Technische Universität Braunschweig

ISF

Capture the Flag

Team 1

Software-Entwicklungspraktikum (SEP)
Sommersemester 2015

Pflichtenheft

Auftraggeber
Technische Universität Braunschweig
Institut für Softwaretechnik und Fahrzeuginformatik
Prof. Dr. Ina Schaefer
Mühlenpfordtstr. 23
38106 Braunschweig

Betreuer: Benjamin Mensing

Auftragnehmer:

Name	E-Mail-Adresse
Lennart Müller	lennart.mueller@tu-braunschweig.de
Pavel Bezwerk	b.pavel@tu-braunschweig.de
Nils-André Forjahn	n.forjahn@tu-braunschweig.de
Robert Drachenberg	r.drachenberg@tu-braunschweig.de
Falk Jacobsen	f.jacobsen@tu-braunschweig.de
Andhika Yopi Setyawan Putera	a.setyawan-putera@tu-braunschweig.de
Nico Scheideler	n.scheideler@tu-braunschweig.de
Ricardo Reck	r.reck@tu-braunschweig.de

Braunschweig, 13. Mai 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Zielbestimmung

Ziel dieses Projektes, welches im Rahmen des Software-Entwicklungspraktikums 2015 stattfindet, ist die Umsetzung des Spiels "Capture the Flag" mit Lego Mindstorms Robotern der Reihe EV3.
Bei dem Spiel "Capture the Flag" versuchen die Roboter eines Teams die gegnerische Flagge aus deren Lager zu erobern und diese ins eigene Lager zu befördern. Parallel dazu muss die eigene Flagge vor dem gegnerischen Team verteidigt werden indem man gegnerische Roboter mittels SSchüssenängreift.
Gewonnen ist das Spiel, wenn sich beide Flaggen im eigenen Lager befinden.

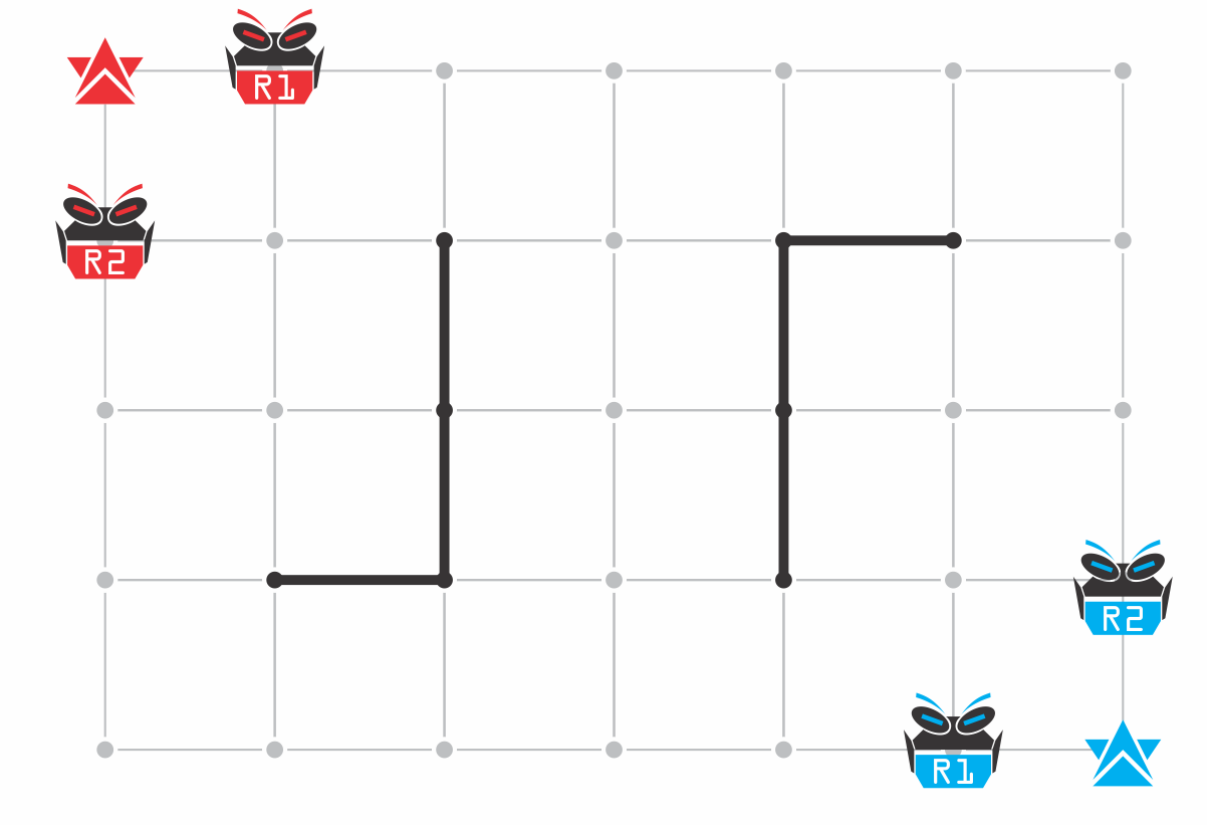

Abbildung 1.1: Spielfeld

Die Roboter sind in der Lage das Spiel mithilfe einer **künstlichen Intelligenz** selbstständig zu spielen.
Ein mit Klebestreifen auf den Boden erzeugtes Gitternetz dient als Spielfeld (sieh Abb. 1.2) auf dem sich die Roboter mittels Lichtsensoren bewegen.
Erweiternd dazu gibt es ein Editorprogramm, auf dem man Spielfeld und Spielregeln konfigurieren kann. Auf einer GUI (Spielsteuerung) wird während des Spielverlaufs ein digitales Abbild des Spielgeschehens dargestellt. Außerdem besteht auf der GUI die Möglichkeit die Roboter durch eine manuelle Steuerung zu bewegen.

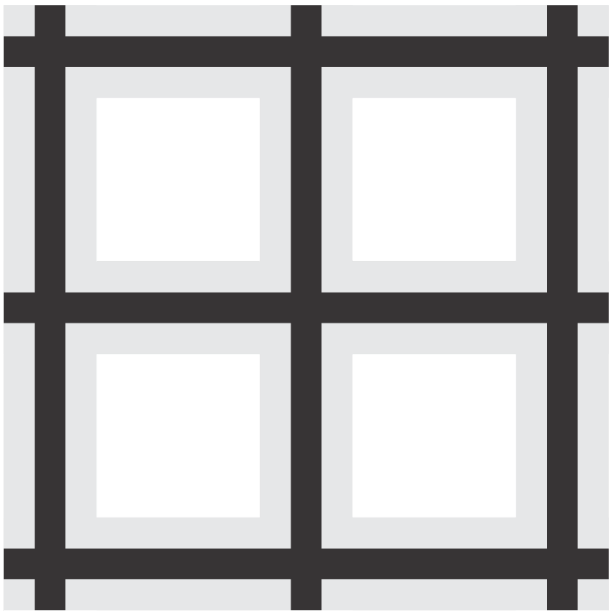

Abbildung 1.2: Gitternetz

1.1 Musskriterien

Hier wird aufgeführt, welche Funktionalitäten/Leistungen das Softwareprodukt in jedem Fall erfüllen muss, damit es genutzt werden kann.

$\langle RM1 \rangle$ - Die Roboter müssen sich mithilfe von NXT-Lightsensoren auf dem Spielfeld bewegen können.
$\langle RM2 \rangle$ - Die Roboter müssen vorwärts fahren können.
$\langle RM3 \rangle$ - Die Roboter müssen sich 90 Grad nach links und rechts drehen können.
$\langle RM4 \rangle$ - Die Roboter müssen eine definierte Sichtweite haben.
$\langle RM5 \rangle$ - Die Roboter müssen gegnerische Roboter angreifen können.
$\langle RM6 \rangle$ - Der Server muss den eigenen Robotern Pakete mit Anweisungen senden können.
$\langle RM7 \rangle$ - Die Roboter müssen dem Server Bestätigungen senden können.

⟨$RM8$⟩ - Der eigene Server sendet Informationen an den Server des anderen Teams.
⟨$RM9$⟩ - Der eigene Server empfängt Informationen von dem Server des anderen Teams.
⟨$RM10$⟩ - Die KI muss sich an die Regeln von "Capture the Flag"halten.
⟨$RM11$⟩ - Die KI muss eine vordefinierte Strategie umsetzen können.
⟨$RM12$⟩ - Die KI muss auf gegnerische Züge reagieren können.
⟨$RM13$⟩ - Der Karteneditor muss dem Benutzer ermöglichen eine Spielkarte zu erstellen.
⟨$RM14$⟩ - Der Karteneditor muss dem Benutzer ermöglichen die Spielparameter (Lebenspunkte, Schussweite, Sichtweite, Wartezeit nach "Todünd Punkte zum Sieg einzustellen.
⟨$RM15$⟩ - Der Karteneditor muss dem Benutzer ermöglichen die Startpunkte der Roboter und Flaggen festzulegen.
⟨$RM16$⟩ - Die Spielsteuerung muss dem Benutzer ermöglichen das Spiel zu verfolgen.
⟨$RM17$⟩ - Die Spielsteuerung muss dem Benutzer ermöglichen die Roboter manuell zu steuern.
⟨$RM18$⟩ - Die Spielsteuerung muss den Punktestand beider Teams anzeigen.
⟨$RM19$⟩ - Die Spielsteuerung muss anzeigen wenn ein Team gewonnen hat.
⟨$RM20$⟩ - Der eigene Server muss eine Schnittstelle für den gegnerischen Server bereitstellen können.
⟨$RM21$⟩ - Der eigene Server muss sich mit dem gegnerischen Server verbinden können.
⟨$RM22$⟩ - Die Roboter müssen mit dem Server über eine Netzwerkverbindung kommunizieren können.

1.2 Sollkriterien

Dies sind Kriterien, die für die Lauffähigkeit des Produkts nicht zwingend erforderlich sind, für die Erreichung der Projektziele aber erfüllt werden sollten.

⟨$RS1$⟩ - Im Karteneditor sollen vor Spielbeginn gespeicherte Karten geladen und bearbeitet werden können.
⟨$RS2$⟩ - Die Spielsteuerung soll intuitiv bedienbar sein.
⟨$RS3$⟩ - Die Anzahl der Leben soll in der GUI angezeigt werden können.
⟨$RS4$⟩ - Die LED der Roboter soll beim letzten Leben rot blinken.
⟨$RS5$⟩ - Die LED der Roboter soll bei Besitz der gegnerischen Flagge grün leuchten.
⟨$RS6$⟩ - Die LED der Roboter soll bei einem Angriff orange blinken.

1.3 Kannkriterien

Die Erfüllung dieser Kriterien ist nicht unbedingt notwendig und sollten nur angestrebt werden, falls noch ausreichend Kapazitäten vorhanden sind.

$\langle RC1 \rangle$ - Die Anzahl der Schüsse könnten auf der Spielsteuerung angezeigt werden.
$\langle RC2 \rangle$ - Die Anzahl der Schüsse könnten auf dem Roboterdisplay angezeigt werden.
$\langle RC3 \rangle$ - Verschiedene Emoticons können bei bestimmten Aktionen auf dem Roboterdisplay angezeigt werden.

1.4 Abgrenzungskriterien

Hier ist zu verdeutlichen, welche Ziele mit dem Produkt bewusst nicht erreicht werden sollen oder werden können. Da die Wünsche an einem Produkt, ohne den technischen Kontext zu beachten, im allgemeinen sehr umfangreich und oft leicht zu formulieren sind, soll dieser Abschnitt dazu dienen, Abgrenzungen des Produktes zu definieren.
Auch Funktionen, die im Allgemeinen von ähnlichen Systemen zu erwarten wären, hier aber explizit nicht umzusetzen sind, sollten erwähnt werden.

$\langle RW1 \rangle$ - Die Roboter können nach Verlassen des Grids nicht selbständig zurückfinden.
$\langle RW2 \rangle$ - Der Benutzer kann die KI nicht verändern oder austauschen.
$\langle RW3 \rangle$ - Fehler in der Software können nicht selbstständig vom Benutzer behoben werden.
$\langle RW4 \rangle$ - Karten können nicht durch einen Zufallsgenerator erstellt werden.
$\langle RW5 \rangle$ - Es wird kein Handbuch für den Benutzer geben, da die Benutzeroberfläche selbsterklärend ist.
$\langle RW6 \rangle$ - Es wird keine Spielmodi über andere Netzwerke als das lokale Netzwerk geben.
$\langle RW7 \rangle$ - Es wird keine Applikation für mobile Endgeräte geben.

2 Produkteinsatz

In diesem Abschnitt werden die Anwendungsbereiche des Produkts, die Benutzer-Zielgruppe, die physikalischen sowie soziotechnischen Bedingungen zur Inbetriebnahme unseres Produkts festgelegt.

2.1 Anwendungsbereiche

Unser Produkt, sowohl die digitale Simulation als auch der analoge Ablauf auf dem Spielfeld mit Robotern, des Spiels "Capture the Flag", wird zum Zweck der Unterhaltung entwickelt. Es dient außerdem zur Veranschaulichung künstlicher Intelligenz von Robotern.

2.2 Zielgruppen

Die Zielgruppe sind Nutzer, die sich für künstliche Intelligenz im Anwendungsbeispiel mit Robotern interessieren. Da die Software intuitiv bedienbar ist, werden vom Benutzer keine besonderen Kenntnisse im Bereich Programmierung gefordert.

2.3 Betriebsbedingungen

Im folgenden Abschnitt wird näher auf die Betriebsbedingungen eingegangen, die gelten müssen, damit ein störungsfreier Ablauf des Produkts gewährleistet werden kann.

- Es muss ein Gitternetz auf dem Boden fixiert sein, welches das Spielfeld korrekt (siehe Abb. 1.2) abbildet.
- Für die Abbildung des Gitternetzes muss matt-schwarzes und weißes Klebeband verwendet werden (siehe Abb. 1.1).
- Das Spielfeld muss ausreichend beleuchtet sein.
- Der Untergrund des Spielfelds muss sauber und eben sein.
- Die Akkumulatoren müssen vor Spielbeginn ausreichend geladen sein.

- Die ständige Beobachtung des Systems findet durch die Bediener und den Server statt.

3 Produktübersicht

In diesem Abschnitt wird die Funktionalität des Systems grafisch mit Hilfe von Use-Case- und Aktivitätsdiagrammen dargestellt.

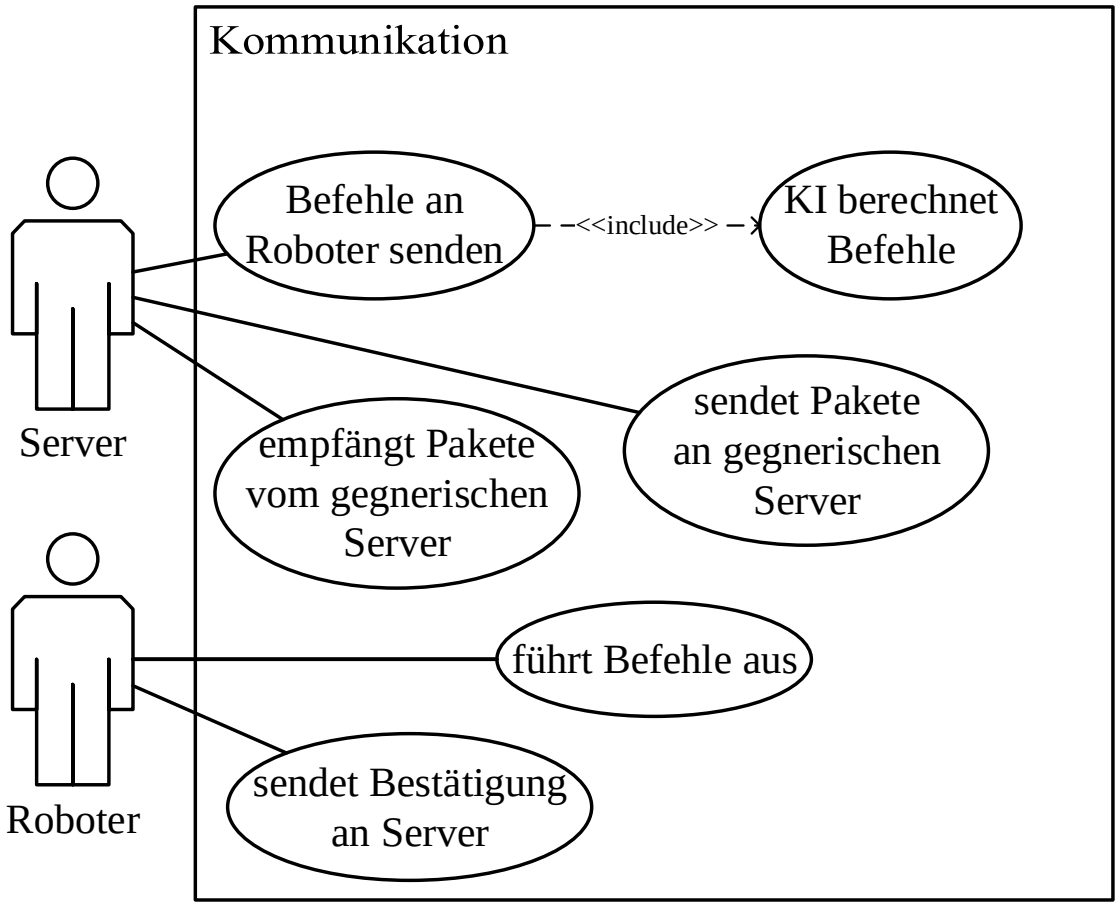

Abbildung 3.1: Use-Case-Diagramm *Kommunikation*

Dieses Diagramm stellt die Kommunikation zwischen eigenem Server und Roboter sowie die Kommunikation zwischen eigenem und gegnerischen Server dar. Der eigene Server empfängt Pakete vom gegnerischen Server mit Spieldaten über dessen letzten Spielzug. Auf Basis der aktualisierten Spieldaten ermittelt die KI dann den nächsten Spielzug. Dieser Spielzug enthält

Anweisungen, die anschließend durch den Server an den Robotern übermittelt werden. Die Roboter führen diese Anweisungen aus und senden daraufhin eine Bestätigung an den Server. Die neuen Spieldaten werden anschließend zurück an den gegnerischen Server gesendet.

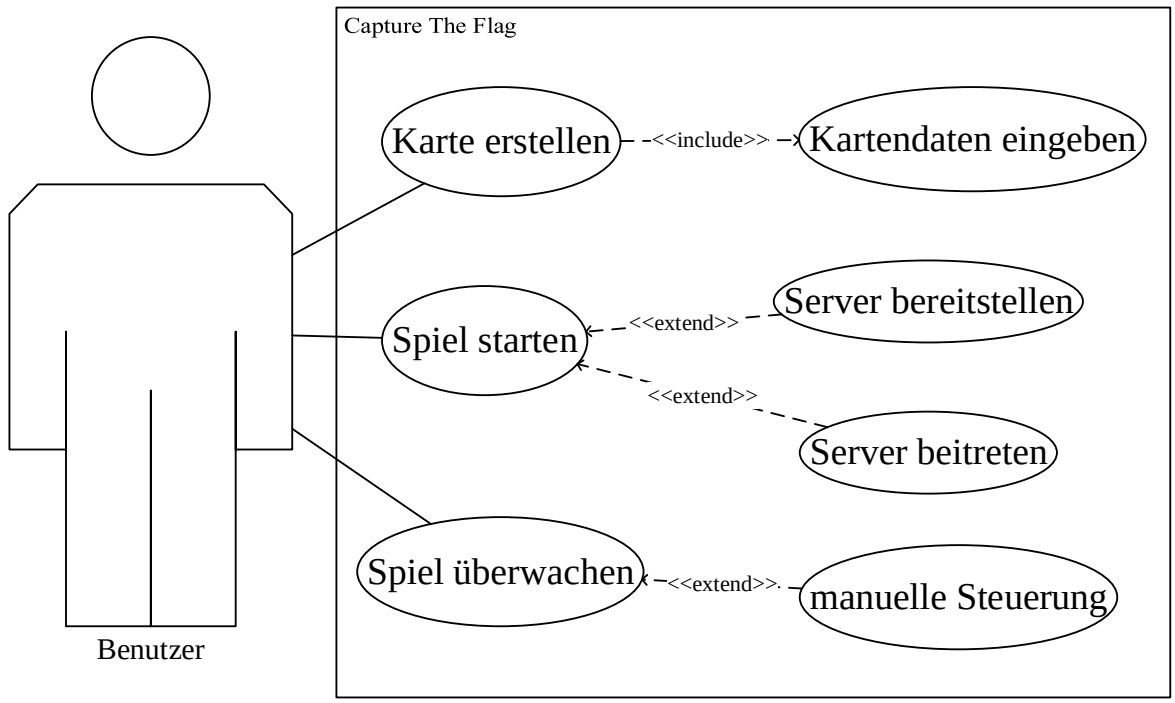

Abbildung 3.2: Use-Case-Diagramm *Capture The Flag*

Dieses Diagramm beschreibt die Anwendung aus Sicht des Benutzers. Der Benutzer kann die Spielkarte erstellen, ein Spiel starten und und es überwachen. Beim Erstellen des Spiels müssen noch weitere Parameter eingegeben werden (siehe Abbildung 3.2). Um ein Spiel zu starten, muss der Benutzer entweder einen Server bereitstellen oder einem bereits bestehenden Server beitreten. Neben der Spielüberwachung bietet die Software noch die Möglichkeit einer manuellen Steuerung.

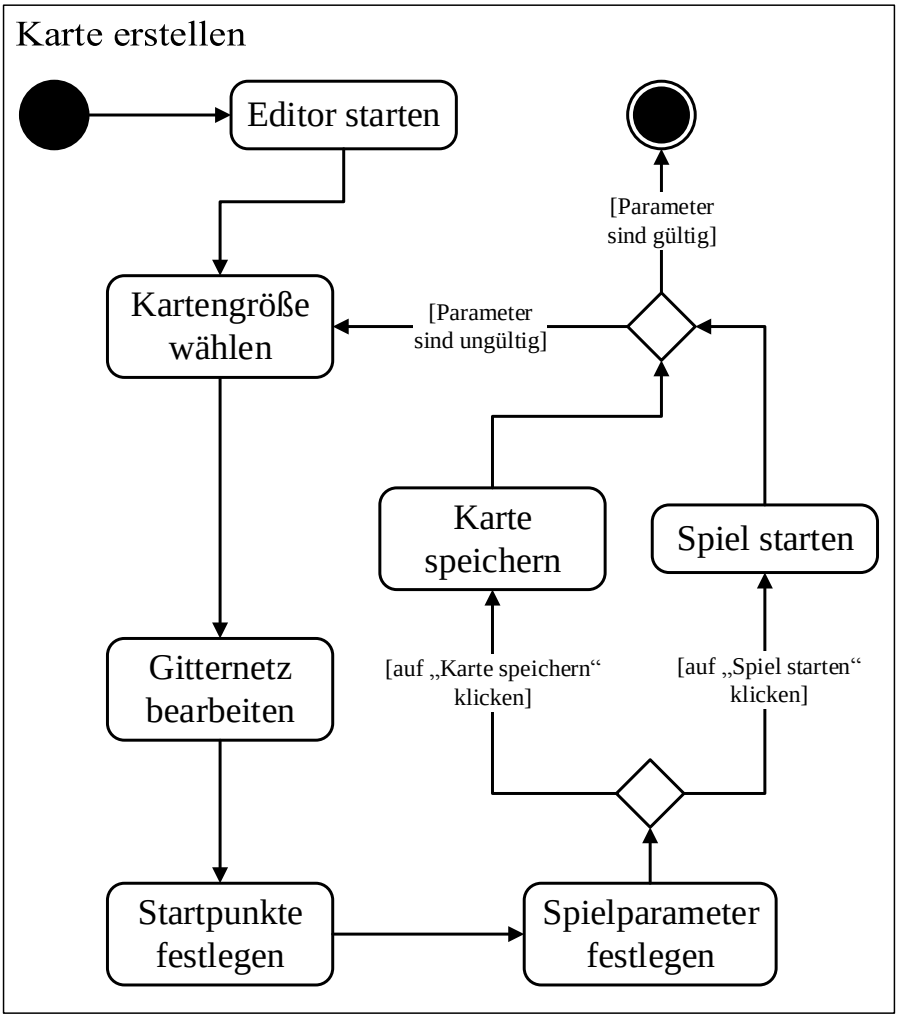

Abbildung 3.3: Aktivitätsdiagramm *Karte erstellen*

Dieses Aktivitätsdiagramm veranschaulicht den Anwendungsfall *Karte erstellen*. Zunächst muss das Programm gestarten werden. Dabei öffnet sich als erstes automatisch der Karteneditor. Hier kann man die Spielkarte bearbeiten. Dazu muss zunächst die Kartengröße gewählt werden. Entsprechend der Benutzerangabe der Kartengröße wird ein Gitternetz im Editor erstellt. Dieses Gitternetz kann nun vom Benutzer mit Hindernissen versehen und die Startpunkte für die Roboter und Flaggen gesetzt werden. Zuletzt muss der Benutzer noch folgende Spielparameter festlegen: Anzahl der Leben, Sichtweite, Schussweite, Punkte zum Sieg und Roboter-Aussetzzeit (nach Tod). Danach hat der Benutzer die Möglichkeit die Karte zu speichern oder mit der gerade erstellen Karte ein Spiel zu starten.

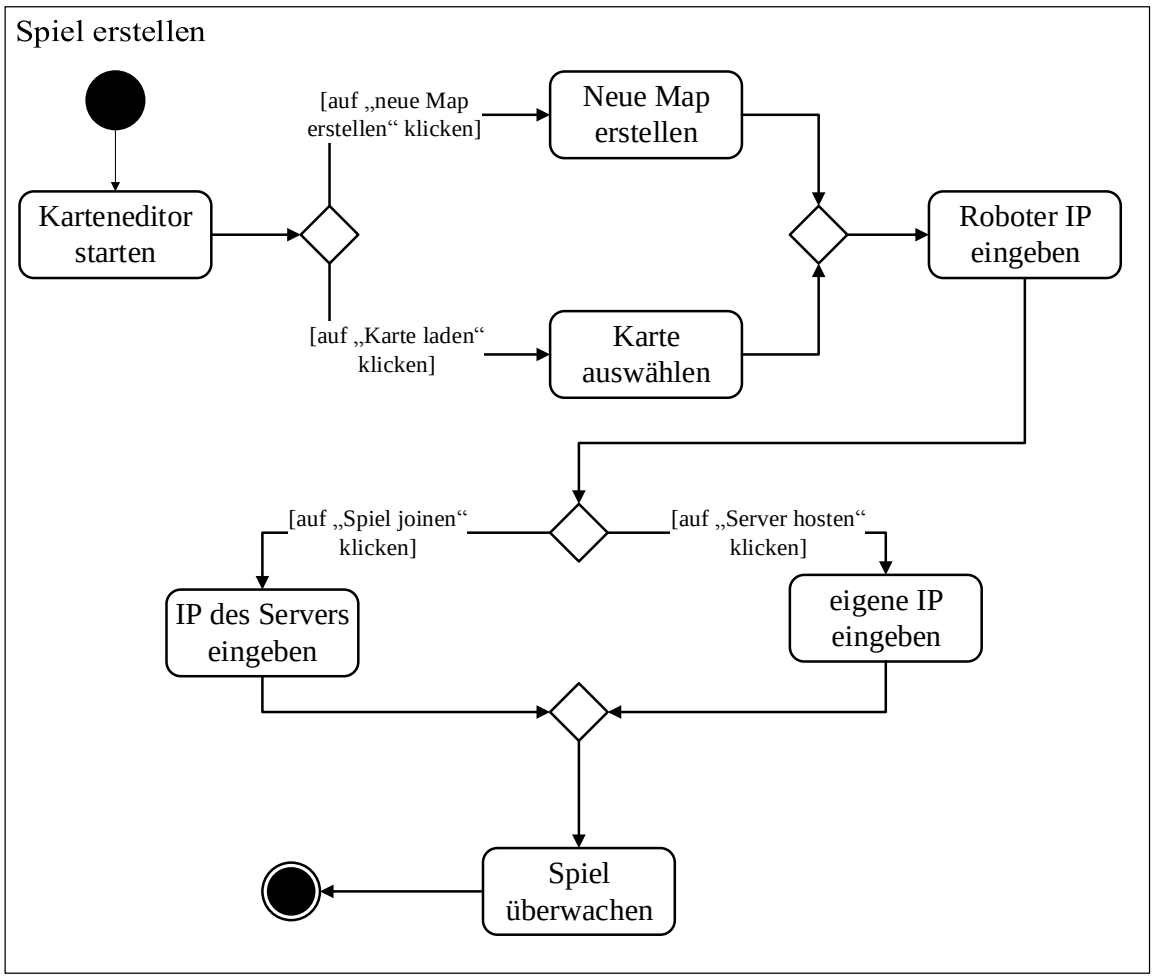

Abbildung 3.4: Aktivitätsdiagramm *Spiel erstellen*

In Abbildung 3.4 wird beschrieben, wie der Benutzer ein Spiel erstellen kann. Zunächst muss der Karteneditor gestartet werden. Danach hat der Benutzer die Möglichkeit eine neue Karte zu erstellen oder eine bereits vorhandene zu laden. Ist eine Karte erstellt oder geladen, gibt man die IP-Adressen für beide Roboter des eigenen Teams ein. Danach hat man die Wahl, ob man ein einen Server bereitstellt oder einem bestehenden Spiel beitritt. Nach der Eingabe der eigenen IP-Adresse, um einen Server bereitzustellen oder der Eingabe der IP-Adresse eines bestehenden Servers, hat man die Möglichkeit das Spiel zu überwachen.

4 Produktfunktionen

Im folgenden Abschnitt wird die funktionale Beschreibung des Produkts konkretisiert.

Karte erstellen $\langle F10 \rangle$

Anwendungsfall: manuell Karte erstellen

Anforderung: RM13, RM14, RM4, RM15

Ziel: Erstellung einer spielbaren Karte

Vorbedingung: Der Karteneditor wurde erfolgreich gestartet.

Nachbedingung Erfolg: Karte wird korrekt gespeichert, die Spielsteuerung wird gestartet und lädt die Karte.

Nachbedingung Fehlschlag: Durch nicht zulässige Parametereingaben im Karteneditor kommt es zu einer Fehlermeldung.

Akteure: Benutzer

Auslösendes Ereignis: Der Benutzer startet den Karteneditor oder hat eine Karte in den Karteneditor geladen.

Beschreibung:

1. Kartengröße eingeben (falls eine neue Karte erstellt wird)
2. Gitternetz bearbeiten
3. Spielparameter festlegen
   - Sicht
   - Schussweite
   - Startpunkte (Roboter und Flagge)
   - Punkte zum Sieg
   - Anzahl der Leben
4. Falls gewünscht: Karte speichern

Karte speichern $\langle F20 \rangle$

Anwendungsfall: Erstellte Karte speichern.

Anforderung: RS2

Ziel: Speichern einer erstellten Karte.

Vorbedingung: Eine Karte wurde erstellt.

Nachbedingung Erfolg: Karte wird korrekt gespeichert.

Nachbedingung Fehlschlag: Durch nicht zulässige Parametereingaben im Karteneditor kommt es zu einer Fehlermeldung und die Karte wird nicht gespeichert.

Akteure: Benutzer

Auslösendes Ereignis: Benutzer wählt im Karteneditor „Karte speichern“ aus.

Beschreibung:

1. Speicherort im Dateisystem wählen
2. Name der Karte eingeben, ggf. vorhandene Karte überschreiben
3. Speichern bestätigen

Karte laden $\langle F30 \rangle$

Anwendungsfall: Karte laden

Anforderung: RS3

Ziel: Laden einer gespeicherten Karte.

Vorbedingung: Der Karteneditor wurde erfolgreich gestartet.

Nachbedingung Erfolg: Die Karte und die Parameter werden im Karteneditor angezeigt.

Nachbedingung Fehlschlag: Beim Laden einer Datei, welche nicht dem geforderten Format entspricht, lädt der Karteneditor keine Karte und gibt eine Fehlermeldung aus.

Akteure: Benutzer

Auslösendes Ereignis: Benutzer wählt im Karteneditor „Karte laden“.

Beschreibung:

1. Kartendatei im Dateisystem suchen und auswählen
2. Karte in den Editor laden

Server bereitstellen $\langle F40 \rangle$

Anwendungsfall: Server wird bereitgestellt

Anforderung: RM20

Ziel: Server bereitstellen

Vorbedingung: Die Roboter sind eingeschaltet, verbunden mit dem Netzwerk und die Spielkarte wurde initialisiert.

Nachbedingung Erfolg: Der Server wird gestartet, verbindet sich mit den eigenen Robotern und wartet auf die Anfrage eines Clients.

Nachbedingung Fehlschlag: Der Server wurde nicht gestartet und eine Fehlermeldung erscheint.
Der Server muss neu gestartet werden.

Akteure: Benutzer

Auslösendes Ereignis: Ein Benutzer wählt „Server bereitstellen“ aus.

Beschreibung:

1. IP-Adressen der Roboter eingeben
2. Server wird gestartet
3. Server verbindet sich mit den Robotern

Server beitreten $\langle F50 \rangle$

Anwendungsfall: Bestehendem Server beitreten.

Anforderung: RM21

Ziel: Als Client mit einem bestehenden Server verbinden.

Vorbedingung: Die Roboter sind eingeschaltet, verbunden mit dem Netzwerk und die Spielkarte wurde initialisiert.

Nachbedingung Erfolg:

1. Die Verbindungen mit den Robotern und dem Server sind hergestellt

Nachbedingung Fehlschlag: Die Verbindung mit den Robotern oder dem Server ist Fehlgeschlagen und eine Fehlermeldung erscheint.

Akteure: Benutzer

Auslösendes Ereignis: Ein Benutzer wählt „Server beitreten“ aus.

Beschreibung:

1. IP-Adressen der Roboter eingeben
2. Zielserveradresse eingeben
3. Verbindung mit Server

Darstellung des Spielgeschehens $\langle F60 \rangle$

Anwendungsfall: Spielverfolgung

Anforderung: RM16, RM18, RM19, RS3

Ziel: Eine digitale Darstellung des analogen Spielgeschehens.
Vorbedingung: Das Spiel muss gestartet sein.
Nachbedingung Erfolg: Spielgeschehen wird korrekt dargestellt.
Nachbedingung Fehlschlag: Das Spielgeschehen wird falsch dargestellt.
Akteure: Benutzer
Auslösendes Ereignis: Ein Benutzer startet den Roboter manuell.
Beschreibung:

1. Der Benutzer verfolgt das Spiel auf der Benutzeroberfläche.
   - Position der Roboter - Position der Flaggen - Anzahl der Leben - Anzahl der Punkte

Manuelle Spielsteuerung $\langle F70 \rangle$

Anwendungsfall: Manuelle Spielsteuerung
Anforderung: RM17, RS1
Ziel: Die Roboter können durch manuelle Eingaben gesteuert werden.
Vorbedingung: Das Spiel wurde gestartet.
Nachbedingung Erfolg: Manuelle Steuerungsbefehle werden korrekt ausgeführt.
Nachbedingung Fehlschlag: Manuelle Steuerungsbefehle werden nicht korrekt ausgeführt.
Akteure: Benutzer, Roboter
Auslösendes Ereignis: „Manuelle Steuerung“ in der Benutzeroberfläche auswählen.
Beschreibung:

1. Spielverfolgung
2. Roboter auswählen
3. Manuelle Steuerung wählen
4. Warten bis Zug beendet ist
5. Manuelle Steuerung bedienen
   - Oben
   - Unten
   - Links
   - Rechts
   -Schießen
6. Manuelle Steuerung aus

Eigener Server empfängt Datenpakete vom gegnerischen Server $\langle F80 \rangle$

Anwendungsfall: Nach Beendigung des gegnerischen Spielzuges werden dem eigenen Server Spieldaten übermittelt.

Anforderung: RM9

Ziel: Die Spieldarstellung wird aktualisiert und die eigene KI kann den nächsten Zug berechnen.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Das Spiel wurde korrekt aktualisiert.

Nachbedingung Fehlschlag: Das Spiel wurde nicht korrekt aktualisiert.

Akteure: Eigener Server, gegnerischer Server

Auslösendes Ereignis: Gegnerischer Server sendet Datenpakete.

Beschreibung:

1. Die vom gegnerischen Server gesendete Datenpakete werden empfangen
2. Die Spieldaten werden der Spielsteuerung und der KI übermittelt

Eigener Server sendet Datenpakete zum gegnerischen Server $\langle F90 \rangle$

Anwendungsfall: Nach Beendigung des eigenen Spielzuges werden dem gegnerischen Server Spieldaten übermittelt.

Anforderung: RM8

Ziel: Die gegnerische Spieldarstellung wird aktualisiert und das gegnerische Team kann seinen Zug ausführen.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Bestätigung des gegnerischen Servers.

Nachbedingung Fehlschlag: Fehlermeldung.

Akteure: Eigener Server, gegnerischer Server

Auslösendes Ereignis: Beendigung des eigenen Spielzuges.

Beschreibung:

1. Die Spieldaten des eigenen Zuges werden dem gegnerischen Server übermittelt
2. Der eigene Server wartet auf eine Bestätigung des gegnerischen Servers

Befehle an Roboter senden $\langle F100 \rangle$

Anwendungsfall: Server übermittelt Anweisungen an den Roboter.

Anforderung: RM6

Ziel: Der Roboter empfängt und befolgt die Anweisungen.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Roboter empfängt die Anweisungen.

Nachbedingung Fehlschlag: Roboter empfängt die Anweisungen nicht bzw. unvollständig.

Akteure: Server, Roboter

Auslösendes Ereignis: Die KI berechnet die Anweisungen für den Roboter.

Beschreibung: Die in der KI errechneten Anweisungen werden vom Server an den Roboter übermittelt.

Roboter führt Befehle aus $\langle F110 \rangle$

Anwendungsfall: Spielzug durchführen

Anforderung: RM1, RM2, RM5, RS4, RS5, RS6

Ziel: Der Roboter führt Aktionen auf dem Spielfeld aus.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Roboter führt Aktionen korrekt aus.

Nachbedingung Fehlschlag: Roboter führt Aktionen nicht korrekt aus.

Akteure: Roboter

Auslösendes Ereignis: Erhalt der Anweisungen vom Server.

Beschreibung:

1. Der Roboter führt folgende Aktionen aus:
   - vorwärts fahren
   - links drehen
   - rechts drehen
   - schießen
   - LED Status
   - LCD Status

Bestätigung an Server senden $\langle F120 \rangle$

Anwendungsfall: Roboter übermittelt nach Ausführung der Aktion eine Bestätigung an den Server.

Anforderung: RM7

Ziel: Der Server soll eine Bestätigung nach der ausgeführten Aktion vom Roboter erhalten.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Die Bestätigung wurde dem Server übermittelt.

Nachbedingung Fehlschlag: Fehlermeldung wird ausgegeben.

Akteure: Server, Roboter

Auslösendes Ereignis: Der Roboter hat die Aktion ausgeführt.

Beschreibung: Der Roboter sendet eine Bestätigung an den Server.

KI berechnet Befehle $\langle F130 \rangle$

Anwendungsfall: Anweisungen und Spielzüge berechnen.

Anforderung: RM10, RM11, RM12

Ziel: Die KI berechnet Anweisungen für die Roboter.

Vorbedingung: Das Spiel muss gestartet sein.

Nachbedingung Erfolg: Anweisungen sind erstellt.

Nachbedingung Fehlschlag: Fehlermeldung wird ausgegeben.

Akteure: Server

Auslösendes Ereignis: Der eigene Server hat Spieldaten vom gegnerischen Server erhalten

Beschreibung:

1. KI analysiert Spieldaten
2. KI errechnet Spielzug
3. KI erstellt Anweisungen für die Roboter

Roboter mit Server verbinden $\langle F140 \rangle$

Anwendungsfall: Roboter initialisieren

Anforderung: RM22

Ziel: Eine Verbindung zwischen Server und Robotern herstellen, so dass diese Befehle erhalten können.

Vorbedingung: Die Roboter sind eingeschaltet und verbunden mit einem lokalen Netzwerk.

Nachbedingung Erfolg: Erfolgreiche Verbindung des Servers mit den Robotern wird durch eine Meldung in der GUI bestätigt.

Nachbedingung Fehlschlag: Ausgabe einer Fehlermeldung in der GUI.

Akteure: Server, Roboter

Auslösendes Ereignis: Der Benutzer gibt die IP-Adressen der Roboter in der GUI ein und und drückt auf verbinden.

Beschreibung:

1. Initialisierungspakete werden an die angegeben IP-Adressen gesendet
2. Antwortpakete werden auf dem Server empfangen

5 Produktdaten

Im Folgenden werden die Daten, welche langfristig zu speichern sind, genannt.

Kartendaten $\langle D10 \rangle$
Daten der Karte (max. 100)
- Kartengröße
- Startpunkte der Flaggen
- Startpunkte der Roboter
- Wege
- Hindernisse

6 Nichtfunktionale Anforderungen

In diesem Kapitel werden Qualitätsmerkmale des Produkts aus den Bereichen „Funktionalität“, „Sicherheit“ und „Benutzbarkeit“ gewichtet. Anschließend werden zu den wichtigsten Qualitätsmerkmalen konkrete Produktanforderungen formuliert.

6.1 Funktionalität

In der folgenden Tabelle werden funktionelle Qualitätsmerkmale gewichtet.

Produktqualität	**sehr gut**	**gut**	**normal**	**nicht relevant**
Angemessenheit		x		
Richtigkeit	x			
Interoperabilität			x	
Ordnungsmäßgkeit				x

6.2 Sicherheit

In der folgenden Tabelle werden Qualitätsmerkmale hinsichtlich der Sicherheit des Produkts gewichtet.

Produktqualität	**sehr gut**	**gut**	**normal**	**nicht relevant**
Zuverlässigkeit		x		
Reife			x	
Fehlertoleranz			x	
Wiederherstellbarkeit			x	

6.3 Benutzbarkeit

In der folgenden Tabelle werden Qualitätsmerkmale hinsichtlich der Benutzbarkeit des Produkts gewichtet.

Produktqualität	sehr gut	gut	normal	nicht relevant
Verständlichkeit		x		
Erlernbarkeit		x		
Bedienbarkeit	x			
Effizienz			x	
Zeitverhalten			x	
Verbrauchsverhalten			x	

6.4 Änderbarkeit

Produktqualität	sehr gut	gut	normal	nicht relevant
Analysierbarkeit				x
Modifizierbarkeit	x			
Stabilität			x	
Prüfbarkeit			x	
Übertragbarkeit			x	
Anpassbarkeit				x
Installierbarkeit		x		
Konformität				x
Austauschbarkeit			x	

6.5 Qualitätsanforderungen

Hier werden zu den wichtigsten Qualitätsmerkmalen konkrete Produktanforderungen formuliert.

- $\langle Q10 \rangle$ Die von der Roboter-KI ausgeführten Züge sollen insoweit korrekt sein, dass die Roboter regelkonform auf dem Spielfeld agieren.

- $\langle Q20 \rangle$ Abgesehen von grundsätzlich korrekt ausgeführten Spielzügen, sollen sich die Roboter hinreichend präzise auf den Spielfeldern bewegen. Bewegungsungenauigkeiten sollten sich in einem Rahmen bewegen, dass sich die Roboter zum einen tatsächlich auf dem Spielfeld befinden, welches auch in der Steuerungsanwendung angezeigt wird, und zum anderen, dass für den Benutzer nicht die Notwendigkeit entsteht den Roboter per Hand zu repositionieren, um künftige Fehlzüge aufgrund von Ungenauigkeiten in der aktuellen Ausgangsposition zu vermeiden.

- ⟨$Q30$⟩ Die Robotersteuerungsanwendung, das Erstellen sowie das Beitreten eines Servers und der Karteneditor sollen für den Benutzer intuitiv verständlich sein.

- ⟨$Q40$⟩ Der Quellcode soll so designt sein, dass nachträgliche Modifizierungen an der KI möglich sind.

- ⟨$Q50$⟩ Die Roboter-Steuerungsanwendung, die Server-Anwendung sowie der Karteneditor sollen plattformunabhängig sein.

7 Benutzeroberfläche/Schnittstellen

In diesem Abschnitt wird die Benutzeroberfläche/Schnittstelle beschrieben.

Benutzeroberfläche:

Die Anwendung wird über den Karteneditor gestartet. Beim Start kann man eine alte Karte laden oder eine neues Grid erstellen. Danach können weitere Spielparameter eingestellt werden. Eine erstellte Karte kann gespeichert werden. Das Spiel wird über den Karteneditor gestartet, der Editor wird dabei beendet und die Spielsteuerung wird aufgerufen. Dabei erscheint ein Pop-up-Fenster, welches die IP der Roboter abfragt. Wenn das geschehen ist, wird der Benutzer wieder per Pop-up-Fenster gefragt, ob er einen Server erstellen oder sich mit einem Server verbinden möchte. Entscheidet sich der Benutzer einen Server zu erstellen, wird ihm seine IP-Adresse solange angezeigt, bis sich der andere Server verbunden hat. Möchte man einem anderen Server beitreten, so muss man die IP-Adresse in einem weiteren Pop-Up-Fenster eingeben. Ist das alles geschehen, kann man das Spiel mit der Spielsteuerung verfolgen und ggf. eingreifen.

Karteneditor $\langle UI10 \rangle$
Erlaubt das Bearbeiten des Spielfeldes. Kanten können per Maus-Click entfernt oder wieder eingefügt werden. Knoten ohne Kanten werden ebenfalls entfernt. Per ComboBox können Eigenschaften wie zum Beispiel Startpunkt Roboter 1 Team Blau, Startpunkt Flagge Rot, etc. ausgewählt und Hindernisse durch das Anklicken der jeweiligen Kante hinzugefügt werden. Spielparameter wie Leben, Schussweite,Höhe, Breite, etc. werden per Tastatureingabe im entsprechenden Feld bestimmt.

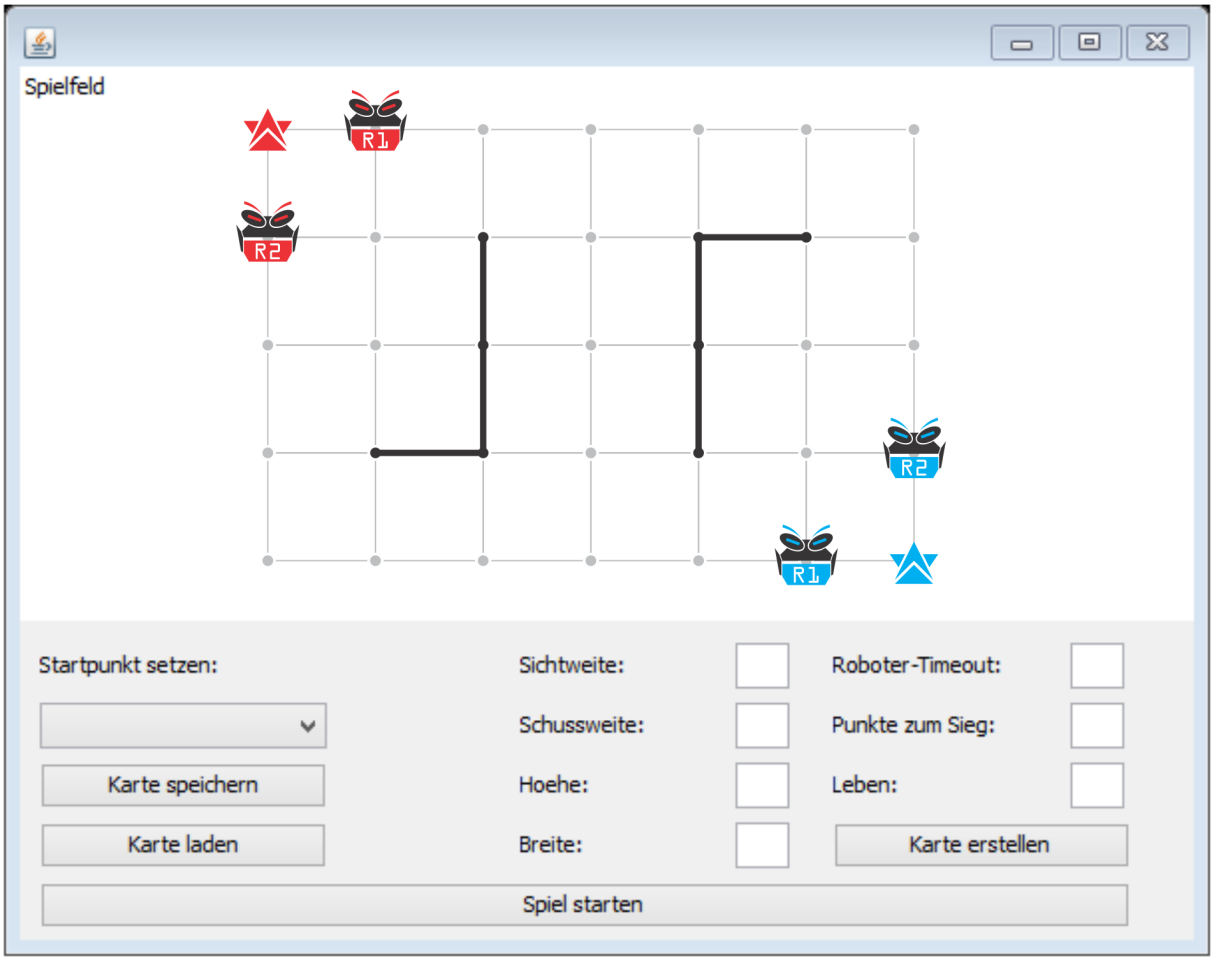

Abbildung 7.1: Karteneditor

Spielsteuerung ⟨*UI*40⟩
Lässt den Benutzer das Spielgeschehen auf dem Bildschirm verfolgen. Die Spielsteuerung erlaubt dem Benutzer die manuelle Steuerung der Roboter, solange der ToggleButton für die manuelle Steuerung aktiv ist. Per ComboBox lässt sich der zu steuernde Roboter auswählen. Mit den Aktionsbuttons "Oben", "Unten", "Links", "Rechts" und "Schießen" lässt sich der Roboter steuern. Diese können sowohl per Maus-Click als auch per Tastatur bedient werden. Wenn man seinen Zug beendet ertönt ein Signal.

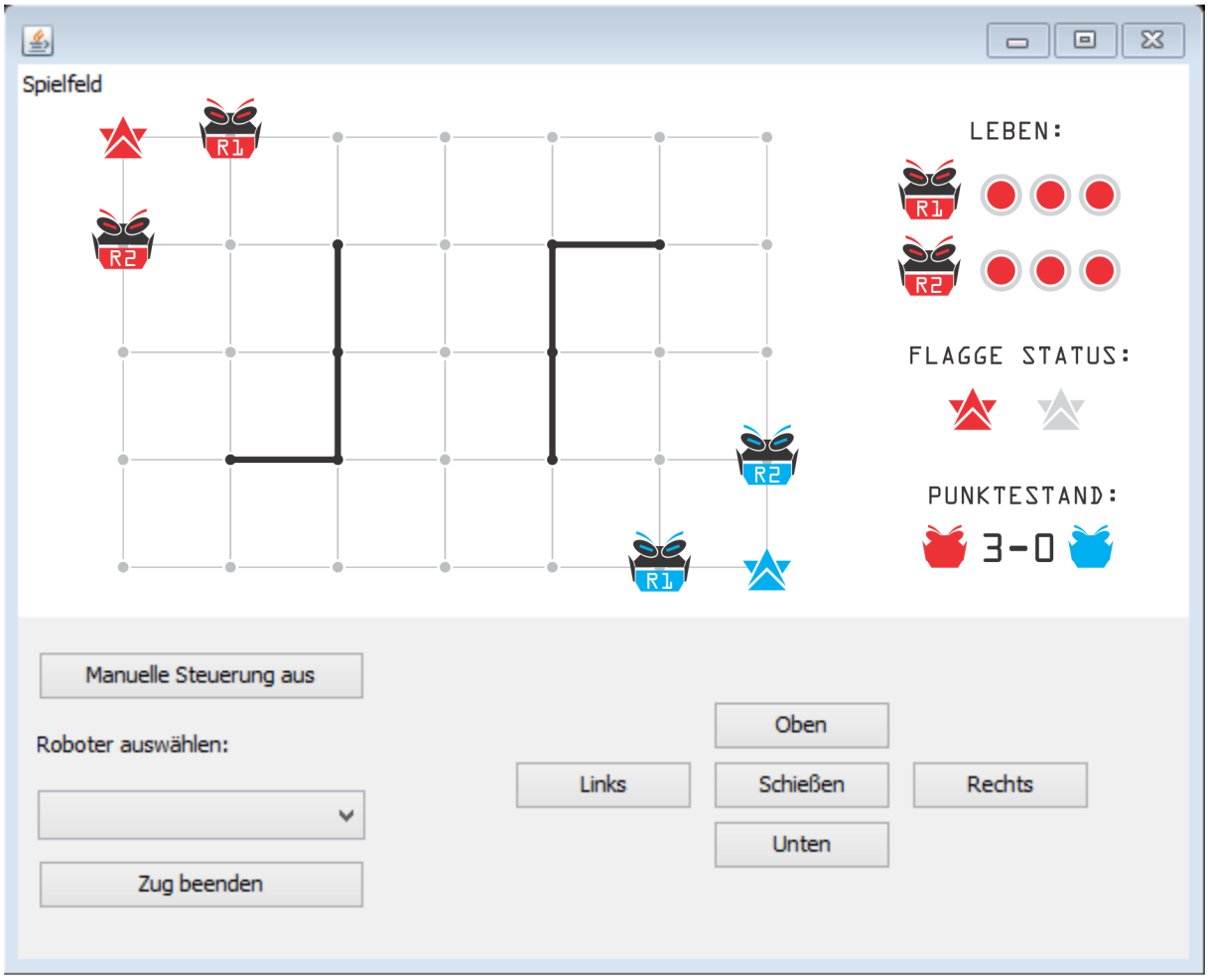

Abbildung 7.2: Spielsteuerung

Schnittstellen:
Eine Schnittstelle wird für die Kommunikation zwischen den Servern zur Spielverwaltung (Synchronisierung der Züge, Positionen, etc.) benötigt. Der Benutzer hat aber keine Interaktionsmöglichkeit außerhalb der Spielsteuerung.

8 Technische Produktumgebung

In diesem Kapitel wird die technische Umgebung des Produktes beschrieben.

8.1 Software

Die Robotersteuerungsanwendung, welche auch die Server-Komponente beinhaltet, sowie der Kartenedior werden in Java geschrieben und sind somit auf allen Betriebssystem, die eine Java-Laufzeitumgebung in der Version 7 unterstützen, lauffähig. Auf den Lego Mindstorms EV3 Robotern ist als Betriebsystem leJOS in der Version 0.9.0 beta installiert.

8.2 Hardware

Es kommen zwei Lego Mindstorm EV3 Roboter mit jeweils zwei NXT-Lichtsensoren und zwei Aktuatoren zum Einsatz. Für die Software-Komponenten wird ein Standard-PC, der die Minimalvoraussetzung für Java erfüllt, vorausgesetzt.

8.3 Produktschnittstellen

Die Robotersteuerungsanwendung enthält innerhalb der Serverkomponente eine Schnittstelle, die verwendet wird um entweder das gegnerische Team an der eigenen Spielsitzung teilnehmen zu lassen oder umgekehrt um sich mit einer existierenden Spielsitzung auf dem Server des gegerischen Teams zu verbinden.

9 Glossar

In diesem Abschnitt werden einige Fachbegriffe erläutert.

JDK - Java Development Kit Sammlung von Werkzeugen und Anwendungen, um eine Software zu erstellen

leJos Java - Betriebssystem für Lego Mindstorm Roboter

Server - Zentrales System, welches auf Verbindungen von Clients reagiert und z.B. Anfagen bearbeitet oder Dateien bereitstellt

API - application programming interface Programmierschnittstelle

JRE - Die Java Runtime Environment führt platformübergreifenden Java-Bytecode in nativen Code über und ermöglicht die Ausführung von Java-Programmen

GUI - Das Graphical User Inteface biete nutzern die Möglichkeit, Software über bspw. Knöpfe und Textfelder zu steuern

Spielsteuerung - Das GUI-Programm zur Visualisierung des Spiels und Steuerung der Roboter Karte Die Karte ist die logische Repräsentation des Spielfeldes im Computer, welches nach dem realen Vorbild durch einen Benutzer modelliert wird

K.I. - Die künstliche Intelligenz ist ein Teilgebiet der Informatik, welches sich mit der Automatisierung intelligenten Verhaltens befasst

Grid - Das Spielfeld in Form eines Gitters