Technische Universität Braunschweig

NeXT Generation Transport Tycoon

Team 0

Software-Entwicklungspraktikum (SEP)
Sommersemester 2013

Systemspezifikation

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

Betreuer: Benjamin Mensing

Auftragnehmer:

Name	E-Mail-Adresse
Dennis Stelter	d.stelter@tu-bs.de
Henrik Lange	henrik.lange@tu-bs.de
Jochen Steiner	jochen.steiner@tu-bs.de
Markus-Björn Meißner	m-b.meissner@tu-bs.de
Patricia-Tatjana Kasulke	p.kasulke@tu-bs.de
Sebastian Eilf	s.eilf@tu-bs.de
Tessa Fabian	tessa.fabian@tu-bs.de

Braunschweig, 15. Mai 2013

Versionsübersicht

Version	Datum	Autor	Status	Kommentar
0.1	05.05.2013	Tessa Fabian, Henrik Lange	i.B.	Analyse von Produktfunktionen 40, 50, 110, 130 hinzugefügt
0.2	06.05.2013	Sebastian Eilf, Markus-Björn Meißner	i.B.	Einleitung hinzugefügt
0.3	06.05.2013	Markus-Björn Meißner	i.B.	Analyse von Produktfunktionen 10, 20, 30, 120 hinzugefügt
0.4	06.05.2013	Tessa Fabian, Jochen Steiner, Dennis Stelter	i.B.	Komponentenspezifikation hinzugefügt
0.5	06.05.2013	Henrik Lange	i.B.	Glossar hinzugefügt
0.6	07.05.2013	Patricia-Tatjana Kasulke, Jochen Steiner	i.B.	Analyse von Produktfunktionen 60, 70, 80, 100 hinzugefügt
0.7	08.05.2013	Tessa Fabian, Jochen Steiner, Sebastian Eilf	i.B.	Softwarearchitektur hinzugefügt
0.8	09.05.2013	Dennis Stelter	i.B.	Analyse von Produktfunktionen 90, 140 hinzugefügt
0.9	09.05.2013	Henrik Lange	i.B.	Layout: Anpassungen und Korrekturen
0.10	09.05.2013	Markus-Björn Meißner	i.B.	Verteilungsentwurf hinzugefügt
0.11	10.05.2013	Patricia-Tatjana Kasulke	i.B.	Kriterienerfüllung hinzugefügt
0.12	13.05.2013	Dennis Stelter, Tessa Fabian, Jochen Steiner, Henrik Lange	i.B.	Korrekturen der Sequenzdiagramme
0.13	14.05.2013	Dennis Stelter, Tessa Fabian, Jochen Steiner, Markus-Björn Meißner	i.B.	Korrektur von Kapitel 3.2
0.14	14.05.2013	Dennis Stelter, Jochen Steiner, Henrik Lange	i.B.	Korrektur von Kapitel 3.1

0.15	14.05.2013	Dennis Stelter, Tessa Fabian, Jochen Steiner, Markus-Björn Meißner	i.B.	Korrektur von Kapitel 3.3
0.16	15.05.2013	Patricia-Tatjana Kasulke, Henrik Lange	i.B.	Korrektur von Kapitel 5.0
0.17	15.05.2013	Dennis Stelter, Tessa Fabian, Jochen Steiner, Markus-Björn Meißner, Patricia-Tatjana Kasulke, Henrik Lange	i.B.	Korrektur der Texte von Kapitel 2.0
1.0	15.05.2013	Henrik Lange	abg.	Abgabe ISF

i.B.: in Bearbeitung, **abg.**: abgeschlossen

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einleitung

Das vorliegende Dokument stellt einen ersten Entwurf (Grobentwurf) des Projekts „NeXT Generation Transport Tycoon“ dar.
In dieser Phase wird primär auf Entwürfe und erste Planungen mit Hauptaugenmerk auf die Systemarchitektur eingegangen. Es werden also zwingend benötigte Funktionalitäten untersucht und es wird festgelegt, welche Architektur für ihre Realisierung benötigt wird. Anschließend erfolgt ein erster Entwurf einer entsprechenden Systemarchitektur (sowohl in Form von Software als auch in Form eines verteilten Systems), der die gefundenen Bedingungen erfüllt (Übersichtsdiagramm siehe Anhang A, zum Betrachten in DIN A3 ausdrucken).

1.1. Projektdetails

Besonders interessante oder komplizierte Sachverhalte werden hier noch weiter vertieft.

1.1.1. Kollisionsvermeidung

Ein komplizierter Sachverhalt des Projektes „NeXT Generation Transport Tycoon“ ist die Vermeidung von Kollisionen der Roboter. Hierfür muss der Server in Echtzeit Informationen über die geplante Route aller aktiven Roboter und ihren Standort erhalten. Mithilfe dieser Informationen müssen „besetzte“ Pfade temporär gesperrt werden und eventuell ganze Routen neu geplant werden. Die Komplexität dieser Aufgabe steigt mit einer größeren Zahl von aktiven Robotern stark an, da es ab einer bestimmten Zahl nur noch schwer möglich ist, für jeden NXT komplette Routen zwischen zwei Punkten herzustellen.

1.1.2. Vermeidung von Deadlocks

Ein weiteres Problem sind die sogenannten Deadlocks, also Situationen, in denen alle beteiligten Roboter handlungsunfähig sind, da alle möglichen Routen durch andere Roboter versperrt sind. Solche Deadlocks können auftreten, wenn Roboter in eine Sackgasse fahren und ein anderer die einzige Zufahrt versperrt. In diesem Fall müssen die Roboter eine Möglichkeit besitzen, solche Situationen einerseits zu erkennen und andererseits entsprechende Lösungsverfahren zu starten, den Robotern evtl. sogar ein Verhalten aufzwingen, dass zwar gegen ihren aktuellen Auftrag

verstößt, aber für das Beheben des Deadlocks unabdingbar ist.
Sobald jede Partei im Spiel mehr als einen Roboter besitzt, kann das bewusste Herbeiführen von Deadlocks jedoch auch ein Bestandteil einer Strategie sein, um den Gegner durch Blockieren zum Ausweichen zu zwingen.

1.1.3. Generieren und Zuweisen von Transportaufträgen

Der Server soll allen Spielern Transportaufträge anbieten können, die zufällig generiert werden und den Transport von Waren zwischen zwei Punkten A und B erfordern.
Diese Aufträge müssen assoziierte Werte wie Entlohnung, Dauer und Kapazität haben. Weiterhin müssen alle Parteien auf diese Aufträge „bieten“, d.h. sie versuchen, sich gegenseitig zu unterbieten. Die Partei, die das günstigste Angebot macht, erhält den Zuschlag und kann den Auftrag ausführen.
Hierbei ist es wichtig, eine entsprechende KI zu implementieren, die dieses Bieten korrekt und strategisch ausführt. Weiterhin muss das Zuweisen von Robotern zu neuen Aufträgen möglich sein. Im Idealfall kann ein derzeit „besetzter“ Roboter seine aktuelle Route anpassen, um mehrere Aufträge zur gleichen Zeit ausführen zu können.

2. Analyse der Produktfunktionen

In diesem Kapitel werden die Zusammenhänge der einzelnen Funktionen über Sequenzdiagramme erläutert. Dadurch soll eine geeignete Architektur auf Basis der im Pflichtenheft gefundenen Produktfunktionen und nicht-funktionalen Anforderungen gefunden werden.

2.1. Analyse von Funktionalität $\langle F10 \rangle$: Bluetooth-Datenverbindung

Roboter und Server kommunizieren miteinander über eine Bluetooth-Verbindung. Die jeweiligen Kommunikations-Komponenten bauen dazu vor Spielbeginn die Verbindung auf. Während des Spiels werden Daten dann in einem Integer-Stream übertragen. Dabei können Daten sowohl vom Server zum Roboter als auch vom Roboter zum Server gesendet werden.

Abbildung 2.1 stellt die Funktionalität in einem Sequenzdiagramm dar. Der Roboter wartet auf eine Verbindung mit dem Server. Der Server baut mit der Verbindungsanfrage zu jedem der Roboter einen eigenen Ein-/Ausgabestrom auf. Während des Spiels sendet der Server den Robotern Auktionsanfragen, Streckenfreigaben und Wegenetzdaten zu. Der Roboter nutzt die Bluetooth-Verbindung zum Bieten auf Auktionen sowie für Freigabeanfragen bezüglich einzelner Streckenabschnitte. Wenn das Spiel beendet wurde, werden die Verbindungskanäle wieder geschlossen.

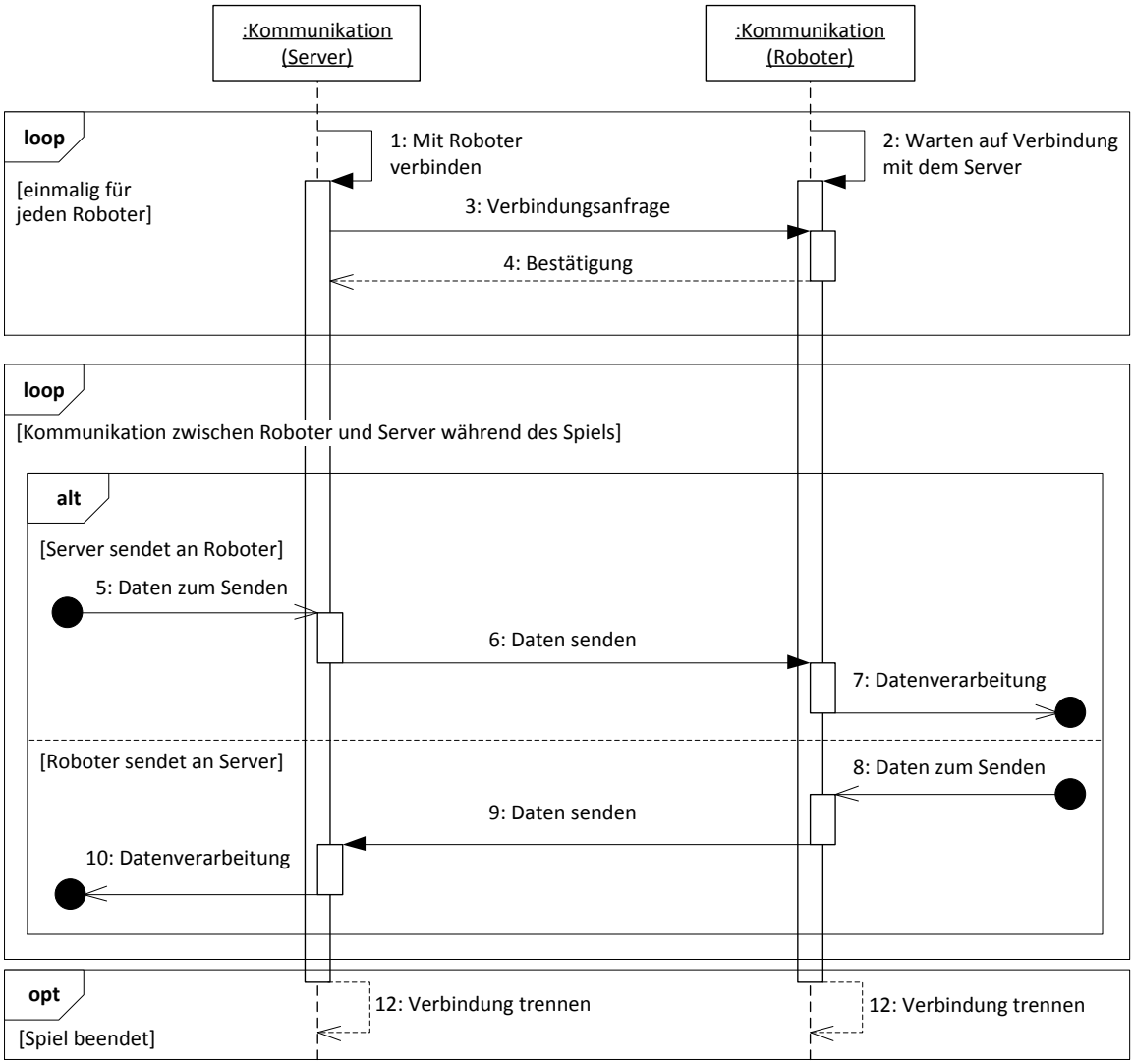

Abbildung 2.1.: Sequenzdiagramm $\langle F10 \rangle$ „Bluetooth-Datenverbindung“

2.2. Analyse von Funktionalität $\langle F20 \rangle$: Datenverwaltung

Vor Spielbeginn müssen die notwendigen Daten (Wegenetz, Ressourcen, Produktionszyklus, Roboter-Startpositionen) auf den Server geladen werden. Erst wenn dieser Vorgang abgeschlossen ist, kann ein Spiel gestartet werden.
Während des Spiels verwaltet der Server die durch ausstehende oder geleistete Aufträge anfallenden Daten und aktualisiert die sich daraus ergebenden Statistiken (Preisbildung, Gewinne). Diese Statistiken werden später sowohl in der Spieler-GUI $\langle F110 \rangle$, als auch in der Benutzer-GUI angezeigt.

Abbildung 2.2 stellt die Initialisierung in einem Sequenzdiagramm dar. Der Benutzer startet das Programm mit einer ausführbaren Datei. Daraufhin hat er die Möglichkeit, alle benötigten Daten über eine spezielle GUI einzugeben. Die Korrektheit wird nach Eingabe geprüft oder es wird eine bestimmte Auswahl vorgegeben. Beispielsweise dürfen sich an einem Standort keine zwei Roboter befinden. Der Server soll nur korrekte Daten speichern und gibt dem Benutzer Rückmeldungen bezüglich dieser. Da sich Benutzer-GUI sowie Verwaltung (Server) auf dem selben Rechner befinden, wird keine besondere Kommunikationskomponente verwendet.

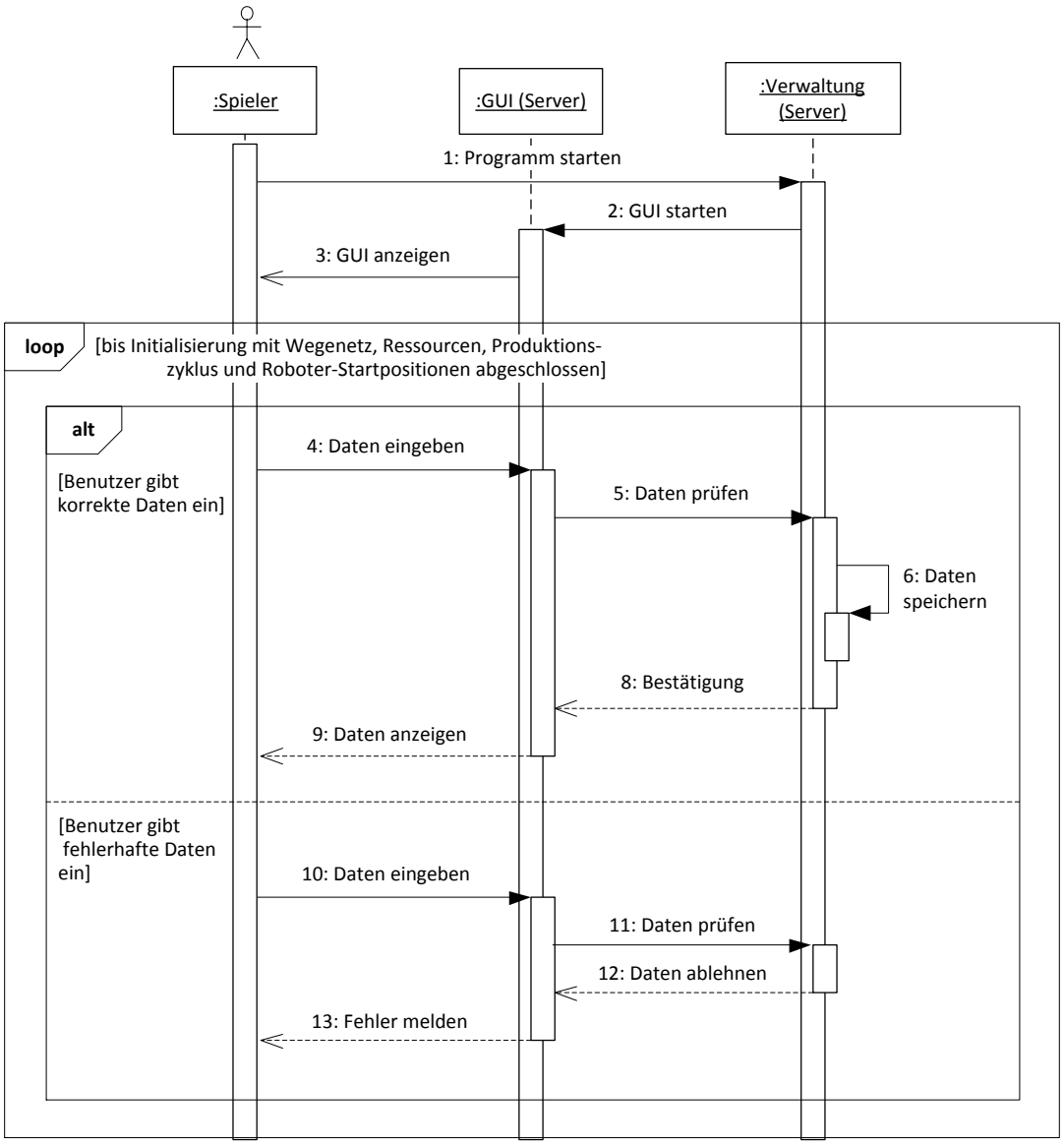

Abbildung 2.2.: Sequenzdiagramm $\langle F20 \rangle$ „Initialisierung“

Die Funktionalität Verwaltung wird in Abbildung 2.3 dargestellt. Nach jedem erfolgreich abgeschlossenen Auftrag muss die Gewinnstatistik aktualisiert werden. Mit Veränderungen der Auftragslage werden Nachfrage und Preise eines Produkts angepasst. Der Benutzer hat die Möglichkeit, Aufträge zu erstellen. Die GUI greift dabei direkt auf Methoden der Serververwaltung zu.

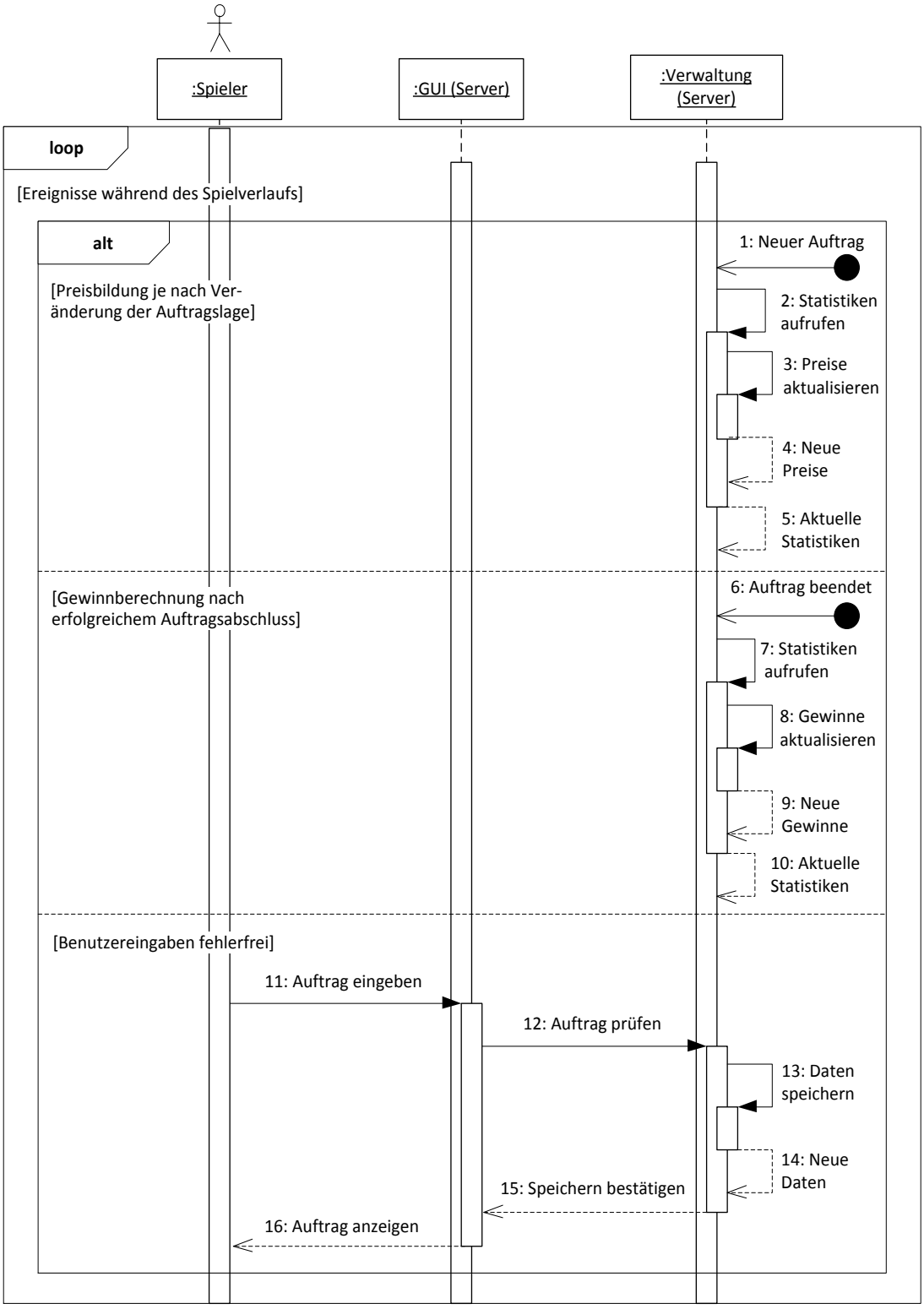

Abbildung 2.3.: Sequenzdiagramm $\langle F20 \rangle$ „Verwaltung“

2.3. Analyse von Funktionalität $\langle F30 \rangle$: Initialisierung der Roboter

Die Roboter werden über Bluetooth-Verbindungen zuerst registriert und später initialisiert.

Abbildung 2.4 stellt die Funktionalität in einem Sequenzdiagramm dar. Sowohl Registrierung als auch Initialisierung nutzen eine Bluetooth-Verbindung $\langle F10 \rangle$. Dabei werden die Daten zwischen den jeweiligen Kommunikationskomponenten als Integer-Stream übertragen.
Jede Anfrage bzw. Übertragung der Verwaltungskomponente des Servers wird von der KI des Roboters bestätigt. Die Initialisierung der Roboter kann erst erfolgen, wenn die Initialisierung des Servers beendet wurde und ist erst dann erfolgreich abgeschlossen, wenn alle registrierten Roboter die Daten des Wegenetzes erhalten haben.

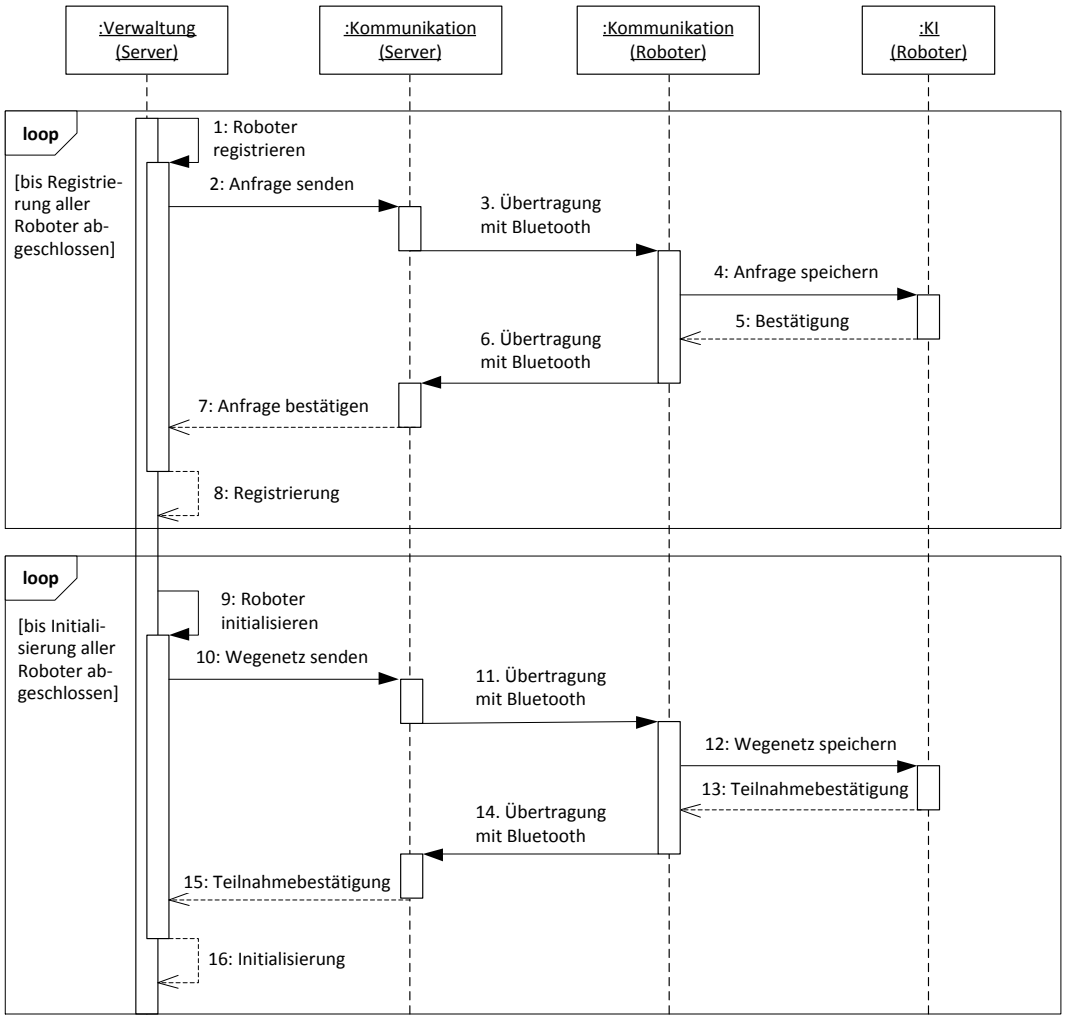

Abbildung 2.4.: Sequenzdiagramm $\langle F30 \rangle$ „Initialisierung der Roboter“

2.4. Analyse von Funktionalität $\langle F40 \rangle$: Straßenerkennung durch Roboter

Der Roboter soll in der Lage sein, dem Straßenverlauf folgen zu können. Um dies zu erreichen, wertet er seine Sensoren aus und anhand deren Daten korrigiert er die Geschwindigkeit seiner Räder.
Als Entscheidungsgrundlage dienen die Helligkeitswerte, die die Sensoren mithilfe einer geeigneten Methode zurückgeben. Es gibt vier mögliche Szenarien:

- Beide Sensoren liefern helle Farbstufen zurück
- Der linke Sensor liefert eine dunkle Farbstufe zurück
- Der rechte Sensor liefert eine dunkle Farbstufe zurück
- Beide Sensoren liefern dunkle Farbstufen zurück

Die Abbildung 2.5 stellt die Funktion als Sequenzdiagramm dar. Der Roboter fragt fortlaufend seine Sensoren ab. Solange beide helle Farben messen, folgt der Roboter der schwarzen Linie. Wenn links oder rechts eine dunkle Farbstufe gemessen wird, wird dementsprechend die Geschwindigkeit des linken oder rechtes Rades heruntergeregelt, um eine dementsprechende Drehung vorzunehmen. Sobald der linke und der rechte Sensor eine dunkle Farbstufe melden, ist das Straßenende erreicht. Eine Drehung um 180° ist erforderlich.

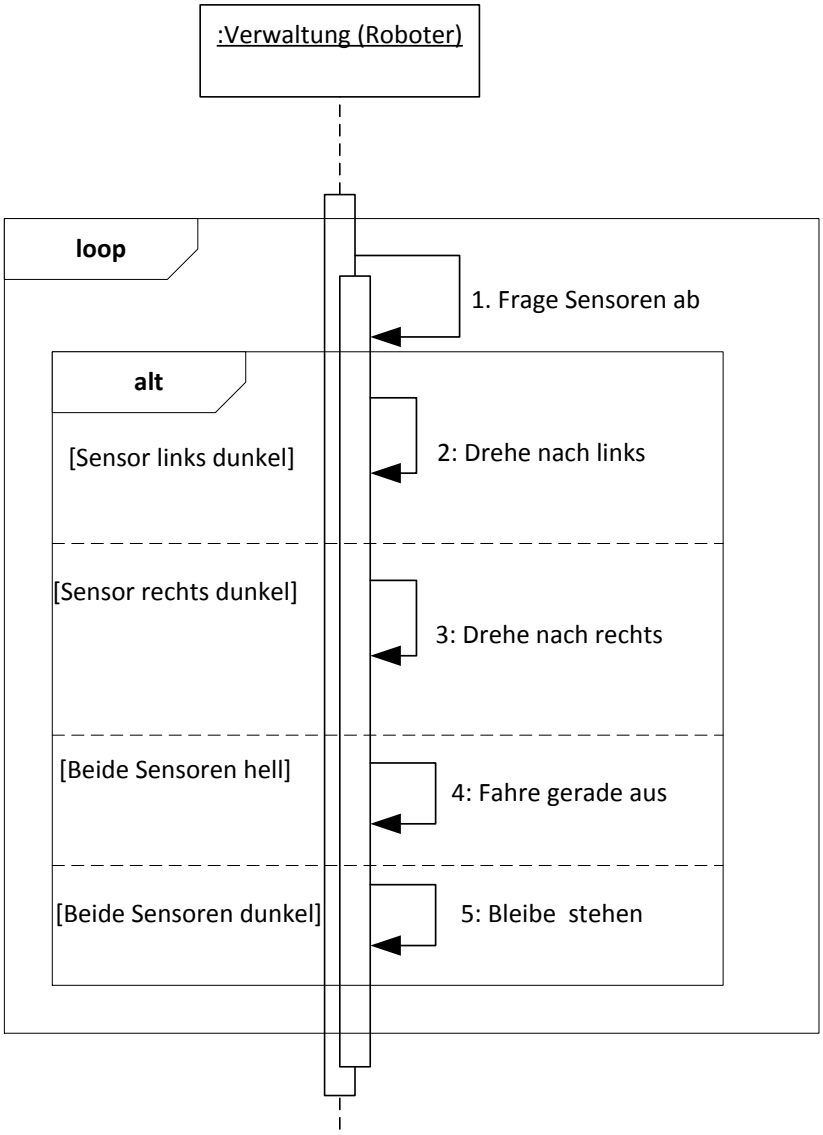

Abbildung 2.5.: Sequenzdiagramm $\langle F40 \rangle$ „Straßenerkennung durch Roboter“

2.5. Analyse von Funktionalität $\langle F50 \rangle$: Transportvolumen der Roboter

Damit ein Roboter nicht beliebig viele Waren zur gleichen Zeit transportieren kann, muss implementiert werden, dass er vor der Abgabe des Auktionsgebotes prüft, ob er genügend Kapazität zur Verfügung hat.

Diese Problematik wird in dem Sequenzdiagramm in Abbildung 2.6 behandelt. Der Roboter überprüft nach Erhalt des Auftragsangebotes, ob genügend Kapazitäten vorhanden sind, um diesen Auftrag durchzuführen. Hat er genügend Volumen, so kann er ein Gebot für den Auftrag abgeben, wenn nicht, muss er den Auftrag ablehnen.

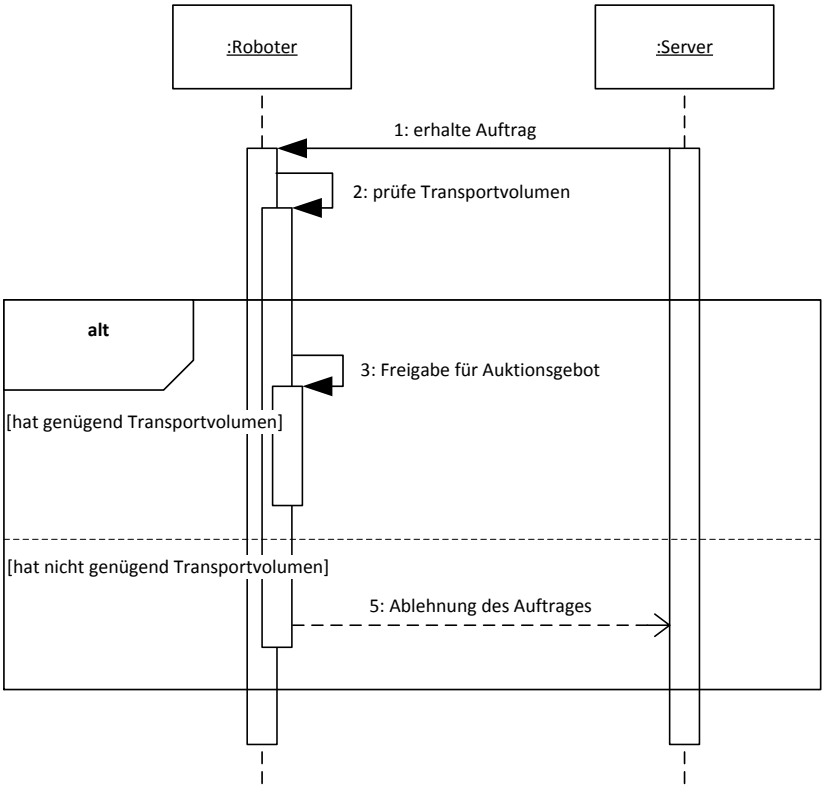

Abbildung 2.6.: Sequenzdiagramm $\langle F50 \rangle$ „Transportvolumen der Roboter“

2.6. Analyse von Funktionalität $\langle F60 \rangle$: Treibstoffverbrauch der Roboter

Der Roboter braucht zur Bearbeitung seiner Transporte einen virtuellen Treibstoff. Er kennt die Restmenge seines Treibstoffvorrats und plant seine Routen unter Berücksichtigung der Treibstoffmenge.
Der Treibstoffverbrauch ist zeitabhängig und ohne Treibstoff fährt der Roboter nicht weiter. Der Roboter erledigt Aufträge und erhält nach erfolgreicher Ausführung eine Betankung.

In dem Sequenzdiagramm in Abbildung 2.7 wird die Funktion näher erörtert. Der Roboter verbraucht seinen Treibstoff. Falls er einen Auftrag erfolgreich erledigt hat, wird er betankt. Ist sein Treibstoffvorrat leer, scheidet er aus.

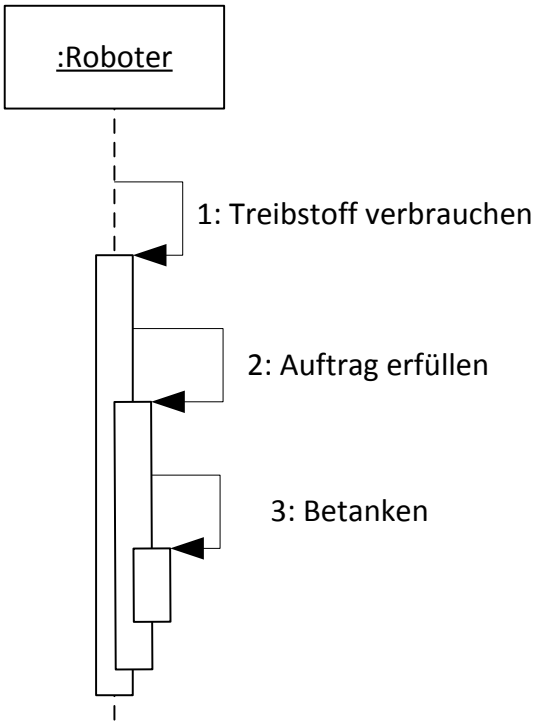

Abbildung 2.7.: Sequenzdiagramm $\langle F60 \rangle$ „Treibstoffverbrauch der Roboter“

2.7. Analyse von Funktionalität $\langle F70 \rangle$: Streckenfreigabe durch Server

Der Roboter bearbeitet einen Auftrag und hat zuvor eine Route berechnet. Danach bittet er um Freigabe eines Streckenabschnitts. Der Server verwaltet die Strecken durch eine Adjazenzmatrix. Falls der Roboter eine Freigabe erhält, darf er in den freigegeben Streckenabschnitt fahren. Für alle anderen Roboter ist dieser Streckenabschnitt dann gesperrt. Der Streckenabschnitt wird freigegeben, wenn der Roboter signalisiert hat, dass er den Abschnitt verlassen hat.

Die genaue Funktionsweise dieser Funktion, wird im Sequenzdiagramm in Abbildung 2.8 erklärt. Der Roboter stellt eine Anfrage, ob ein Streckenabschnitt, den er befahren will, frei ist. Diese Anfrage wird durch die Kommunikationsmodule von Roboter und Server an das Spielkoordinationsmodul weitergegeben. Die Spielkoordination leitet die Anfrage an das Wegenetzverwaltungsmodul weiter. Dort wird geprüft, ob der Streckenabschnitt frei ist.
Falls der Streckenabschnitt frei ist, wird eine Freigabe an den Roboter gesendet und der Roboter fährt in den Streckenabschnitt. Wenn der Roboter den alten Streckenabschnitt verlassen hat, wird dieser in der Wegenetzverwaltung freigegeben.
Falls der Streckenabschnitt nicht frei ist, erhält der Roboter keine Freigabe und wartet solange, bis er eine Freigabe erhält. Danach verfährt der Roboter nach demselben Muster wie bei sofortiger Freigabe.

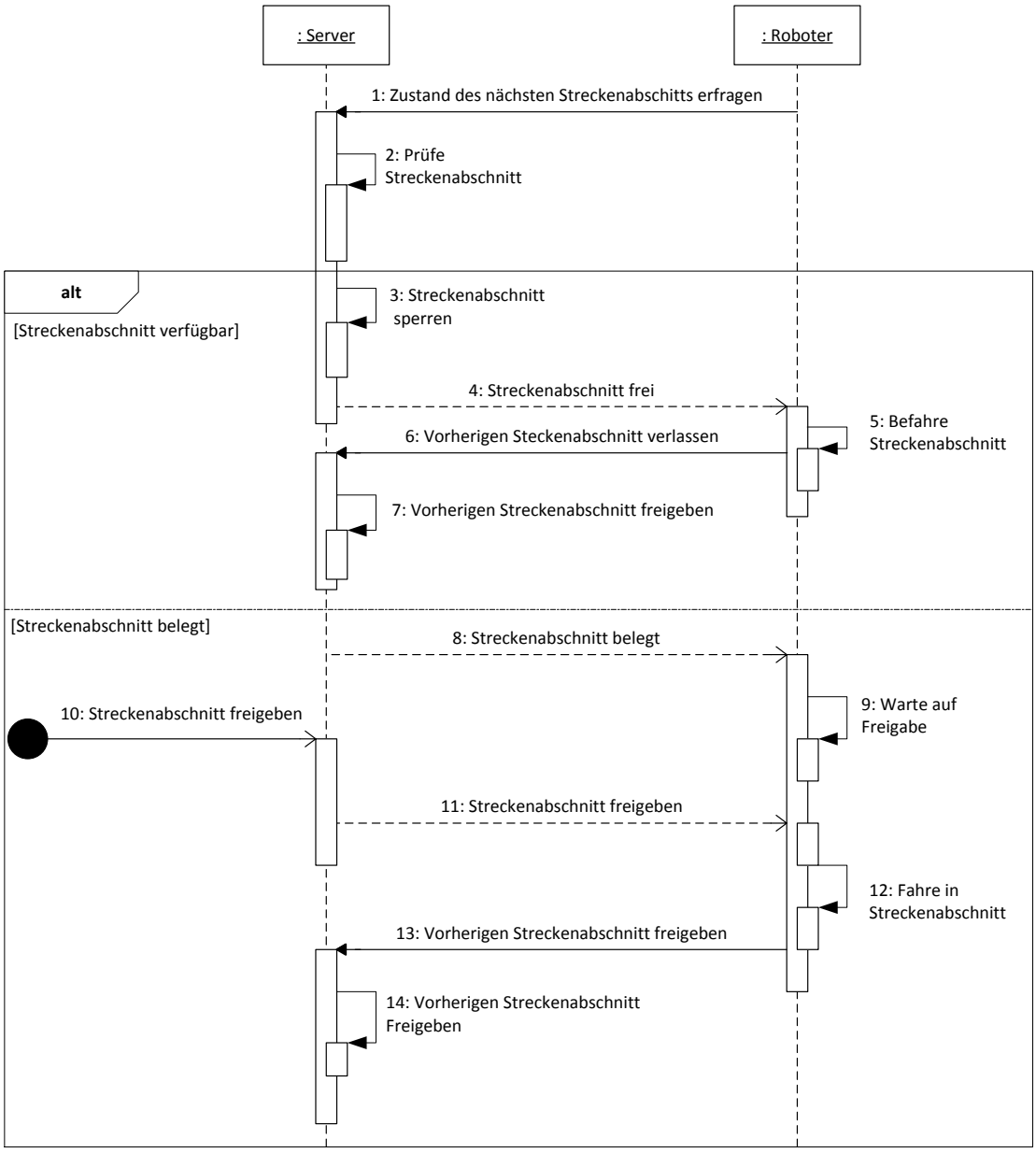

Abbildung 2.8.: Sequenzdiagramm $\langle F70 \rangle$ „Streckenfreigabe durch Server“

2.8. Analyse von Funktionalität $\langle F80 \rangle$: Routenberechnung durch Roboter

Selbständige Wegewahl des Roboters. Der Roboter sucht sich selbständig den Weg durch das Wegenetz und kommuniziert stetig mit Server. Will der Roboter einen Abschnitt befahren, muss dieser eine Reservierungsanfrage an den Server senden. Dieser schaut, ob bereits eine Reservierung vorliegt oder der Abschnitt frei ist. Beim zweiten Fall reserviert der Server diesen Streckenabschnitt für diesen Roboter. Bis er dem Server das Verlassen des Abschnitts mitteilt. Nun kann der Abschnitt wieder freigegeben werden.

In dem Sequenzdiagramm, welches in der Abbildung 2.9 zu sehen ist, wird der zeitliche Ablauf der Kommunikation während der Routenberechnung durch den Roboter dargestellt.
Der Roboter fordert die Auftragsdaten vom Server an. Nach dem Erhalt der Auftragsdaten werden die Wegenetzdaten ausgewertet, damit der Roboter mit der Hilfe eines Algorithmus die jeweilige Strecke berechnen kann.
Darauf folgt eine Berechnung der für den Auftrag notwendigen Ressourcen, wie zum Beispiel der Treibstoffverbrauch. Nachdem die Berechnung abgeschlossen ist, sendet der Roboter die Daten an dem Server, der die Route auf Inkonsistenz überprüft. Wurde keine Inkonsistenz gefunden, sendet der Server eine positive Rückmeldung. Der Roboter bestätigt den Empfang der Nachricht und fährt die Route ab.
Bei einer negativen Rückmeldung, behebt der Roboter die Inkonsistenz und sendet dem Server die veränderten Daten zu. Der Server führt eine erneute Überprüfung durch.

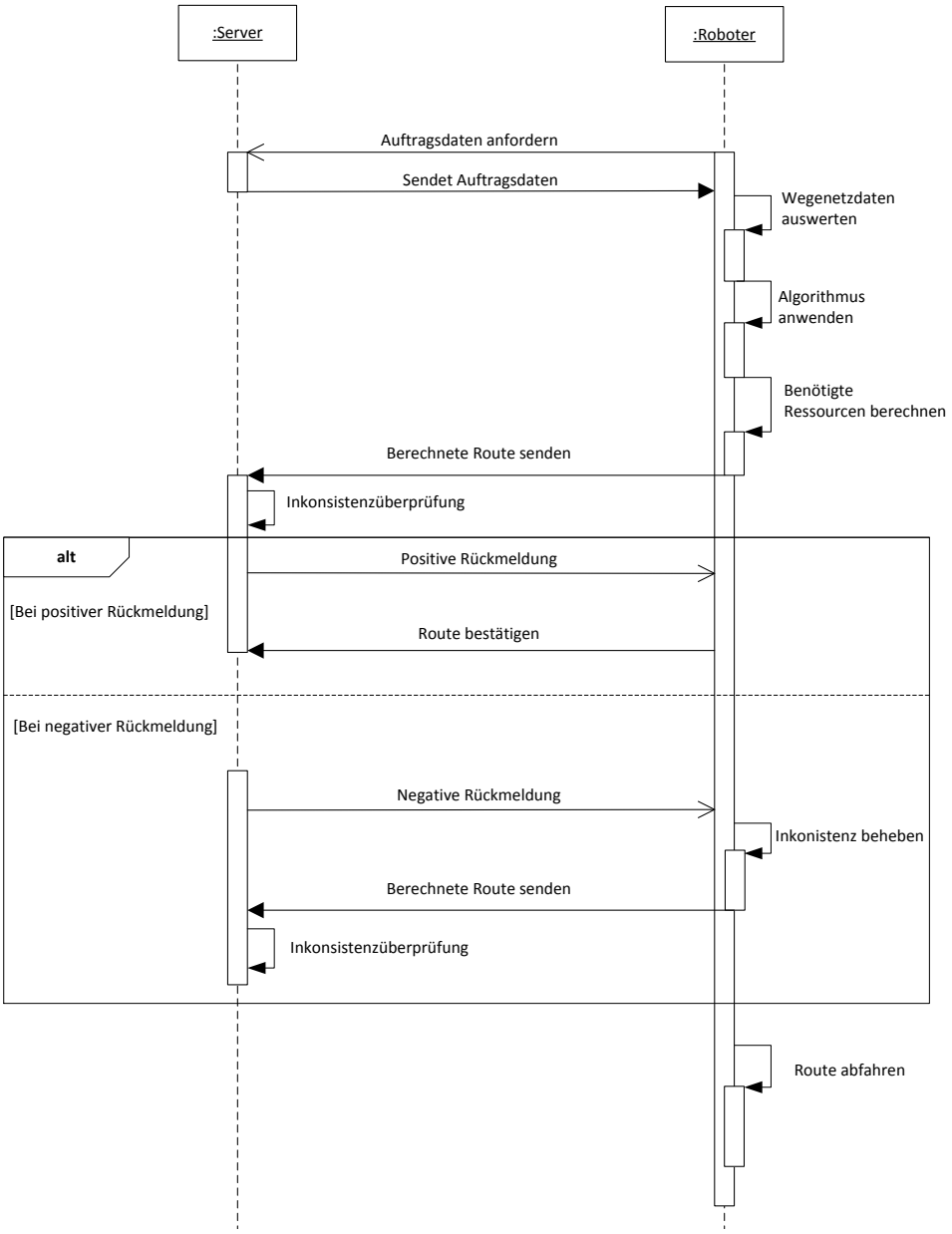

Abbildung 2.9.: Sequenzdiagramm $\langle F80 \rangle$ „Routenberechnung durch Roboter“

2.9. Analyse von Funktionalität $\langle F90 \rangle$: Spielstart durch Benutzer

Das Spiel wird durch den Benutzer gestartet.

Im Sequenzdiagramm, in der Abbildung 2.10, wird näher auf die Funktion eingegangen. Nachdem der Benutzer den Startknopf gedrückt hat, erfolgt die Initialisierung ($\langle F20 \rangle$). Wenn diese erfolgreich verläuft, wird das Spiel gestartet. Schlägt die Initialisierung fehl, wird das Spiel nicht gestartet.

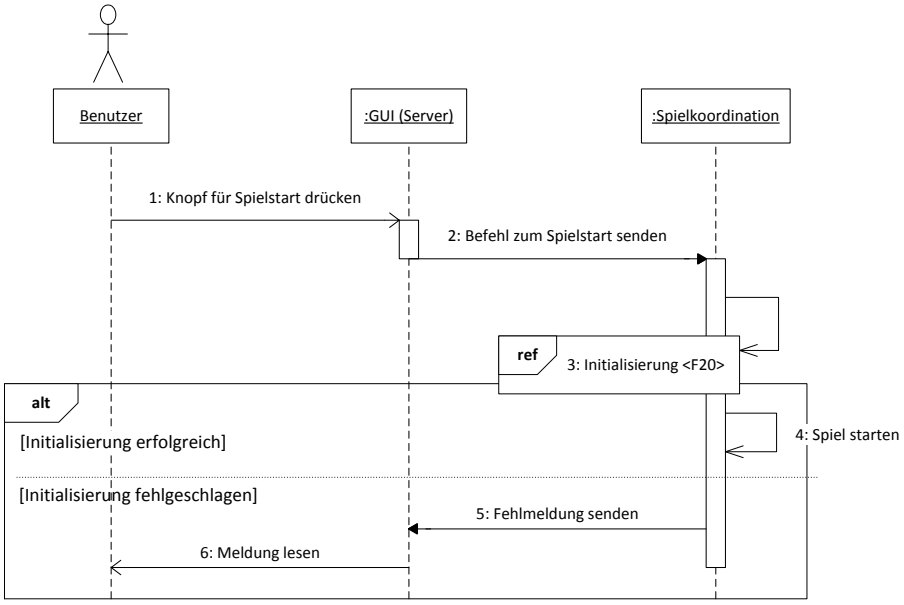

Abbildung 2.10.: Sequenzdiagramm $\langle F90 \rangle$ „Spielstart durch Benutzer“

2.10. Analyse von Funktionalität $\langle F100 \rangle$: Spieldarstellung der Spieleroberfläche

Die Spieleroberfläche zeigt das Wegenetz mit den Standorten von Industrien und Robotern an, dabei werden zusätzlich Statistiken geladen, die den Spielern über den derzeitigen Stand der jeweiligen Objekte informiert.

In der Abbildung 2.11 wird die Spieldarstellung wie folgt dargestellt.
Wenn das Spiel vom Spieler gestartet wurde, wird die Spieleroberfläche geladen. Auf der Spieleroberfläche sieht er das aktuelle Wegenetz, die Standorte und Status der Roboter sowie die Industriestandorte, die visualisiert in Echtzeit dargestellt werden.
Dabei zeigt sie auch die Details, wie zum Beispiel vorhandene Waren zum Transport und gelagerte Waren zur Weiterverarbeitung, der einzelnen Industrien an.
Um das Wegenetz anzeigen zu können, werden Wegenetzdaten (Standorte der Industrie usw.) benötigt, die der Server bereithält und der GUI zur Verfügung stellt. Diese werden auf der Spieleroberfläche angezeigt.
Die Darstellung der Roboterstandorte wird durch die ständige Kommunikation zwischen Robotern und Server aktuell gehalten.

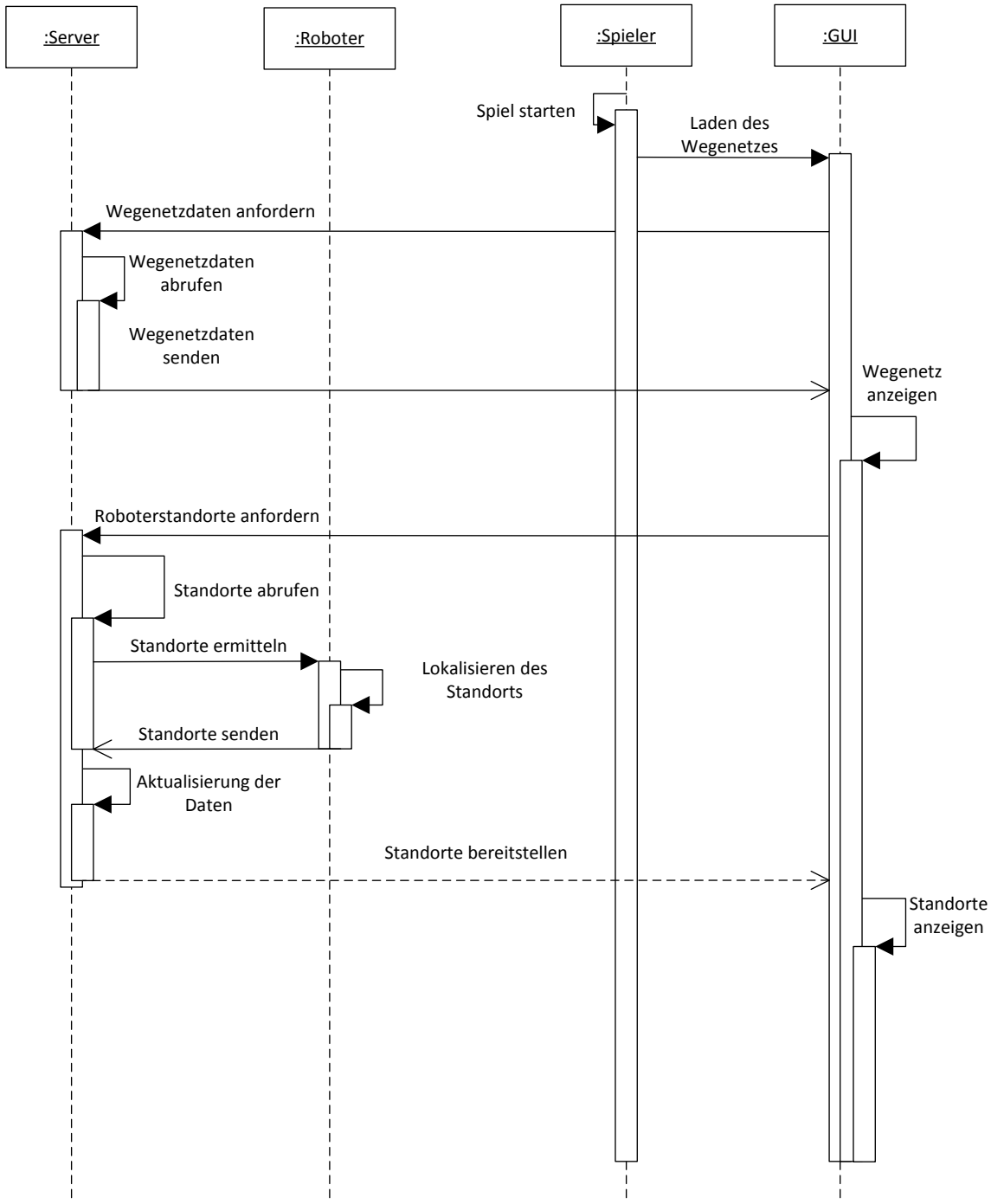

Abbildung 2.11.: Sequenzdiagramm ⟨$F100$⟩ „Spieldarstellung der Spieleroberfläche“

2.11. Analyse von Funktionalität $\langle F110 \rangle$: Statistikanzeige der Spieleroberfläche

Die Spieleroberfläche gibt Informationen über laufende Aufträge aus und zeigt zusätzlich Statistiken der Roboter an. Nachdem der Roboter seine Statistiken generiert und diese zur weiteren Verarbeitung an den Server übermittelt hat, werden dort Berechnungen durchgeführt. Das Ergebnis wird anschließend an den Client übertragen. Der Spieler kann die Ergebnisse dort aufrufen.

Das Sequenzdiagramm in der Abbildung 2.12, zeigt die Interaktion des Spielers beim Abruf von Statistiken der Roboter. Er kann diese am Client erst dann abrufen, wenn sie zuvor vom Server berechnet und an den Client übermittelt wurden. Dabei kommt es zwischen den Akteuren Server, Client und Roboter auf eine funktionierende Netzwerkverbindung an. Der Spieler arbeitet direkt am Client.

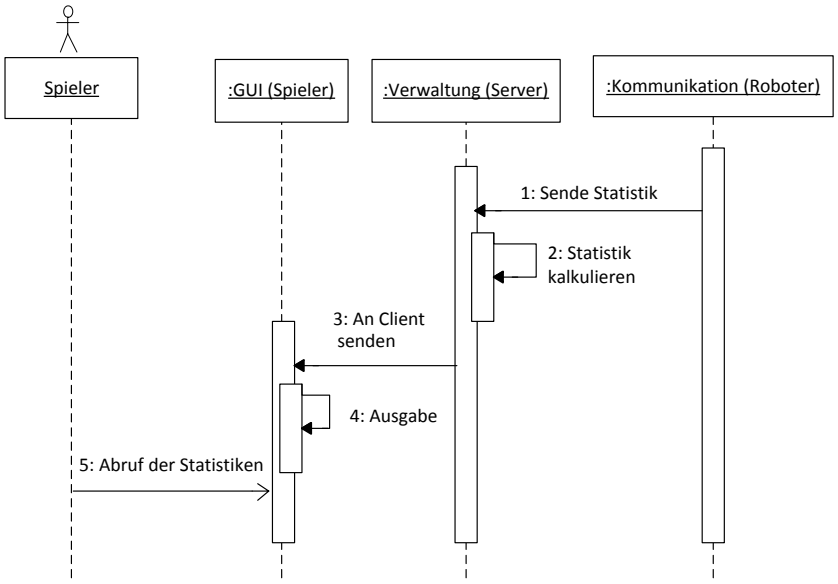

Abbildung 2.12.: Sequenzdiagramm $\langle F110 \rangle$ „Statistikanzeige der Spieleroberfläche“

2.12. Analyse von Funktionalität $\langle F120 \rangle$: Auktionen für Transportaufträge

Der Server veranstaltet in bestimmten zeitlichen Abständen Auftrags-Auktionen. Solche Auktionen sind die einzige Möglichkeit, an neue Aufträge zu gelangen. Dabei müssen alle Teilnehmer entweder ihr Gebot abgeben oder den Auftrag ablehnen. Der Teilnehmer mit dem niedrigsten Gebot gewinnt die Auktion.

Abbildung 2.13 stellt die Funktionalität in einem Sequenzdiagramm dar. Die Verwaltungkomponente des Servers steuert die Vergabe der Aufträge. Ein Auftrag wird entweder aus der Warteliste gewählt oder vom Benutzer vorgegeben. Eine neue Auktion wird gestartet und alle teilnehmenden Roboter bzw. Spieler werden mittels Bluetooth $\langle F10 \rangle$ bzw. TCP-Verbindung informiert. Bei eigenständigen Robotern berechnet die KI das bestmögliche Gebot. Im Falle eines Spielerroboters wird das Gebot vom Spieler selbst über die Spieler-GUI abgegeben. Nachdem alle Teilnehmer ein Gebot bzw. eine Ablehnung, beispielsweise im Falle von zu wenig Transportvolumen, abgegeben haben, ermittelt die Serververwaltung den Auktionsgewinner. Wurde der Gewinner gefunden, endet die Auktion. Spieler sollen über den Ausgang der Auktion benachrichtigt werden. Dem Roboter des Gewinners muss der Auftrag zugeteilt werden. Mit der Bestätigung, dass dieser in der Bearbeitungsliste des Roboters gespeichert wurde, wird der Auftrag daraufhin aus der Liste des Servers gelöscht.

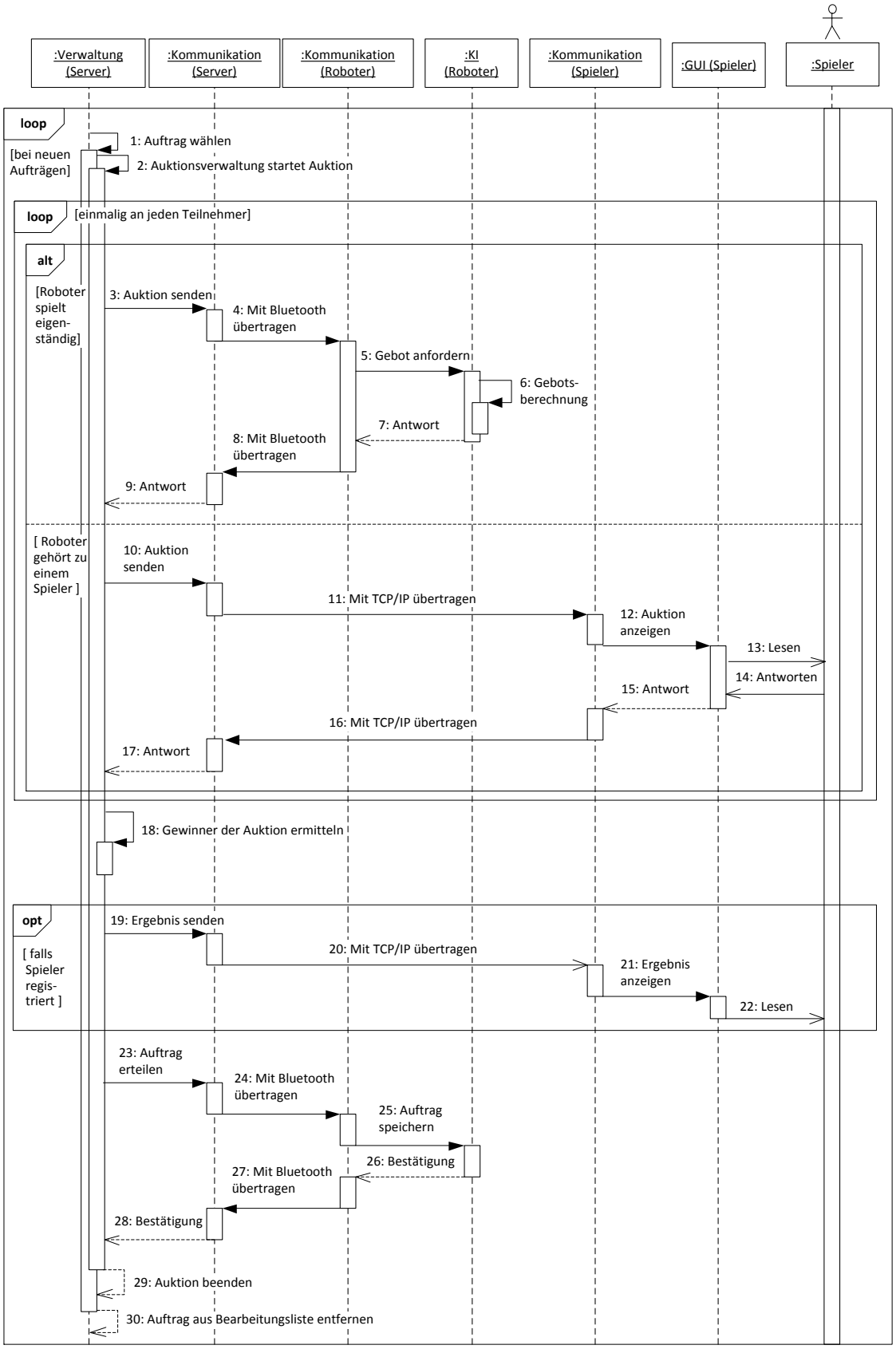

Abbildung 2.13.: Sequenzdiagramm $\langle F120 \rangle$ „Auktionen für Transportaufträge“

2.13. Analyse von Funktionalität $\langle F130 \rangle$: Vermeidung von Deadlocks

Damit es einen reibungslosen Spielverlauf gibt, muss es eine Möglichkeit geben wie Deadlocks vermieden werden können, dazu wurde ein Algorithmus gefunden der dies tut.

Im Sequenzdiagramm in der Abbildung 2.14 wird dieser näher dargestellt. Um Deadlocks zu vermeiden, müssen verschiedene Dinge geprüft werden: so muss geschaut werden, ob der Roboter eine Alternativ-Route zum Ziel finden kann, ohne die blockierte Kante zu verwenden. Ist dies möglich, so wird sie als aktuelle Route verwendet und die alte Route zum Ziel ersetzt. Da es nicht immer möglich ist, eine alternative Route zu finden, muss der Roboter für die Situation gesagt bekommen, was zu tun ist.
So wartet der Roboter eine definierte Zeit ab, bevor er nochmals nachfragt, ob die Route freigegeben wurde. Dies muss getan werden, da es gut sein kann, dass der andere Roboter einfach vorbei fährt, da dieser nicht diesen Streckenabschnitt befahren wollte oder schon eine alternative Route berechnet hat. Ist dies aber nicht der Fall und der benötigte Streckenabschnitt ist immer noch belegt, so wird geprüft, ob es eine anliegende Kante gibt, die derzeitig frei ist und zum Ausweichen benutzt werden kann. Beim Ausweichen fährt der Roboter eine gewisse Länge auf die freie Kante, wendet dort und fragt den Server, ob der benötige Streckenabschnitt nun zum Befahren freigegeben ist.
Falls aber keine Kante frei zum Ausweichen war, muss der Roboter, falls er sich nicht in einer Sackgasse befindet, zum vorherigen Knoten zurück fahren und dort ausweichen. Dadurch wird der Deadlock aufgelöst, da die Auflösung auch bei mehreren Robotern schrittweise erfolgt.

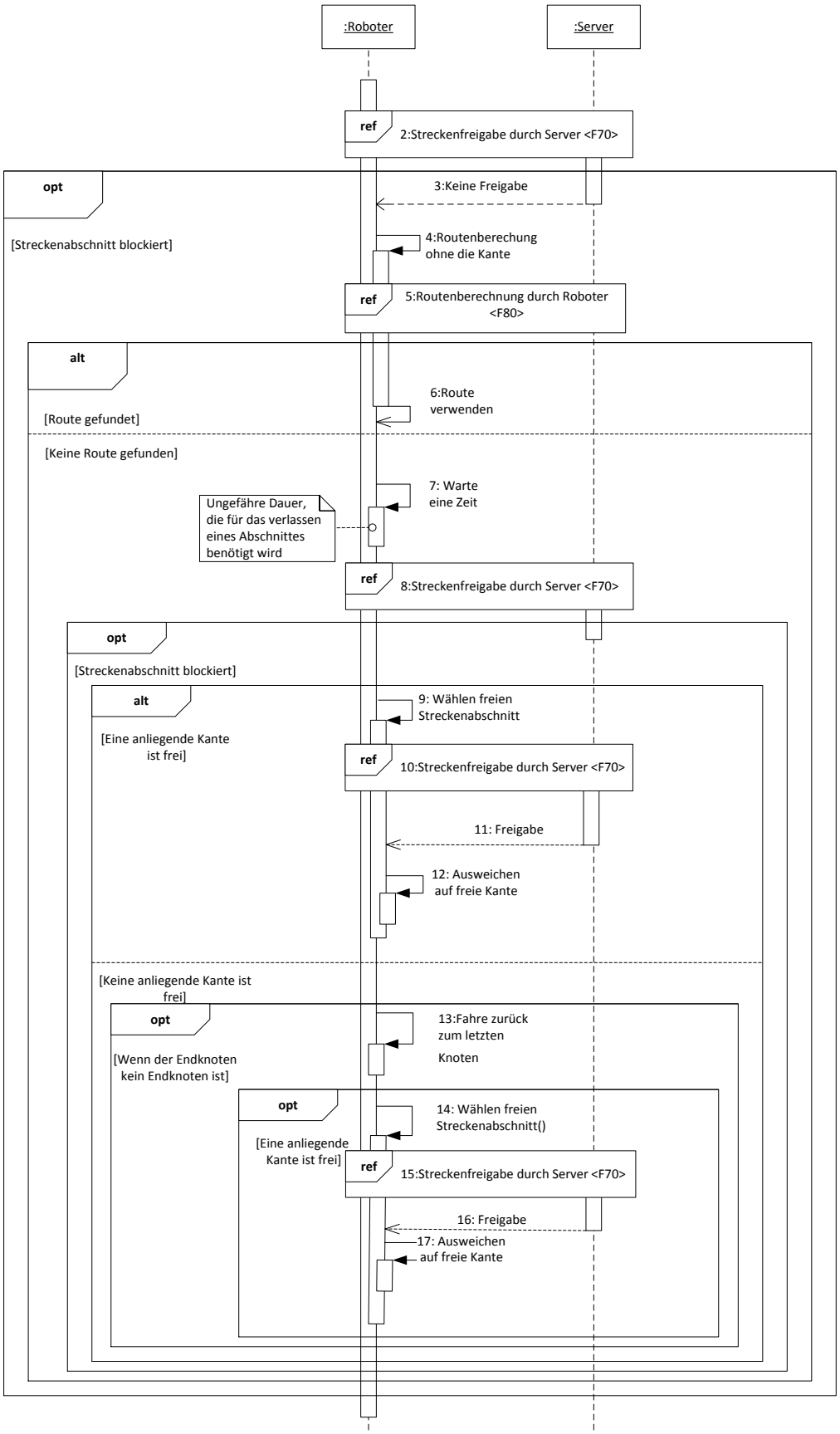

Abbildung 2.14.: Sequenzdiagramm $\langle F130 \rangle$ „Vermeidung von Deadlocks“

2.14. Analyse von Funktionalität $\langle F140 \rangle$: Spieldarstellung über Benutzeroberfläche

Die Benutzeroberfläche zeigt das Wegenetz, Aufträge und Statistiken an.
In der Spieldarstellung werden Aufträge über die GUI an den Server gesendet, Roboter können aus dem Spiel entfernt werden und das Wegenetz wird vom Server aktualisiert und angezeigt.
Aufträge: Der Benutzer erstellt über die GUI des Servers einen Auftrag. Dieser wird von der GUI an die Spielkoordination übermittelt. Die Spielkoordination leitet den Auftrag anschließend an die Auktionsverwaltung weiter, welche diesen Auftrag dann zur Auktion anbietet ($\langle F120 \rangle$).

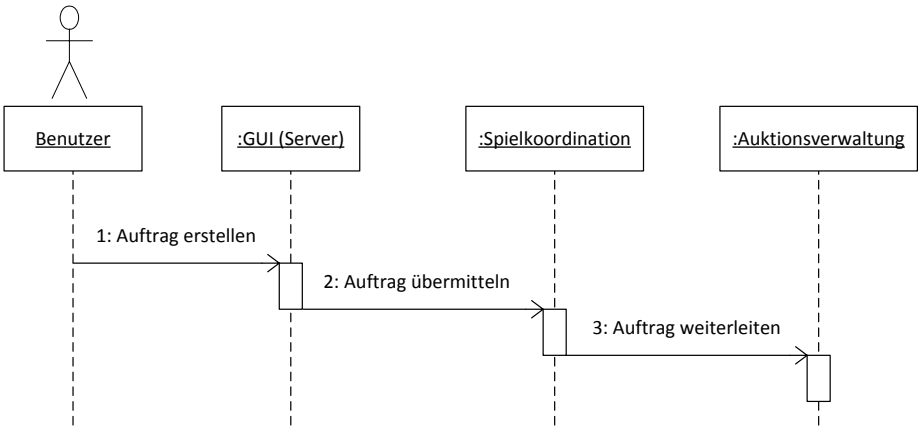

Abbildung 2.15.: Sequenzdiagramm $\langle F140 \rangle$ zum Senden eines Auftrages

Roboter: Wenn der Benutzer einen Roboter aus dem Spiel entfernen möchte, wählt er diesen über die GUI des Servers aus. Dieser übermittelt anschließend den Befehl zur Entfernung des Roboters an die Spielkoordination.

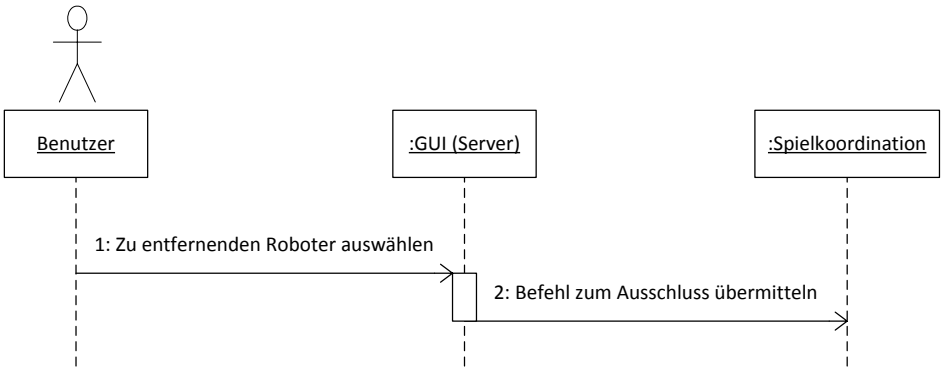

Abbildung 2.16.: Sequenzdiagramm $\langle F140 \rangle$ zum Entfernen eines Roboters

Wegenetz: Damit der Benutzer einen aktuellen Überblick über das Wegenetz mit belegten und freien Streckenabschnitten und den Positionen der Roboter hat, wird die Visualisierung in der GUI in regelmäßigen Abständen aktualisiert. Dazu schickt die Spielkoordination eine Anforderung zur Aktualisierung an die Wegenetzverwaltung. Diese ruft das aktuelle Wegenetz ab und sendet die Informationen an die Spielkoordination, damit sie an die GUI übertragen werden können.

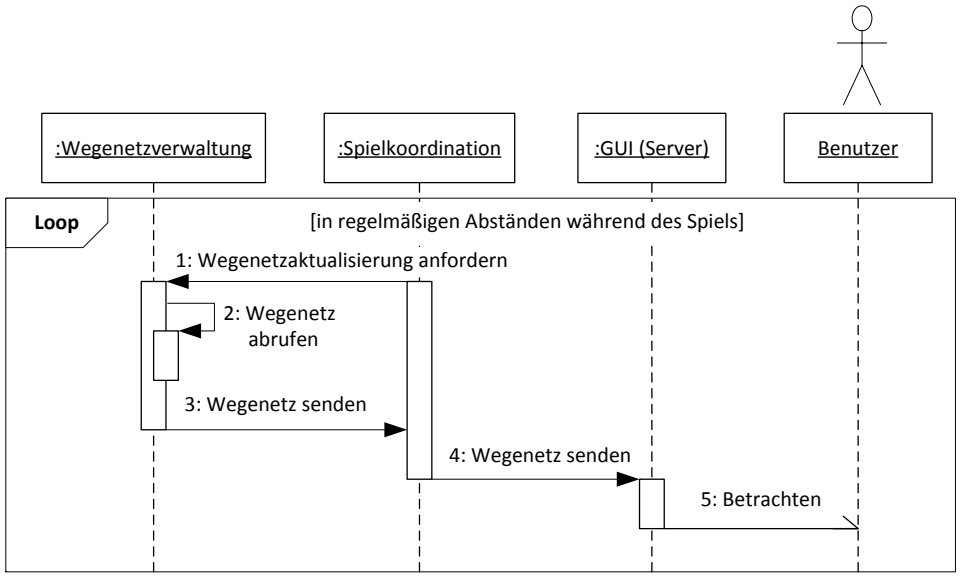

Abbildung 2.17.: Sequenzdiagramm $\langle F140 \rangle$ zur Visualisierung des Wegenetzes

3. Resultierende Softwarearchitektur

Eine gute Softwarearchitektur ist grundlegend für eine gelungene Umsetzung der im Pflichtenheft formulierten Produktfunktionen. In diesem Kapitel wird die aus ihnen resultierende Softwarearchitektur vorgestellt.

3.1. Komponentenspezifikation

Zunächst werden die Komponenten der Software mit Hilfe eines Diagramms und einer näheren Beschreibung vorgestellt.

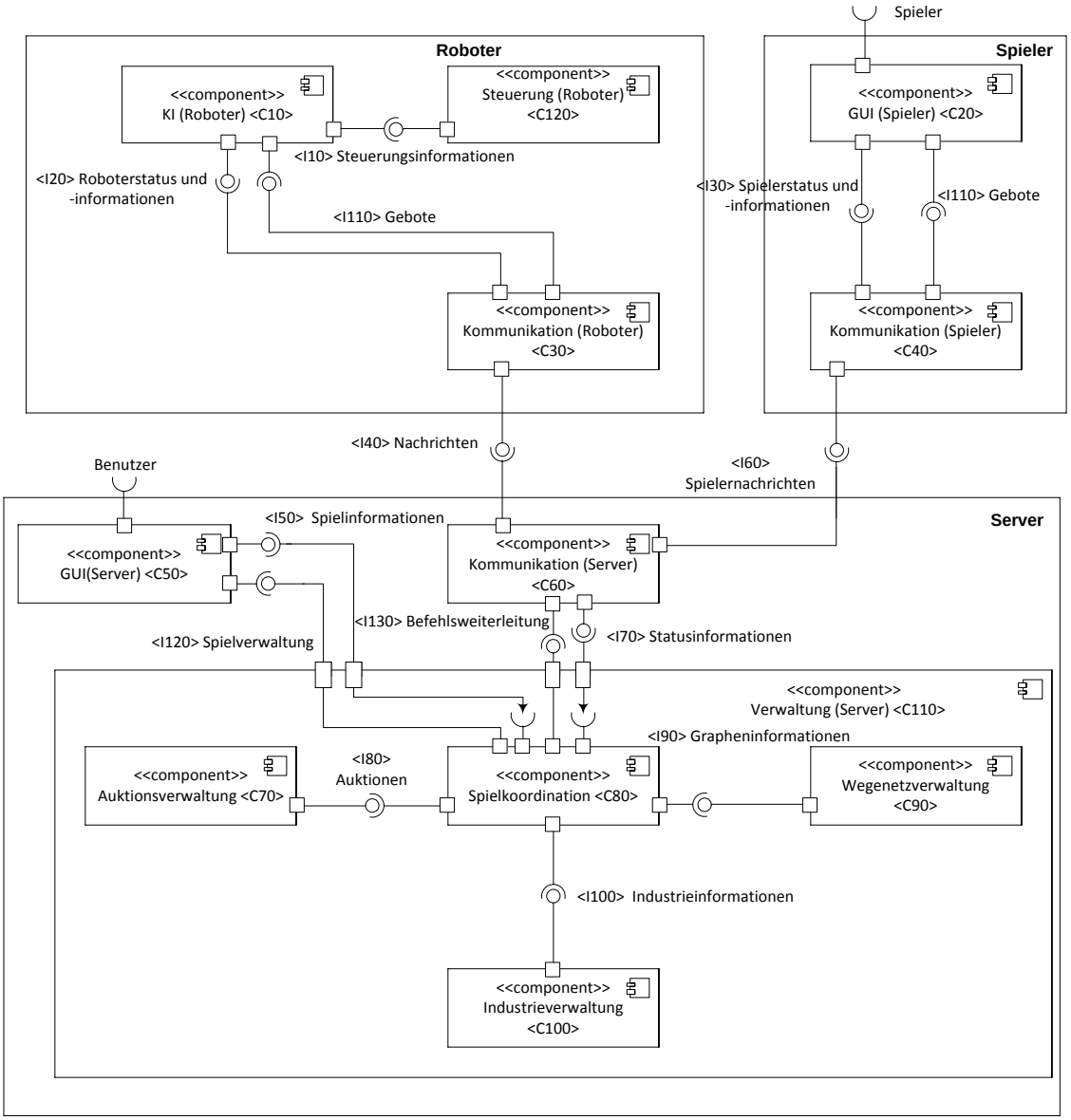

Abbildung 3.1.: Komponentendiagramm

Es haben sich insgesamt zwölf Komponenten herauskristallisiert, die die komplette Funktionalität des Programms abdecken. Sie werden nachfolgend vorgestellt.

Komponente $\langle C10 \rangle$**: <KI (Roboter)>**
Die KI ist für die Entscheidungen zuständig, die der Roboter als Teilnehmer treffen muss.

Komponente $\langle C20 \rangle$**: <GUI (Spieler)>**
Die GUI visualisiert für den Spieler den Ablauf des Spiels und ermöglicht Eingaben wie z.B Gebote und Routenänderungen.

Komponente $\langle C30 \rangle$**: <Kommunikation (Roboter)>**
Das Kommunikationsmodul des Roboters ermöglicht die Kommunikation zum Server via Bluetooth.

Komponente $\langle C40 \rangle$**: <Kommunikation (Spieler)>**
Das Kommunikationsmodul des Spielers ermöglicht die Kommunikation zum Server via TCP/IP.

Komponente $\langle C50 \rangle$**: <GUI (Server)>**
Die GUI erlaubt dem Benutzer die Administration des Spiels.

Komponente $\langle C60 \rangle$**: <Kommunikation (Server)>**
Das Kommunikationsmodul des Servers ermöglicht die Kommunikation zum Roboter sowie zum Spieler über das jeweilige Protokoll.

Komponente $\langle C70 \rangle$**: <Auktionsverwaltung>**
Die Auktionsverwaltung nimmt Auftragsauktionen und Gebote entgegen und ermittelt einen Gewinner von den Auktionen.

Komponente $\langle C80 \rangle$**: <Spielkoordination>**
Die Spielkoordination ist für den reibungslosen Ablauf des Spiels zuständig.

Komponente $\langle C90 \rangle$**: <Wegenetzverwaltung>**
Die Wegenetzverwaltung speichert das zugrunde liegende Wegenetz. Sie ist für die Standorterfassung der Roboter sowie für die Streckenfreigaben und -sperrungen zuständig.

Komponente $\langle C100 \rangle$**: <Industrieverwaltung>**
Die Industrieverwaltung administriert die Industrien. Dies bedeutet, dass die hergestellten Produkte durch Aufträge wegtransportiert werden können. Zur Herstellung von Produkten werden Waren, die angeliefert werden, benötigt.

Komponente $\langle C110 \rangle$**: <Verwaltung (Server)>**
Die Verwaltung des Servers vereinigt die Komponenten $\langle C80 \rangle$, $\langle C90 \rangle$, $\langle C70 \rangle$ und $\langle C100 \rangle$. Damit dient sie als Verwaltungskomponente des Spiels.

Komponente $\langle C120 \rangle$**: <Steuerung (Roboter)>**
Die Steuerung des Roboters sorgt dafür, dass der Roboter seine Strecke gemäß der Befehle der KI abfährt.

3.2. Schnittstellenspezifikation

Innerhalb des Komponentendiagramms kristallisierten sich nachfolgende Schnittstellen sowie deren Operationen heraus. Sie werden nachfolgend erläutert.

Schnittstelle $\langle I10 \rangle$: <Steuerunginformationen>

Operation	Beschreibung
Steuerungsinformationen senden	Die KI sendet Befehle zur Manövrierung an die Steuerung.
Aktionspunkte zurückgeben	Die Steuerung übermittelt die Ankunft an Aktionspunkten, an denen die KI neue Befehle übermitteln muss.

Schnittstelle $\langle I20 \rangle$: <Roboterstatus und -informationen>

Operation	Beschreibung
Standort übermitteln	Die KI gibt über das Kommunikationsmodul ihre Position an.
Grunddaten senden	Die KI erhält vom Kommunikationsmodul zu Beginn Grunddaten.
Auftragsangebot weiterleiten	Das Kommunikationsmodul des Roboters leitet Auftragsangebote weiter.
Auftragsbestätigungen weiterleiten	Das Kommunikationsmodul des Roboters leitet Auftragsbestätigungen weiter.

Schnittstelle $\langle I30 \rangle$: <Spielerstatus und -informationen>

Operation	Beschreibung
Spielerstatus und -informationen übertragen	Die Kommunikation aktualisiert Spielstatus und -informationen (z.B. Wegenetzdaten).

Schnittstelle $\langle I40 \rangle$: <Nachrichten>

Operation	Beschreibung
Nachricht senden	Operationen von $\langle I20 \rangle$ und $\langle I110 \rangle$ werden zwischen den beiden Kommunikationsmodulen von Roboter und Server weitergeleitet.

Schnittstelle $\langle I50 \rangle$: <Spielinformationen>

Operation	Beschreibung
Spielinformationen übermitteln	Die Verwaltung bzw. Spielkoordination aktualisiert Spielstatus und -informationen (z.B. Wegenetzdaten).

Schnittstelle $\langle I60 \rangle$: <Spielernachrichten>

Operation	Beschreibung
Spielernachrichten übertragen	Operationen von $\langle I30 \rangle$ und $\langle I110 \rangle$ werden zwischen den beiden Kommunikationsmodulen von Spieler und Server weitergeleitet.

Schnittstelle $\langle I70 \rangle$: <Statusinformationen>

Operation	Beschreibung
Statusinformationen anfragen	Die Spielkoordination stellt Anfragen zur Ermittlung von Statusinformationen an das Kommunikationsmodul.

Schnittstelle $\langle I80 \rangle$: <Auktionen>

Operation	Beschreibung
Erstellte Auktionen weiterleiten	Die Spielkoordination sendet vom Benutzer oder von der Industrieverwaltung eingegebene Aufträge an die Auktionsverwaltung.
Auktionen an Teilnehmer empfangen	Die Auktionsverwaltung übergibt zu versteigernde Aufträge an die Spielkoordination, die diese an die Roboter bzw. Spieler weiterleitet.
Gebote von Teilnehmern senden	Die Spielkoordination sendet Gebote an die Auktionsverwaltung.
Auktionsgewinner empfangen	Die Spielkoordination empfängt den Auktionsgewinner von der Auktionsverwaltung.

Schnittstelle $\langle I90 \rangle$: <Grapheninformationen>

Operation	Beschreibung
Roboterposition senden	Die Spielkoordination sendet die Standorte der Roboter an die Wegenetzverwaltung.
Streckensperrung senden	Die Spielkoordination sendet die Sperrung einer Strecke an die Wegenetzverwaltung.
Streckenfreigabe senden	Die Spielkoordination sendet die Freigabe einer Strecke an die Wegenetzverwaltung.
Grunddaten senden	Die Spielkoordination sendet zu Beginn des Spiels die grundlegenden Daten des Wegenetztes an die Wegenetzverwaltung.
Status einer Strecke anfragen	Die Spielkoordination fragt den Status einer Strecke bei der Wegenetzverwaltung an.

Schnittstelle $\langle I100 \rangle$: <Industrieinformationen>

Operation	Beschreibung
Grunddaten senden	Die Spielkoordination sendet zu Beginn des Spiels die grundlegenden Daten der Industrien an die Industrieverwaltung.
Warenankunft übermitteln	Die Spielkoordination sendet nach erfolgreicher Abarbeitung eines Auftrags die Ankunft der transportierten Waren an die Industrieverwaltung.
Aufträge weiterleiten	Die Spielkoordination leitet Aufträge von der Industrieverwaltung an die Auktionsverwaltung weiter.

Schnittstelle $\langle I110 \rangle$: <Gebote>

Operation	Beschreibung
Gebote senden	Die GUI des Spielers bzw. die KI des Roboters sendet Gebote an das jeweilige Kommunikationsmodul zur Weiterleitung an den Server.

Schnittstelle $\langle I120 \rangle$: <Spielverwaltung>

Operation	Beschreibung
Aufträge erstellen	Die GUI gibt neu erstellte Aufträge an die Spielkoordination weiter.
Roboter entfernen	Die GUI gibt an die Spielkoordination den passenden Befehl weiter, dass ein Roboter entfernt werden soll.
Spiel starten	Die GUI gibt den Befehl zum Spielstart an die Spielkoordination weiter.
Streckensperrungen senden	Die GUI sendet zu sperrende Strecken an die Spielkoordination.

Schnittstelle $\langle I130 \rangle$: <Befehlsweiterleitung>

Operation	Beschreibung
Befehle weiterleiten	Befehle von Roboter bzw. Spieler, die durch das Kommunikationsmodul empfangen werden, werden an die Spielkoordination weitergeleitet$\langle I110 \rangle$.

3.3. Protokolle für die Benutzung der Komponenten

Grundsätzlich sind viele Softwarekomponenten dieses Projekts entweder spezifisch auf die entsprechende Anwendungssituation bezogen (z.B. Spielkoordination) oder so gestaltet, dass eine Adaptation auf einen neuen Anwendungsfall einer Neuentwicklung der Komponenten gleichkäme (z.B. GUI). Zwei Komponenten fallen hierbei jedoch heraus und lassen sich relativ einfach aus dem vorliegenden Projekt isolieren und könnten dann anderweitig verwendet werden.
Zunächst ist dies mit der Auktionsverwaltung möglich, die das Erschaffen von Auktionen, das Überwachen von Geboten und das Feststellen das „Auktionssiegers“ zum Ziel hat.

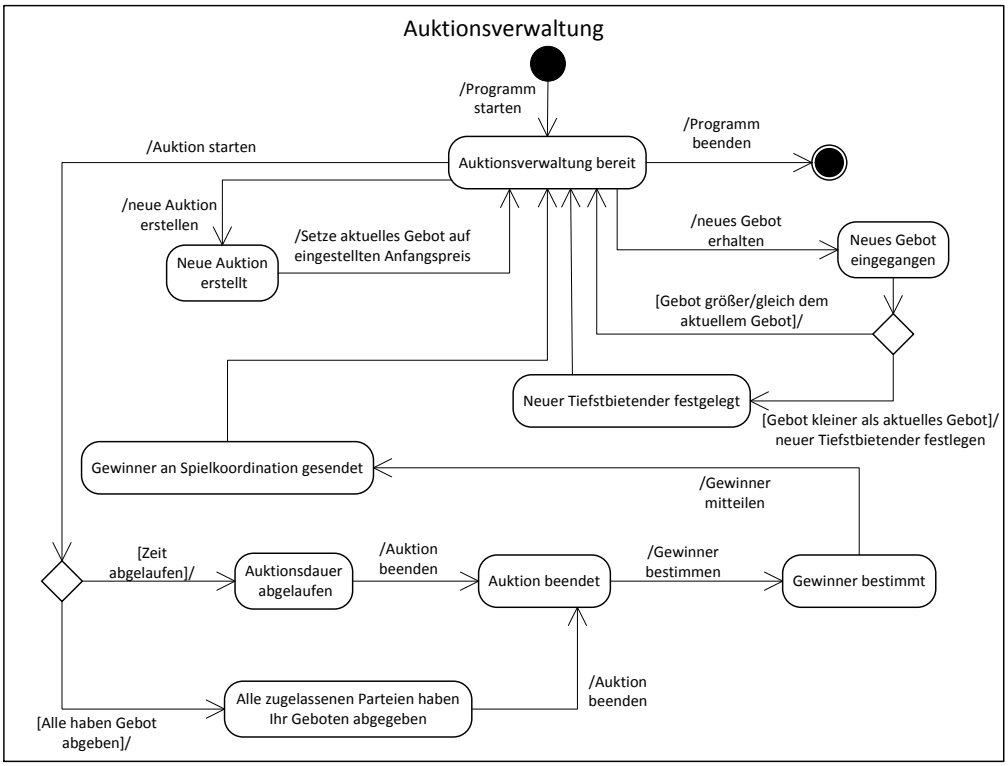

Abbildung 3.2.: Auktionsverwaltung

Diese Komponente könnte zur generellen Auktionserstellung für z.B. Programme die ein Angebot-Nachfrage-System implementieren wollen (z.B Wirtschaftssimulation).

Die zweite Komponente ist das grundlegende Kommunikationsinterface, das in folgender (oder ähnlicher) Form in den drei Komponenten Kommunikation (Server), (Roboter) und (Spieler) vorkommen muss.

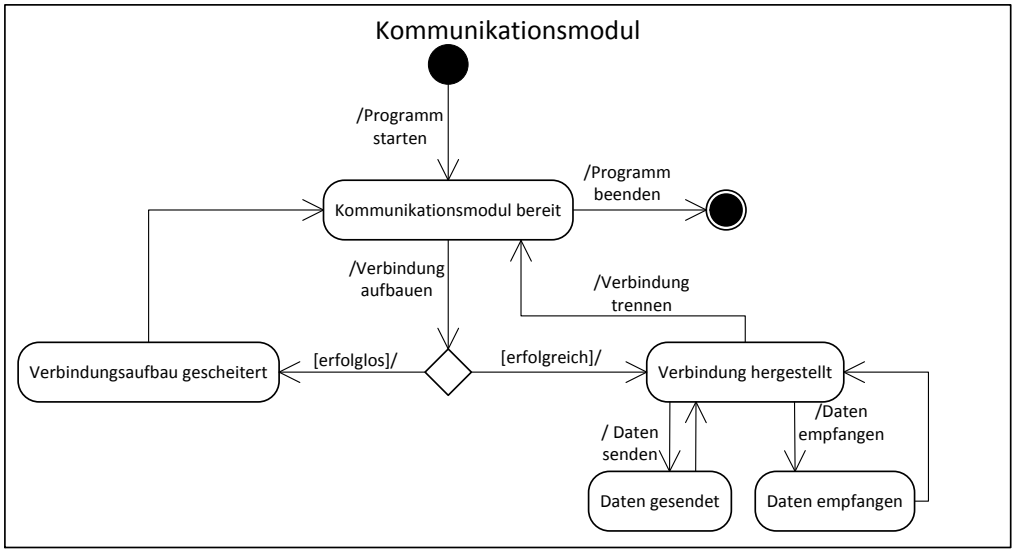

Abbildung 3.3.: Kommunikationsmodul

Mithilfe dieser Komponente kann die Kommunikation zwischen verschiedenen Systemen (NXT-Roboter ⇔ Server) gewährleistet werden.

Eine weitere Komponente ist die Spielkoordination $\langle C80 \rangle$, dargestellt durch folgendes Statechartdiagramm (Abbildung 3.4):

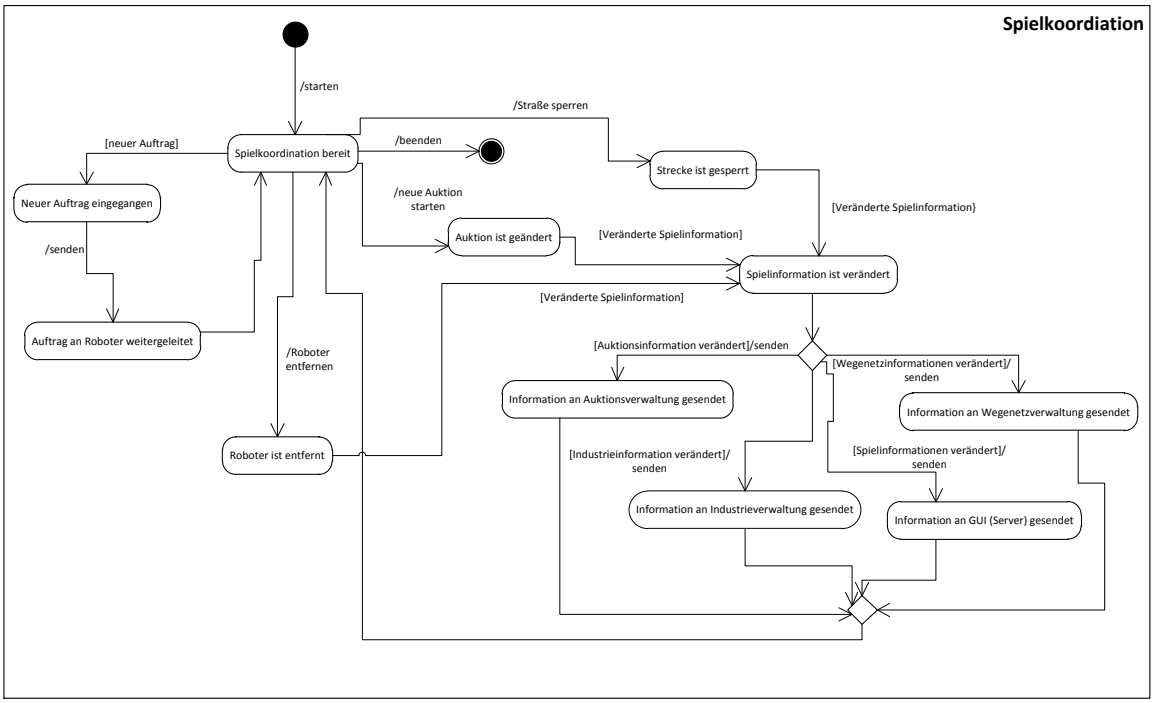

Abbildung 3.4.: Spielkoordination

Sobald ein neuer Auftrag existiert, wird er durch die Spielkoordination an den entsprechenden Roboter übermittelt. Eine Änderung der Spielinformation kann durch drei Aktionen ausgelöst werden:

- Ein Roboter wird entfernt
- Eine Straße wird gesperrt
- Eine Auktion findet statt

Die Resultate dieser Aktionen müssen zur GUI des Servers übermittelt werden. Zusätzlich wird jede Information zur jeweilig zuständigen Verwaltungskomponente übermittelt. Neue Aufträge werden von der Spielkoordination an die betreffenden Roboter weitergeleitet. Sobald ein Roboter entfernt oder eine Strecke gesperrt wurde, werden die Verwaltungskomponenten der Industrie, des Wegenetzes, der Auktion sowie die GUI des Servers aktualisiert.

Eine weitere Komponente ist die Industrieverwaltung $\langle C100 \rangle$, dargestellt durch folgendes Statechartdiagramm (Abbildung 3.5):

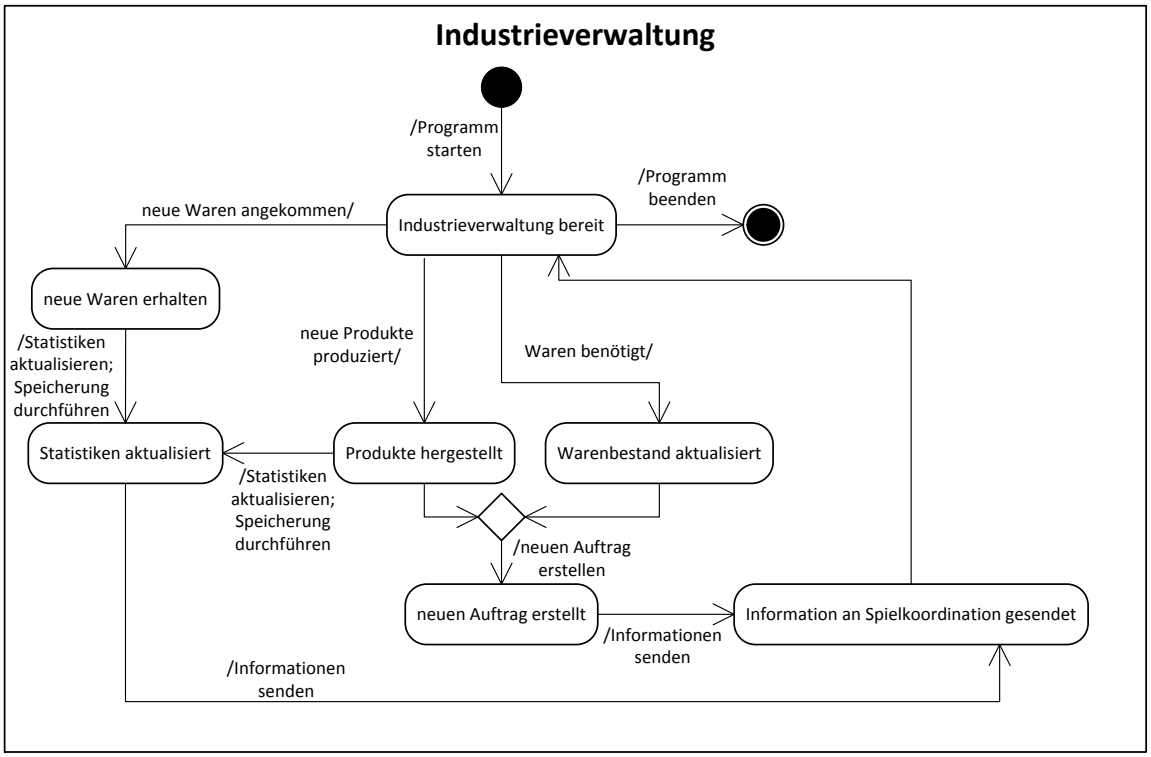

Abbildung 3.5.: Industrieverwaltung

Diese verwaltet die Industrien und benötigt Waren zu Erzeugung von Produkten. Falls Produkte hergestellt wurden oder Waren benötigt werden erstellt Sie neue Aufträge, welche sie an die Spielkoordination weitergibt. Zusätzlich aktualisiert sie die Statistiken, wenn neue Waren eintreffen oder Produkte hergestellt wurden.

Die im folgenden Zustandsdiagramm (Abbildung 3.6) darstellte Komponente ist die GUI(Spieler) $\langle C20 \rangle$:

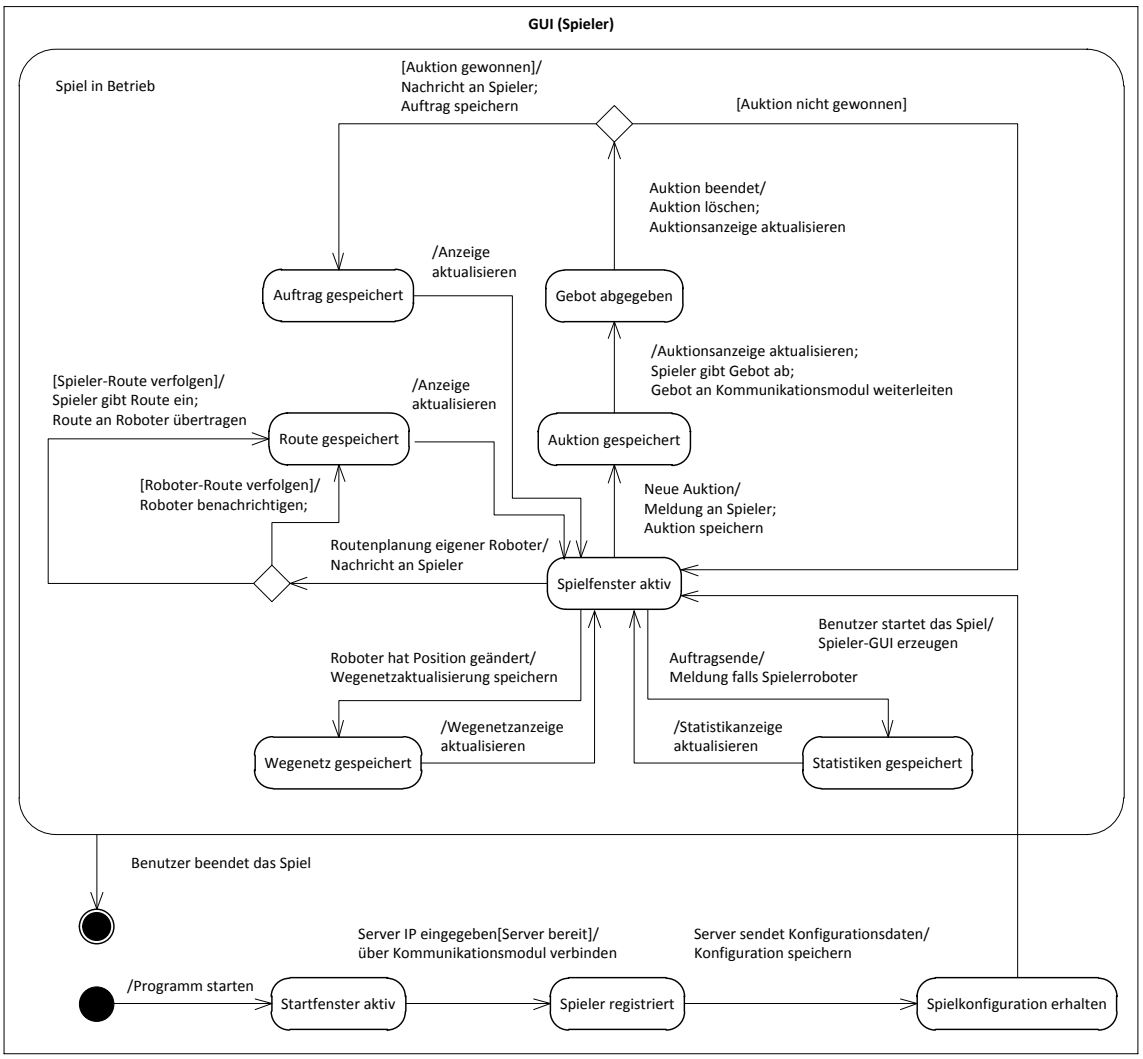

Abbildung 3.6.: Spieleroberfläche

Es wird dargestellt, wie der Spieler auf bestimmte Aktionen anderer Komponenten im Spiels reagieren kann. Neue Auktionen verlangen das Handeln des Spielers, indem er Gebote abgibt. Im Fall einer Routenberechnung hat er aber auch die Möglichkeit dem Roboter die Routenwahl zu überlassen. Die Spieler-GUI liefert eine ähnliche Darstellung wie die Benutzer-GUI. Jedoch besitzt der Spieler nur einen eingeschränkten Zugriff auf Informationen beispielsweise andere Roboter betreffend. Die GUI(Spieler) beinhaltet zudem noch einen verwaltenden Teil, der sich um das Erkennen und Verarbeiten der empfangenen und zu sendenden Objekte kümmert.

Das nachfolgende Zustandsdiagramm (Abbildung 3.7) stellte die Komponente GUI(Benutzer) $\langle C50 \rangle$ dar:

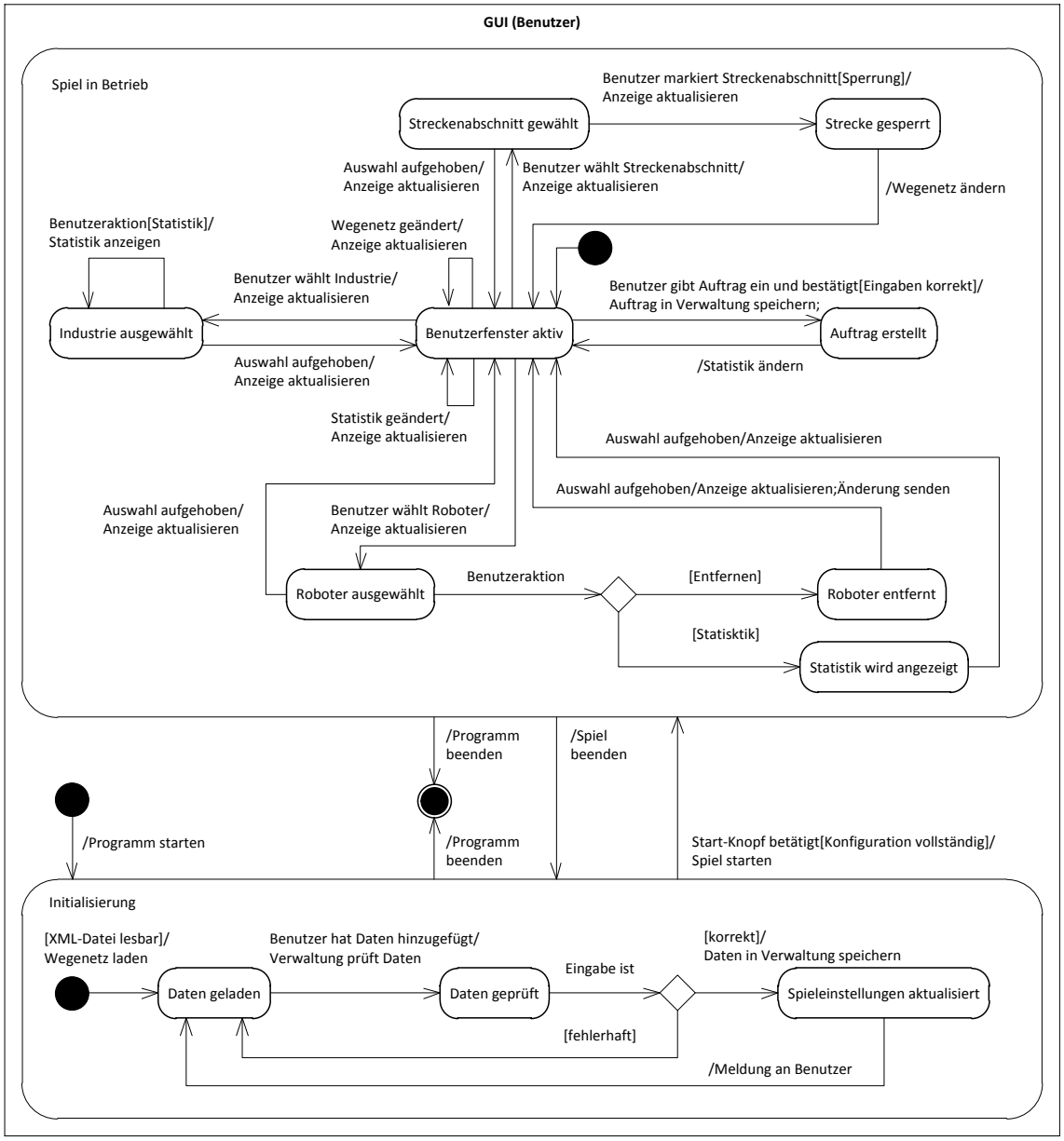

Abbildung 3.7.: Benutzeroberfläche

Der Benutzer kann während des Spiels jederzeit Aktionen durchführen, die anderen Komponenten als Ereignis dienen, in einen anderen Zustand zu wechseln. So kann er z. B. Aufträge erstellen oder Strecken sperren, wodurch zum einen die Auktionsverwaltung und zum anderen die Wegenetzverwaltung aktiv werden.

Eine weitere Komponente ist die Wegenetzverwaltung $\langle C90 \rangle$, dargestellt durch folgendes Statechartdiagramm (Abbildung 3.8):

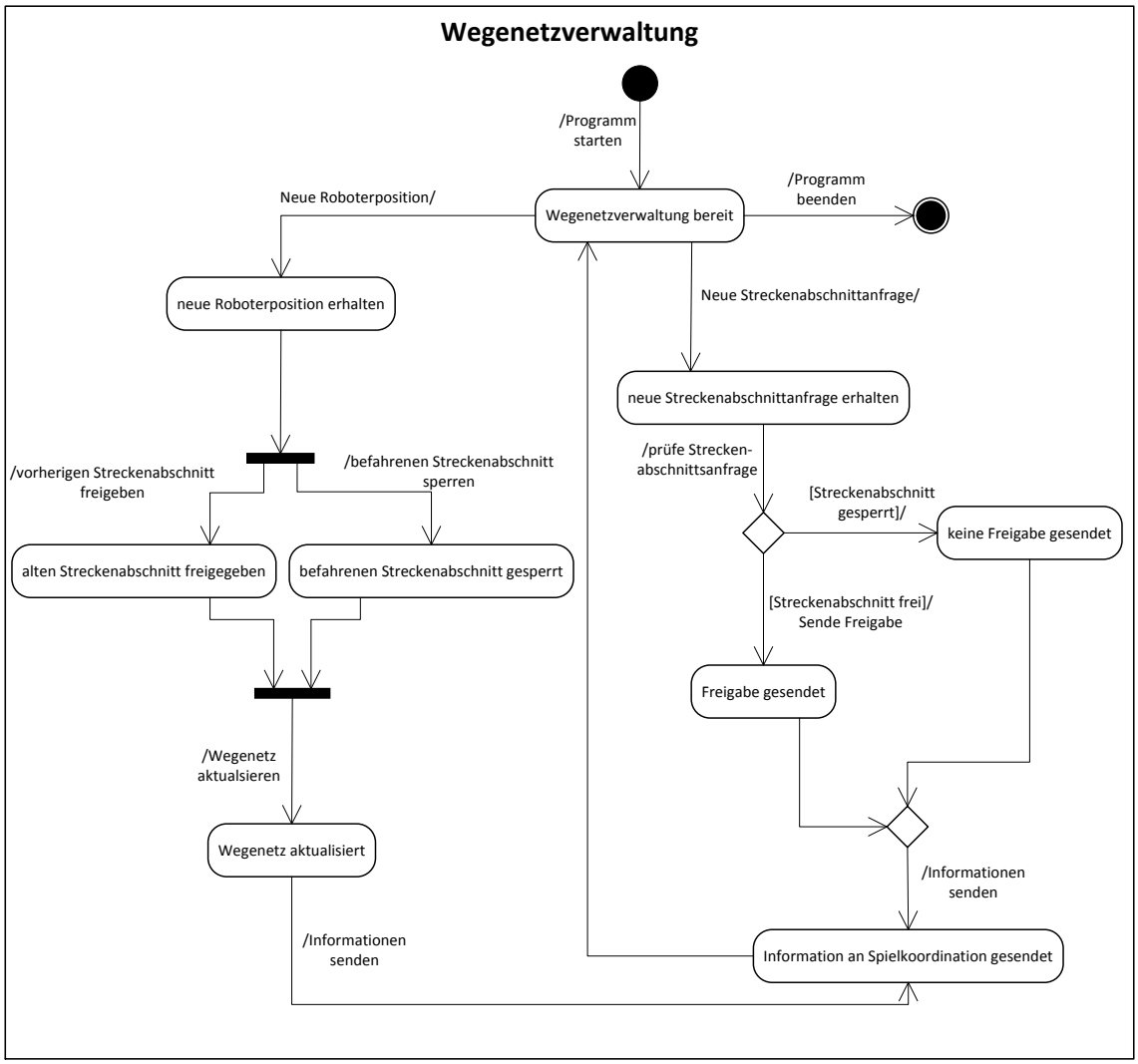

Abbildung 3.8.: Wegenetzverwaltung

Diese verwaltet das Wegenetz und gibt Freigaben zum befahren einer Strecke an die Roboter. Sie erhält die neuen Positionen der Roboter und aktualisiert dann das Wegenetz und gibt die aktuellen Informationen an die Spielkoordination weiter. Außerdem erhält sie Anfragen von den Robotern, ob ein Streckenabschnitt befahrbar ist. Falls dieser frei ist wird eine Freigabe an den Roboter über die Spielkoordination gesendet. Falls der Streckenabschnitt gesperrt ist wird keine Freigabe gesendet.

Die primäre Komponente des Roboters ist die KI $\langle C10 \rangle$, dargestellt durch folgendes Statechartdiagramm (Abbildung 3.9):

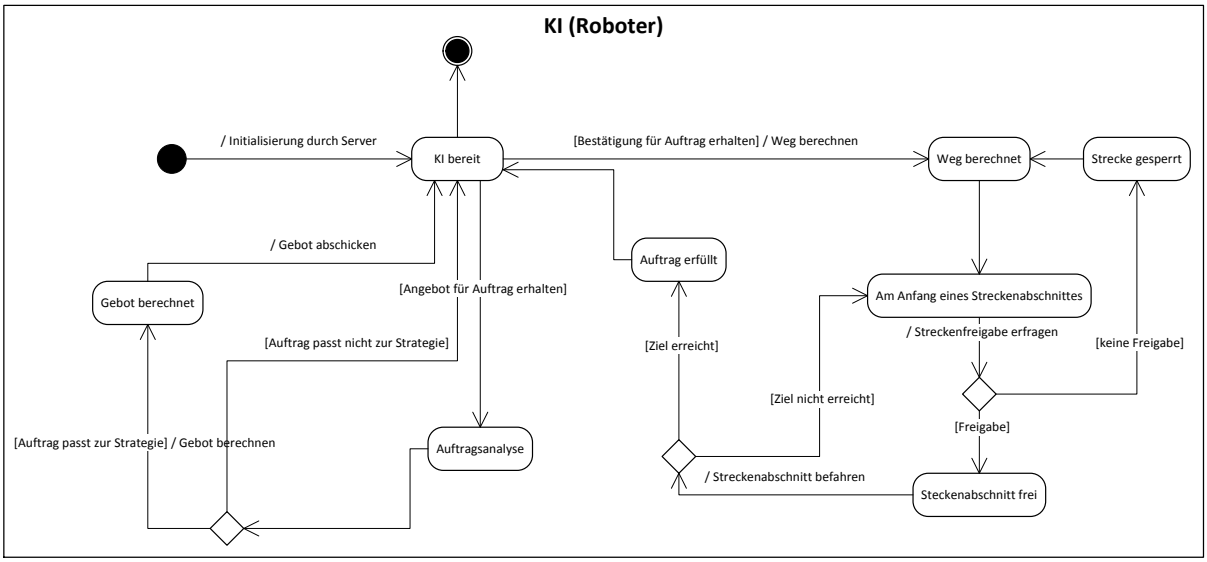

Abbildung 3.9.: KI (Roboter)

Sie betrachtet Angebote zu Aufträgen und entscheidet gemäß ihrer Strategie, ob sie auf diesen Auftrag bieten möchte. Wenn sie die Bestätigung für einen Auftrag erhält, berechnet sie die zu fahrende Strecke, um ihren Auftrag abzuarbeiten. Für jede neue Strecke erfragt sie eine Freigabe, damit es nicht zur Kollision mit anderen Robotern kommt. Sie hat einen Auftrag erfolgreich abgeschlossen, wenn sie am Ziel angekommen ist und die transportierten Waren abliefern kann. Die letzte Komponente die Steuerng des Roboters $\langle C120 \rangle$, dargestellt durch folgendes Statechartdiagramm (Abbildung 3.10):

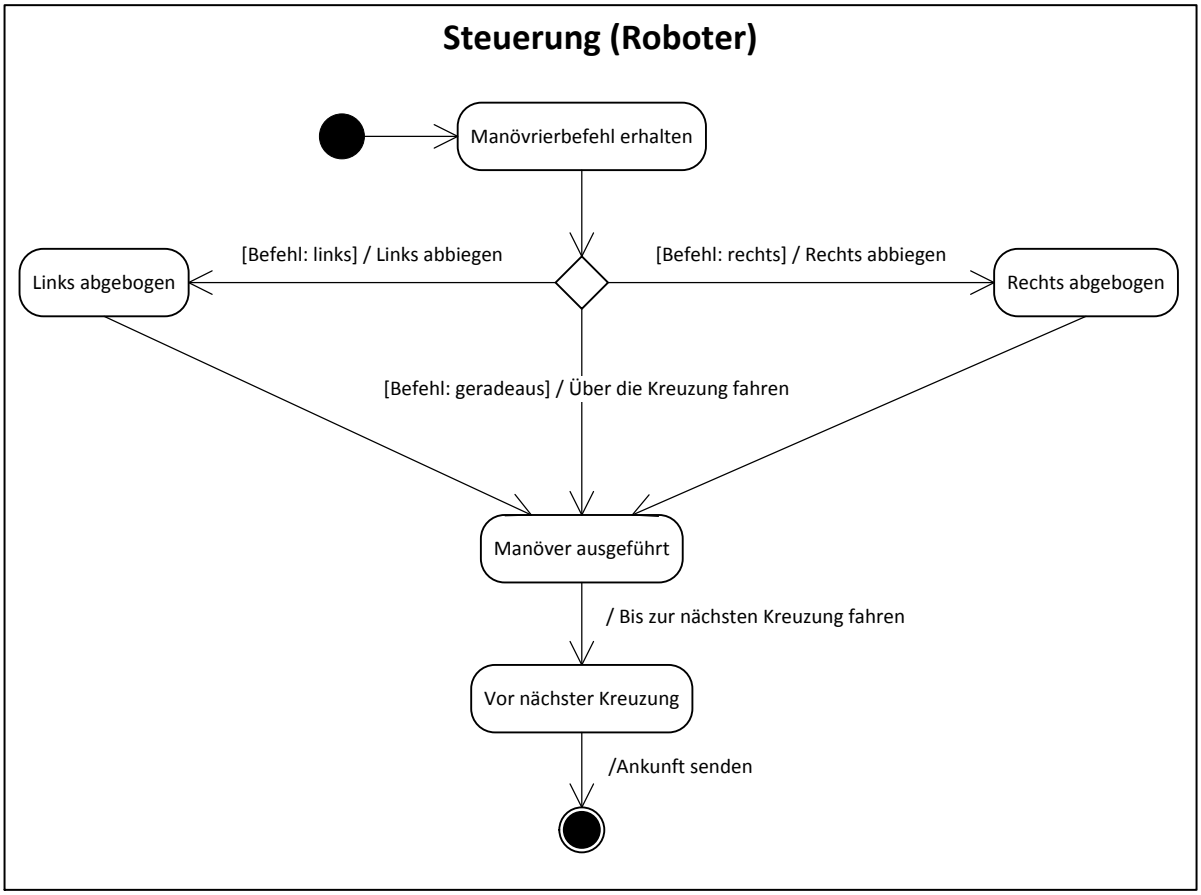

Abbildung 3.10.: Steuerung des Roboters

Diese dient lediglich zur Manövrierung des Roboters auf dem Wegenetz. Wenn die Steuerung vor einer neuen Kreuzung mit einem Manövrierbefehl aufgerufen wird, prüft sie zuerst die Richtung, die sie einschlagen muss und biegt ab bzw. fährt über die Kreuzung geradeaus. Danach fährt sie über den nächsten Abschnitt bis vor die nächste Kreuzung und übermittelt die Ankunft an die KI.

4. Verteilungsentwurf

In der Abbildung 4 sieht man, dass das System in 3 Teile eingeteilt ist. Die Verbindung zwischen Server und NXT-Roboter geschieht über Bluetooth, hierbei werden Statuswerte und Kommandos übertragen. Die Kommunikation zwischen Spieler PC und Server PC, erfolgt über TCP/IP, dabei werden hauptsächlich Statistiken und Einstellungen übertragen.

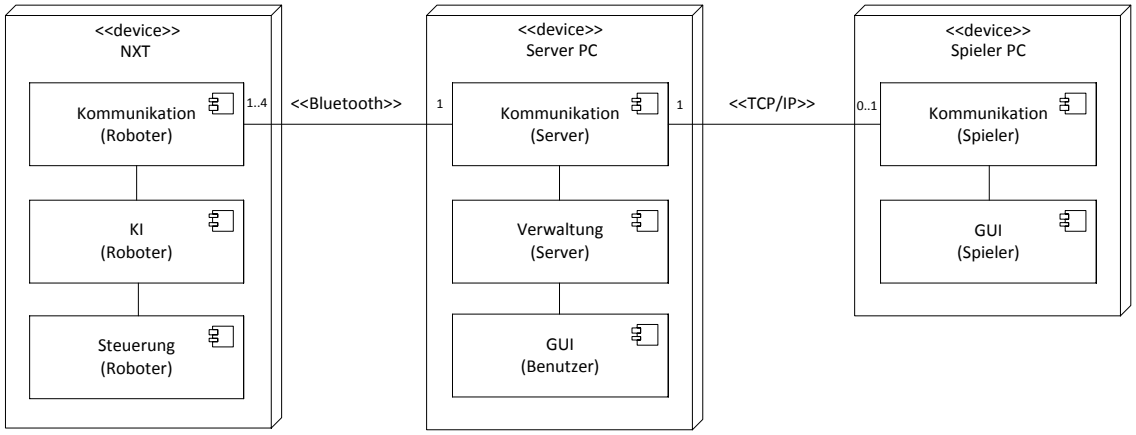

Abbildung 4.1.: Verteilungsdiagramm

5. Erfüllung der Kriterien

In diesem Abschnitt wird beschrieben, wie die einzelnen Kriterien des Pflichtenheftes erfüllt werden und worauf geachtet wird. Es wird zunächst auf die Musskriterien, dann auf die Soll- und Kann-Kriterien und zuletzt auf die Abgrenzungskriterien eingegangen.

5.1. Musskriterien

Die folgenden Kriterien sind unverzichtbar und müssen durch das Produkt erfüllt werden:

$\langle RM1 \rangle$ „*Der Roboter erkennt den Straßenverlauf in Form von schwarzen Linien und folgt ihnen.*“
Dieses Kriterium wird folgendermaßen umgesetzt: Der Roboter besitzt zwei Lichtsensoren, die die Helligkeit misst. Der Straßenverlauf wird durch dunkle Linie dargestellt, die der Roboter von dem hellen Untergrund unterscheiden kann. Der Weiße Untergrund wird als Helligkeitsstufe „0“ und der dunkle Untergrund misst einen Wert „>0“. Zur Markierung von Straßenenden und Kreuzungen wurden ebenfalls schwarze quer zur Straßenrichtung verlaufende Streifen geklebt. Kreuzungen oder Industriestandorte erkennt der Roboter, indem einer der beiden Sensoren einen dunklen Untergrund erkennt.

$\langle RM2 \rangle$ „*Der Roboter führt angenommene Aufträge selbständig durch.*“
Wenn der Server einen Auftrag zu vergeben hat, wird dieser für alle im Spiel vorhandenen Roboter freigegeben. Der Server startet eine Auktion an der alle Roboter, die genug Ressourcen haben (z.B. genügend Treibstoff) teilnehmen können. Mit Hilfe einer Methode, die die Gebote anfordert, werden alle getätigten Gebote gesammelt und ausgewertet. Bevor der Roboter ein Gebot abgibt, berechnet er selbständig mit einer Methode, ob er den Auftrag annehmen kann und welches das bestmögliche Gebot ist. Erst wenn er diese Berechnungen durchgeführt hat, nimmt er an der Auktion teil und sendet dem Server sein Gebot zu. Nach einem Zeitfenster wird die Auktion geschlossen und der Gewinner ermittelt. Der Server sendet ihm den Auftrag zu. Der Roboter führt den Auftrag aus.

$\langle RM3 \rangle$ „*Der Roboter kann zwischen zwei Punkten den Weg selbstständig finden.*“
Dieses Kriterium wird einer Methode, die den Weg zwischen zwei Knotenpunkten mit der Hilfe von einem implementierten Algorithmus findet, umgesetzt.

⟨*RM*4⟩ „*Der Roboter muss Aufträge ersteigern können.*“
Über eine Bluetooth-Verbindung kommuniziert der Roboter mit dem Server, der ihm mit Hilfe der Auktionsverwaltung die Auktion freigibt und ein Gebot anfordert. Dieser teilt dem Server nach seiner Berechnung, ebenfalls über die Bluetooth-Verbindung, seine Antwort mit. Das Gebot wird in einer „Liste“ gespeichert und ausgewertet. Erhält der Roboter den Auftrag und hat somit die Auktion gewonnen, speichert er den Auftrag und teilt den Erhalt dem Server mit.

⟨*RM*5⟩ „*Der Roboter muss mit dem Server kommunizieren können.*“
Die Kommunikation zwischen Server und Roboter wird via Bluetooth hergestellt. Beide Komponenten sind über ihre Kommunikationsmodule miteinander vernetzt.

⟨*RM*6⟩ „*Der Roboter hat eine maximale Ladekapazität.*“
Die maximale Ladekapazität wird durch den Administrator oder einem Standardwert festgelegt. Dieser wird während des Spiels nicht verändert. Die KI des Roboters überprüft bei jeder Gebotsberechnung, ob die Größe, der bereits angenommen Aufträge, addiert mit der Größe, des zu ersteigernden Auftrags, mehr als die maximale Ladekapazität ergibt. Nur wenn der Wert niedriger ist, kann der Roboter den neuen Auftrag annehmen und ersteigern. Wurde ein Gebot abgegeben, so überprüft der Server die Korrektheit des Transportvolumens, damit möglich Manipulationen ausgeschlossen sind.

⟨*RM*7⟩ „*Der Roboter muss Deadlock-Situationen erkennen und vermeiden können.*“
Diese Funktion wird wie folgt umgesetzt: ist der Weg blockiert, so versucht er eine Alternativroute zu berechnen, geht dies nicht und es vergeht eine gewisse Zeit, bemerkt der Roboter, dass es sich hierbei um einen Konflikt handeln muss. Er versucht dann auf einen anliegenden Weg auszuweichen und somit den anderen Roboter den Weg freizumachen. Ist ihm das nicht möglich, so muss er zurückfahren wenn er nicht an einem Endknoten sich befindet.
Eine Vermeidung der Deadlocks wird durch die stetige Kommunikation zwischen Server und allen auf dem Spielfeld vorhandenen Robotern umgesetzt.

⟨*RM*8⟩ „*Der Server muss das Wegenetz verwalten.*“
Der Graph ist in einer .xml-Datei gespeichert und wird vom Server importiert. Mit Hilfe von Methoden werden alle Graphendetails ausgelesen und Attributen zugeordnet.

⟨*RM*9⟩ „*Der Server muss Aufträge verwalten.*“
Die Verwaltungskomponente des Servers vergibt Aufträge, die sich in entweder in einer Warteliste befinden oder vom Benutzer vorgegeben werden. Via Bluetooth bzw. TCP-Verbindung werden die Spielteilnehmer über eine Auktion informiert. Aufträge werden in einer Liste gespeichert, die die jeweiligen Attribute (Auftragsnummer, Größe, Art usw.) enthält.

⟨*RM*10⟩ „*Der Server muss den Roboter hinsichtlich der Kartendaten initialisieren.*“
Der Roboter wird zunächst über eine Bluetooth-Verbindung registriert. Die Daten zwischen den jeweiligen Kommunikationskomponenten werden als Stream versendet. Bei der Initialisierung des Roboters wird dem Roboter das Wegenetz zugesendet, der dieses speichert und seine Teilnahmebestätigung mitteilt.

⟨*RM*11⟩ „*Der Server muss die Preisbildung verwalten.*“
Die Serververwaltung bildet den Preis je nach Auftragslage. Dabei schaut er, wie hoch die Warenverfügbarkeit des Produktes ist. Dafür wird geschaut, auf welchem Verfügbarkeitslevel, sich die Ware befindet. Ist nur eine gewisse Menge verfügbar, gilt sie als knapp. Somit wird der Preis erhöht. Der neue Preis wird gespeichert und freigegeben.

⟨*RM*12⟩ „*Der Server verwaltet die Kommunikation zwischen Benutzeroberfläche und Roboter.*“
Während des Spiels verwaltet der Server die durch ausstehende oder geleistete Aufträge anfallenden Daten und aktualisiert die sich daraus ergebenden Statistiken (Preisbildung, Gewinne). Diese Statistiken werden später sowohl in der Spieler-GUI ⟨*F*110⟩, als auch in der Benutzer-GUI angezeigt. Die Verwaltung zwischen der Benutzeroberfläche und den Robotern wird via Bluetooth bzw. TCP/IP-Verbindung über das Kommunikationsmodul des Servers ermöglicht.

⟨*RM*13⟩ „*Der Server muss mit mehreren Robotern gleichzeitig kommunizieren.*“
Für die gleichzeitige Kommunikation mit allen teilnehmenden Robotern, stellt der Server zu Beginn mit jedem einzelnen Roboter eine Verbindung her, die bis zum Ende des Spieles aufrecht gehalten wird. ⟨*RM*14⟩ „*Der Server muss den Standort der Roboter kennen.*“
Durch die stetige Kommunikation mit dem Server, wird an jedem Abschnitte einer Kante die Position gesendet, dadurch ist die Verfolgung des Roboters möglich.

⟨*RM*15⟩ „*Der Server muss Auftragsauktionen anbieten.*“
Über die Kommunikationsmodule wird den Robotern bzw. Spieler mitgeteilt, ob eine neue Auftragsauktion freigegeben wurde. Die Verwaltungskomponenten des Servers steuert die Vergabe der Aufträge. Bei jedem Start einer Auktion werden alle mitspielenden Roboter informiert. Ist ein neuer Auftrag vorhanden, werden die Daten vom Server bereitgestellt und an das Kommunikationsmodul der Roboter versendet, damit sie an der Auktion teilnehmen können. Darauf folgt eine Bestätigung oder Ablehnung der Roboter, die an den Server gesendet wird. Der Server entfernt daraufhin die Kennung der Roboter, für diesen Auftrag. Wenn alle Roboter eine Antwort an den Server gesendet haben, werden die Gebote verglichen und der Roboter mit dem niedrigsten Gebot wird sein Gewinn mitgeteilt.

⟨*RM*16⟩ „*Der Server muss Geschäftsstatistiken für die Roboter führen.*“
Während des Spiels sendet der Roboter, bei Änderung, seine Daten an den Server, aus diesen Daten wird eine Geschäftsstatistik berechnet. Über die Kommunikationsmodule findet ein stetiger Datenabgleich statt.

⟨*RM*17⟩ „*Der Server muss es ermöglichen, dass Ressourcen und Produkte ergänzt werden können.*“
Das Ergänzen von Ressourcen und Produkten wird durch die Eingabemöglichkeit der Benutzeroberfläche ermöglicht. Der Benutzer erstellt über die GUI des Servers einen Auftrag, der an die Spielkoordination übermittelt wird. Diese leitet den Auftrag an die Auktionsverwaltung weiter, die den Auftrag in einer Auktion anbietet. Industrien können ebenfalls über die Benutzeroberfläche ergänzt werden. Dies ist allerdings nur während der Spielpause möglich. Wird eine neue Industrie angelegt, wird sie in die vorhandene Liste eingefügt.

⟨*RM*18⟩ „*Die Spieleroberfläche muss das Wegenetz visualisieren.*“
Der Server übermittelt die gespeicherten Wegenetzdaten an das Spielerinterface. Zudem übermittelt er die reservierten Abschnitte des Wegenetzes mit den Kennungen der Roboter. Die Darstellung erfolgt in der Form einer 2D-Ansicht.

⟨*RM*19⟩ „*Die Spieleroberfläche muss eine Übersicht über die Aufträge bieten.*“
Dies wird folgendermaßen umgesetzt. Mit Hilfe der Spieler GUI wird ein Fenster anzeigt, welches alle notwendigen Daten, z.B. das Wegenetz und die Auftragsinformationen des eigenen Roboters, beinhaltet. Der Spieler kann aussuchen, welche Aufträge angezeigt werden (zurzeit verfügbare, schon abgearbeitete oder zurzeit bearbeitete Aufträge). Anhand eines Filters, der gesetzt wird, werden die gewünschten Aufträge aufgelistet. Klickt man auf den Auftragsdetails-Button, wird eine Anfrage an den Server gestellt, der die aktuellen Daten heraussucht und an die Spieler GUI übergibt.

⟨*RM*20⟩ „*Die Spieleroberfläche muss eine Statistik über die Roboter anzeigen.*“
Der Server sammelt Spieldaten jedes mitspielenden Roboters und speichert diese zwischen. Anhand dieser Daten werden durchschnittliche Gewinne und der gesamte Gewinn errechnet und der GUI zur Verfügung gestellt. Die berechneten Werte werden mit der Kennung des Roboters auf der Spieleroberfläche angezeigt.

⟨*RM*21⟩ „*Die Spieleroberfläche muss die Teilnahme an Auftragsauktionen ermöglichen.*“
Durch eine Eingabemaske auf der Spieleroberfläche ist es möglich, an einer Auktion teilzunehmen. Dazu wird der gewünschte Auftrag ausgewählt. Bei einem Gebot wird die Auftragsnummer mit der Kennung des Roboters verknüpft und an den Server gesendet. Dieser registriert das Gebot des Roboters/Spielers und beendet die Auktion, wenn er von allen mitspielenden Robotern eine Antwort erhalten hat und teilt dem Sieger den Gewinn der Auktion mit.

⟨*RM*22⟩ „*Die Visualisierung des Wegenetz muss den Standort und Status von Robotern und Industriestandorten enthalten.*“
Der Server stellt die Status des Roboters und der Industriestandorte bereit. Diese werden auf der Oberfläche dargestellt. Die Aktualität der Daten wird durch die stetige Kommunikation zwischen Server und Roboter gewährleistet.

⟨*RM*23⟩ „*Die Karte des Wegenetzes kann leicht vom Benutzer an die örtlichen Bedingungen (aufgeklebtes Straßennetz) angepasst werden.*“
Dies wird durch eine einfache Oberfläche gewährleistet, die die Möglichkeit bietet durch einfache Handhabung Knoten und Kanten hinzuzufügen oder diese zu entfernen. Dies ist nur vor dem Spielbeginn möglich, um eine Inkonsistenz des Spiel zu vermeiden.

⟨*RM*24⟩ „*Die Spieloberfläche muss Details (vorhandene Waren zum Transport, gelagerte Waren zur Weiterverarbeitung) einzelner Industrien anzeigen.*“
Dieses Kriterium wird mit der Hilfe des „Industriedetails anzeigen“-Button gewährleistet. Dieser führt eine Anfrage aus, sobald er gedrückt wurde, und listet alle Details der Industrien auf.

⟨*RM*25⟩ „*Die Anordnung und der Produktionszyklus von Industrien soll leicht änderbar bzw.anpassbar sein.*“
Die Anpassung der Industrien wird durch den Administrator durchgeführt. Über die Administratoroberfläche kann ein Industriestandort mit all seinen Details angeschaut werden und die Attribute abgeändert und an den Server gesendet werden. Dies ist nur vor dem Spielbeginn möglich.

5.2. Sollkriterien

Die Erfüllung folgender Kriterien für das abzugebende Produkt wird angestrebt:

⟨*RS*1⟩ „*Der Roboter verwaltet seinen (virtuellen) Benzinverbrauch.*“
Wird mit Hilfe eines Counters, der die Zählmenge dekrementiert, umgesetzt. Zum Start des Spiels wird die Treibstoffmenge aller Roboter auf den gleichen Wert gesetzt. Im Laufe des Spiels zählt der Counter zeitabhängig nach unten. Bei jedem erfolgreich abgeschlossenen Auftrag wird die Treibstoffmenge wieder erhöht. Ist der Counter bei Null angekommen, hat der Roboter/das Team verloren.

⟨*RS*2⟩ „*Der Roboter zeigt eine Statusmeldung mit Transportinformationen an.*“
Über die Displayausgabe des Roboters werden die Statusmeldungen angezeigt. Hier werden der aktuelle Gewinn, Ladekapazität, Anzahl der angenommenen Aufträge, das aktuelle Ziel sowie Treibstoffmenge angezeigt.

⟨*RS*3⟩ „*Der Roboter kann eigenständig lukrative Aufträge wählen.*“
Bei der Auftragsannahme entscheidet der Roboter je nach Ausgangslage, ob der Auftrag mehr oder weniger lukrativ ist. Dies wird folgendermaßen umgesetzt. Zur Berechnung des möglichen Gewinns bei erfolgreicher Auftragsdurchführung werden mehrere Faktoren betrachtet. Dabei spielen die Entfernung des Roboters vom aktuellen Standort zum Ziel, die Auftragsdauer sowie der Preis des Auftrags eine Rolle. Befindet sich ein Roboter in der Nähe des Auftrags und ist der Produktionszyklus kurz und dadurch dieser Auftrag schnell durchführbar ist, wird dieser Auftrag als sehr lukrativ betrachtet.

⟨*RS*4⟩ „*Der Server soll eine Administrator-Oberfläche haben.*“
Die Administratoroberfläche wird mit Java erzeugt und beim Starten des Spiels ausgewählt. Sie enthält wie die Spieleroberfläche die Statistiken und eine Übersicht der Roboter und Industrien. Im Gegensatz zur Spieleroberfläche kann der Administrator neue Roboter hinzufügen, die vom Server initialisiert werden. Er alleine kann zudem das Spiel starten, indem er den „Spiel-Starten“-Button drückt.

⟨*RS*5⟩ „*Der Server soll Benutzeraufträge annehmen können.*“
Über die Administratoroberfläche legt der Benutzer neue Aufträge an, die an den Server geschickt werden. Dieser überprüft die Richtigkeit der Daten und sendet eine Bestätigung. Sind die Daten korrekt fügt der Server den Auftrag an das Ende der Warteliste ein. Die Warteliste wird aktualisiert und an die Spieleroberfläche gesendet.

⟨*RS*6⟩ „*Der Spieler soll die vom Roboter vorgeschlagene Route verändern können.*“
Diese Kriterium wird über die Spieleroberfläche ermöglicht. Der Roboter sendet die Daten der vorgeschlagenen Route an den Server, der die Daten weiter an die Spieleroberfläche leitet. Die Route wird visualisiert dargestellt. Will der Benutzer die Route verändern, klickt er auf den Button „Bearbeiten-der-Route“. Die Route wird zur Manipulation freigegeben. Nun kann der Benutzer durch Klicken auf verschiedene Abschnitte die Route verändern. Hat er die endgültige Route gefunden, speichert er sie über den „Speichern“-Button. Die Route wird an den Server gesendet, der sie auf Konsistenz überprüft und danach an den Roboter weiterleitet. Der Roboter überschreibt die Route mit der neuen und führt den Auftrag aus.

5.3. Kannkriterien

Die Erfüllung folgender Kriterien für das abzugebende Produkt wird angestrebt: ⟨*RC*1⟩ „*Der Roboter soll die Verderblichkeit der Waren berücksichtigen.*“
Dies würde bei der Routenplanung in Betracht gezogen werden. Das bedeutet das verderbliche Waren eine höhere Priorität hätten.

⟨*RC*2⟩ „*Der Server kann eine Kalibrierung der Roboter vornehmen.*“
Dies würde durch ein Button in Benutzer-GUI realisiert werden, welchen den Roboter während des Spiels zum Kalibrieren zwingt.

⟨*RC*3⟩ „*Die Spieleroberfläche muss von einem anderen PC aus aufrufbar sein*“
Dies würde über TCP/IP realisiert werden. Dann könnte die Spieler-GUI sich über die IP des Servers in das Spiel einklinken und bei einer neuen runde dran teilnehmen.

5.4. Abgrenzungskriterien

Folgende Funktionalitäten werden nicht durch das Produkt, sondern wie folgt beschrieben anderweitig erfüllt:

⟨*RW*1⟩ „*Nach dem Spielstart darf das Wegenetz nicht geändert werden können.*“
Da das Wegenetz statisch während des Spielbetriebs existiert, darf das Wegenetz nicht nach dem Spielstart geändert werden.

⟨*RW*2⟩ „*Die Roboter transportieren Waren nicht physisch.*“
Da für die Aktion Be- und Entladen keine Vorrichtung existiert und die Entwicklung keinen Informationstechnischen Mehrwert hat, wird dies nicht umgesetzt.

⟨*RW*3⟩ „*Die grafische Darstellung erfolgt nicht in Echtzeit.*“
Da die dauernde Übertragung nicht viel mehr Information enthält, als eine Abschnittsweise Ortsbestimmung, soll die grafische Darstellung nicht in Echtzeit erfolgen. Außerdem existiert die Darstellung des Spiels mit den Robotern.

⟨*RW*4⟩ „*Die Karte wird nicht dreidimensional visualisiert.*“
Da eine Visualisierung durch die Roboter existiert, soll diese nicht dreidimensional visualisiert werden.

⟨*RW*5⟩ „*Zur Spielzeit dürfen die Eigenschaften von Ressourcen und Produkte nicht verändert werden.*“
Dies ist notwendig da eine Änderung während der Spielzeit zu Inkonsistenzen führen könnte.

Glossar

Auftrag

Der Auftrag setzt sich aus Auftragnehmer, Auftraggeber, der zu transportierenden Ware, Start- und Zielpunkt sowie dem zu erwartenden Lohn zusammen.

Benutzer

Menschlicher Anwender, der auf den Server zugreift. Der Benutzer (Admin) verwaltet den Server. Der Benutzer kann (muss aber nicht) Spieler sein.

Broadcast

Die identische Nachricht an alle teilnehmenden Roboter.

Deadlock

Beim Deadlock können die Roboter nicht mehr ihrer Aufgabe nachkommen, weil sie sich gegenseitig behindern. Ein Deadlock ist also ein zu vermeidender Spielzustand.

FCFS-KI

Diese KI arbeitet nach dem Prinzip first-come, first-served, nimmt also immer den zuerst zur Verfügung stehenden Auftrag an.

GUI

Die GUI ist die grafische Benutzeroberfläche für den Server und den Spielerclient.

Kartendaten

Mit den Kartendaten sind die Informationen über das Wegenetz gemeint, welches als Graph implementiert wird.

Konflikt

Ein Konflikt tritt auf, wenn eine Straße von einem Roboter versperrt ist. Andere Roboter können diese zur selben Zeit nicht befahren.

Künstliche Intelligenz (KI)

Das Programm auf den Robotern, welches das Handeln des Roboters festlegt.

NXT

NXT ist ein Roboterbausatz der Firma Lego.

Ressourcen

Unter Ressourcen fallen Industrien, Waren und der Kraftstoff für die Roboter.

Roboter

Die Roboter arbeiten jeweils mit einer NXT-2.0-Einheit der Firma Lego und stellen die Transportmittel für die Waren dar.

Server

Der Server dient hier als Koordinationsschnittstelle und übernimmt somit die Aufgabe des Spielleiters.

Spieler

Menschlicher Anwender, der auf die Spieleroberfläche (nicht den Server) zugreift.

Spieleroberfläche

Graphische Benutzeroberfläche für den Spieler, um einerseits das Spielgeschehen zu beobachten, andererseits aktiv am Spiel teilzunehmen.

Spielzeit

Die Spielzeit bezeichnet den Zeitraum, der nach Initialisierung des Spiels und dessen Start bis zum Erreichen des Spielziels abläuft.

A. Projektübersichtsdiagramm

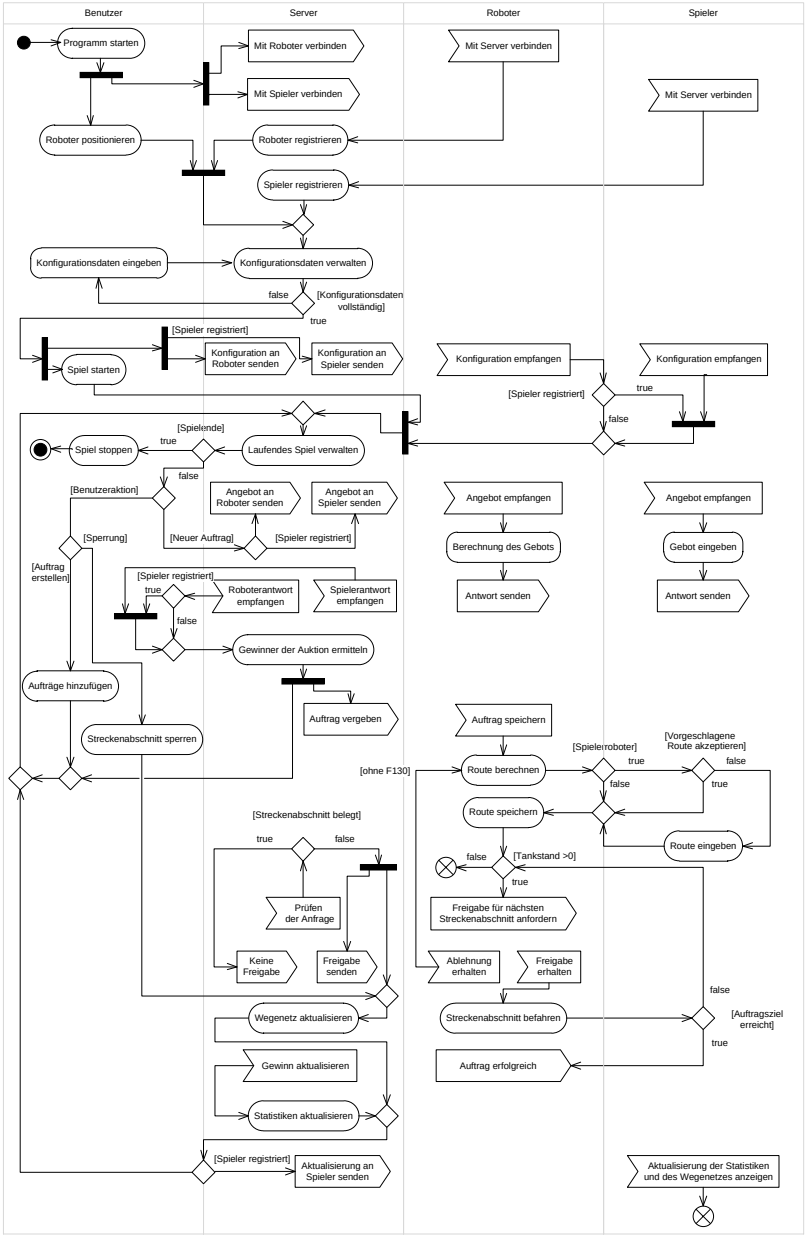

Abbildung A.1.: Aktivitätsdiagramm zur Projektübersicht