Technische Universität Braunschweig

RoboSoccer - Road to France...

LeGoal

Software-Entwicklungspraktikum (SEP)
Sommersemester 2016

Fachentwurf

Auftraggeber
Technische Universität Braunschweig
Institut für Programmierung und Reaktive Systeme
Prof. Dr. Ursula Goltz
Mühlenpfordtstrasse 23
38106 Braunschweig

Betreuer: Sascha Lity, Ken Rieke

Auftragnehmer:

Name	E-Mail-Adresse
Sören Christmann	s.christmann@tu-braunschweig.de
Alexander Joost	a.joost@tu-braunschweig.de
Mohamad Karaki	mo.karaki@tu-braunschweig.de
Bengt Kensy	b.kensy@tu-braunschweig.de
Anna Lörke	a.loerke@tu-braunschweig.de

Braunschweig, 1. Juni 2016

Bearbeiterübersicht

Kapitel	**Autoren**
1	Mohamad Karaki
2	Anna Lörke Alexander Joost
3	Sören Christmann
4	Bengt Kensy
5	alle

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Das Ziel unseres Projekts ist die Entwicklung einer künstlichen Intelligenz für die „Lego Mindstorms EV3 - Roboter“, deren Aufgabe es ist, ein Fußballspiel gegen ein anderes Team zu gestalten. Dabei sollen die Roboter miteinander über Raspberry Pis kommunizieren und dadurch gemeinsam Entscheidungen für eine Spielstrategie anhand der Spielsituation treffen. Diese wird anhand Positionsdaten der Roboter und des Balls erkannt. Diese Positionsdaten werden an unsere Raspberry Pis mittels einer Kamera, welche sowohl die Roboter als auch den Ball erkennt, übermittelt. Zusätzlich steht eine graphische Benutzeroberfläche (GUI) zur Verfügung, welche dem Benutzer zum einen eine Visualisierung des Spielfeldes, zum anderen auch die Steuerung des Spiels ermöglicht. Daneben bietet unsere GUI die Möglichkeit das Spiel manuell zu steuern.

Die Roboter agieren dabei nach im Vorfeld festgelegten Regeln. Diese Regeln sind im erstem Abschnitt dieses Kapitels erläutert.

In diesem Dokument werden wir außerdem die Funktionen sowie ihre Zusammenhänge darstellen. Die Visualisierung unseres Fachentwurfs wird mittels Funktionsmodellen und Objektmodellen erfolgen, beispielsweise mit Aktivitätsdiagrammen, Sequenzdiagrammen und Klassendiagrammen. In diesem Kapitel wird dem Leser zusätzlich mitgeteilt, welche Vorgänge zum Starten des Spiels bzw. für die manuelle Steuerung durchgeführt werden müssen. Außerdem werden wir die Funktionen der künstlichen Intelligenz erläutern. Daneben werden die Produktfunktionen aus dem Pflichtenheft im zweiten Kapitel erläutert und mit Sequenzdiagrammen visualisiert. Ferner wird unser Datenmodell im dritten Kapitel durch ein Klassendiagramm dargestellt. Letztlich wird die Konfiguration des Netzwerks im vierten Kapitel erläutert.

Am wichtigsten ist es, dem Leser eine Übersicht des Systems zu geben. Dies ermöglicht das folgende Aktivitätsdiagramm:

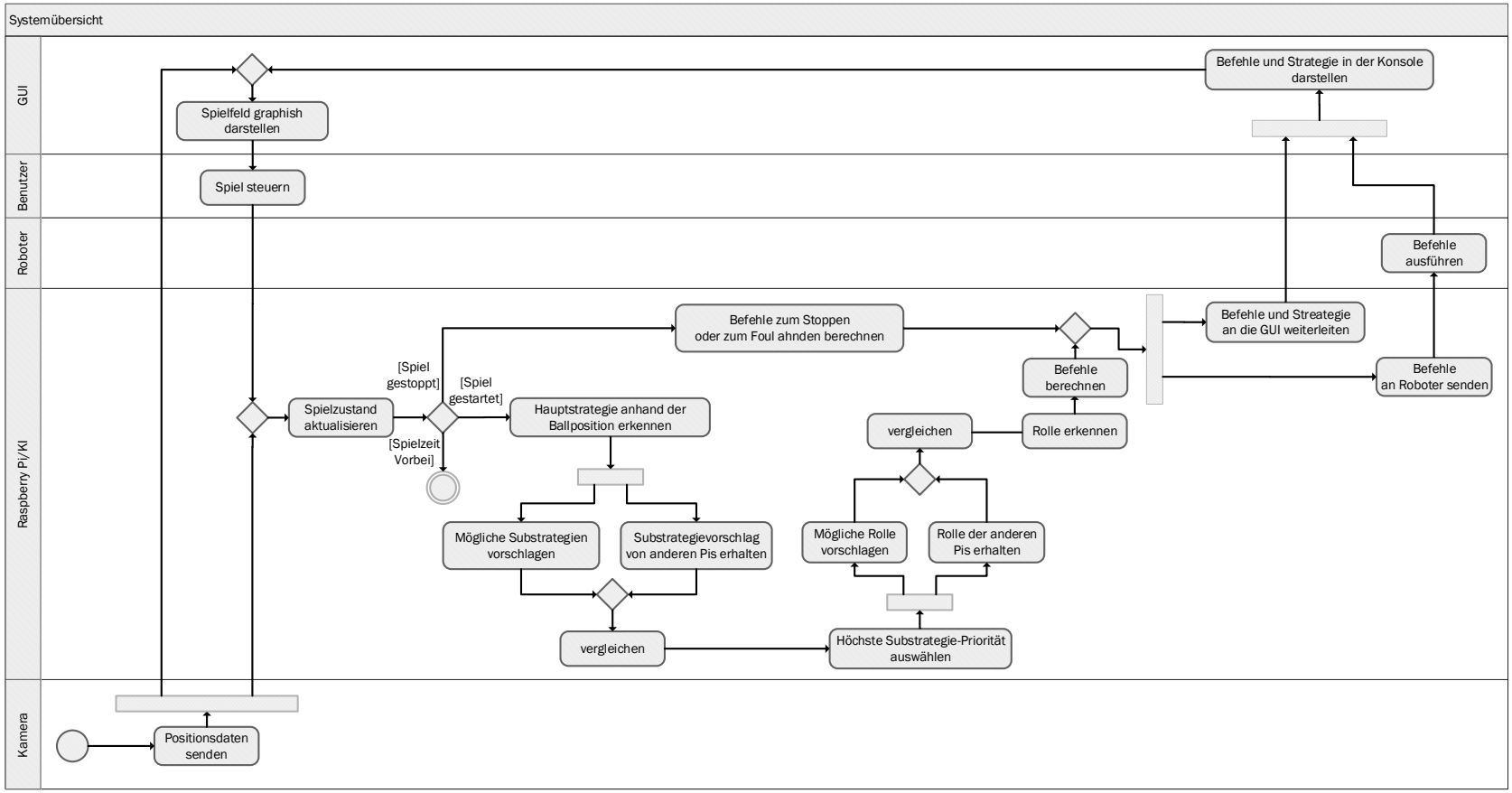

Abbildung 1.1: Aktivitätsdiagramm: *Systemübersicht*

Beschreibung zum Aktivitätsdiagramm *Systemübersicht*:

Dieses Aktivitätsdiagramm zeigt eine Übersicht der Funktionen des Systems, die während einer Runde durchgeführt werden. Zum Start schickt die Kamera die Positionsdaten an die GUI und an den Raspberry Pis. In der GUI wird das Spielfeld visualisiert. Danach kann der Benutzer das Spiel steuern. Bei der Steuerung des Spiels können mehrere Befehle an den Raspberry Pis gesendet werden. Diese sind „Starten“, „Stoppen“, „Foul ahnden“ und „Spiel neu starten“, welche genauer im Aktivitätsdiagramm *LeGoal Benutzeroberfläche* im Pflichtenheft erklärt wurden. Auf alle Befehle reagieren die Pis und sammeln dazu die Kameradaten. Anhand dieser Daten wird der Spielzustand aktualisiert. Wenn das Spiel gestartet ist, entscheidet sich die KI anhand der Ballposition für eine der drei Hauptstrategien, die wir in Abschnitt „Die künstliche Intelligenz“ erläutern werden. Danach schlägt der Raspberry Pi eine Substrategie vor und bekommt gleichzeitig Substrategievorschläge von den anderen Pis. Darüber hinaus werden Substrategievorschläge verglichen und für die Substrategie mit der höchsten Priorität entschieden. Nach einer Substrategieentscheidung müssen die Rolle der Roboter verteilt werden. Hierfür schlägt auch der Raspberry Pi eine Rolle vor und erhält dazu die Rollenvorschläge von den anderen Pis. Diese Rollenvorschläge werden verglichen, danach erkennt der Pi seine Rolle. Anhand dieser Analyse werden die Befehle berechnet und anschließend an den Roboter gesendet. Nachdem der Roboter die Befehle erhält, führt er diese aus. Die berechneten Befehle werden an die GUI für eine Darstellung der ausgeführten Befehle weitergeleitet. Anschließend werden die Befehle an die Roboter gesendet und von den Robotern ausgeführt. Nach der Ausführung und der Darstellung der Befehle startet eine neue Runde.

1.1 Projektdetails

Im folgenden Abschnitt werden die 16 mit dem Gegnerteam vereinbarten Spielregeln näher erläutert. Sie sollen dafür sorgen, dass beide Teams am Ende ein gemeinsames Spiel abhalten können.

1.1.1 Regeln

1. Das Spiel besteht aus zwei Halbzeiten. Jede Halbzeit dauert fünf Minuten.
2. Ein Zufallsgenerator entscheidet über die Startseiten der beiden Teams.
3. Die Roboter beider Teams stehen auf den im Vorfeld abgesprochenen, einheitlichen Startpositionen.
4. Für jeden Anstoß gibt es ein Vorrangteam: Hat ein Team Vorrang, so darf dieses sich mit einem Roboter innerhalb der ersten fünf Sekunden nach Spielbeginn frei bewegen. Alle anderen Roboter bleiben auf ihren jeweiligen Startpositionen. Nach Ablauf der fünf Sekunden dürfen sich alle Roboter frei bewegen. Die Sekundenzahl kann vor jedem Spiel in Absprache mit dem Gegnerteam verändert werden.
5. Zum Start entscheidet ein Zufallsgenerator welches der beiden Teams Vorrang hat.
6. Nach einem erfolgreichen Torschuss eines Teams wird dieser vom Schiedsrichter gewertet. Die Roboter setzen das Spiel an ihren jeweiligen Startpositionen fort, wobei das entsprechend andere Team Vorrang erhält.
7. Nach der ersten Halbzeit werden die Seiten gewechselt.
8. Ein Austausch der Roboter während des Spiels ist untersagt.
9. Alle Roboter müssen zeitgleich starten. Ausnahme ist hierbei ein Roboter des Vorrangteams, welcher bereits fünf Sekunden vor allen anderen startet.
10. Das Team, welches am Ende des Spiels mehr Tore erzielt hat, gewinnt.
11. Sollte am Ende des Spiels ein Gleichstand der Punkte herrschen, so ist dies ein gültiges Ergebnis.
12. Ein Foul kann nur dann entstehen, wenn sich zwei gegnerische Roboter rammen.

13. Sollte ein Roboter ein Foul ausüben, so sorgt der im Vorfeld von beiden Teams aus festgelegte Schiedsrichter für einen erneuten Spielstart. Hierbei erhält das gefoulte Team Vorrang.

14. Sollte der Foul-Verursacher nicht eindeutig zu erkennen sein, so wird das Vorrangteam für den erneuten Spielstart durch einen Zufallsgenerator bestimmt.

15. Gelangt der Ball durch die Aktionen eines Roboters in einen nicht erreichbaren Bereich, so müssen alle Roboter an ihre Startposition zurückkehren, sodass ein erneuter Anstoß- durchgeführt werden kann und Die . Vorrang hat hierbei das entsprechend andere Team.

16. Eine Manipulation des Spielfeldes, des Balls, der Roboter oder sonstigen zum Spiel gehörigen Dingen ist untersagt.

1.2 Vorgänge zum Starten des Spiels

In diesem Abschnitt werden die Vorgänge zum Starten des Spiels erläutert.

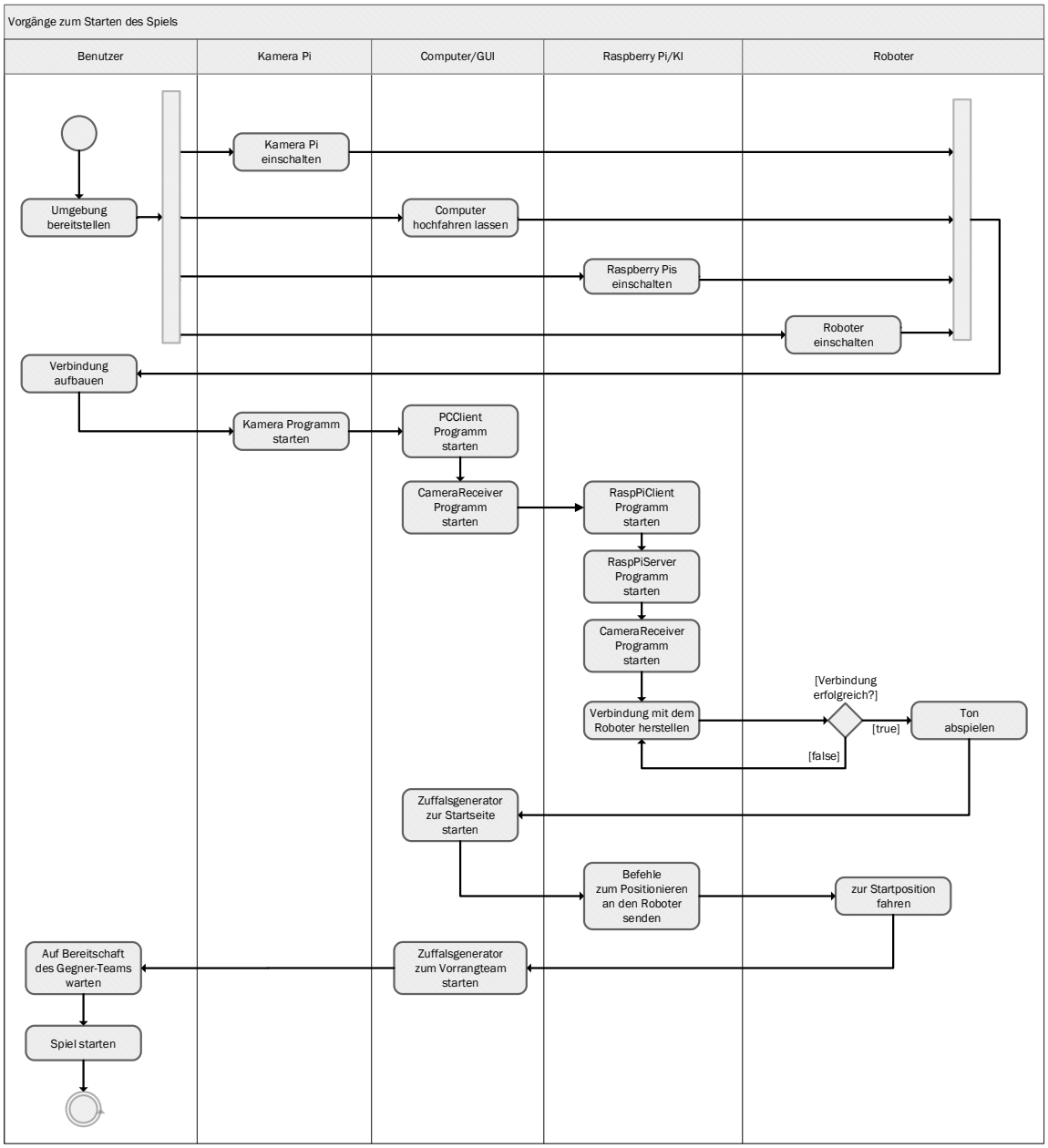

Abbildung 1.2: Aktivitätsdiagramm: *Vorgänge zum Starten des Spiels*

Beschreibung zum Aktivitätsdiagramm *Vorgänge zum Starten des Spiels*:

Dieses Aktivitätsdiagramm zeigt die Vorgänge, die vom Benutzer vor dem Spielbeginn durchgeführt werden müssen. In diesem Diagramm gibt es fünf Funktionen. Jede Funktion ist ein Teil des Systems. Um das Spiel zu starten, muss der Benutzer die Spielumgebung bereitstellen. Hierfür muss er den Kamera Pi, den Computer (GUI) und die Raspberry Pis sukzessive hochfahren lassen. Zuletzt müssen alle drei Roboter, die zu unserem Team gehören, gestartet werden. Nachdem alle Systemkomponenten gestartet sind, muss der Benutzer die zugehörige Software ausführen. Um die Kameradaten verteilen zu können, muss im Raspberry Pi der Kamera das Kameraprogramm gestartet werden. Daneben muss im Computer die GUI gestartet werden und die Verbindung mit den Raspberry Pis aufgebaut werden. Die Verbindung erfolgt dadurch, dass das PCClient Programm und das CameraReceiver Programm, deren Funktionen im Kapitel 4 erklärt sind, gestartet werden. Danach müssen in den Raspberry Pis die Programme RaspPiClient, RaspPiServer und CameraReceiver, deren Funktionen auch in Kapitel 4 erklärt sind, gestartet werden. Darüber hinaus soll die Verbindung zwischen den Pis und den zugehörigen Roboter durch ein Programm aus der Lejos-EV3-Bibliothek aufgebaut werden. Bei einer erfolgreichen Verbindung spielt der Roboter einen Ton ab. Zuletzt hat der Benutzer die Möglichkeit das Spiel zu starten aber er muss dafür auf die Bereitschaft des Gegner-Teams warten. Nach dem Spielstart können die Raspberry Pis durch die integrierte KI Befehle berechnen, welche an die Roboter gesendet und von den Robotern ausgeführt werden.

1.3 Vorgänge zum manuellen Steuern des Spiels

In diesem Abschnitt werden die Vorgänge zum manuellen Steuern des Spiels erläutert.

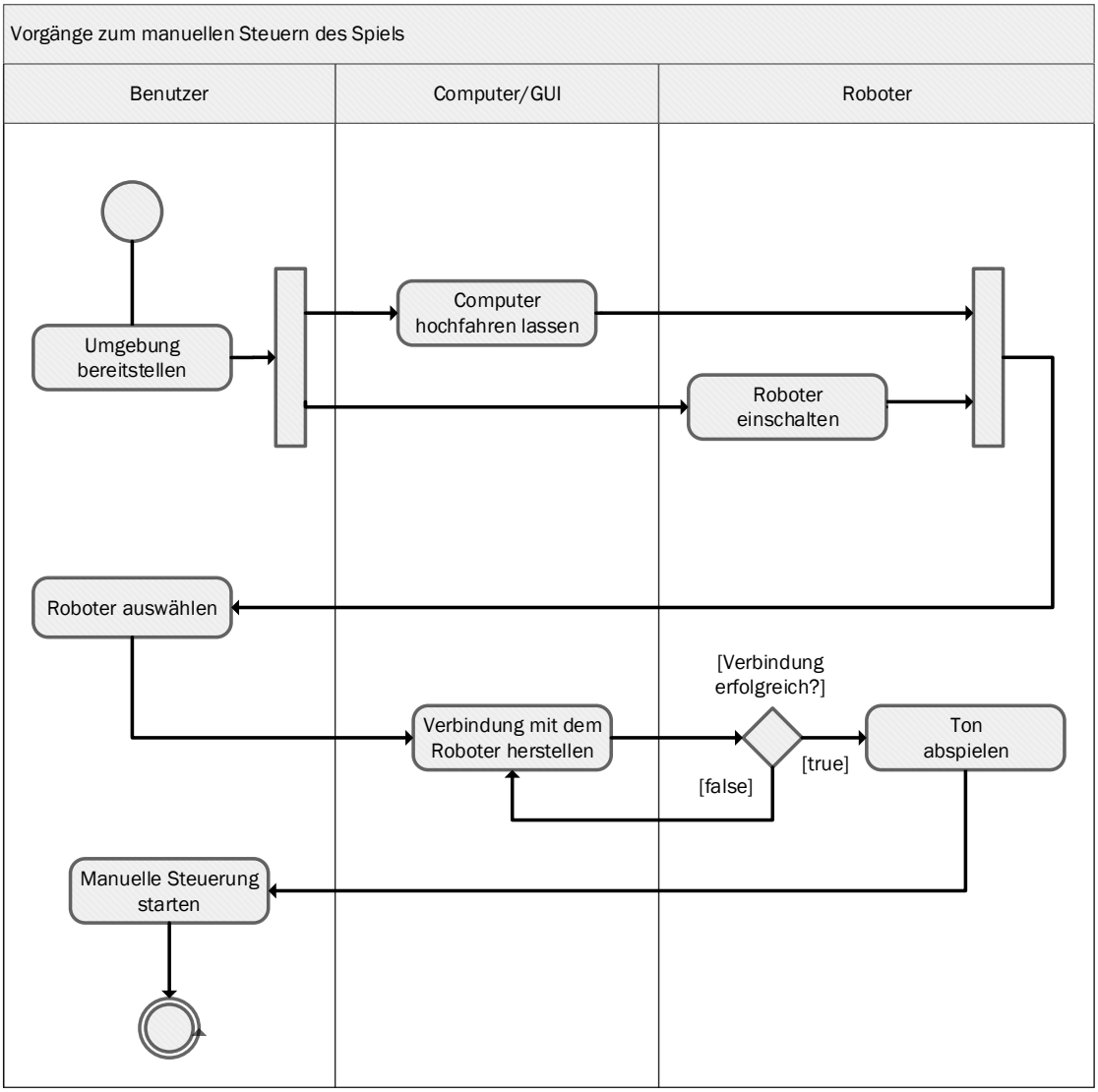

Abbildung 1.3: Aktivitätsdiagramm: *Vorgänge zum manuellen Steuern des Spiels*

Beschreibung zum Diagramm *Vorgänge zum manuellen Steuern des Spiels*:

Dieses Aktivitätsdiagramm zeigt die Vorgänge, die vom Benutzer vor dem Start der manuellen Steuerung durchgeführt werden müssen. In diesem Diagramm gibt es drei Funktionen. Jede Funktion ist ein Teil des Systems. Um die manuelle Steuerung zu starten, muss der Benutzer die Spielumgebung bereitstellen. Hierfür muss er den Computer und die Roboter hochfahren lassen. Nachdem der Computer und die Roboter gestartet sind, kann der Benutzer genau einen Roboter gleichzeitig steuern. Hierfür muss er einen Roboter auswählen. Außerdem soll die Verbindung zwischen dem Computer und dem Roboter durch ein Programm aus der Lejos-EV3-Bibliothek aufgebaut werden. Bei einer erfolgreichen Verbindung spielt der Roboter einen Ton ab. Zuletzt kann der Benutzer den Roboter manuell steuern und mithilfe der Tastaturtasten Befehle senden.

1.4 Die künstliche Intelligenz

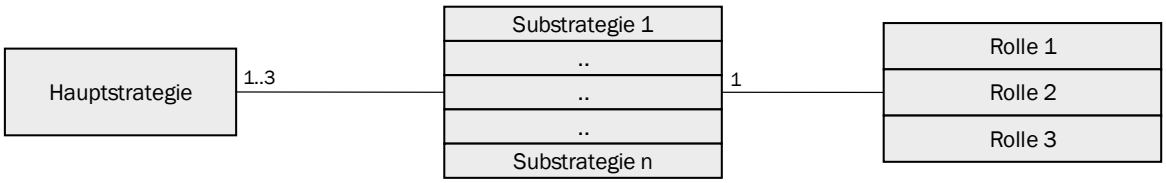

Abbildung 1.4: *Spielstrategie*

Dieses Diagramm zeigt, wie sich die künstliche Intelligenz der Raspberry Pis für eine Strategie entscheidet. Um die Künstliche Intelligenz zu entwickeln, haben wir uns für eine effiziente Methode zum Wechseln der Strategien anhand der Spielsituation entschieden. Es wird in unserem Spiel drei Hauptstrategien geben. Diese sind „ZUM BALL GEHEN“, „ANGRIFF“ und „ABWEHR“. Diese Strategien sind lediglich Hauptstrategien. Es wird sich anhand der Ballposition für eine von diesen entschieden. Die Funktionen der Hauptstrategien sind im Aktivitätsdiagramm *Roboter KI* visualisiert. Für jede Hauptstrategie wird es mehrere Substrategien geben. Die Substrategien haben Prioritäten. Es wird gemeinsam im Team anhand der Roboterposition für eine Substrategie mit der höchsten Priorität entschieden. Folgendes Szenario beschreibt wie alle drei Roboter ihre möglichen Substrategien vorschlagen und sich für die Substrategie mit der höchsten Priorität entscheiden. S3 hat in diesem Fall die höchste Priorität.

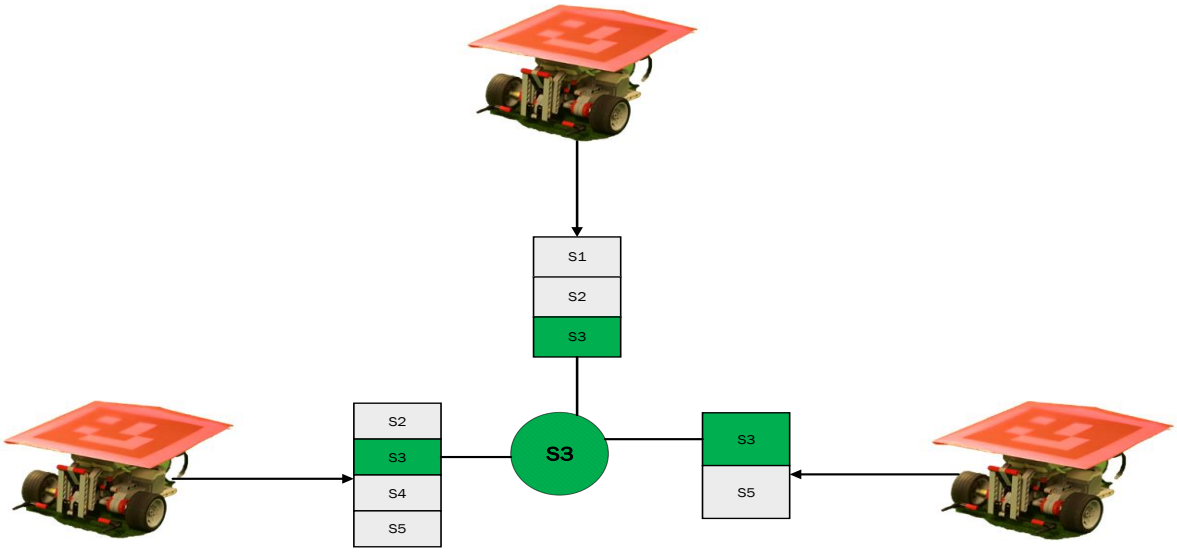

Abbildung 1.5: *Beispielszenario*

Die Prioritäten der Substrategien werden von unserer Gruppe festgelegt und dementsprechend implementiert. Außerdem hat jedes Roboter eine Rolle in der Substrategie. Die Rolle eines Roboters wird auch im Team bestimmt.

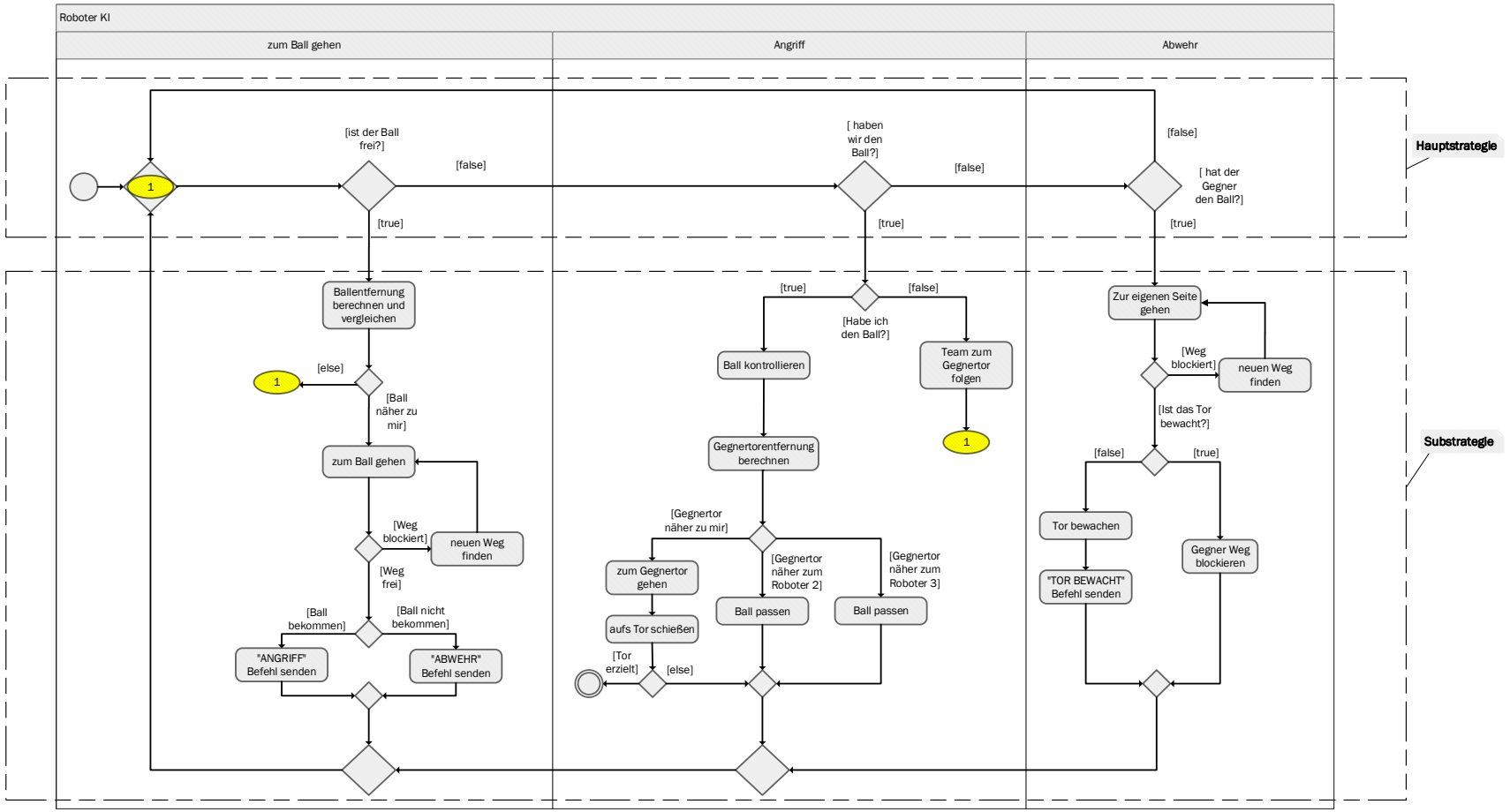

Abbildung 1.6: Aktivitätsdiagramm: *Roboter KI*

Die 1 Markierung hat die Aufgabe, die letzte Aktivität mit dem markierten Mergenode zu verbinden.

Beschreibung zum Aktivitätsdiagramm *Roboter KI*:

Dieses Aktivitätsdiagramm beschreibt die drei Hauptstrategien. Alle Strategien werden anhand der Ballposition im Spielfeld durchgeführt. Außerdem wird bei jeder Hauptstrategie eine Substrategie beschrieben. Bei einem freien Ball im Spielfeld liegt der Fokus der Roboter daran, den Ball zu kriegen. Deswegen müssen die Roboter zum Ball fahren. Allerdings wird sich nur ein Roboter gleichzeitig zum Ball bewegen, um Konflikte zu vermeiden. Beim Erlangen des Balls, sendet der Roboter den „ANGRIFF“ Befehl an die anderen Roboter. Beim missglückten Versuch den Ball zu erlangen, sendet der Roboter den „ABWEHR“ Befehl an die anderen Roboter. Beim Angriff, muss ein Roboter von unserem Team den Ball haben. Hierfür dürfen alle Roboter gleichzeitig in den gegnerischen Bereich des Spielfeldes fahren. Beim Verlieren des Balls oder wenn der Ball vom anderen Team besetzt ist, müssen die Roboter zum eigenen Bereich zurückfahren und im Abwehrzustand stehen. Daneben muss genau ein Roboter im Torhalbkreis stehen und das Tor bewachen.

2 Analyse der Produktfunktionen

In den nachfolgenden Abschnitten werden die Produktfunktionen des Pflichtenheftes näher ausgeführt und analysiert. Mittels Sequenzdiagrammen werden diese graphisch veranschaulicht.

Die folgende Skizze soll den allgemeinen, technischen Aufbau zwischen Kamera, den Raspberry Pis und dem Computer bzw. der GUI näher erläutern. Die Kamera sendet in kurzen Abständen Bilder an ihren Raspberry Pi (Kamera-Pi). Dieser nutzt die Bilder um Positionsdaten zu berechnen. Anschließend werden diese Daten an die Raspberry Pis der Roboter und an den Computer gesendet. Den Pis ist es möglich, untereinander zu kommunizieren, um eine schwarmbasierte, künstliche Intelligenz zu gewährleisten. Der Computer und die einzelnen Pis tauschen bezüglich der GUI, insbesondere der Darstellung der Befehle, Daten aus.

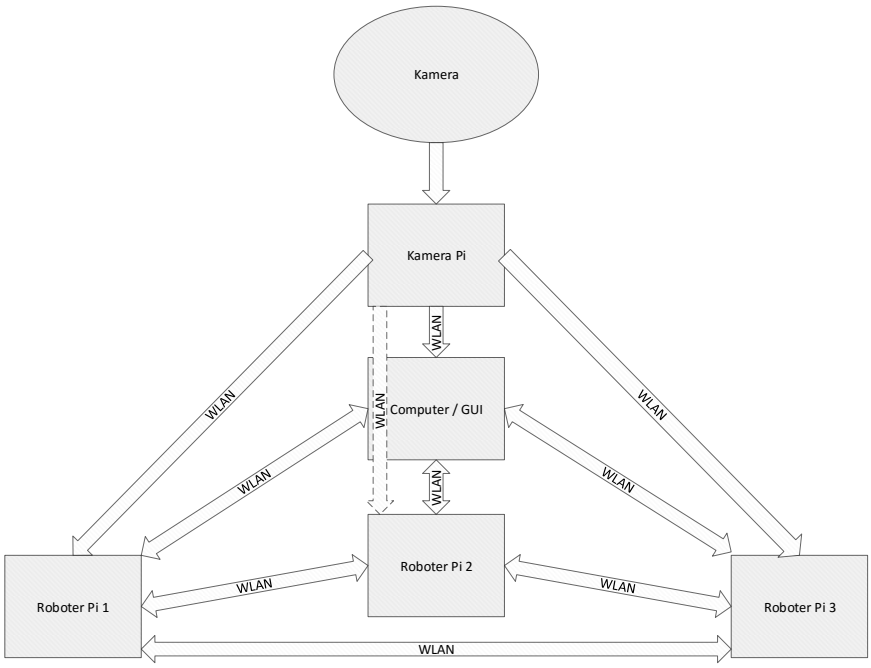

Abbildung 2.1: Skizze: *Überblick des Aufbaus*

2.1 Analyse von Funktionalität F10: Spiel starten

Die Funktion F10 gibt dem Nutzer die Möglichkeit das RoboSoccer-Spiel zu beginnen. Dafür wird der „Spiel starten“- Button auf der GUI gedrückt und diese zeigt dem Nutzer dann alle wichtigen Spieldaten an. Bevor die Daten jedoch angezeigt werden, wird intern eine Start-Nachricht von der GUI (dem Computer) an jeden der drei Raspberry Pis verschickt. Daraufhin wird die Information zur Startvorbereitung an den jeweils zugeordneten Roboter weitergeben und anschließend eine Bestätigung an die GUI zurückgegeben. Die Roboter selber sind nicht in der Lage Bestätigungen zu senden, führen jedoch die empfangenen Befehle zum Spielstart aus.

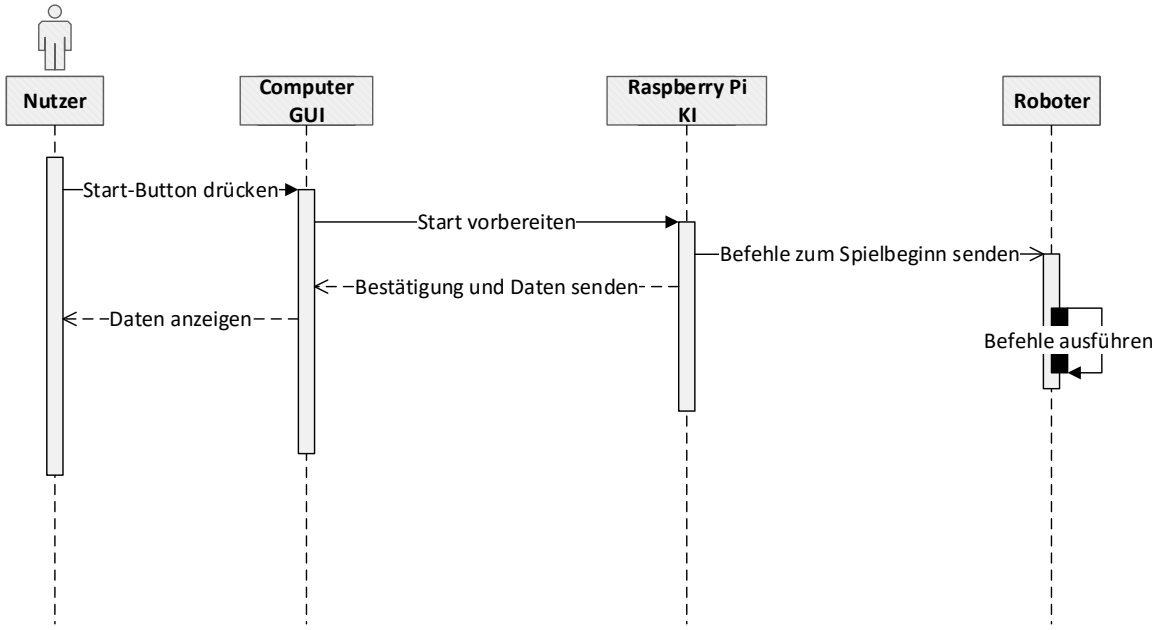

Abbildung 2.2: Sequenzdiagramm: *F10 - Spiel starten*

2.2 Analyse von Funktionalität F20: Spiel überwachen / Schiedsrichterfunktionen ausführen

Die Funktion F20 ermöglicht dem Spieler die Parameter des RoboSoccer-Spiels auf der GUI zu verfolgen und eventuell als Schiedsrichter einzugreifen. Mögliche Interaktionen sind dabei „Pausieren“, „Fortsetzen“, „Stoppen“, „Neustart“, „Foul“ und „Tor“ mit jeweiliger Teamauswahl. Dabei betreffen alle Aktionen jeweils sämtliche am Spiel teilnehmende Roboter und nicht nur die des LeGoal Teams, wobei die Datenverarbeitung im Raspberry Pi und die Kommunikation mit dem jeweiligen Roboter unterschiedlich gestaltet sein kann.

2.2.1 Analyse von Funktionalität F20a

Das folgende Sequenzdiagramm beruht auf der Annahme, dass das Spiel zuvor schon gestartet wurde. Drückt der Nutzer nun den „Pausieren“-Button in der GUI, so sendet der ausführende Computer eine Nachricht, dass das Pausieren ausgeführt werden soll, an die Raspberry Pis. Jeder Pi mit der darauf installierten KI empfängt diese Nachricht und sendet einen Befehl zum Pausieren an den jeweils zugehörigen Roboter. Nachdem der Roboter den Befehl ausführt hat, empfängt der Pi Positionsdaten vom Kamera-Pi und überprüft damit, ob wirklich alle Roboter stehen geblieben sind. Zu beachten ist dabei, dass der Kamera-Pi nahezu sekündlich Positionsdaten versendet und diese von den Raspberry Pis der Roboter und dem GUI-Computer entsprechend empfangen und ausgewertet werden. In sämtlichen Sequenzdiagrammen wurde aber, um die Übersichtlichkeit zu wahren, diese Verbindung nur dargestellt, wenn sie von Bedeutung für die auszuführende Funktion ist. Wenn alle Roboter stehen, sendet der Pi eine Bestätigung an die GUI zurück, die die aktualisierten Daten für den Nutzer anzeigt. Danach soll das Spiel fortgesetzt werden. Nach dem Auslösen des entsprechenden Buttons durch den User, passiert intern wieder nahezu das Gleiche wie schon beim Pausieren, mit dem Unterschied, dass ein „Fortsetzen“-Befehl im System weitergegeben wird und die Überprüfung der Positionsdaten ergeben sollte, dass die Roboter sich wieder bewegen. Die Schiedsrichterfunktion „Stoppen“, welche im Diagramm aus Gründen der Übersichtlichkeit nicht dargestellt ist, funktioniert analog zum „Pausieren“ und ist Voraussetzung für die „Foul“-Funktion, auf die im Folgenden noch eingegangen wird.

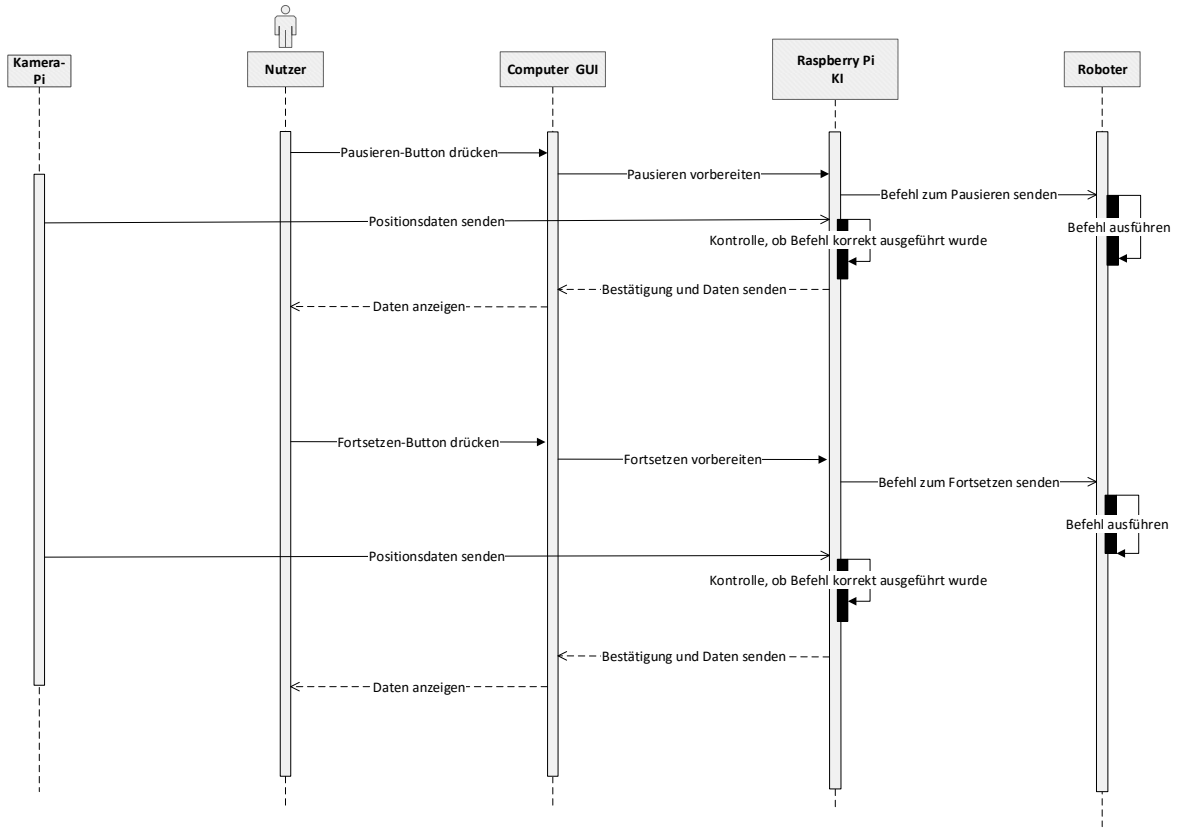

Abbildung 2.3: Sequenzdiagramm: *F20a - Spiel überwachen / Schiedsrichterfunktionen ausführen*

2.2.2 Analyse von Funktionalität F20b

Im weiteren Sequenzdiagramm wird das Neustarten des RoboSoccer-Spiels dargestellt. Nachdem der Nutzer den „Neustart“- Button in der GUI gedrückt hat, sendet der ausführende Computer eine entsprechende Nachricht an alle Raspberry Pis. Diese geben den „Startposition“-Befehl weiter an die Roboter, welche dann zur gespeicherten Startposition zurückfahren. Der Roboter-Pi erhält nach kurzer Zeit neue Positionsdaten vom Kamera-Pi und wertet diese aus, ebenso der Computer mit der GUI. Ergibt die Analyse der Positionen, dass alle Roboter auf ihre Startposition zurückgekehrt sind, wird vom Roboter-Pi eine Nachricht an die GUI versendet und diese zeigt die neuen Positionen dem Nutzer an. Ferner gibt der jeweilige Raspberry-Pi den Befehl zum Neustart, der aussagt, dass das Spiel von neuem beginnt, jedoch sonst keine Auswirkungen auf die Technik hat. Der Roboter empfängt den entsprechenden Befehl und führt ihn aus. Der GUI wird erneut eine Bestätigung übermittelt, die an den Nutzer weitergegeben wird.

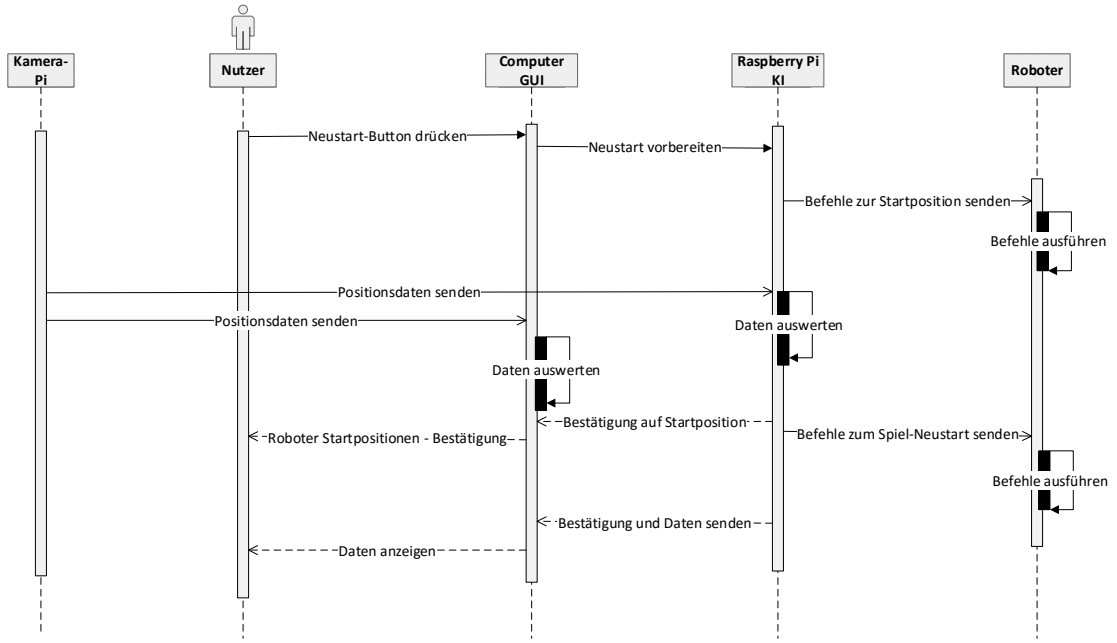

Abbildung 2.4: Sequenzdiagramm: *F20b - Spiel überwachen / Schiedsrichterfunktionen ausführen*

2.2.3 Analyse von Funktionalität F20c

Folgendes Sequenzdiagramm zeigt die Situation, wenn das gegnerische Team ein Tor geschossen hat. Zuerst drückt der Nutzer den „+1 Tor - Gegner “-Button in der LeGoal-GUI. Danach verschickt der dazugehörige Computer eine Meldung zum Stoppen und Anstoß vorbereiten, welchen der Raspberry Pi mit der KI empfängt. Nun sendet der Pi zuerst den schon bekannten „Stop“-Befehl an den Roboter, welcher diesen ausführt. Der Pi empfängt zwischenzeitlich neue Positionsdaten vom Kamera-Pi und überprüft damit, ob die Roboter wirklich alle zum Stehen gekommen sind. Anschließend übermittelt er den Befehl zur Startposition zurückzukehren. Zwischenzeitlich erhält der Raspberry Pi schon neue Positionsdaten vom Kamera-Pi, welche er verarbeitet und, falls alle Roboter die Startposition erreicht haben, senden die jeweiligen Pis den Befehl zum Anstoß an das LeGoal-Team. Auch der GUI-Computer empfängt Positionsdaten der Kamera und wertet diese zur späteren Darstellung für den Nutzer aus. Kurz darauf wird eine Bestätigung an den GUI-Computer übermittelt, welcher diese an den Nutzer weitergibt. Hat das LeGoal-Team ein Tor geschossen, sieht der Ablauf sehr ähnlich aus. Die Ausnahme bildet der Anstoß, welcher vom gegnerischen Team ausgeführt wird. Auch ein Foul wird systemintern ähnlich verarbeitet. Das Spiel ist zu dem Zeitpunkt schon gestoppt und es wir nur die Information wer gefoult hat weitergegeben. Daraufhin fahren die Roboter zu ihren Startpositionen und das Team, welches gefoult wurde, erhält Anstoß.

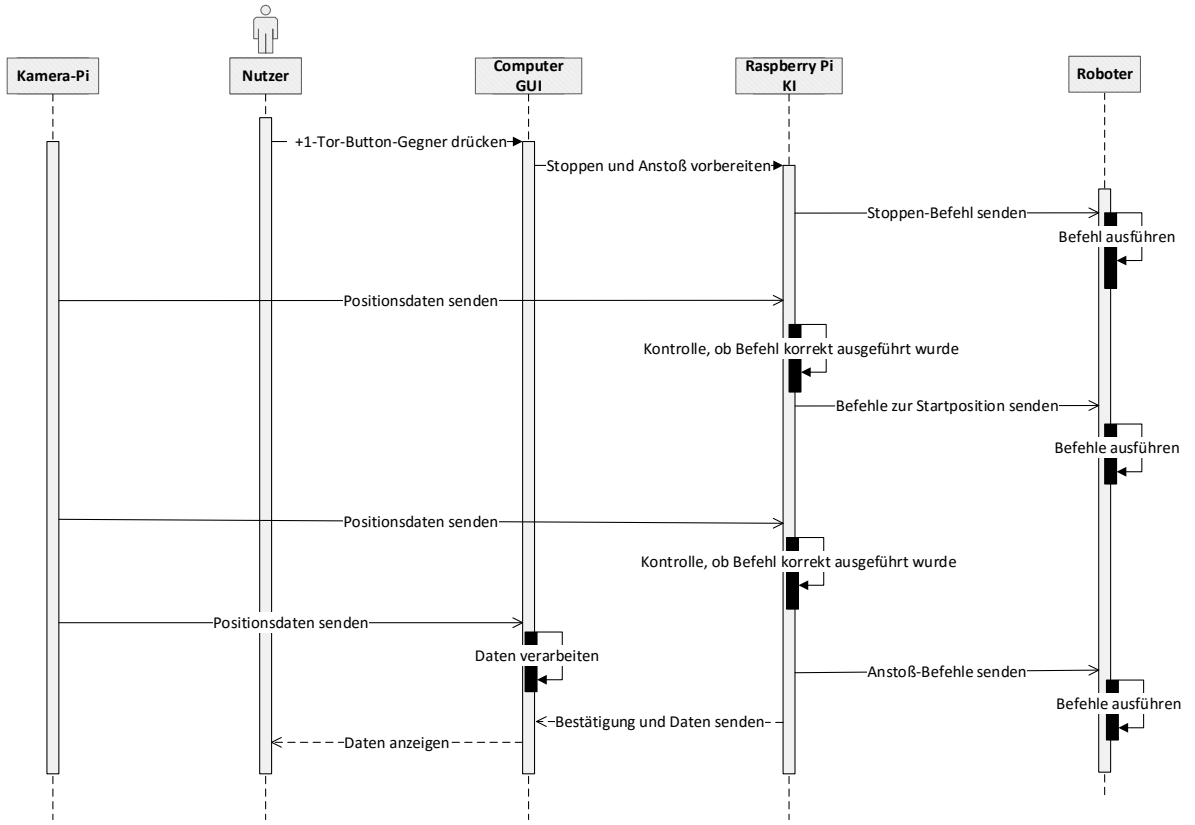

Abbildung 2.5: Sequenzdiagramm: *F20c - Spiel überwachen / Schiedsrichterfunktionen ausführen*

2.3 Analyse von Funktionalität F30: Spielfeld graphisch darstellen

Die Funktion F30 hat die Aufgabe das Spielfeld in der LeGoal-GUI darzustellen. Dafür werden vom Kamera-Pi über UDP die berechneten Positionsdaten an die GUI gesendet, welche diese dann verarbeitet und graphisch veranschaulicht.

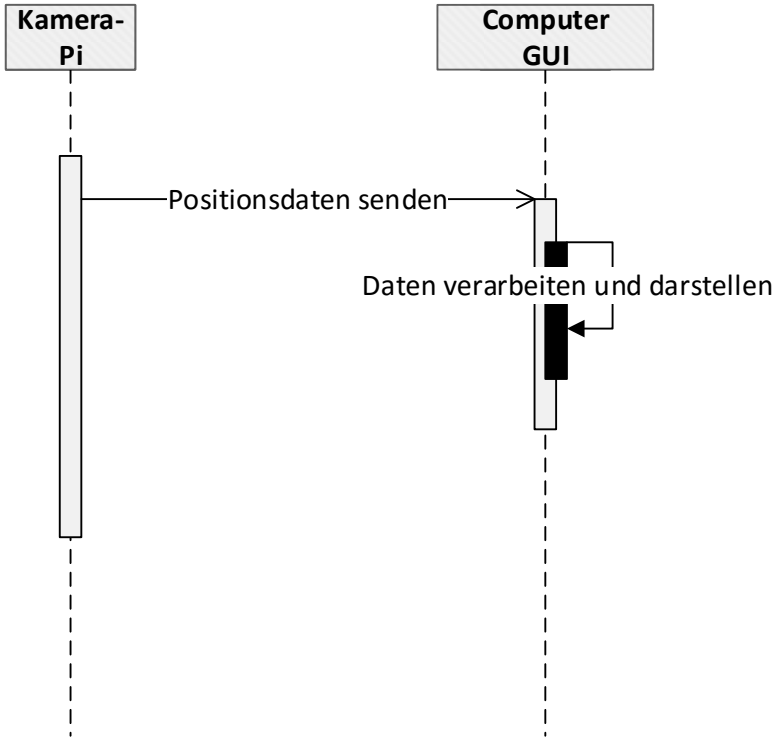

Abbildung 2.6: Sequenzdiagramm: *F30 - Spielfeld graphisch darstellen*

2.4 Analyse von Funktionalität F40 und F70: KI berechnet Befehle / Roboterinteraktion

Die Funktionen F40 und F70 werden gemeinsam betrachtet, da das Berechnen der Befehle in der KI eng mit der Roboterinteraktion zusammenhängt. Zuerst erhalten die Raspberry Pis der drei Roboter des LeGoal-Teams Positionsdaten vom Kamera-Pi. Diese werten sie jeweils aus. Anhand der Position des Balles entscheiden sie sich für eine Oberstrategie und jeder Roboter-Pi entscheidet sich dann aufgrund der eigenen Position für Substrategien. Diese teilt jeder Pi den anderen teameigenen Pis mit. Diese Kommunikation kann, anders als im Diagramm, auch zeitgleich geschehen. Die Substrategie mit der höchsten Priorität, welche von allen Roboter-Pis vorgeschlagen wurde, wird gewählt und die Pis senden entsprechende Bestätigungen untereinander. Danach erfolgt nach gleichem Muster eine Kommunikation über die Wahl der Rollen der einzelnen Roboter. Diese wurde aber, um die Übersichtlichkeit zu wahren, nicht dargestellt. Mithilfe der gewählten Strategie und Rolle berechnen die Pis schließlich die Befehle für die Roboter, die entsprechend übertragen werden.

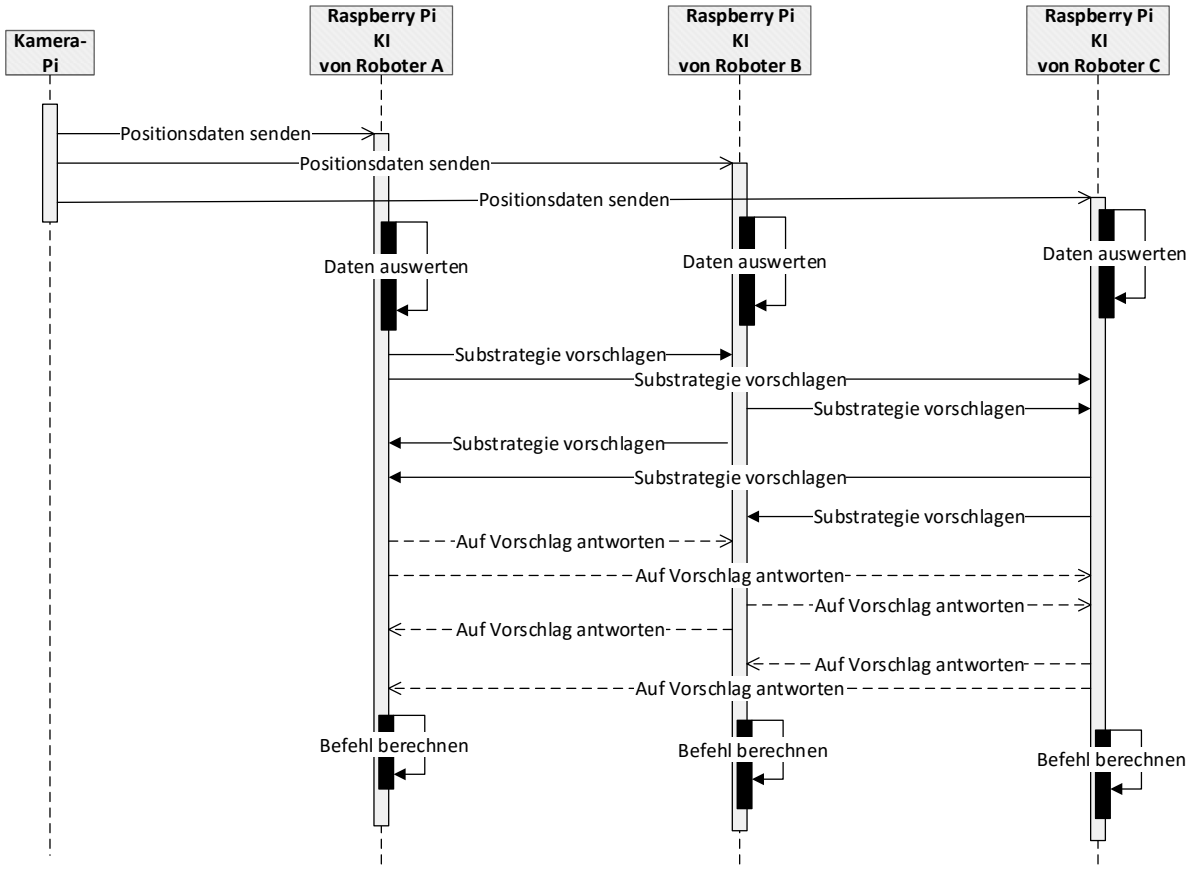

Abbildung 2.7: Sequenzdiagramm: *F40 und F70 - KI berechnet Befehle / Roboterinteraktion*

2.5 Analyse von Funktionalität F50 und F60: Raspberry Pis senden Befehle an Roboter / Befehle ausführen

Die Funktionen F50 und F60 werden gemeinsam betrachtet, da sie stark voneinander abhängig sind. Sobald der Raspberry Pi die von der KI berechneten Befehle (vgl. F40) versendet hat und diese vom Roboter erhalten wurden, führt der Roboter diese aus. Es wird jedoch vom Roboter keine Bestätigung übermittelt. Eine Kontrolle erfolgt über die vom Kamera-Pi versendeten Positionsdaten, die der Raspberry-Pi des Roboters empfängt und auswertet.

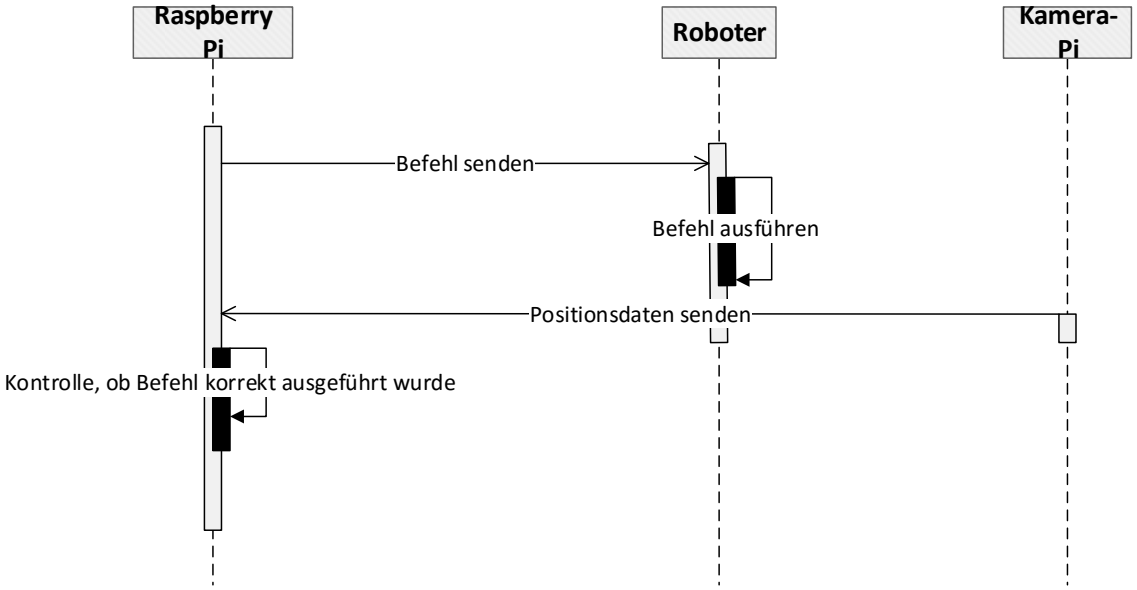

Abbildung 2.8: Sequenzdiagramm: *F50 und F60 - Raspberry Pis senden Befehle an Roboter / Befehle ausführen*

2.6 Analyse von Funktionalität F80: Roboter fährt zum Ball

Die Funktion F80 ermöglicht, dass jeweils ein Roboter zum Ball fährt. Dabei werden vom Kamera-Pi die Positionsdaten verschickt und der Raspberry Pi des Roboters empfängt diese. Das Sequenzdiagramm zeigt exemplarisch, wie der Roboter, der am nächsten am Ball steht, sich zum Ball bewegen soll. Dazu werden wie vormals erwähnt die Positionsdaten empfangen, daraus die Befehle zum Bewegen berechnet und diese dann an den Roboter versendet, welcher sie ausführt. Danach werden erneut aktuelle Positionsdaten vom Kamera-Pi empfangen und ausgewertet, ob die Aktion erfolgreich war, und gegebenenfalls das weitere Vorgehen berechnet.

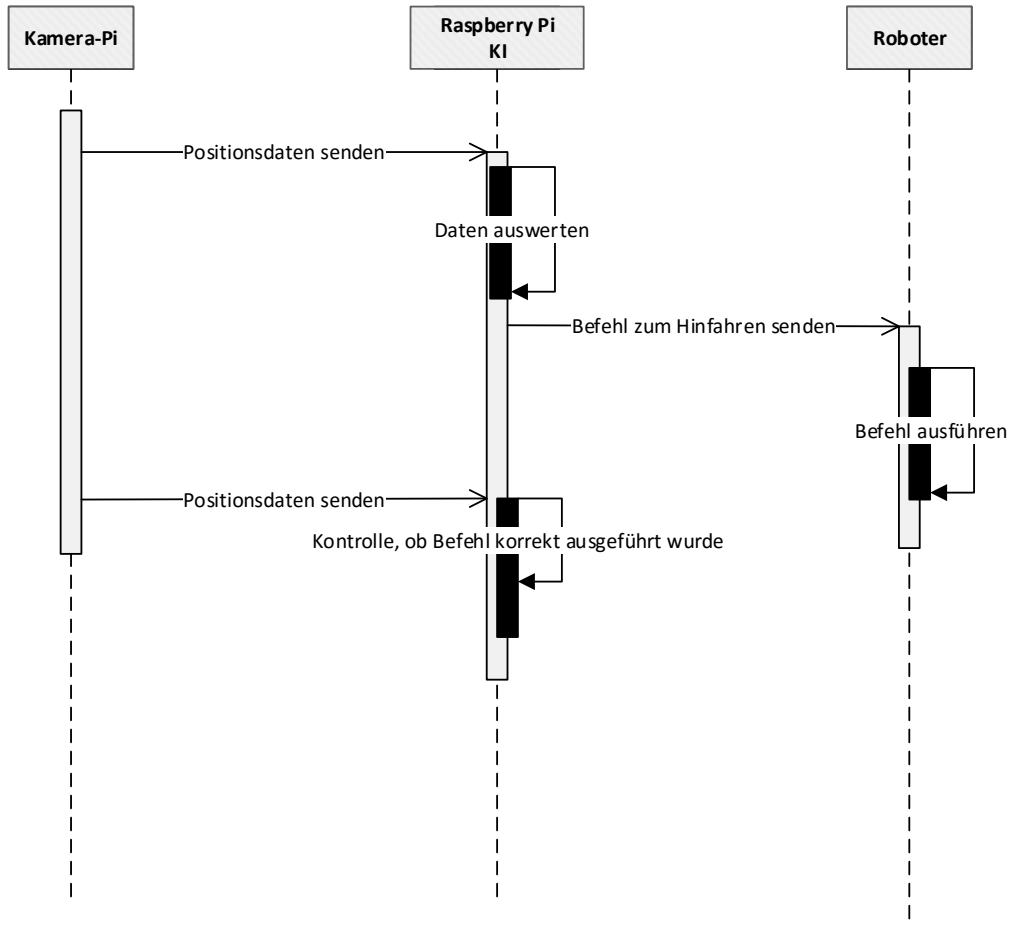

Abbildung 2.9: Sequenzdiagramm: *F80 - Roboter fährt zum Ball*

2.7 Analyse von Funktionalität F90: Roboter schießt den Ball

Die Funktion F90 sorgt dafür, dass die Roboter Schüsse ausführen können. Dabei wird vom jeweiligen Raspberry Pi der Befehl zum Schuss an den Roboter gesendet, der diesen dann ausführt. Danach überprüft der Raspberry Pi des Roboters anhand der aktuellen Positionsdaten vom Kamera-Pi, ob der Befehl ausgeführt wurde.

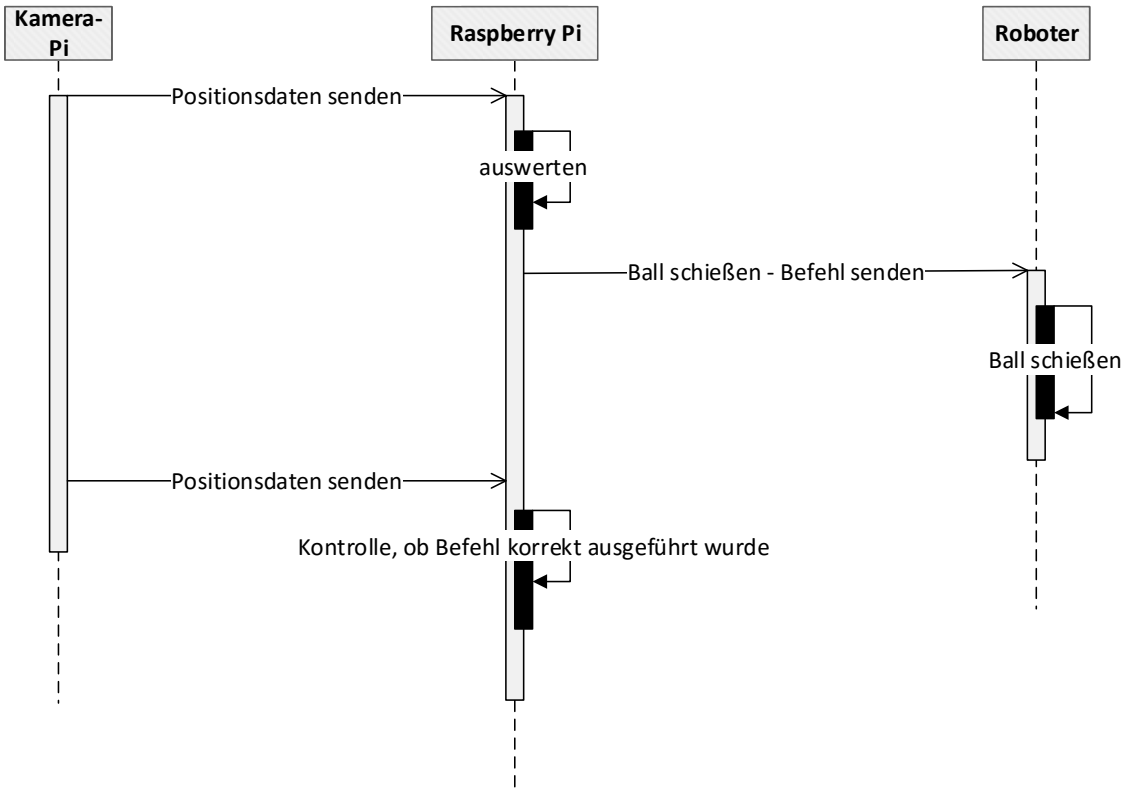

Abbildung 2.10: Sequenzdiagramm: *F90 - Roboter schießt den Ball*

2.8 Analyse von Funktionalität F100: Ball passen

Die Funktion F100 ermöglicht das Zuspielen des Balls von einem Roboter zum anderen. Als erstes sendet der Kamera-Pi die Positionsdaten an den Raspberry Pi des Roboters. Nun sendet der Raspberry Pi des Roboters mit Ball einen Befehl an den Roboter, damit dieser sich zum Roboter ohne Ball dreht. Äquivalent dazu erhält der andere Roboter einen Befehl zum Ausrichten und führt diesen aus. Zwischenzeitlich sendet der Raspberry Pi der Kamera wieder die Positionsdaten der Roboter und des Balls. Nun empfängt der Roboter mit Ball von seinem Raspberry Pi den Befehl zum Schießen des Balls und der Roboter ohne Ball den Befehl zum Entgegennehmen des Balls. Beide führen den entsprechenden Befehl aus. Nach jeder ausgeführten Aktion überprüft der Raspberry Pi des entsprechenden Roboters mithilfe der Positionsdaten der Kamera, ob diese ausgeführt wurde.

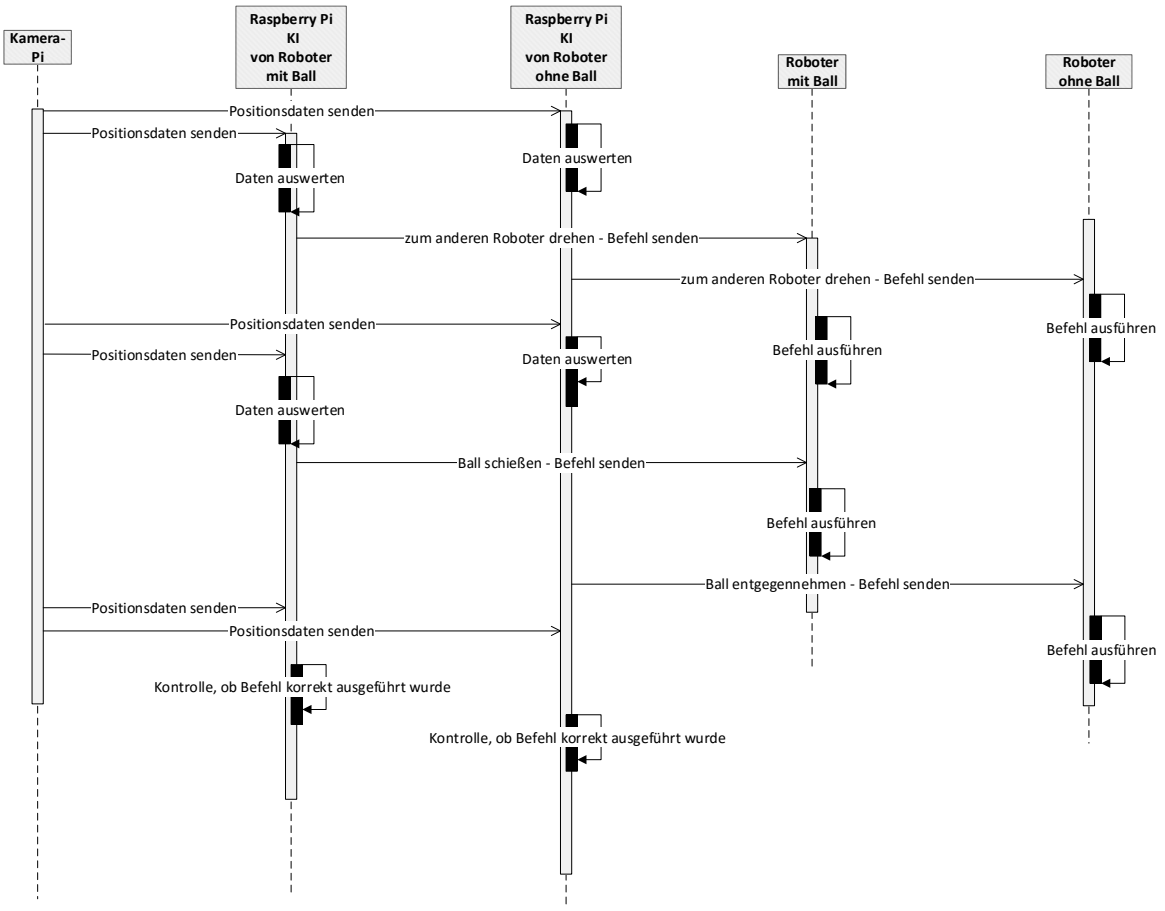

Abbildung 2.11: Sequenzdiagramm: *F100 - Ball passen*

2.9 Analyse von Funktionalität F110: Ball kontrollieren

Die Funktion F110 beschreibt das Kontrollieren des Balls durch einen Roboter. Der Kamera-Pi sendet zuerst die Positionsdaten. Der Raspberry Pi wertet diese Daten danach aus. Anschließend wird an den Roboter der Befehl zum Kontrollieren des Balls gesendet. Der Roboter führt diesen Befehl nun aus. Abschließend sendet der Kamera-Pi wieder aktuelle Positionsdaten, damit der Raspberry Pi überprüfen kann, ob die Aktion des Roboters erfolgreich ausgeführt wurde.

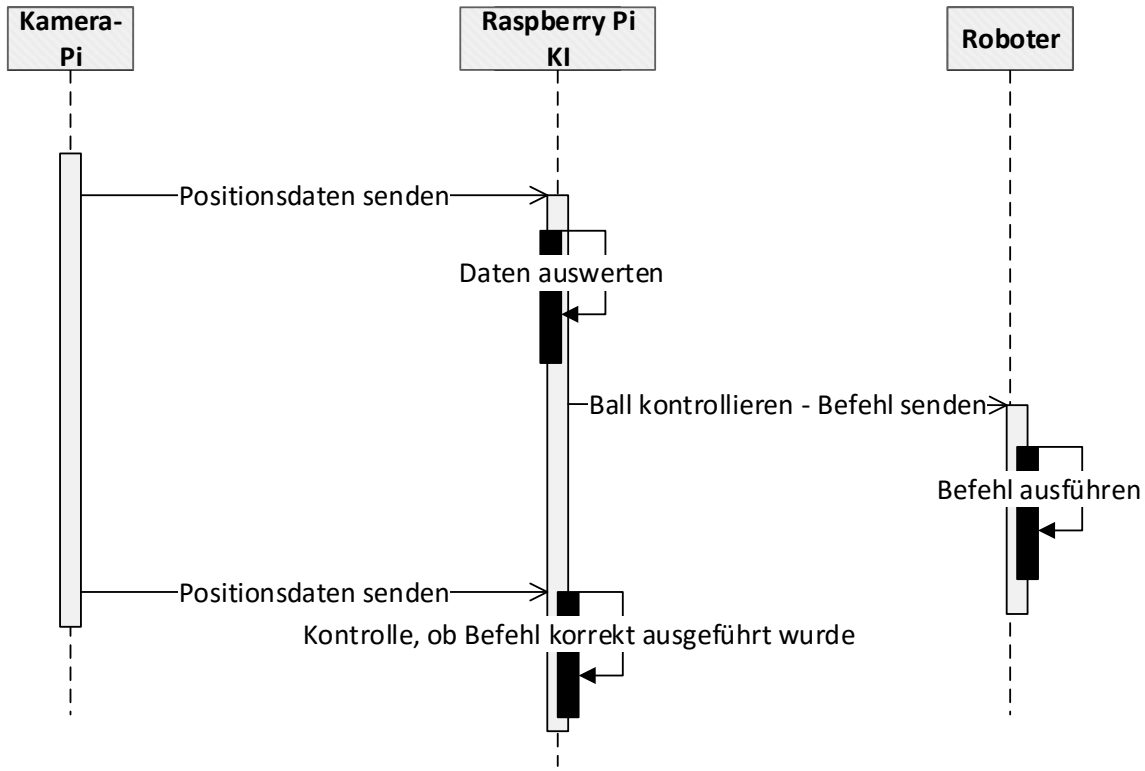

Abbildung 2.12: Sequenzdiagramm: *F110 - Ball kontrollieren*

2.10 Analyse von Funktionalität F120 und F130: Manuelle Steuerung

Die Funktionen F120 und F130 beschreiben jeweils die manuelle Steuerung der Roboter zum einen mit einem Controller, zum anderen mit einer Tastatur. Der Nutzer wählt als erstes einen Roboter in der GUI aus. Der Computer der GUI stellt anschließend die Verbindung zum Roboter her und zeigt dem Nutzer bei erfolgreicher Verbindung die Anleitung für die Steuerung. Danach kann der Nutzer zum Steuern eine der empfohlenen Tasten drücken. Der Computer der GUI sendet den zugehörigen Befehl dem Roboter. Dieser führt den entsprechenden Befehl aus. Dieser Vorgang wiederholt sich so lange, bis der Benutzer die manuelle Steuerung in der GUI beendet. Der Computer der GUI trennt anschließend die Verbindung zum Roboter und zeigt dem Nutzer schlussendlich wieder den Startbildschirm der GUI an.

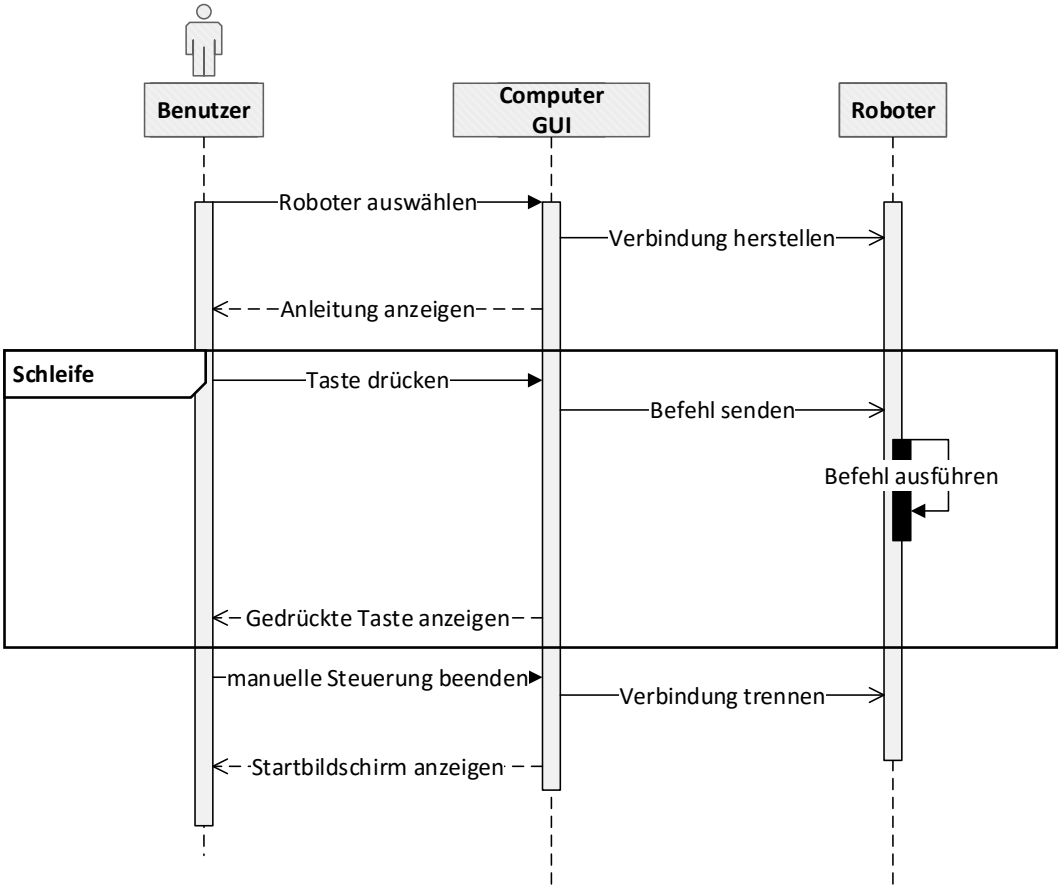

Abbildung 2.13: Sequenzdiagramm: *F120/F130 - Manuelle Steuerung*

3 Datenmodell

In diesem Kapitel wird erläutert, welche dauerhaft zu speichernden Daten unsere Anwendung benötigt. Dies wird unter Verwendung eines UML-Klassendiagramms visualisiert.

3.1 Diagramm

Abbildung 3.1: Klassendiagramm: *Dauerhaft zu speichernde Daten*

3.2 Erläuterung

Durch das kleine UML-Klassendiagramm ist leicht zu erkennen, dass es bei unserer Anwendung nicht viele dauerhaft zu speichernde Daten gibt. Da Fußball ein dynamischer Sport ist, besitzen nur wenige Parameter permanent den gleichen Wert. Eine Klasse, die dauerhaft gleich bleiben sollte, ist unser Spielfeld. Hier würden sich Veränderungen stark auf unsere Roboter sowie deren Spielverhalten auswirken. Außerdem hat unsere künstliche Intelligenz einige langfristig zu speichernde Werte um eine intelligente Strategie entwickeln zu können.

In der Klasse „Spielfeld“ werden alle in der Realität vorhandene Daten gespeichert, also die Spielfeldbegrenzung und die speziellen Bereiche des Feldes. Zudem werden besonders wichtige Koordinaten, wie der Startpunkt auf dem Feld, gespeichert. Andere auf dem Spielfeld liegende

Bereiche sind der Mittelkreis, die Torkreise und natürlich die Tore. Ein vorherige Speicherung ermöglicht einen direkten Zugriff auf die Daten.

In unsere Klasse „Roboter KI“ sollen in erster Linie die Daten dauerhaft gespeichert werden, die wichtig für eine Entscheidungsfindung sind. Diese werden dann mit den agilen Parametern kombiniert, um eine optimale Strategie auszuwählen. Hierzu gehört auch der momentane Spielstand. Liegt man beispielsweise mit 3:1 weit zurück, so sollte offensiver gespielt werden als mit einer Führung von 3:1. Vorher fest definiert sind unsere Bewegungs- und Aktionsmethoden.

Die KI nutzt die Daten des Spielfeldes zur Orientierung und zur Berechnung der Strategie.

4 Konfiguration

In diesem Kapitel werden alle Dateien aufgeführt, die die Kommunikation der einzelnen Komponenten über das Netzwerk konfigurieren. Diese umfassen die Kommunikation des ausführenden Computers zu den Raspberry Pis, die Kommunikation zwischen den Pis, die Einbindung der Kameradaten und die Übertragung über das UDP.

Die folgenden Dateien müssen alle im Unterordner „Communication“ liegen und dürfen nicht verschoben werden. Für die Ausführung muss die Entwicklungsumgebung Eclipse vorliegen und das leJos EV3 Plug-In in der Version „0.9.1 beta“ für selbige Entwicklungsumgebung. Das Plug-In kann über den internen Marktplatz von Eclipse abgerufen und installiert werden. Damit Eclipse nutzbar ist, muss die Java JDK und die Java JRE in der Version „1.8.0_77“ auf dem Computer vorliegen. Die erwähnte LeJos EV3-Version muss außerdem auf der SD-Karte in den Robotern vorhanden sein. Ansonsten werden keine weiteren Programme benötigt.

In der Datei **„PCClient.java“** wird eine Verbindung zwischen dem ausführenden Computer und den Raspberry Pis hergestellt. Dabei werden feste IP-Adressen der Pis benötigt. Diese stehen noch nicht fest, können aber in der Datei unter den Variablen „Pi1Ip“, „Pi2Ip“ und „Pi3Ip“ eingetragen werden.

In der Datei **„CameraReceiver.java“** wird eine Verbindung zur Spielfeldkamera hergestellt. Die Datei „CameraReceiver.java“ bekommt die Daten über den Port 61001. Die Kamera liefert Positionsdaten der sechs Roboter und des Balls sowie die Rotation der Roboter. Die Positionsdaten sehen wie folgt aus: Zunächst wird die Position des Balls zurückgegeben (int X-Koordinate,int Y-Koordinate). Anschließend folgen die Daten der Roboter mit folgendem Schema: (int Roboter-ID, int X-Koordinate, int Y-Koordinate, double Rotation).

In den Dateien **„RaspPiClient.java“** und **„RaspPiServer.java“** wird das Server-Client System zwischen den Raspberry Pis konfiguriert. Für die Pis werden drei Ports benötigt, die zu diesem Zeitpunkt noch nicht feststehen. Die Ports können unter der Datei „RaspPiClient.java“ eingegeben werden. Außerdem wird ein Port für den Pi-Server benötigt. Dieser steht ebenfalls noch nicht fest und kann in der Datei „RaspPiServer.java“ festgelegt werden. Die IP-Adresse des Empfängers wird in der Datei „RaspPiClient.java“ konfiguriert.

5 Glossar

Aktivitätsdiagramm: Ist ein Verhaltensdiagramm in UML. In einem Aktivitätsdiagramm wird der Ablauf eines konkreten Anwendungsfalls modelliert.

Controller: Ist ein Eingabegerät, dass hauptsächlich für die Steuerung von Computerspielen vorgesehen ist.

Funktionsmodelle: Ein Funktionsmodell wird dafür genutzt die Systemteile und die Kommunikation unter den Systemteilen zu modellieren.

GUI: Grafische Benutzeroberfläche. Hat die Aufgabe Anwendungssoftware mit Symbolen, Steuerelemente und Widgets bedienbar zu machen.

IP-Adressen: Eine IP-Adresse ist eine Adresse in einem Computernetz und stellt sicher, dass die einzelnen Geräte im Netzwerk erreichbar sind.

JDK: Das Java Delevelopment Kit ist eine Sammlung fertiger Programme in der Programmiersprache Java, um neue Programme entwickeln zu können.

JRE: Die Java Runtime Enviroment ist die Laufzeitumgebung von Java.

KI: Künstliche Intelligenz. Übertragung von menschenähnlicher Intelligenz auf Roboter, Computer und oder Maschinen.

Klassendiagramm: Ein Klassendiagramm ist ein Strukturdiagramm zur Modellierung von Klassen, Schnittstellen und deren Beziehungen.

LeJos: Ist eine Programmbibliothek und auch Betriebssystem, dass eine Programmierung/Steuerung von Lego Mindstorm Robotern ermöglicht.

Objektmodelle: Objektmodelle werden dafür benutzt statische Strukturen von Klassen und Objekten, sowie deren Verhalten zu modellieren.

Plug-In: Eine optionale Softwarekomponente für ein bestehendes Programm.

Port: Ermöglichen den Zugriff von Client zu Server, sowohl Server als auch Client haben bei einer bestehenden Verbindung mindestens einen Port. Der Server besitzt einen festen Port und der Client einen ausgehandelten unbenutzten Port.

Raspberry Pi: Ein Minicomputer mit einem ARM-Mikroprozessor, auf dem als Betriebssystem hauptsächlich Linux eingesetzt wird. Er kann für verschiedene Zwecke eingesetzt werden.

Roboter: Sind eine technische Apparatur, die dazu dient menschliche Aufgaben zu übernehmen, sie werden meist über Computerprogramme gesteuert. In diesem Projekt hingegen wird als Roboter lediglich der „Lego Mindstorm EV3 – Roboter“ verstanden.

RoboSoccer: Fußballspiel zwischen Robotern mit angepassten Regeln.

Schwarmbasiert: Eine schwarmbasierte KI arbeitet ohne direkten Anführung oder Befehlsgeber, sondern als kollektive Intelligenz.

Sequenzdiagramm: Ein Sequenzdiagramm ist ein Verhaltensdiagramm in UML. In Sequenzdiagrammen wird der Austausch von Nachrichten modelliert.

Systemkomponenten: Einzelne Elemente eines Gesamtsystems.

UDP: User Datagram Protocol ist ein Netzwerkprotokoll, bei dem nicht sichergestellt wird, dass gesendete Pakete auch wirklich beim Empfänger ankommen und kein Verbindungsaufbau stattfindet. Dadurch wird der Overhead reduziert, aber auch kein Schutz der Daten gewährleistet.

UML: Die Unified Modelling Language ist eine grafische Modellierungssprache zur Spezifikation, Konstruktion und Dokumentation von Software-Teilen.