration visualisieren
Beschreibung:	Der Konfigurator kann eine geöffnete Team- / Partie-Konfiguration grafisch darstellen und anzeigen, damit ein Nutzer diese bearbeiten kann.
Begründung:	Konfiguration eines Teams / Einer Partie.
Abhängigkeiten:	Quidditchteam-Konfiguration, Partie-Konfiguration
Priorität:	-
Akteure:	Nutzer, Konfigurator

ID:	FA71
Titel:	Team- / Partie-Konfiguration erstellen und speichern
Beschreibung:	Der Konfigurator kann eine Team- / Partie-Konfiguration erstellen und speichern.
Begründung:	Konfiguration eines Teams / Einer Partie.
Abhängigkeiten:	Quidditchteam-Konfiguration, Partie-Konfiguration
Priorität:	-
Akteure:	Nutzer, Konfigurator

ID:	FA72
Titel:	Team- / Partie-Konfiguration bearbeiten
Beschreibung:	Der Konfigurator kann eine bestehende Team- / Partie-Konfiguration öffnen und bearbeiten.
Begründung:	Konfiguration eines Teams / Einer Partie.
Abhängigkeiten:	Quidditchteam-Konfiguration, Partie-Konfiguration
Priorität:	-
Akteure:	Nutzer, Konfigurator

3.6 Funktionale Anforderungen: KI-Client

Die folgenden funktionalen Anforderungen betreffen nur die KI-Client-Anwendung.

ID:	FA73
Titel:	Schwierigkeitsgrad einstellen
Beschreibung:	In der KI-Clientanwendung hat ein Nutzer die Möglichkeit, die Spielstärke der KI einzustellen.
Begründung:	Um dem Nutzer unterschiedlich starke KI-Gegner zur Verfügung zu stellen.
Abhängigkeiten:	-
Priorität:	-
Akteure:	Nutzer, KI

ID:	FA74
Titel:	Serverkonfiguration einstellen
Beschreibung:	In der KI-Clientanwendung kann ein Nutzer einstellen, mit welchem Server sich der KI-Client verbinden soll.
Begründung:	Damit sich der KI-Client mit dem gewünschten Server verbinden kann.
Abhängigkeiten:	-
Priorität:	-
Akteure:	Nutzer, KI

ID:	FA75
Titel:	Team-Konfiguration laden
Beschreibung:	Der Nutzer kann der KI eine gewünschte Team-Konfiguration zuweisen, indem er eine bereits erstellte Konfiguration lädt.
Begründung:	Um der KI ein gewünschte Team-Konfiguration zuzuweisen.
Abhängigkeiten:	Team-Konfiguration
Priorität:	-
Akteure:	Nutzer, KI

Teil III
Softwarespezifikation

1 Schnittstellenarten, Dialoge und Dialogstruktur

1.1 Client

Der Client stellt die Anwendung dar, mit der ein Nutzer aktiv als Spieler oder auch passiv als Gast an einer Partie teilnehmen kann.

1.1.1 Schnittstellenarten

Als Benutzerschnittstelle wird eine grafische Benutzeroberfläche verwendet.
Begründung: Um das Spiele so intuitiv wie möglich zu gestalten, ist es sinnvoll, dem Nutzer alle für das Spielgeschehen relevanten Informationen und Aktionen grafisch in einer

GUI darzustellen. Hinzu kommt, dass neben dem eigentlichen Spiel auch die zugehörigen Funktionen, wie zum Beispiel das Verbinden mit einem Server, benutzerfreundlich und leicht zu bedienen sein sollte. Dies lässt sich am leichtesten durch eine grafische Benutzeroberfläche bewerkstelligen.

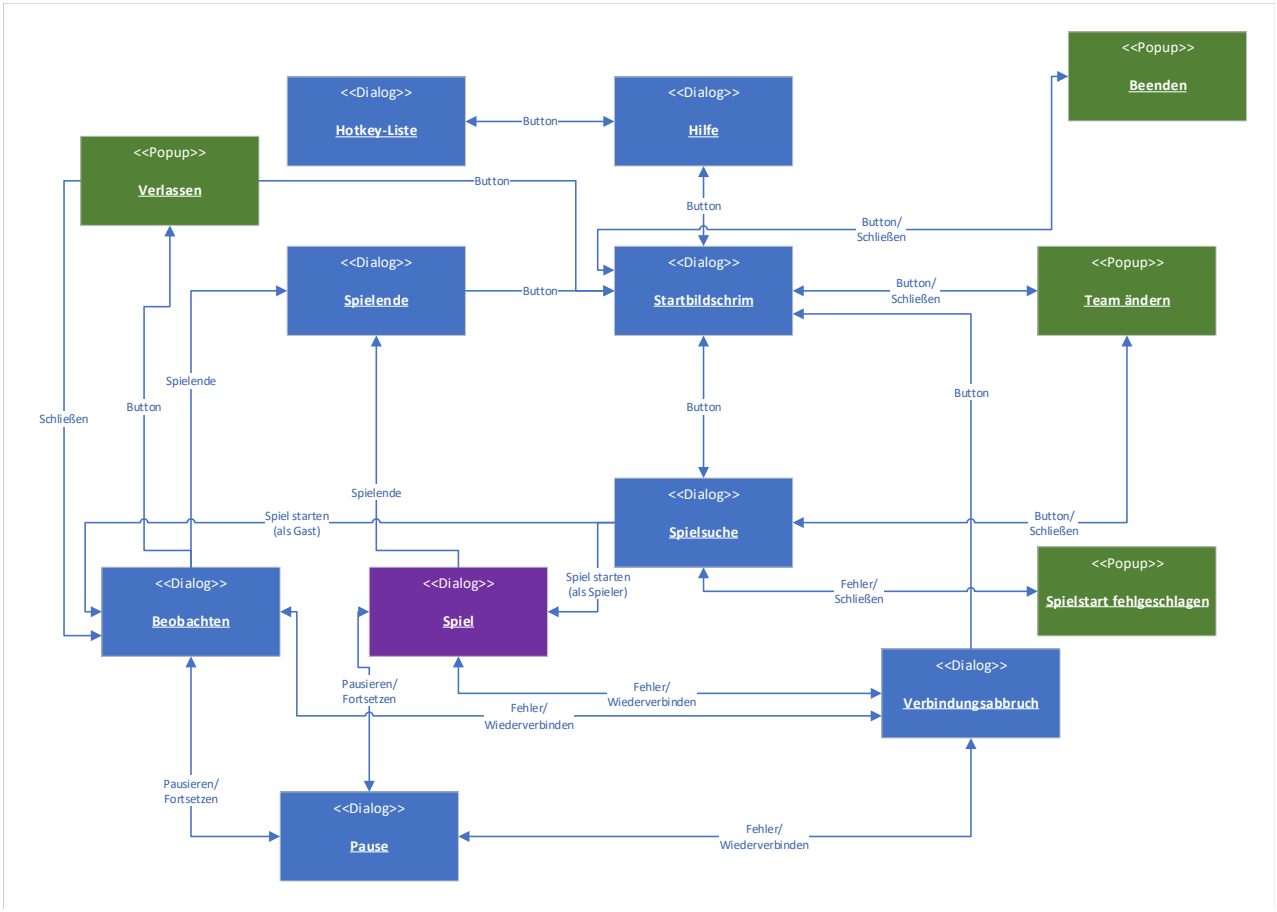
1.1.2 Dialoge

Im Folgenden sind alle Dialoge, die während der Nutzung des Clients benötigt werden, den zugehörigen Anwendungsfällen zugeordnet.

Name	**Typ**	**Abgedeckte Anforderungen**	
Startbildschirm	Dialog	FA60	Hauptmenü [Ansicht]
Hilfe	Dialog	FA65	Hilfe [Ansicht]
		QA18	Benutzerfreundlichkeit
Hotkey-Liste	Dialog	FA65	Hilfe [Ansicht]
		FA68	Hotkeys
Team ändern	Popup	FA63	Team-Konfiguration importieren [Ansicht]
		FA15	Teams
		FA54	Quidditchtem-Konfiguration
		FA61	Spiel beitreten [Ansicht]
Beenden	Popup		
Spielstart fehlgeschlagen	Popup	QA16	Zuverlässigkeit
Verlassen	Popup		
Spielsuche	Dialog	FA61	Spiel beitreten [Ansicht]
		FA55	Netzwerkschnittstelle
Spielende	Dialog	FA62	Spiel Ende [Ansicht]
		FA49	Spielende
Beobachten	Dialog	FA66	Beobachter [Ansicht]
Verbindungsabbruch	Dialog	QA16	Zuverlässigkeit
		QA17	Robustheit
Pause	Dialog	FA69	Pausieren

Spiel	Dialog	FA64	Spiel[Ansicht]
		FA1	Spielfeld
		FA2	Mittelkreis
		FA3	Mittelzelle
		FA4	Hüterzone
		FA5	Zelle
		FA6	Torring
		FA8	Punkte erzielen
		FA10	Ball
		FA11	Quaffel [Ball]
		FA12	Klatscher [Ball]
		FA13	Goldener Schnatz [Ball]
		FA14	Besen
		FA16	Spielfiguren
		FA17	Jäger
		FA18	Treiber
		FA19	Hüter
		FA20	Sucher
		FA31	Fans
		FA32	Elfen [Fantyp]
		FA33	Kobolde [Fantyp]
		FA34	Trolle [Fantyp]
		FA35	Niffler [Fantyp]
		FA36	Schiedsrichter
		FA44	Runde
		FA46	Spielerphase

1.1.3 Dialogstrukturdiagramme



1.2 Server

Diese Komponente hostet Spiele und enthält die Spiellogik. Zu Beginn wird der Server einmalig vom Systemadministrator gestartet und steht anschließend den Spielern von *Fantastic Feasts* zur Verfügung.

1.2.1 Schnittstellenarten

Als Benutzerschnittstelle wird ein CLI verwendet. **Begründung:** Der Nutzer kommt mit dieser Komponente über keine Benutzerschnittstelle in Berührung. Deswegen spielen Look and Feel keine Rolle. Zusätzlich wird der Server nur einmal mit wenigen Parametern gestartet, benötigt zur Laufzeit dann keine weiteren Eingaben und muss auch keinerlei graphische Ausgabe zur Verfügung stellen, weswegen das CLI ausreicht.

1.2.2 Dialoge

Im Folgenden werden die Anforderungen an den Server einem CLI-Dialog zugeordnet.

Name	**Typ**	**Abgedeckte Anwendungsfälle**	
ServerInit	CLI Befehl mit Params	FA57	Partie-Konfiguration laden
ServerRunning	Response	FA58, FA59	Spielmechanik.
ServerError	Response	allgemein	Feedback

1.2.3 Dialogstruktur

Die Dialogstruktur des Servers lässt sich wie folgt beschreiben: **ServerInit:** Der Server lässt sich in der Konsole mit dem Namen der Server-Anwendung, dem Namen einer gültigen Partie-Konfiguration und einem Parameter aufrufen. Der Parameter ist die Port-Nummer, über die der Server erreichbar ist. Darauf gibt es zwei mögliches Antworten. **ServerRunning:** War die Initialisierung erfolgreich, antwortet der Server mit einer entsprechenden Nachricht. **ServerError:** Im Falle eines Fehlers bei der Initialisierung wird mit einer Fehlernachricht geantwortet.

1.2.4 Zulässige Optionen

Flag	Erklärung
-p	Legt die Portnummer fest.
--help	Zeigt eine Liste möglicher Optionen an.

1.3 Team- und Partiekonfigurator

Diese Komponente enthält einen Konfigurator, mit dem sich sowohl Quidditch-Teams, als auch Partie-Konfigurationen erstellen und bearbeiten lassen.

1.3.1 Schnittstellenarten

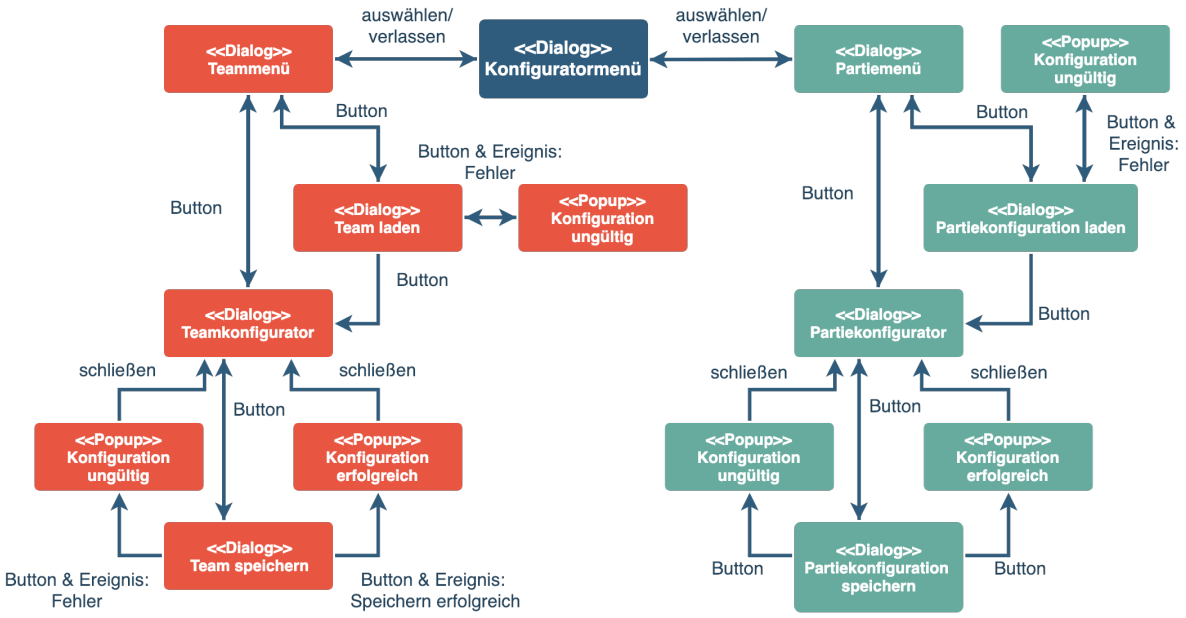
Als Benutzerschnittstelle wird, wie im Lastenheft vorgeschrieben, eine GUI verwendet. **Begründung:** Der Nutzer möchte alle Informationen zu einer gegebenen Konfiguration übersichtlich dargestellt bekommen und anhand dieser Darstellung direkt Änderungen vornehmen können. Hierfür ist eine GUI die intuitivste und sinnvollste Variante, da eine grafische Darstellung eine übersichtliche Visualisierung erlaubt und eine Änderung direkt anhand dieser Visualisierung möglich ist.

1.3.2 Dialoge

Im Folgenden werden die bereits formulierten Anforderungen und Anwendungsfälle der Komponente den entsprechenden Dialogen zugeordnet.

Name	**Typ**	**Abgedeckte Anwendungsfälle**	
Konfiguratormenü	Dialog	QA18	implizit aus Benutzerfreundlichkeit
Teammenü	Dialog	QA18	implizit aus Benutzerfreundlichkeit
Team laden	Dialog	FA72	Konfiguration öffnen
Teamkonfigurator	Dialog	FA70-72	Konfiguration erstellen/bearbeiten
Team speichern	Dialog	FA72	Konfiguration speichern
Partiemenü	Dialog	–	implizit, da Strukturierung erforderlich
Partiekonfiguration laden	Dialog	FA72	Konfiguration öffnen
Partiekonfigurator	Dialog	FA70-72	Konfiguration erstellen/bearbeiten
Partiefonfiguration speichern	Dialog	FA72	Konfiguration speichern
Konfiguration erfolgreich	Popup	QA17-18	Benutzerfreundlichkeit und Robustheit
Konfiguration ungültig	Popup	QA17-18	Benutzerfreundlichkeit und Robustheit

1.3.3 Dialogstrukturdiagramme



1.4 KI-Client

Diese Komponente simuliert einen Menschlichen Gegner, der sich wie ein normaler Client bei einem Spielserver anmeldet und dann autonom seine Spielentscheidungen trifft.

1.4.1 Schnittstellenarten

Es besteht kein Grund, für diese Komponente eine grafische Oberfläche bereitzustellen, da die Anwendung zur Laufzeit keine Eingabe von einem menschlichen Benutzer erwartet und eine Partie mittels des Clients verfolgt werden kann.
Für eine Kommandozeilenanwendung ist es einfacher, Plattformunabhängigkeit sicherzustellen. Außerdem wird es damit problemlos möglich, den KI-Client aus einem anderen Programm zu starten. Beispielsweise kann dem Client eine Funktion hinzugefügt werden, gegen die KI zu spielen, ohne dass der Benutzer den KI-Client extern starten muss.

1.4.2 Dialoge

Im Folgenden werden die Anforderungen an den KI-Client einem CLI-Dialog zugeordnet.

Name	Typ	Abgedeckte Anwendungsfälle	
Init	CLI Befehl mit Params	FA73 - FA75	Schwierigkeit, Server- und Team-Konfiguration einstellen.
InitFailure	Response	FA73 - FA75	Feedback.
WaitingForGame	Response	FA55	Netzwerkschnittstelle, allgemeine Kommunikation.
Playing	Response	allgemein	Feedback.
AttemptingReconnect	Response	FA55	Netzwerkschnittstelle, allgemeine Kommunikation.
ConnectionLost	Response	FA55	Netzwerkschnittstelle, allgemeine Kommunikation.

1.4.3 Dialogstruktur

Init: Der KI-Client wird über die Kommandozeile gestartet. Serverkonfiguration, Team-Konfiguration, die maximale Anzahl an Reconnect-Versuchen und der Schwierigkeitsgrad werden beim Start der Anwendung mittels Kommandozeilenparametern gehandhabt. Der Server, mit dem sich der KI-Client verbinden soll wird als Argument übergeben. Die Team-Konfiguration und der Schwierigkeitsgrad können mittels Optionen verändert werden und nehmen ansonsten einen Standardwert an.
InitFailure: Wird eine ungültige Option angegeben, ist der Server nicht erreichbar oder wurde eine ungültige Team-Konfigurationsdatei geladen, so erscheint eine Fehlermeldung mit entsprechenden Hinweisen.
WaitingForGame: War die Initialisierung erfolgreich und der KI-Client konnte sich mit dem Server verbinden, so erscheint eine Nachricht, die darauf hinweist, dass noch auf den Beginn der Partie gewartet wird.
Playing: Während einer laufenden Partie wird ein Hinweis angezeigt.
AttemptingReconnect: Bei einem Verbindungsabbruch zeigt der KI-Client eine Meldung an und versucht automatisch, die Verbindung wiederherzustellen.
ConnectionLost: Konnte sich der KI-Client nicht innerhalb der angegebenen Anzahl von Versuchen erneut mit dem Server verbinden, so erscheint eine Fehlermeldung und das Programm beendet sich.
Der KI-Client beendet sich außerdem nach Abschluss einer Partie durch ein reguläres Spielende.

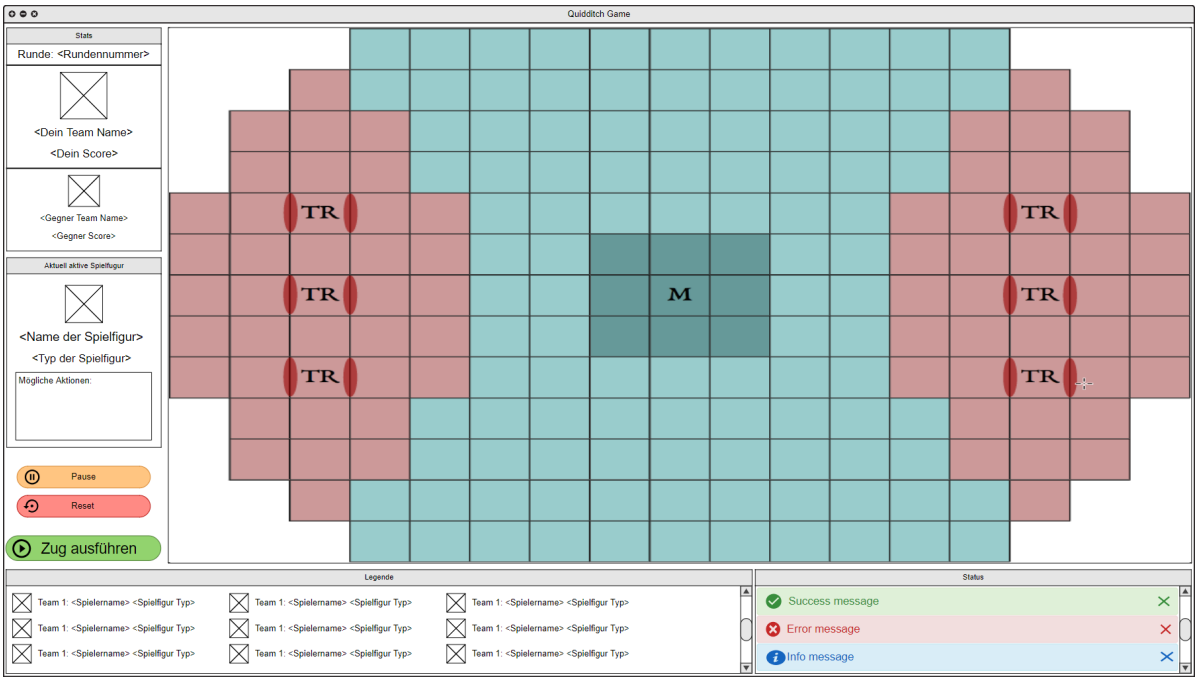
1.4.4 Zulässige Optionen:

Flag	Erklärung
-s	Legt den Schwierigkeitsgrad fest. Akzeptiert eine ganze Zahl zwischen 0 und 2, wobei 0 für einfach, 1 für mittelschwer und 2 für schwer steht. Bei einer ungültigen Eingabe wird eine Fehlermeldung ausgegeben.
-t	Legt die Team-Konfiguration fest. Akzeptiert einen String als Pfad zu einer JSON-Datei. Existiert der angegebene Pfad nicht oder ist die Datei keine gültige Konfigurationsdatei, wird eine entsprechende Fehlermeldung ausgegeben.
-r	Reconnects: Spezifiziert, wie oft der KI-Client versucht, die Verbindung nach einem Verbindungsabbruch wiederherzustellen. Standardwert ist 5.
--help	Zeigt eine Liste möglicher Optionen an.

2 Grafische Gestaltung und Nutzungskonzept

2.1 Client

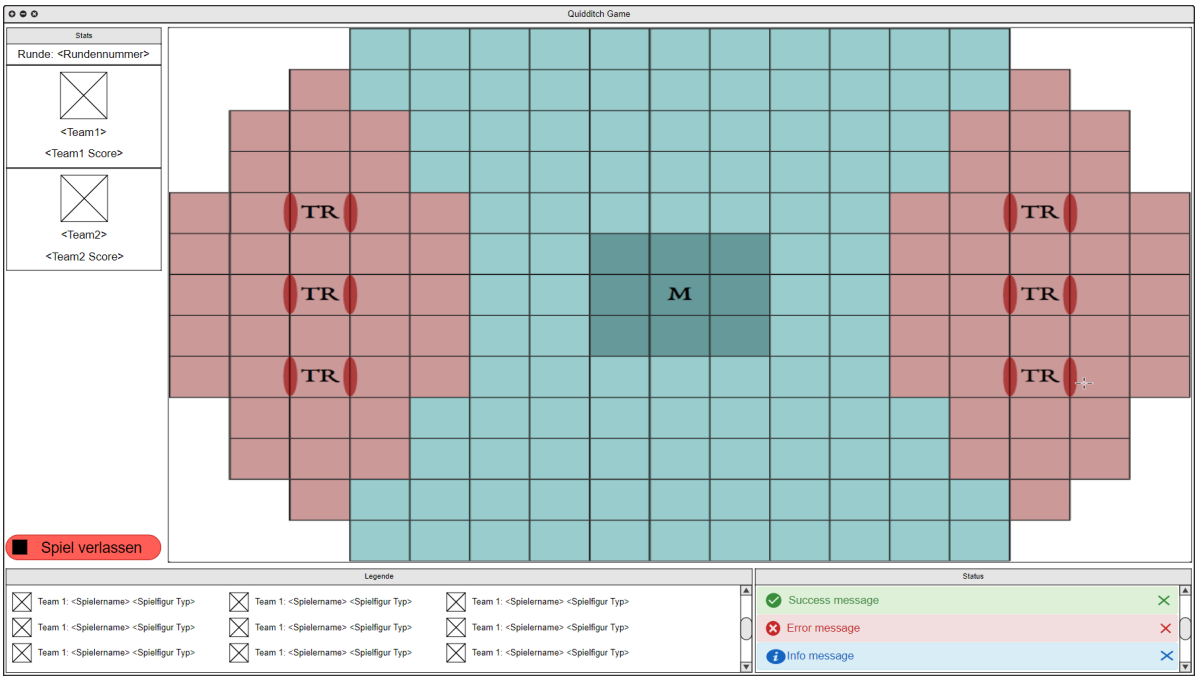
2.1.1 Spielansicht für einen Spieler



In der Spielansicht kann ein Spieler das aktuelle Spielgeschehen verfolgen und seine Züge ausführen. Dabei ist die Oberfläche in mehrere Teile unterteilt.
Im *Stats* Bereich werden grundlegende Informationen über die aktuelle Partie und die beiden Teams dargestellt. Das Team, das an oberster Position steht, ist momentan am Zug. Im darunter liegende Bereich wird die *aktuell ausgewählte Spielfigur* hervorgehoben. Zusätzlich wir aufgeführt, um welchen Typen von Spielfigur es sich handelt und welche grundlegenden Züge diese Figur ausführen kann. In den Fan-Phasen werden hier vergleichbare Informationen zu den Fans angezeigt. Darunter befinden sich drei Buttons mit dem entweder das Spiel *pausiert* werden, alle Veränderungen, die man in der aktuellen Runde getätigt hat *zurücksetzen* oder seinen Zug endgültig *ausführen* kann. Am unteren Rand der Oberfläche ist eine *Legende* mit einer Übersicht über alle Spielfiguren des eigenen und des gegnerischen Teams zu sehen. Daneben befindet sich ein Feld, in dem Statusmeldungen angezeigt werden können. Beispiele für solche Statusmeldungen sind eine Benachrichtigung über ein Foul oder über das erfolgreiche Ausführen eines Spielzuges. Den größten Teil der Oberfläche nimmt das eigentliche Spielfeld ein. Hier

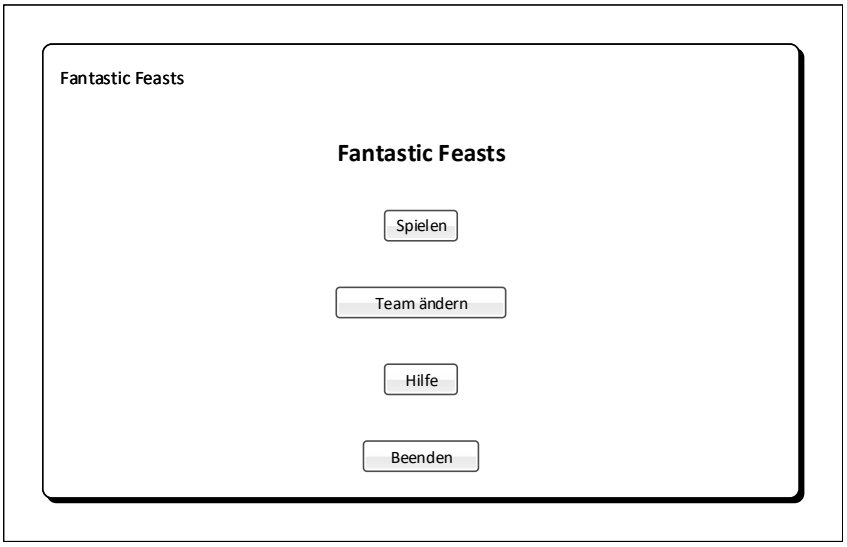
werden alle Spielfiguren in den Feldern angezeigt, auf denen sie sich gerade befinden. Ist man am Zug, so werden alle Züge, die von der aktuell ausgewählten Spielfigur ausgeführt werden können, farblich hervor gehoben. Der Spieler kann diese Aktionen dann durch Klicken auf die passende Zelle ausführen. Ist man nicht am Zug, so werden alle Eingabemöglichkeiten, mit Ausnahme des *Pause* Buttons, deaktiviert.

2.1.2 Spielansicht für einen Beobachter



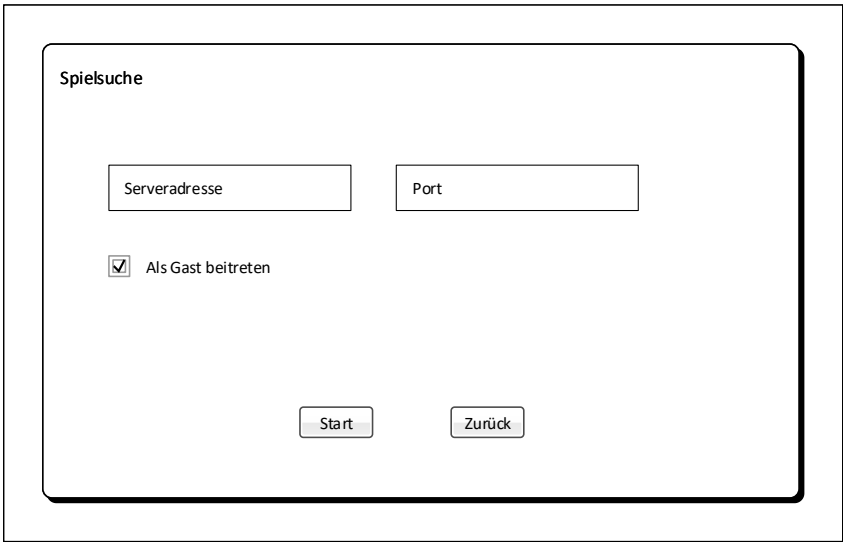
In der Spielansicht kann ein Beobachter eine Partie zwischen zwei anderen Gegnern passiv verfolgen. Die Oberfläche ist im wesentlichen gleich aufgebaut wie die Oberfläche, die die Spieler sehen. Jedoch sind beim Beobachter alle Felder, die zur Eingabe dienen, deaktiviert. Die einzige Interaktion, die durch einen Button ermöglicht wird, ist das vorzeitige *Verlassen* einer Partie.

2.1.3 Hauptmenü



Dieser Dialog erscheint nach dem Start der Anwendung. Der „Spielen“-Button öffnet den Spielsuche-Dialog. Der „Team ändern“-Button öffnet das „Team ändern“-Popup. Der „Hilfe“-Button öffnet den Hilfe-Dialog. Der „Beenden“-Button öffnet ein Bestätigungs-Popup und beendet bei positiver Antwort die Anwendung.

2.1.4 Spielsuche

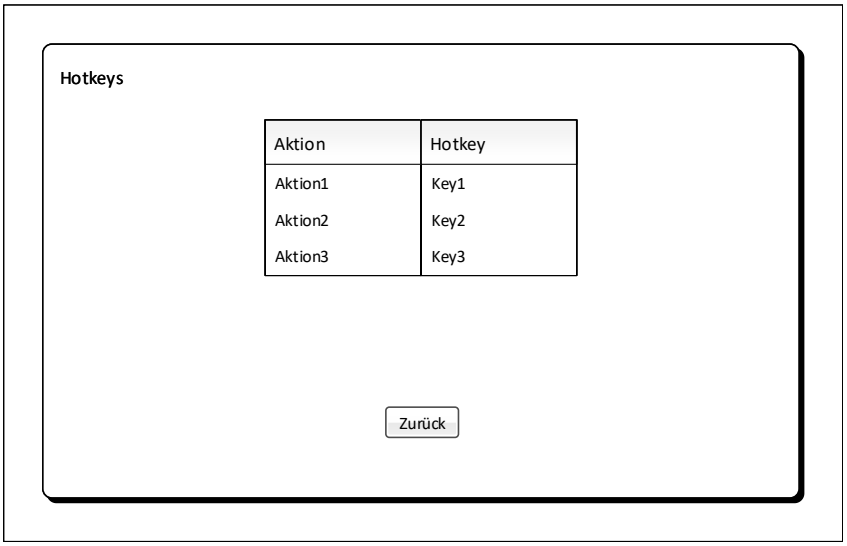


Der Benutzer gibt zuerst die Adresse und den Port des Spielservers ein, mit dem er sich verbinden möchte. Wenn er die Partie beobachten will, wählt er „Als Gast beitreten“ aus. Drückt er anschließend auf den „Start“-Button, versucht sich der Client mit dem angegebene Server zu verbinden. Mit dem „Zurück“-Button gelang man zurück auf den Startbildschirm.

2.1.5 Hilfe

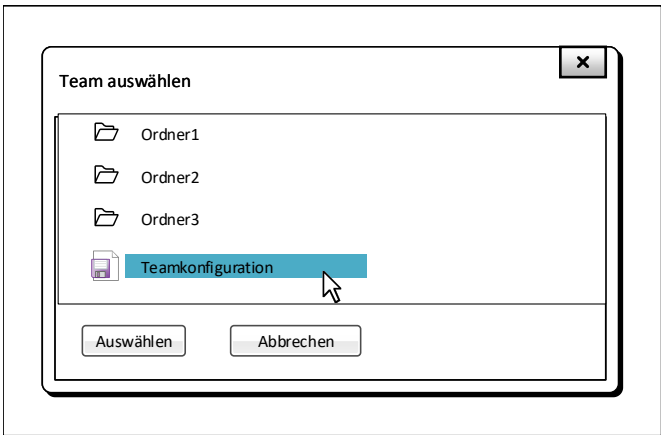
In einem Textfeld, gegebenenfalls mit Scrollbar, wird ein Hilfetext angezeigt. Der „Zurück“-Button öffnet den Startbildschirm, der „Hotkey“-Button den Hotkey-Dialog.

2.1.6 Hotkeys



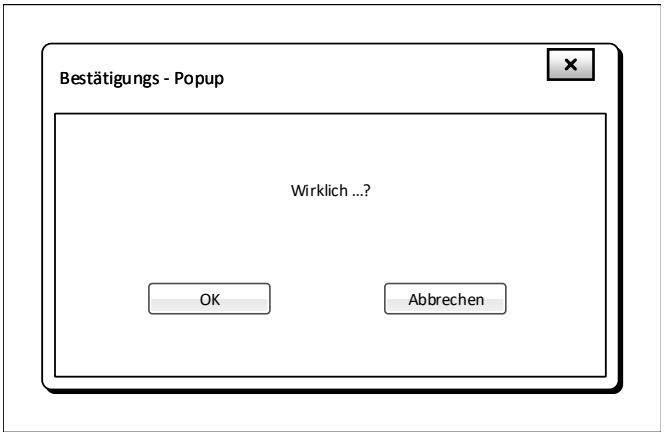
Hier werden alle verfügbaren Hotkeys in Tabellenform aufgelistet. Der „Zurück“-Button öffnet den Hilfe-Dialog.

2.1.7 Team ändern



In diesem Popup kann der Benutzer durch sein Dateisystem navigieren und eine JSON-Datei auswählen. Hat er eine gültige Datei ausgewählt und betätigt den „Auswählen“-Button, wird das Popup geschlossen und die Team-Konfiguration für das Spiel verwendet. Der „Abbrechen“-Button schließt das Popup und es werden keine Änderungen vorgenommen.

2.1.8 Bestätigungsaufforderung

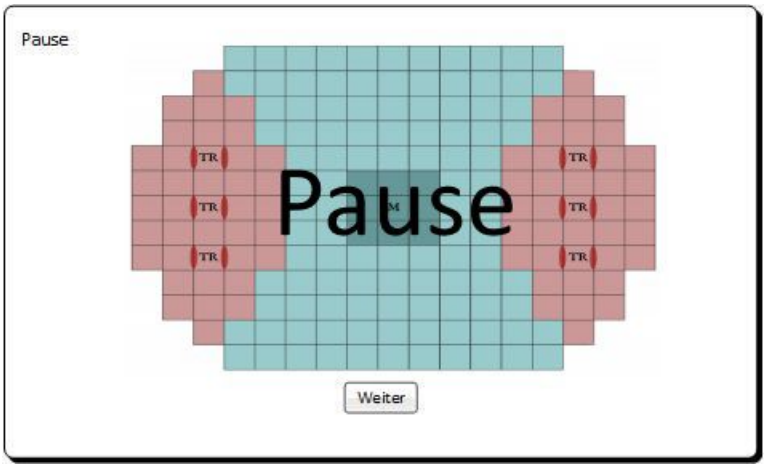


Dieser Aufbau wird für die Popups „Beenden“ und „Verlassen“ verwendet. Der „Abbrechen“-Button schließt das Popup und der Benutzer gelangt zurück in den Dialog, in dem er vor Öffnen des Popups war. Der „OK“-Button führt dazu, dass eine Aktion ausgeführt wird.

2.1.9 Fehler

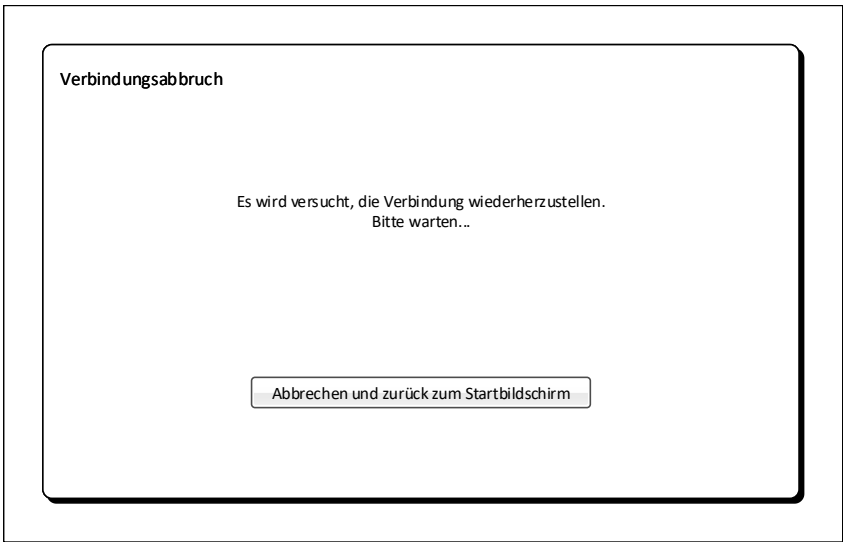
Dieser Aufbau wird für das Popup „Spielstart fehlgeschlagen“ verwendet. Der angezeigt Text richtet sich nach dem aufgetretenen Fehler. Der „OK“-Button schließt das Popup.

2.1.10 Pause



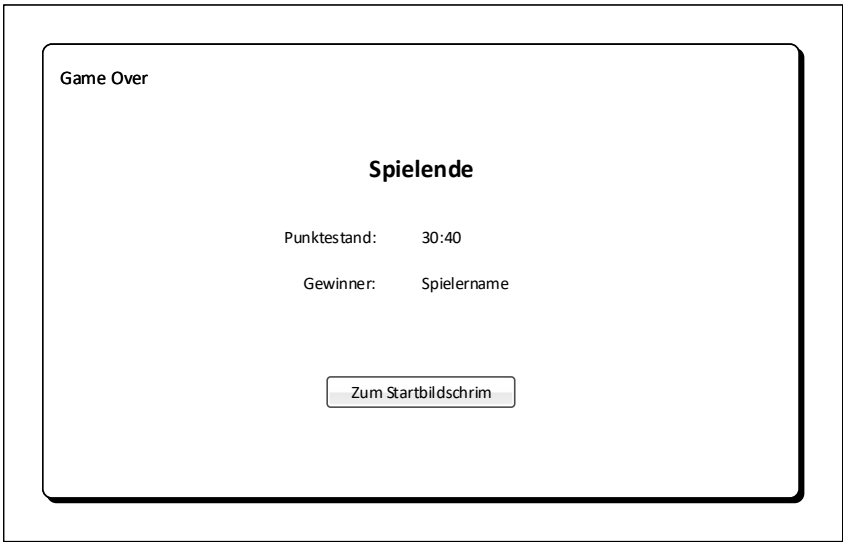
Wird angezeigt, wenn das Spiel von einem Spieler pausiert wird. Der „Weiter“-Button setzt die Partie fort und kann nicht von einem Gast betätigt werden.

2.1.11 Verbindungsabbruch



Im Falle eines Verbindungsabbruchs zwischen Client und Server wird dieser Dialog angezeigt. Wird die Verbindung wiederhergestellt, gelangt der Benutzer automatisch wieder zurück in den vorherigen Dialog. Alternativ kann er durch betätigen des Buttons zum Startbildschirm gelangen. Ist er ein Spieler, kann er die Partie nicht weiterführen und sein Gegner gewinnt nach Ablauf einer Zeitdauer die Partie.

2.1.12 Spielende



Bei Spielende wird dieser Dialog geöffnet. Hier werden der Punktestand bei Spielende und der Name des Gewinners angezeigt. Der Button öffnet den Startbildschirm-Dialog.

2.2 Server

2.2.1 ServerInit

```
$ server -p1230 standard_partie.json
... pending
```

Das Beispiel stellt einen Aufruf des Servers mit entsprechenden Argumenten dar. Mit dem Parameter wird eine Port-Nummer angegeben. Das zweite Argument gibt den Pfad zu einer gültigen Partie-Konfigurationsdatei an.

2.2.2 ServerRunning

```
Fantastic Feasts server is running on Port 1230 ...
```

Im Falle einer erfolgreichen Initialisierung wird dem Systemadministrator diese Nachricht mitgeteilt. Es können hier auch weitere Informationen über den initialisierten Server ausgegeben werden.

2.2.3 ServerError

```
Error: Initialization failed
```

Im Falle eines Fehlers bei der Initialisierung wird der Systemadministrator mit dieser Fehlermeldung benachrichtigt.

2.3 Team- und Partiekonfiguration

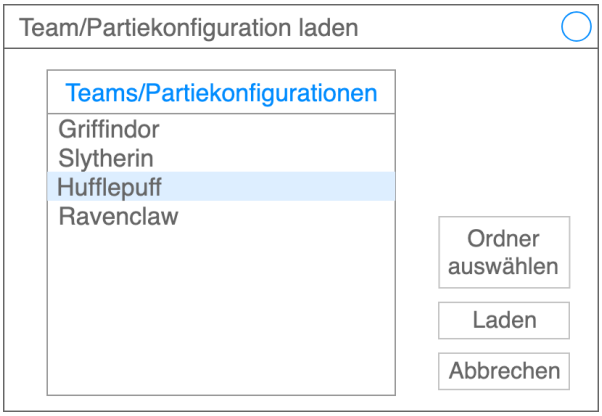
2.3.1 Konfiguratormenü

Über das Konfiguratormenü kann eine Auswahl zwischen dem Teammenü und dem Partiemenü über die entsprechenden Buttons getroffen werden. Diese führen zu den Menüs der jeweiligen Konfiguratoren. Über den Button *Verlassen* kann der Konfigurator verlassen werden.

2.3.2 Teammenü

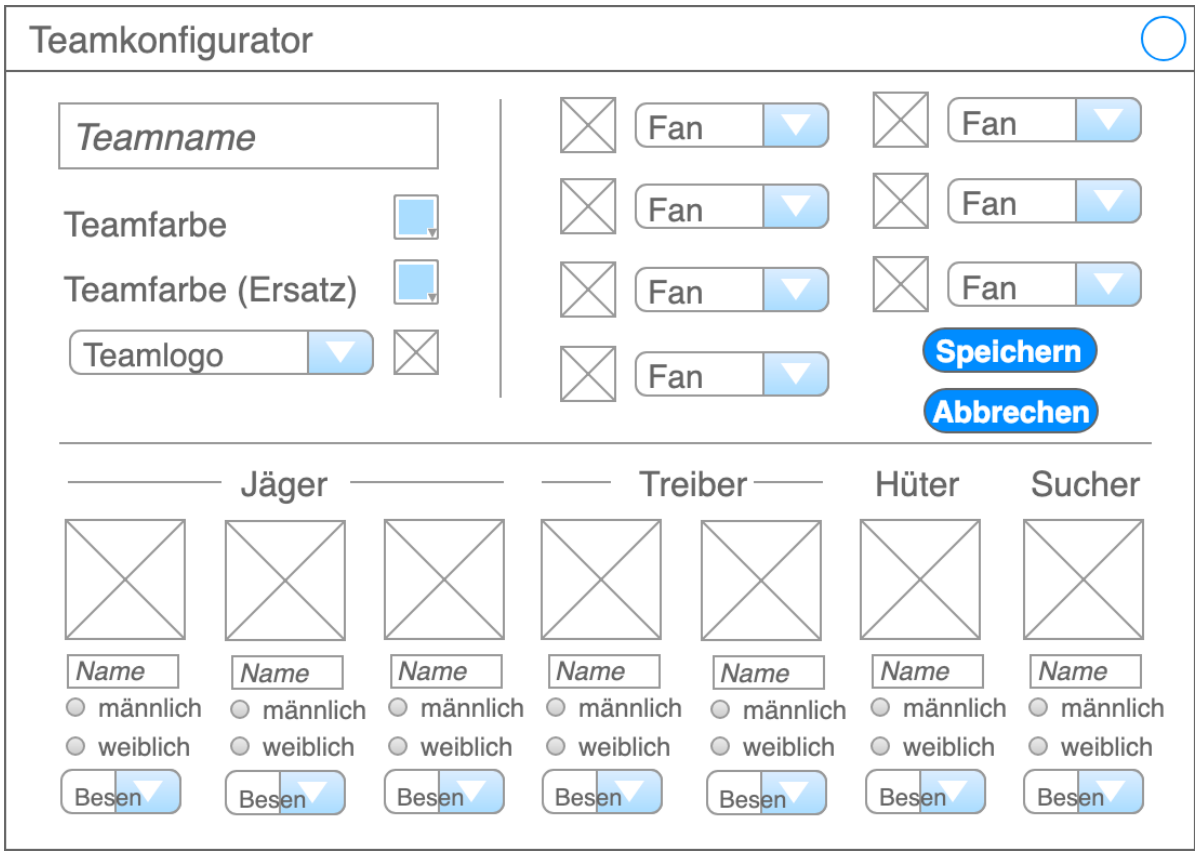
Über das Teammenü kann eine Auswahl zwischen dem Laden und dem Erstellen einer Teamkonfiguration getroffen werden. Das erfolgt über die entsprechenden Buttons. Über den Button *Verlassen* kann das Teammenü verlassen werden.

2.3.3 Team bzw. Partiekonfiguration laden



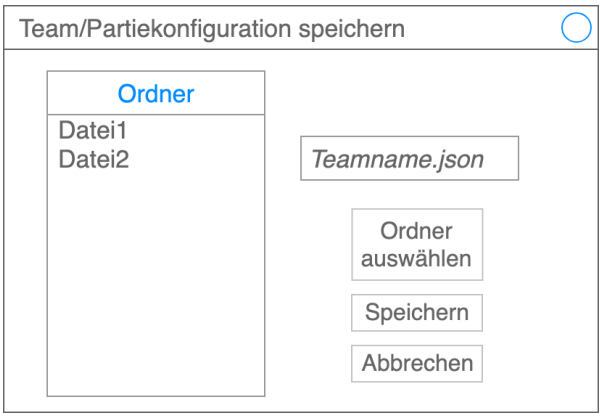
Über diesen Dialog kann eine bereits vorhanden Konfigurationsdatei ausgewählt und im Konfigurator geladen werden. Das erfolgt über ein Dateiauswahlelement und die entsprechenden Buttons. Über den Button *Abbrechen* gelangt man zurück zu den entsprechenden Menüs. Da das Laden einer Teamkonfiguration nahezu identisch zum Laden einer Partiekonfiguration ist, wurde diese beiden Fälle in einem zusammen gefasst. Sollte eine zu ladende Konfigurationsdatei ungültig sein, öffnet sich das Popup *Konfiguration ungültig*.

2.3.4 Teamkonfigurator



Im Teamkonfigurator können alle Parameter eines Teams eingestellt werden. Team- und Spielernamen lassen sich durch ein Textfeld bearbeiten. Teamfarben sind über eine Farbauswahl einstellbar. Das Teamlogo lässt sich aus einer Liste vorhandener Logos auswählen. Die Kästen mit den Kreuzen dienen als Platzhalter für Icons, die die verschiedenen Fans und Spieler darstellen und somit unterscheidbar machen. Fans sowie Besen der Spieler sind über eine Dropdown-Auswahl einstellbar. Das Geschlecht der Spieler lässt sich über Radio-Buttons einstellen. Bei jeder Änderung werden die entsprechenden Bedingungen für eine gültige Konfiguration geprüft und der Nutzer erhält visuelles Feedback (z. B. in Form von roter Schriftfarbe in den entsprechenden Feldern). Über den Button *Abbrechen* lässt sich der Konfigurator jederzeit verlassen. Ist eine gültige Auswahl eingestellt, kann die Konfiguration über *Speichern* in einem separaten Speicherdialog persistiert werden.

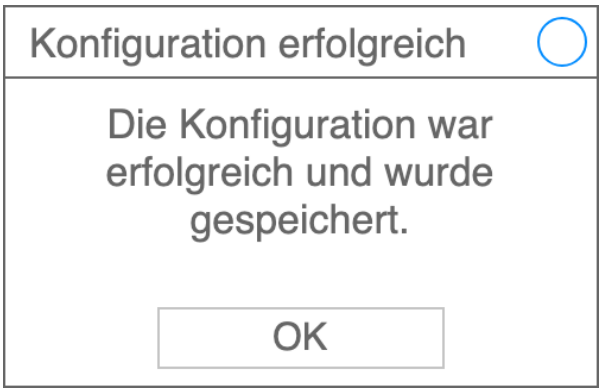
2.3.5 Team bzw. Partiekonfiguration speichern



In diesem Dialog kann Dateiname und Speicherort der Konfiguration festgelegt werden. Der Button *Abbrechen* bringt den Nutzer direkt zurück in den entsprechen Konfigurator. Durch Klicken auf den Button *Speichern* wird die Datei mit dem gewählten Namen und Speicherort gespeichert.

Wurde versucht eine Konfiguration mit ungültigen Parametern zu speichern oder trat beim Speichervorgang ein Fehler auf, wird der Nutzer über dieses Popup darüber informiert. Der *Ok*-Button führt zurück zum entsprechenden Konfigurator.

2.3.6 Konfiguration erfolgreich

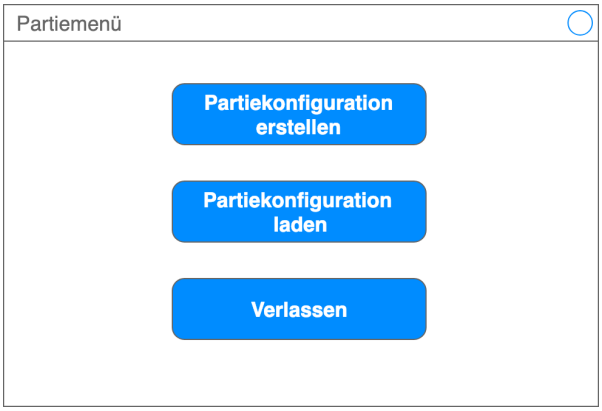


Wenn alle Paramter einer Konfiguration gültig waren, wird dieser Dialog angezeigt. Der *Ok*-Button führt zurück zum entsprechenden Menü.

2.3.7 Konfiguration ungültig

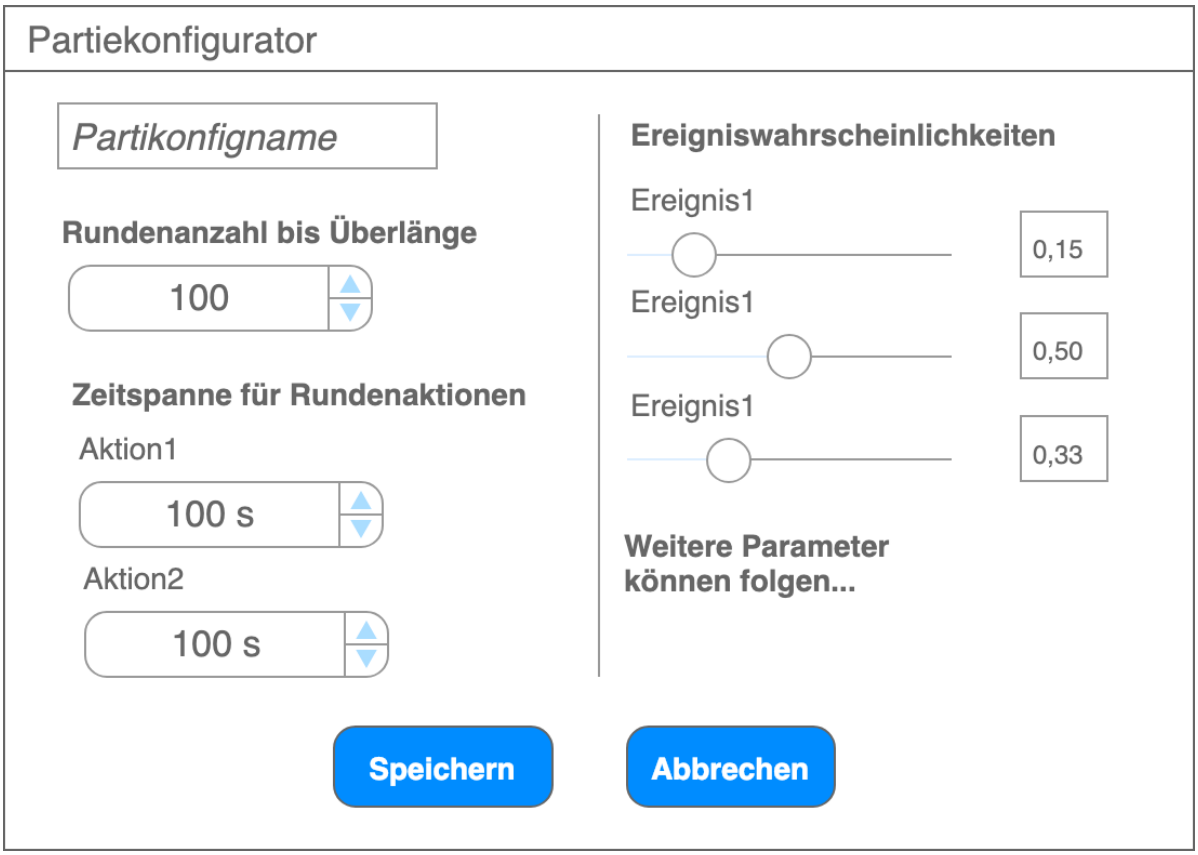
Wurde versucht eine Konfiguration mit ungültigen Parametern zu speichern oder trat beim Speichervorgang ein Fehler auf, wird der Nutzer über dieses Popup darüber informiert. Der *Ok*-Button des Popups führt zurück zum entsprechenden Dialog.

2.3.8 Partiemenü



Über das Partiemenü kann eine Auswahl zwischen dem Laden und dem Erstellen einer Partiekonfiguration getroffen werden. Das erfolgt über die entsprechenden Buttons. Über den Button *Verlassen* kann das *Partiemenü* verlassen werden.

2.3.9 Partiekonfigurator



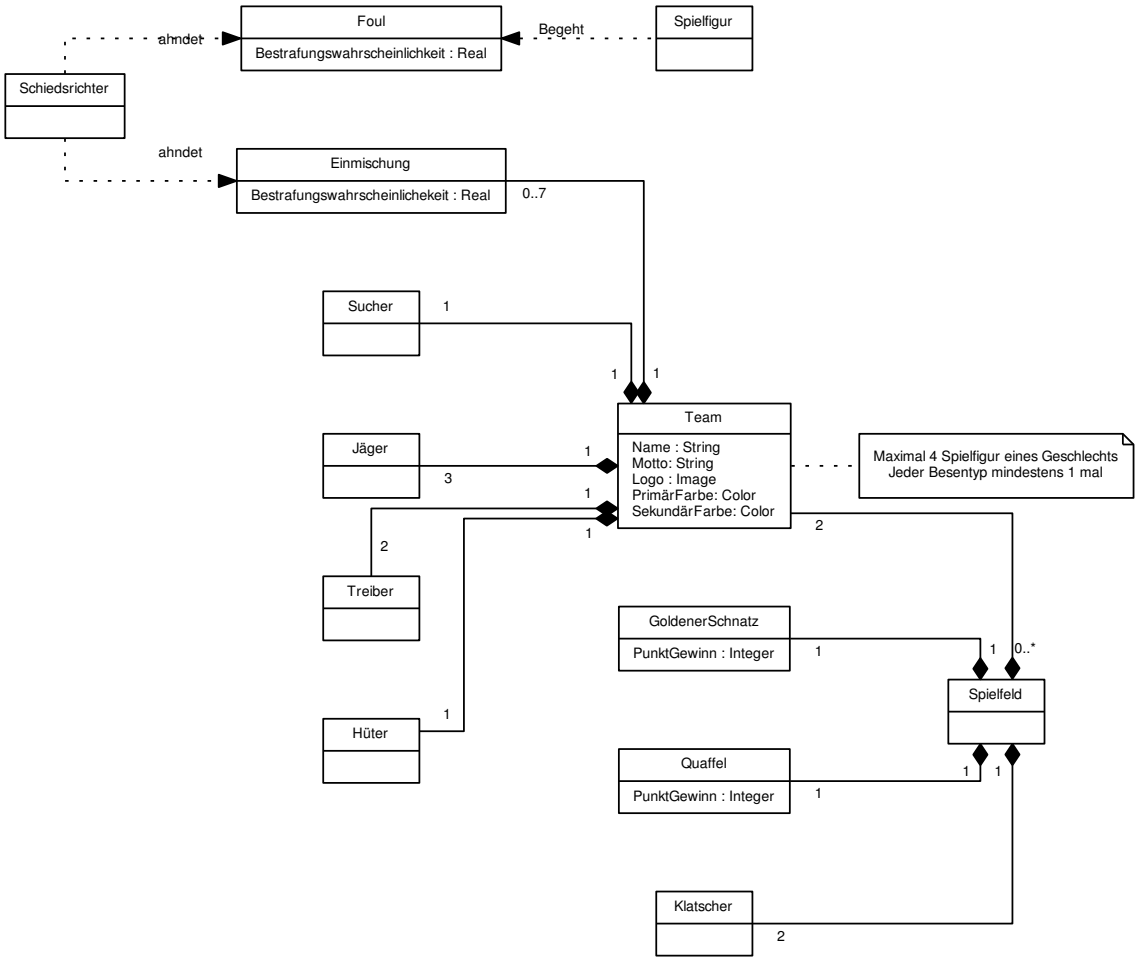
Im Partiekonfigurator können alle Paramter für eine gültige Partiekonfigurationsdatei eingestellt werden. Dazu gehören unter anderem die Rundenzahl, bis Überlänge erreicht ist, die Zeitspannen für die jeweiligen Spielaktionen und Ereigniswahrscheinlichkeiten. Je nach Art des Parameteres sind Spinner, Slider, Textfelder oder im weiteren Verlauf der Implementierung noch andere Auswahlelemente vorhanden.

Über den Button *Speichern* gelangt man in den *Partiekonfiguration speichern*-Dialog. Der Button *Abbrechen* führt zurück ins *Partiemenü*.

3 Datenmodell

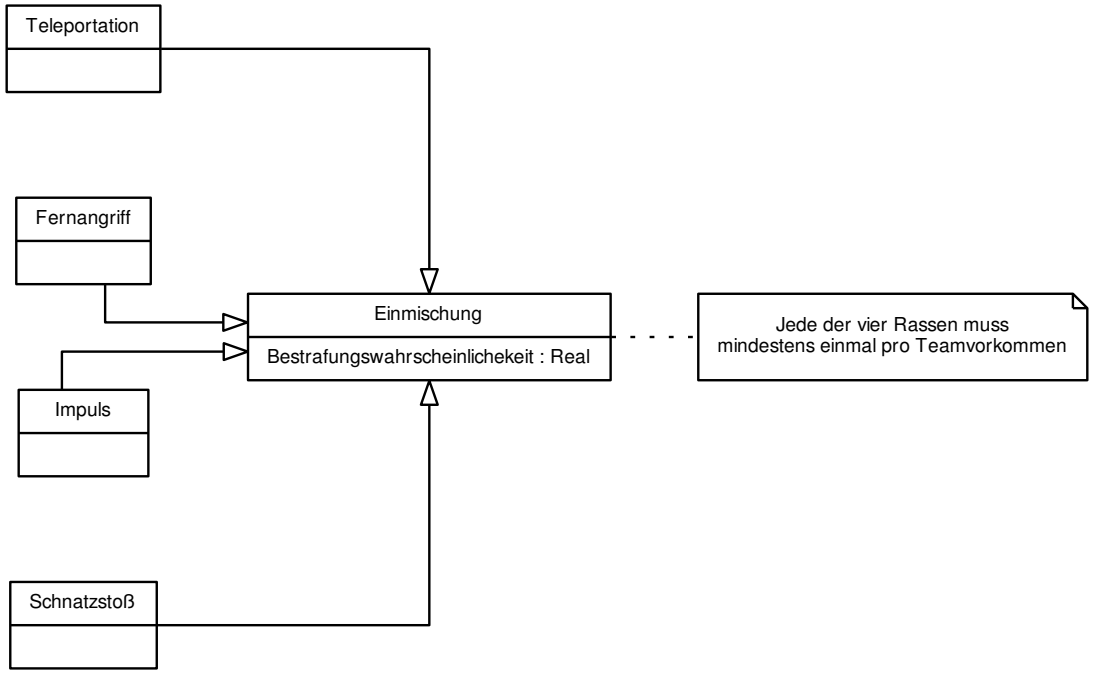
3.1 Gesamtübersicht

Im folgenden werden die Domänen des Spiels modelliert. Die Diagrammstruktur orientiert sich an einem UML-2 Klassendiagramm. Das erste Diagramm gibt einen Überblick über die Domäne, die folgenden Diagramme bilden die spezifischen Aspekte der Domäne detailierter ab.



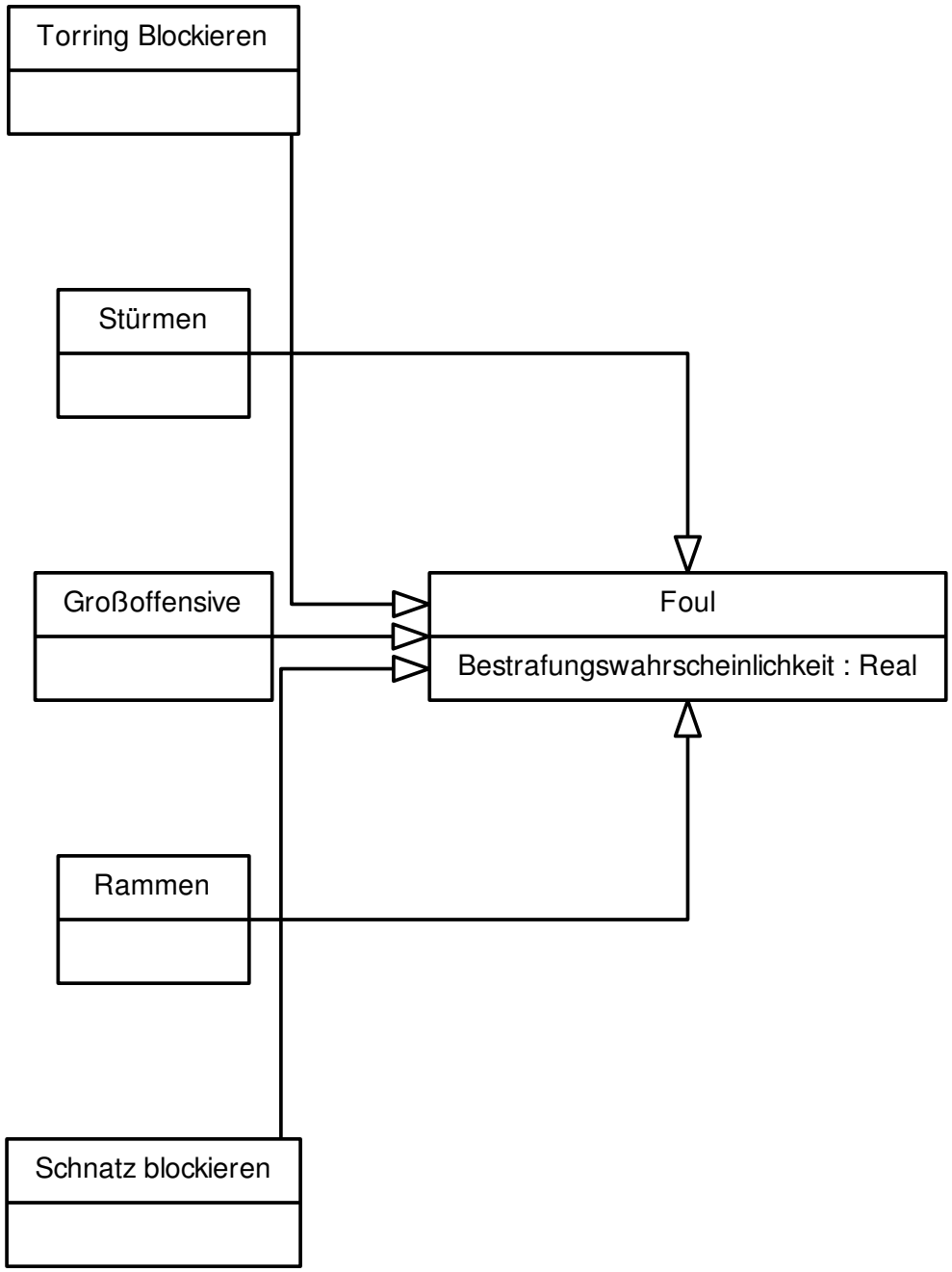
3.2 Einmischung

Gibt den Fantyp an.



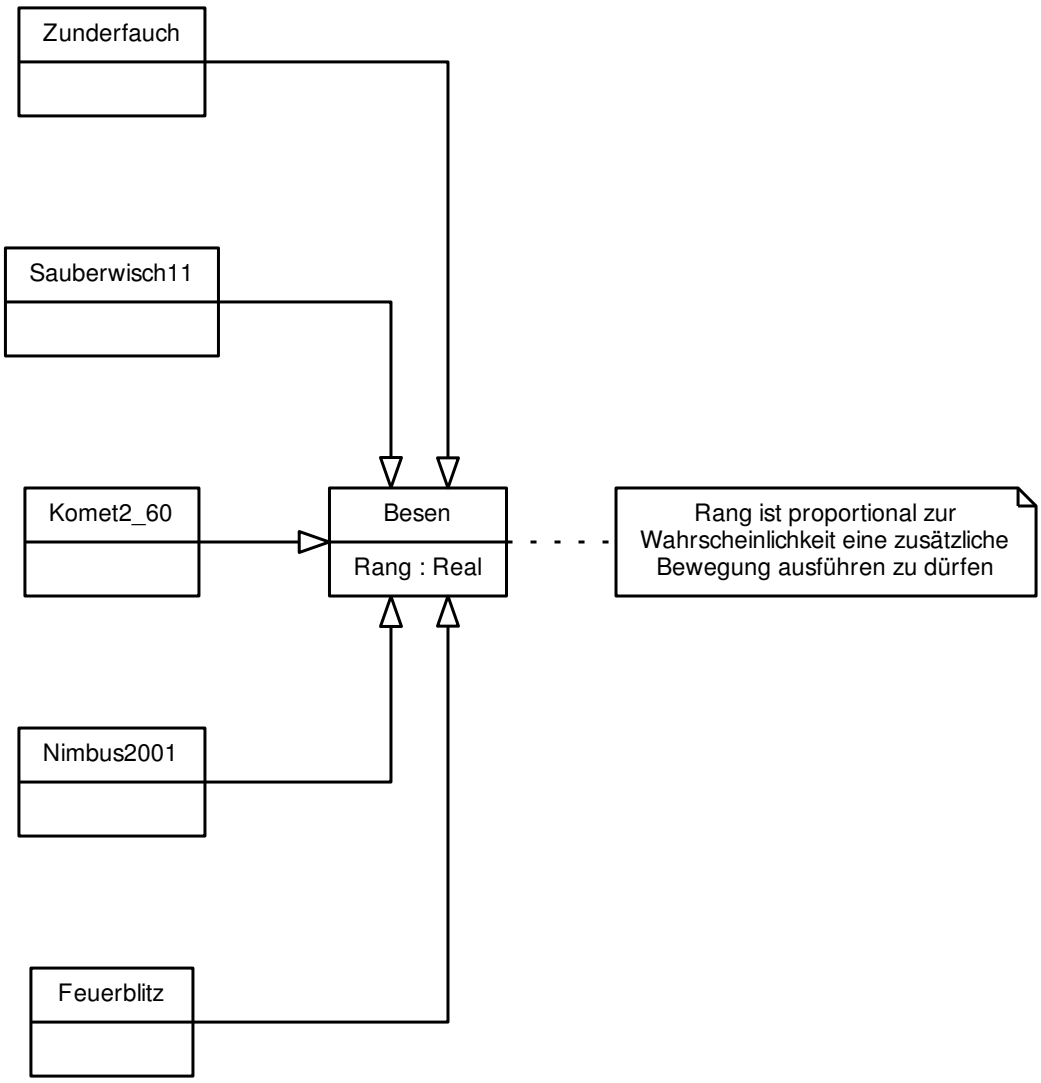
3.3 Foul

Es gibt verschiedene Fouls, welche in dem Domänenmodell alle aufgeführt werden und möglich sind.



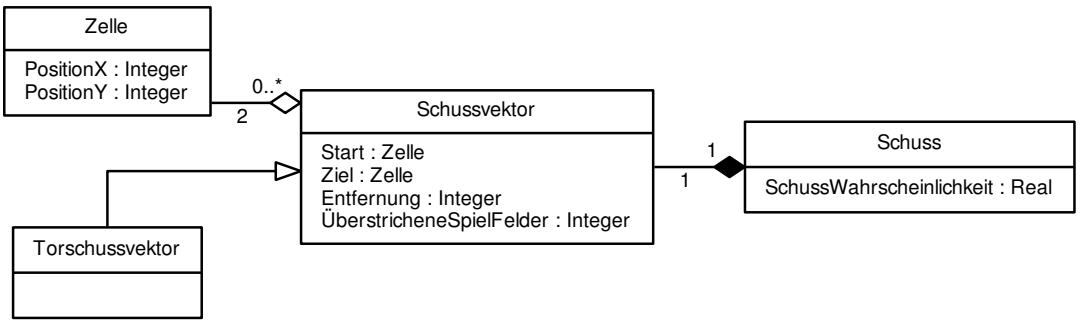
3.4 Besen

Alle in dem Domänenmodell verschiedenen Besen sind möglich zum auswählen.



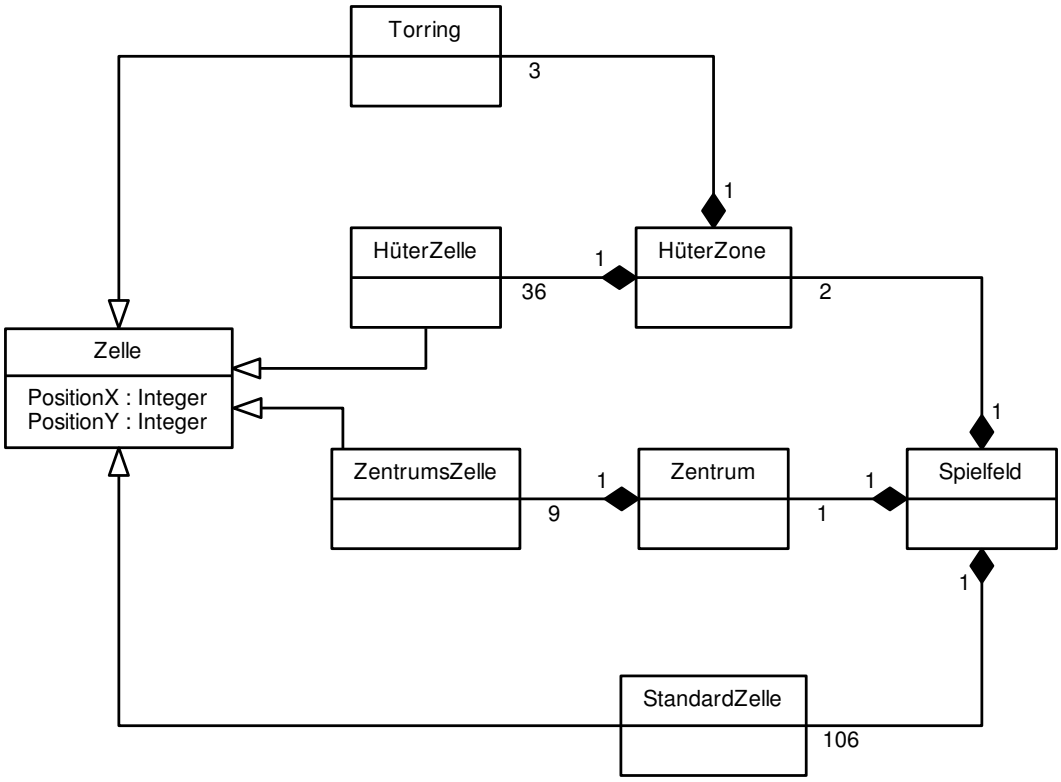
3.5 Schuss

Ablauf eines Schusses im Domänenmodell.



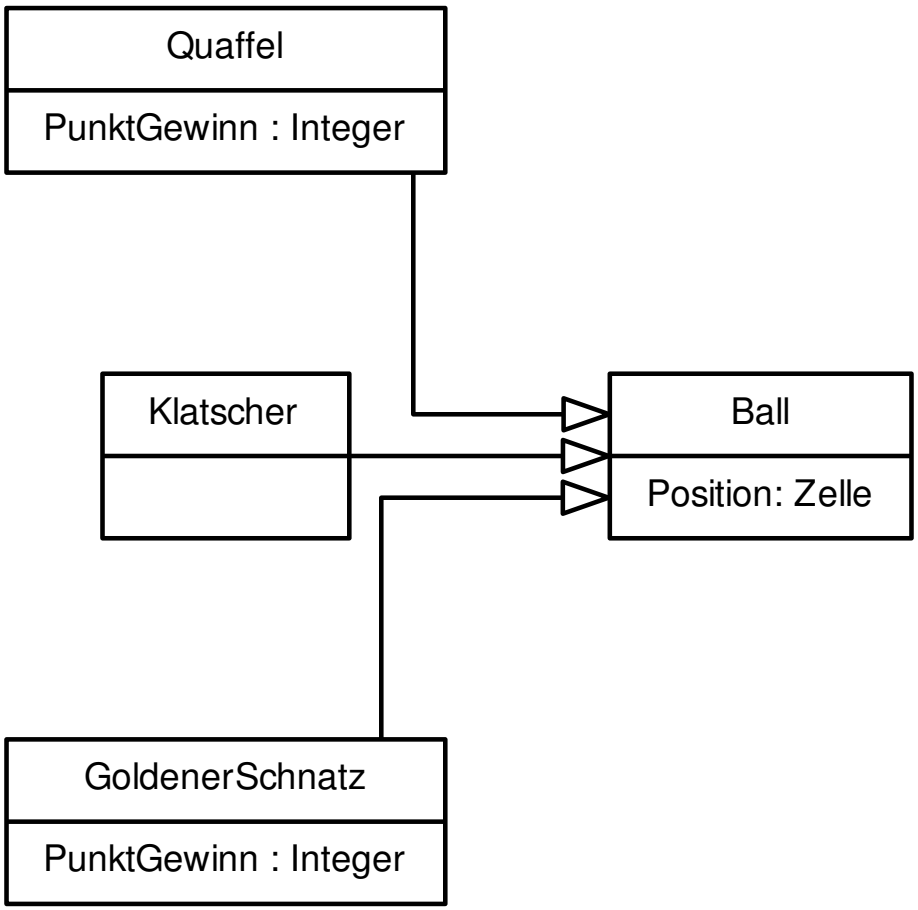
3.6 Zelle

Die verschiedenen Zelltypen, die in dem Spiel vorkommen.



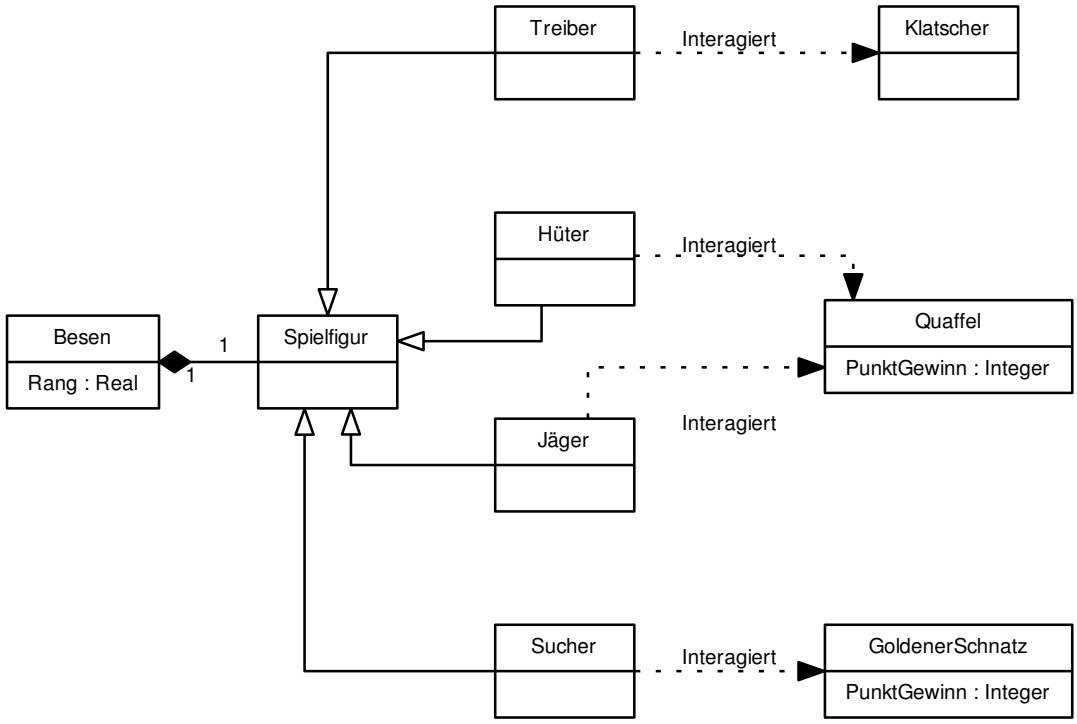
3.7 Ball

Die verschiedenen Ballarten des Spiels.



3.8 Interaktion der Spielerfiguren mit den Bällen

Die möglichen Interaktionen, die Spielfiguren jeweils mit den Bällen machen können.



4 Funktionen

Hier wird angegeben, welche grundlegenden Funktionen jede Komponente erfüllen muss. Die Angaben sind nicht implementierungsspezifisch.

4.1 Client

Name	Parameter	Vorbedingung	Nachbedingung	Beschreibung
Status anzeigen	Team1, Team2, Punktestand, Rundennummer, aktuelle Spielfigur	-	-	Zeigt den aktuellen Status der Partie in der Spielansicht an.
Spielfeld rendern	mögliche Aktionen	-	-	Zeigt das Spielfeld, die Spielfiguren und mögliche ausführbare Aktionen der aktuell aktiven Spielfigur in der Spielansicht an.
Aktion registrieren	Aktion	Aktion ist in der gegebenen Situation möglich.	-	Fügt eine Aktion einer Liste von auszuführenden Aktionen zu.

Pausieren	-	-	-	Wechselt in den Pause-Dialog.
Aktionen zurücksetzen	Liste registrierter Aktionen	Liste der registrierter Aktionen ist nicht leer	-	Leert die Liste der registrierten Aktionen
Zug ausführen	Liste registrierter Aktionen	Liste registrierter Aktionen ist nicht leer.	-	Sendet die Liste registrierter Aktionen an den Server zur Verarbeitung.
Legende anzeigen	Team1, Team2	-	-	Zeigt alle Spielfiguren in der Legende am unteren Rand der Spielansicht an.
Statusmeldung anzeigen	Statusmeldung	-	-	Fügt den Statusmeldungen am unteren Rand der Spielansicht die übergebene Statusmeldung an.
Spiel verlassen	-	-	Positive Antwort in dem sich öffnenden Bestätigungs-Popup	Wechselt in den Hauptmenü-Dialog.
Spiel beenden	-	-	Positive Antwort in dem sich öffnenden Bestätigungs-Popup	Schließt die Anwendung.
Hilfe anzeigen	-	-	-	Wechselt in den Hilfe-Dialog.

Team ändern	-	-	Eine gültige Team-Konfiguration ist ausgewählt.	Öffnet das „Team ändern“-Popup und ändert die aktive Team-Konfiguration.
Spielsuche öffnen	-	Eine gültige Team-Konfiguration ist aktiv.	-	Wechselt in den Spielsuche-Dialog
Hauptmenü öffnen	-	-	-	Wechselt in den Hauptmenü-Dialog.
Serververbindung aufbauen	Serveradresse, Port, als Gast	-	Auf dem jeweiligen Port des Rechners mit der Adresse ist ein Spielserver initialisiert.	Verbindet sich mit dem angegebenen Server.
Hotkeys öffnen	-	-	-	Wechselt in den Hotkey-Dialog.
Spiel fortsetzen	-	-	-	Wechselt vom Pause-Dialog zurück in die Spielansicht.
Spielende-Dialog öffnen	Punktestand, Name des Gewinners	-	-	Wechselt in den Spielende-Dialog und zeigt darin den Endpunktestand und den Namen des Gewinners an.

4.2 Server

Name	Parameter	Vorbedingung	Nachbedingung	Beschreibung
Initialisieren	Port, Partie-Konfiguration	Gültiger Port angegeben	Initialisierung erfolgreich	Startet den Server, damit sich Clients damit verbinden können.
Spiel starten	-	Zwei Spieler sind verbunden.	-	Benachrichtigt die verbundenen Clients, dass das Spiel gestartet wurde.
Aktionen empfangen	Partie ist nicht pausiert	Clients sind verbunden.	-	Wartet auf Informationen vom Client eines Spielers darüber, welche Aktionen durchgeführt werden sollen und verarbeitet diese Informationen zu einer Liste von durchzuführenden Aktionen.
Spielsituation aktualisieren	Liste von Aktionen	Clients sind verbunden, Spiel ist nicht pausiert.	-	Ändert die Spielsituation anhand der durchzuführenden Aktionen und schickt die aktuelle Spielsituation an alle verbundenen Clients.

Partie pausieren	-	-	-	Stoppt alle laufenden timer und verhindert das Verändern der Spielsituation.
Partie fortsetzen	-	Spiel ist pausiert.	-	Lässt alle timer weiterlaufen und ermöglicht das Verändern der Spielsituation.

4.3 Konfigurator

Name	Parameter	Vorbedingung	Nachbedingung	Beschreibung
Beenden	-	-	-	Schließt die Anwendung.
Teammenü öffnen	-	-	-	Wechselt in den Teammenü-Dialog.
Partiemenü öffnen	-	-	-	Wechselt in den Partiemenü-Dialog
Konfiguratormenü öffnen	-	-	-	Wechselt in den Konfiguratormenü-Dialog.
Teamkonfigurator öffnen	-	-	-	Wechselt in den Temkonfigurator-Dialog.
Pertiekonfigurator öffnen	-	-	-	Wechselt in den Partiekonfigurator-Dialog.
Konfiguration laden	Dateipfad	Datei im Pfad ist eine Konfigurations-Datei.	-	Öffnet eine Konfigurationsdatei und lädt sie in den Konfigurator zur Bearbeitung.
Konfiguration speichern	Konfiguration, Pfad	Konfiguration erfüllt alle Bedingungen.	-	Speichert die übergebene Konfiguration als Konfigurations-Datei.

4.4 KI-Client

Name	Parameter	Vorbedingung	Nachbedingung	Beschreibung
Serververbindung herstellen	Adresse, Port	Auf dem jeweiligen Port des Rechners mit der Adresse ist ein Spielserver initialisiert.	Verbindung ist erfolgreich.	Verbindet den KI-Client mit dem angegebenen Server.
Spielsituation empfangen	-	-	-	Wartet auf Informationen zur Spielsituation vom Server.
Spielerphase durchführen	-	-	-	Berechnet, welche Aktionen in der aktuellen Spielerphase durchgeführt werden sollen und schickt sie an den Server.
Endphase durchführen	-	-	-	Berechnet, welche Einmischungen in der aktuellen Endphase ausgeführt werden sollen und schickt sie an den Server.

Teil IV
Randbedingungen

1 Qualität

1.1 Nicht funktionale Anforderungen

Bei den nachfolgenden nicht funktionalen Anforderungen handelt es sich in erster Linie um qualitative und messbare bzw. vergleichbare Anforderungen an die Anwendungen. Da diese Anforderungen die Qualität der Software sicherstellen sollen, ist laufend zu prüfen, ob sie eingehalten werden.

ID:	**QA1**
Titel:	Plattformunabhängigkeit
Beschreibung:	Die Serveranwendung und der KI-Client müssen auf mindestens zwei gängigen Computerbetriebssystem-Plattformen (z.B. Linux, Windows) uneingeschränkt benutzbar sein.
Begründung:	Die Plattformunabhängigkeit ist von großer Bedeutung, da die Anwendungen auf möglichst vielen Zielsystemen funktionieren sollen, um die Menge an Endnutzer so wenig wie möglich einzuschränken.
Abhängigkeiten:	Programmiersprache, Docker-Container
Priorität:	+
Akteure:	Nutzer, Entwickler

ID:	**QA2**
Titel:	Version-Controlling
Beschreibung:	Beim Verwalten des Quellcodes soll ein Git basiertes Version-Controlling Werkzeug (*GitHub / GitLab*) verwendet werden.
Begründung:	Durch das Verwenden eines Versionierungswerkzeuges wird das zusammenarbeiten unterschiedlicher Entwickler erleichtert, da das zusammenführen des Codes größtenteils automatisiert abläuft.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Entwickler

ID:	QA3
Titel:	Continuous Integration
Beschreibung:	Jeder Commit soll automatisch mit Hilfe der CI Unit-Tests und der Statischen Codeanalyse unterzogen werden. Zudem soll eine automatisierte Code-Dokumentation angestoßen werden. Bei erfolgreichem Abschließen aller Tests soll zum Schluss der aktuelle Stand deployed werden.
Begründung:	Die CI nimmt den Entwicklern Arbeit ab und kann dazu beitragen, dass Fehler frühzeitig erkannt und behoben werden können.
Abhängigkeiten:	Version-Controlling
Priorität:	0
Akteure:	Entwickler

ID:	QA4
Titel:	Statische Codeanalyse
Beschreibung:	Mit Hilfe des Tools „SonarQube“ bzw. „SonarCloud“ soll der gesamt Quellcode einer statischen Analyse unterzogen werden. Dabei darf die technische Codequalität von diesem Tool nicht schlechter als „B“ bewertet werden.
Begründung:	Quellcode mit einer hohen Codequalität ist weniger anfällig für Fehler und Probleme.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Entwickler

ID:	QA5
Titel:	Automatisierte Unit-Tests
Beschreibung:	Alle definierten Unit-Tests müssen fehlerfrei bestanden werden.
Begründung:	Da alle Komponenten fehlerfrei funktionieren müssen, ist es unerlässlich, die einzelnen Teil der Software ständig auf ihre Funktionalität zu prüfen.
Abhängigkeiten:	-
Priorität:	0
Akteure:	Entwickler

ID:	QA6
Titel:	Docker Container
Beschreibung:	Um die Plattformunabhängigkeit zu gewährleisten, soll sowohl die Server-Komponente, als auch die KI-Komponente mit Hilfe eines Docker Containers veröffentlicht werden.
Begründung:	Docker Container bieten den Vorteil, dass die Software nicht auf jedem Zielsystem neu compiliert werden muss. Sobald sie auf einem System in einem Docker-Container lauffähig gemacht wurde, lässt sich dieser Container in der Regel auf diversen anderen Zielsystemen ausführen.
Abhängigkeiten:	Plattformunabhängigkeit
Priorität:	+
Akteure:	Entwickler

ID:	QA7
Titel:	Dokumentation
Beschreibung:	Alle Klassen und Methoden der Software müssen dokumentiert werden. Dabei sollen mindestens alle Übergabeparameter und Rückgabewerte genau spezifiziert werden. Zudem sind komplexe Algorithmen detailliert zu dokumentieren. Die gesamte Dokumentation ist dabei mit dem Tool Doxygen zu erstellen.
Begründung:	Gut dokumentierte Software vereinfacht die Fehlersuche, die Wartung und das hinzufügen von neuen Features.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Entwickler

ID:	QA8
Titel:	Benutzerhandbuch
Beschreibung:	Zu jeder Komponente des Projektes muss eine Benutzerhandbuch existieren, in welchem alle Features unmissverständlich erklärt sind, sodass ein neuer Benutzer auf Basis des Benutzerhandbuches die Software bedienen kann.
Begründung:	Das Benutzerhandbuch erleichtert die Bedienung der Anwendung.
Abhängigkeiten:	Dokumentation
Priorität:	0
Akteure:	Nutzer, Entwickler

ID:	QA9
Titel:	Anwendungssprache
Beschreibung:	Das User-Interface der Anwendungen soll in deutscher Sprache gestaltet werden.
Begründung:	Die Zielkundschaft spricht überwiegend Deutsch.
Abhängigkeiten:	-
Priorität:	0
Akteure:	Nutzer, Entwickler

ID:	QA10
Titel:	Implementierungssprache
Beschreibung:	Die Anwendung soll in englischer Sprache implementiert werden.
Begründung:	Die Implementierungssprache ist im Lastenheft vorgegeben.
Abhängigkeiten:	-
Priorität:	0
Akteure:	Entwickler

ID:	QA11
Titel:	Dokumentationssprache
Beschreibung:	Die Dokumentation der Software kann in englischer oder deutscher Sprache gestaltet sein.
Begründung:	Die Dokumentationssprache ist im Lastenheft vorgegeben.
Abhängigkeiten:	-
Priorität:	0
Akteure:	Entwickler, Kunde

ID:	QA12
Titel:	Programmiersprache
Beschreibung:	Die verwendete Programmiersprache ist C++.
Begründung:	C++ bietet mehr Features als Java, stellt aber trotzdem Plattformunabhängigkeit sicher.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Entwickler

ID:	QA13
Titel:	Format für Konfigurationsdateien
Beschreibung:	Alle Konfigurationsdateien müssen den *JSON* Standard erfüllen. Des Weiteren sind alle vom Komitee festgelegten weiteren Standards einzuhalten.
Begründung:	Durch einheitliche Formate der Konfigurationsdateien lässt sich sicherstellen, dass einzelne Komponenten zwischen den Entwicklungsteams ausgetauscht werden können und diese miteinander kompatibel sind.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Entwickler, Nutzer

ID:	QA14
Titel:	Netzwerkkommunikation
Beschreibung:	Die Netzwerkkommunikation zwischen Client und Server soll über sogenannte *Web-Socket-Sessions* realisiert werden, sodass Client und Server ortsunabhängig von einander betrieben werden können. Die Nachrichten sollen im *JSON* Format formatiert sein.
Begründung:	Die Netzwerkkommunikation muss gewisse Standards erfüllen, damit Client- und Serveranwendungen von unterschiedlichen Entwicklerteams mit einander kompatibel sind.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Server, Client, KI-Client

ID:	QA15
Titel:	Funktionalität
Beschreibung:	Die Anwendungen müssen alle im Lastenheft als Minimalanforderungen aufgeführten Anforderungen erfüllen.
Begründung:	Um die Abnahmen zu bestehen, müssen die Minimalanforderungen erfüllt werden.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Kunde, Entwickler

ID:	QA16
Titel:	Zuverlässigkeit
Beschreibung:	Bei 100 Spielen darf maximal eine Partie aufgrund eines Fehlers in der Anwendung abgebrochen werden müssen.
Begründung:	Durch zu häufige Ausfälle der Software ist das Benutzererlebnis massiv beeinträchtigt.
Abhängigkeiten:	Robustheit
Priorität:	+
Akteure:	Nutzer, Entwickler

ID:	QA17
Titel:	Robustheit
Beschreibung:	Die Anwendungen dürfen nicht aufgrund einer falschen oder ungültigen Benutzereingabe abstürzen, sondern müssen den Benutzer auf seinen Fehler hinweisen.
Begründung:	Um das Benutzererlebnis nicht zu beeinträchtigen und keine Sicherheitslücken zu verursachen ist es notwendig, dass die Funktion der Software nicht durch fehlerhafte Benutzereingaben beeinträchtigt wird.
Abhängigkeiten:	-
Priorität:	++
Akteure:	Nutzer, Entwickler

ID:	QA18
Titel:	Benutzerfreundlichkeit
Beschreibung:	Dem Endnutzer muss es möglich sein, alle Komponenten des Projektes nur auf Basis des mitgelieferten Benutzerhandbuches und den Hilfeseiten die Software ohne Einschränkungen bedienen zu können.
Begründung:	Wenn es für die Endnutzer der Software zu kompliziert ist, die Software zu Benutzen, dann ist das Benutzererlebnis erheblich gestört und die Software wird nicht benutzt werden, da die Endbenutzer unzufrieden sind.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Nutzer

ID:	QA19
Titel:	Wartbarkeit
Beschreibung:	Die Software muss so aufgebaut sein, dass einzelne Teilstücke bei Bedarf ohne Umbauten der übrigen Software ersetzbar sind.
Begründung:	Im Falle einer Fehlfunktion in einem Teilstück der Software muss dieses einfach austauschbar sein, um den Fehler schnellstmöglich beheben zu können. Zudem sollte das Hinzufügen weiterer Features möglich sein, um das Produkt stetig weiterentwickeln zu können.
Abhängigkeiten:	-
Priorität:	-+
Akteure:	Entwickler

ID:	QA20
Titel:	Effizienz
Beschreibung:	Die Software sollte ressourcenschonend arbeiten. Keine Komponente darf mehr als ein Gigabyte Arbeitsspeicher benötigen. Zudem darf keine Komponente mehr als 50% der auf dem System zur Verfügung stehenden Prozessorleistung benötigen. Im Durchschnitt darf während einer Partie nicht mehr als $1\frac{MBit}{s}$ an Netzwerkbandbreite benötigt werden, um das Spiel ohne Einschränkungen nutzen zu können.
Begründung:	Eine ressourcenschonende Anwendung ist auch auf älteren Zielsystemen problemlos nutzbar.
Abhängigkeiten:	-
Priorität:	-
Akteure:	Nutzer, Entwickler

ID:	QA21
Titel:	Kurze Ladezeiten
Beschreibung:	Systembedingte Ladezeiten der Software dürfen auf einem aktuellen Computer eine Sekunde pro geladener Ansicht nicht überschreiten.
Begründung:	Bei längeren Ladezeiten ist das Benutzererlebnis massiv beeinträchtigt.
Abhängigkeiten:	-
Priorität:	+
Akteure:	Nutzer, Entwickler

2 Betriebskonzept

2.1 Allgemein

Um die Funktionalität der jeweilgen Anwendung zu nutzen, muss der verwendete Computer über eine Internetanbindung verfügen, die eine durchschnittliche Datenrate von $1\frac{\text{mbit}}{\text{s}}$ mit einer Latenz von unter 200ms zur Verfügung stellt. Des weiteren sollte der verwendete Computer über eine moderne CPU und mindestens 2GB RAM verfügen.

Da die Anwendung keine alte Anwendung ersetzt, ist ein Ablösungskonzept nicht von nöten. Auch das Einführungskonzept gestaltet sich einfach, da die Anwendung primär von Privatpersonen genutzt werden soll. Diese können sich bei passendem Zielsystem eine kompilierte Fassung der Anwendung herunterladen oder die Anwendung selber kompilieren.

Die angestrebte Lebensdauer der Anwendung beträgt fünf Jahre. Für die Anwendung wird jedoch keine Haftung übernommen, insbesondere auch nicht für die angestrebte Lebensdauer.

Für die Verwendung des jeweiligen Programms fallen keine Kosten an. Das in diesem Dokument aufgeführte Konzept wird als final angesehen, es sind keine weiteren Erweiterungen geplant. Auch Instandhaltungen sind nach Abnahme des Projekts nicht geplant.

2.2 Client

Der Client soll auf einem aktuellen Desktopbetriebssystem mit graphischer Oberfläche eingesetzt werden. Das heißt im speziellen, aber nicht ausschließlich, auf einer aktuellen Linux-Distribution.

Die Anwendung hängt von der SFML-Bibliothek ab, große Veränderungen an dieser Bibliothek können zu einer Verkürzung der Lebensdauer führen.

2.3 Server

Die Serveranwendung soll auf einem aktuellen Betriebssystem mit Unterstützung für Docker eingesetzt werden. Das heißt im speziellen, aber nicht ausschließlich, auf einer aktuellen Linux-Distribution, auf macOS und auf Windows 10.

Durch eine Beschreibung der Konfigurationsoptionen der Applikation, also der Kommandozeilenargumente sowie der Konfigurationsdateien, in Form einer Dokumentation ist es dem Administrator möglich, sich über die Benutzung der Anwendung zu informieren. Dadurch sind gesonderte Schulungen nicht vonnöten.

Die Serveranwendung läuft in einem Docker-Container, weshalb ihre Funktionalität von Docker abhängig ist.

2.4 Team- und Partiekonfigurator

Der Team- und Partiekonfigurator soll auf einem aktuellen Desktopbetriebssystem mit graphischer Oberfläche eingesetzt werden. Das heißt im speziellen, aber nicht ausschließlich, auf einer aktuellen Linux-Distribution.

Die Anwendung hängt von der SFML-Bibliothek ab, große Veränderungen an dieser Bibliothek können zu einer Verkürzung der Lebensdauer führen.

2.5 KI-Client

Die Serveranwendung soll auf einem aktuellen Betriebssystem mit Unterstützung für Docker eingesetzt werden. Das heißt im speziellen, aber nicht ausschließlich, auf einer aktuellen Linux-Distribution, auf macOS und auf Windows 10.

Der KI-Client läuft in einem Docker-Container, weshalb seine Funktionalität von Docker abhängig ist.

3 Entwicklungsvorgaben

Alle Teile der Anwendung werden in C++17 geschrieben. Als Compiler wird dafür entweder der GNU-C++-Compiler (GCC) oder Visual C++ (MSVC) verwendet, als Buildsystem wird CMake genutzt. Zur statischen Codeanalyse wird Clang-Tidy verwendet, Unit Tests werden mit Google-Test bzw. Google-Mock implementiert. Um undefiniertes Verhalten zu vermeiden wird das Programm während der Ausführung mit Adress Sanitizer überprüft.

Der Entwicklungsprozess der Softwaremodule gestaltet sich als agiler Scrum-Prozess mit einer Sprintdauer von zwei Wochen. Für jede Komponente wird ein Ansprechpartner definiert, der die Koordination der Entwicklung übernimmt. Das Entwicklungsteam besteht aus sechs Entwicklern, zusätzlich agiert der Tutor als Scrummaster. Der Quellcode wird mithilfe von Git versioniert, wobei für jede Komponente ein eigenes Repository existiert. Die Entwicklung der Features erfolgt auf dedizierten Feature-Branches, sodass der master-Branch immer in einem funktionierenden Zustand bleibt.

Alle Teile des Quellcodes werden mithilfe von Doxygen-Kommentaren dokumentiert. Im wesentlichen soll sich während des Entwicklungsprozess an den Linux-Kernel-Styleguide

[1] gehalten werden. Zwei wesentliche Abweichungen von diesem Styleguide sind eine Einrückungstiefe von 4 Zeichen, sowie die Platzierung der öffnenden Klammer bei Funktionsaufrufen [2] in der selben Zeile.

4 Abnahmekriterien

Die einzelnen Anwendungen gelten als vollendet, wenn alle Funktionalen- und Nichtfunktionalen Anforderungen mit einer Priorität größer gleich „0“, also „0“, „+“ sowie „++“, erfüllt sind.

[1] https://www.kernel.org/doc/html/v4.20/process/coding-style.html
[2] Siehe Absatz 3 des Styleguides