Universität Koblenz-Landau
Fachbereich Informatik
Institut für Softwaretechnik

Anpassung eines Leitstandes für autonome Fahrzeuge an die Rollende Landstraße

Diplomarbeit

zur Erlangung des Grades eines Diplom-Informatikers im Studiengang Diplom Informatik vorgelegt von

Daniel Schüller und **Ralf Töppner**

Betreuer:
Prof. Dr. Dieter Zöbel, Institut für Softwaretechnik, FB4
Dipl.-Inf. Philipp Wojke, Institut für Softwaretechnik, FB4

Erklärungen

Ich versichere, dass ich die Abschnitte 1, 2.1, 2.2, 3, 4.1.1, 4.1.2, 4.1.3.3, 4.1.3.4, 4.1.3.5, 4.1.3.6, 4.1.3.7, 4.1.4, 4.1.5, 4.3.1.2, 4.3.2.1, 4.4.3, 5.1, 5.10.3, 6.1.1, 6.1.5, 6.2, 6.3, 6.4.7 und die Abschnitte 7.1, 7.2, 7.6.2, 7.7 des Anhangs der vorliegenden Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, Januar 2007

Daniel Schüller

Ich versichere, dass ich die Abschnitte 4.1.3.1, 4.1.3.2, 4.2, 4.3.1.1, 4.3.1.3, 4.3.1.4, 4.3.2.2, 4.4.1, 4.4.2, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10.1, 5.10.2, 6.1.2, 6.1.3, 6.1.4, 6.4.1, 6.4.2, 6.4.3, 6.4.4, 6.4.5, 6.4.6, 6.5 und die Abschnitte 7.3, 7.4, 7.5, 7.6.1 des Anhangs der vorliegenden Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Koblenz, Januar 2007

Ralf Töppner

Danksagung

„Der Abschied von einer langen und wichtigen Arbeit ist ebenso traurig wie erfreulich.“
(Friedrich Schiller)

Der Abschied von dieser Arbeit ist für uns mehr als nur die Vollendung des letzten Kapitels unseres Studiums. Während wir das eine Buch zuschlagen, wartet bereits ein weiteres darauf von uns neu geschrieben zu werden. Neben neuen Perspektiven bedeutet dies aber auch den Abschied von vielen Arbeitsweisen und Personen, die uns in den letzten Jahren begleitet und geprägt haben. Daher möchten wir nun an dieser Stelle die Gelegenheit nutzen uns zu bedanken, bei ...

... Prof. Dr. Dieter Zöbel, für die unermüdliche und zuverlässige Unterstützung während des gesamten Studiums, die ständige Bereitschaft sich unsere Probleme anzuhören und seinen Einsatz Lösungen zu finden, seinen unkomplizierten und freundschaftlichen Umgang mit den Studenten sowie die Möglichkeit für uns die Diplomarbeit in diesem Rahmen durchzuführen.

... Diplominformatiker und Betreuer Philipp Wojke, für seinen ausdauernden Einsatz, die kompetente Betreuung, das zielorientierte Vorgehen, die durch ihn gelegten Grundlagen der Arbeit und das freundschaftliche Arbeitsverhältnis auch über die Uni hinaus.

... allen die im Vorfeld dieser Diplomarbeit an den Grundlagen EZsped (René Lotz und Vanessa Thewalt) und deren Visualisierung (Jörg Sesterhenn), der Bibliothek EZauto (Philipp Wojke) und dem PProLa beteiligt waren.

... allen Freunden, Bekannten, Verwandten und Kommilitonen, welche uns in der ganzen Zeit unterstützten, sei es durch die unangenehme Arbeit des Korrekturlesens oder die bloße mentale Unterstützung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Kapitel 1

Hintergründe/Grundlagen der Arbeit

In diesem Kapitel sollen zunächst die Grundlagen der vorliegenden Diplomarbeit erläutert und die Zusammenhänge der einzelnen Gebiete aufzeigt werden. Es soll ein Überblick gegeben werden, auf welcher Basis die vorliegende Arbeit verfasst worden ist und welchen informationstechnischen, wirtschaftlichen und politischen Hintergrund die einzelnen Themengebiete besitzen. Dabei geht es zunächst nicht nur um die Sichtweise der Informatik, sondern es soll ein Gesamtüberblick entstehen, der die Motivation und die Grundgedanken dieser Arbeit darlegt. Ebenso wird die Position und die Bedeutung der Arbeit innerhalb der Forschungsgruppe Echtzeitsysteme aufgezeigt. Dies wird zunächst übersichtlich dargestellt, ehe die einzelnen Grundlagen in jeweils gesonderten Abschnitten genauer erläutert werden. Zudem soll die Themenstellung ausführlich erklärt und die Ziele definiert werden.

1.1 Arbeitsgruppe Echtzeitsysteme

Die Arbeitsgruppe Echtzeitsysteme unter der Leitung von Prof. Dr. Dieter Zöbel an der Universität Koblenz-Landau befasste sich Mitte der 90er Jahre überwiegend mit zeitkritischen Echtzeitsystemen und deren Planungsverfahren. Es sollte ein Bewusstsein dafür geschaffen werden, wie bedeutend und bereits allgegenwärtig diese Systeme den Menschen in seinem täglichen Leben begleiten und wie fundamental deren Weiterentwicklung für die Zukunft nicht nur innerhalb des

Fachgebietes der Informatik ist. Auf der Suche nach einem geeigneten Experiment zur Demonstration eines Echtzeitsystems konstruierte die Gruppe unter anderem das System „Balancieren eines Balls auf einer Wippe“. Dort ging es darum, einen Ball auf einer um zwei Achsen bewegbaren Wippe mittels angesteuerter Motoren zu bewegen und so zu balancieren, dass der Ball niemals die Wippe verlässt. Dieses Balancieren wurde mittels eines Echtzeitsystems, verteilt auf zwei Rechnern, berechnet und umgesetzt. Jeder, der dieses Vorhaben, einen Ball auf einer Wippe zu balancieren, schon mal in seiner Kindheit in verschiedenen Spielen umzusetzen versucht hat, kann sich nun vorstellen, innerhalb welcher kurzen Zeit das System reagieren muss, um den Ball auf der Wippe kontrollieren zu können.

Ein weiteres Beispiel für ein Echtzeitsystem fand die Arbeitsgruppe auch im Bereich des autonomen Fahrens. Die Idee dahinter war zunächst simpel: Ein technisch modifiziertes Fahrzeug soll in der Lage sein sich autonom auf einer Fläche zu bewegen. Dabei sollte es diverse Anforderungen einhalten, z. B. die Umgebung erkennen, Kollisionen vermeiden oder Befehle empfangen und ausführen etc. Dieses autonome Fahren war z. B. aus Lagertransportsystemen bekannt und eröffnete der Arbeitsgruppe zugleich eine Vielzahl neuer Forschungsgebiete. So wurde im Jahr 1998 das Projekt EZauto[1] ins Leben gerufen, das sich mit dem autonomen Fahren von Gespannen beschäftigen sollte. Am Anfang der Forschung standen die kinematischen Modelle von Fahrzeugen und die Berechnung von fahrbaren Trajektorien im Vordergrund. Hieraus entwickelte sich im Laufe der Jahre schnell eine Vielzahl von Anwendungen. Mit deren steigender Anzahl wuchs die Notwendigkeit für ein einheitliches System zur Entwicklung und Erforschung von Anwendungen im Bereich des autonomen und assistierten Fahrens. Aufgrund dieser Erkenntnis entwarf die Forschungsgruppe eine Softwarearchitektur, die diverse Fragestellungen in diesem Gebiet durch ihre hohe Flexibilität und Wiederverwendbarkeit beantworten konnte. Diese Architektur wurde in einer Bibliothek implementiert und liefert derzeit das Fundament für viele Aktivitäten der Forschungsgruppe. Vorgestellt und weiterhin betreut wird die Bibliothek im Wesentlichen durch Diplom-Informatiker Philipp Wojke, der diese im Rahmen seiner Dissertation[2] kontinuierlich ausbaut.

Ebenso im Kontext des autonomen Fahrens wurde im Jahr 2003 das Projektpraktikum Rollende Landstraße (kurz PProla) veranstaltet, in welchem die Simulation eines Verladebahnhofes

[1] vgl. Abschnitt 1.4
[2] vgl. [Woj-b]

der Rollenden Landstraße geschaffen und welches von dort an unter dem Projektnamen EZrola geführt wurde. Der Grundgedanke des Projektes war folgender: Es sollte ein Prototyp konstruiert werden, welcher Lkw mittels autonomer Fahrt auf Züge verlädt, alle Bewegungen auf dem Verladebahnhof initiiert sowie kontrolliert und die komplette Leitung über das Gelände übernimmt. Das wirtschaftliche Ziel dahinter war es, den Schritt weg von einem begleiteten zum unbegleiteten Güterverkehr zu schaffen. Es entstand dabei ein funktionales System, welches zwar den Anforderungen gerecht wurde, aber verschiedene Gesichtspunkte der Softwaretechnik nicht erfüllen konnte. Dadurch wurde eine weitere Verwendung erschwert. Diese Erkenntnisse und Einsichten wurden zur Basis des Projektes EZrola, welches Diplom-Informatiker Philipp Wojke in seiner Dissertation fortführt und weiterentwickelt. Dabei sollen die aus dem PProla erkannten Defizite durch eine neue Softwarearchitektur beseitigt und die einzelnen Strukturen und Abläufe optimiert werden.

Mit den Erkenntnissen des PProla wurde in Zusammenarbeit mit dem Automobilkonzern Daimler-Chrysler ein weiteres Projekt mit dem Namen EZspedition (kurz EZsped) auf dem gleichen Grundgedanken des autonomen Fahrens angesetzt. Ziel war es, eine flexible Architektur zu erstellen, die autonomes Fahren, unabhängig vom Kontext, ermöglichen sollte. Diesmal, im Gegensatz zum Projektpraktikum RoLa, sollte auf dem Projekt EZauto aufgebaut werden.
Diese Zielsetzung wurde in der gemeinsamen Diplomarbeit von René M. Lotz und Vanessa Thewalt umgesetzt, in der sie die Architektur eines Leitstandes für autonomes Fahren entwarfen und diese beispielhaft implementierten. Als konkretes Fallbeispiel diente hierzu der Speditionshof in Großheppach bei Stuttgart.

Nach Abschluss dieses Projektes stellte sich nun die Frage, wie flexibel und anpassungsfähig dieser Leitstand für autonomes Fahren ist. Um dies genau zu spezifizieren, bot sich die einmalige Gelegenheit an, das Konzept des Leitstandes aus EZsped in das Projekt EZrola zu portieren. Somit kann zum einen auf eine wohlbekannte Basis aufgebaut, diese aber auch zum anderen gleichzeitig erweitert und an mögliche Grenzen geführt werden. Als Fallbeispiel diente hier das Areal des RoLa-Terminals in Regensburg, das durch den Bayernhafen Regensburg, sowie die Ökombi GmbH[3] unterhalten wird und das Teil der aktuell verwendeten RoLa-Strecke Graz-Regensburg ist.

[3] vgl. Abschnitt 3.1

1.2 Thema

In diesem Abschnitt soll nun die Themenstellung dieser Arbeit erläutert werden. Dies wird zunächst auf allgemeiner Ebene durchgeführt, ehe der genaue wissenschaftliche Hintergrund und die zu erarbeitenden Fragestellungen dargestellt werden.

1.2.1 Allgemeine Definition und Zielsetzung

Das Thema der vorliegenden Arbeit basiert auf der Intention des Projektpraktikums Rollende Landstraße der Forschungsgruppe Echtzeitsysteme der Universität Koblenz-Landau. Somit wird es unter dem Projektnamen EZRollendeLandstraße (kurz EZrola) geführt. In diesem Projekt geht es um die Automatisierung von Verladebahnhöfen (auch Terminals genannt) der Rollenden Landstraße. Ziel dieser Automatisierung ist es, dass die Fahrer der Lkw diese nicht mehr beim Schienentransport begleiten müssen und die RoLa somit zum unbegleiteten kombinierten Güterverkehr heranreift. Des Weiteren sollen die Effizienz und die Auslastung der RoLa-Terminals durch eine vollständige Automatisierung gesteigert werden. Dazu sollen die Fahrer ihre Sattelauflieger oder Anhänger am Terminal übergeben, wo sie auf autonome Terminaltraktoren umsatteln und mit diesen auf die Bahnwaggons verladen werden. Ebenso sollen zukünftig technisch modifizierte Kundenfahrzeuge ihre Sattelauflieger und Anhänger nicht mehr abkuppeln müssen, sondern das gesamte Fahrzeug soll einer autonomen Fahrt auf dem Gelände folgen können.

Das in Abschnitt 1.5 vorgestellte Projekt EZsped ähnelt der Zielsetzung von EZrola sehr, allerdings stellt es einen anderen Anwendungsfall von autonomer Fahrt dar. Hier werden speziell ausgerüstete Kundenfahrzeuge auf einem autonomen Speditionshof rangiert. Im Rahmen dieses Projektes wurde ein Leitstand für autonomes Fahren entworfen, welcher jegliche Bewegungen auf einem definierten Gelände und alle damit verbundenen notwendigen Entscheidungen koordinieren und initiieren soll. Um diese Funktionalität zu gewährleisten, wurde der Leitstand in verschiedene Module unterteilt, die sich den unterschiedlichen Aufgabenstellungen widmen. Diese Module sind aktuell nur beispielhaft im Rahmen der Simulation eines autonomen Speditionshofes implementiert.

Ziel dieser Arbeit ist es, den Leitstand von EZsped so anzupassen und zu ergänzen, dass dieser für ein EZrola-Terminal eingesetzt werden kann. Die Leitstände von EZrola und EZsped

sollen dabei zu zwei Spezialisierungen des „Allgemeinen Leitstandes“ werden. Dazu bedarf es einer intensiven Analyse, inwieweit sich die Anforderungen an den Leitstand zwischen EZsped und EZrola unterscheiden und welche Ergänzungen und Änderungen vorgenommen werden müssen. Dementsprechend sind Lösungen zu entwickelten, diese in die Architektur des Leitstandes einzupassen und abschließend zu implementieren. Dabei soll auch untersucht werden, inwieweit sich die Architektur des EZsped-Leitstandes für EZrola eignet und welche Einschränkungen sich für diesen Anwendungsfall ergeben.

Aufgaben, die der EZrola-Leitstand durchführen muss, sind unter anderem das An- und Abkuppeln von Sattelaufliegern, Anhängern und autonomen Fahrzeugen, eine Verwaltung der Park- und Wartebereiche, sowie die Generierung von Fahraufträgen für den Transport von Sattelaufliegern und Anhängern auf dem Terminal, wie auch für das Be- und Entladen der Züge.

Im Rahmen von EZsped wurde auch eine Visualisierung entwickelt, welche durch den Leitstand verwaltete Vorgänge in einer dreidimensionalen Ansicht darstellt. Diese Visualisierung ist so anzupassen, dass sie auch für EZrola verwendet werden kann.

1.2.2 Wissenschaftlicher Hintergrund

Nach der allgemeinen Definition des Themas, sollen nun die zentralen Aspekte des wissenschaftlichen Hintergrundes aufgeführt werden. Diese sind aus Sicht der Informatik im Wesentlichen im Bereich der Softwaretechnik anzusiedeln. Doch nicht nur in diesem Gebiet, sondern auch aus wirtschaftlicher Sicht können durch die entstandene Simulation konkrete Fragestellungen zur Nutzung der Rollenden Landstraße beantwortet werden. Diese Antworten werden nicht Gegenstand der vorliegenden Arbeit sein, dennoch wird kurz aufgezeigt, welche Möglichkeiten sich ergeben und wie diese eine Basis für weitere wirtschaftliche Beurteilungen bieten könnten.

1.2.2.1 Softwaretechnische Aspekte

Den Mittelpunkt der Anpassung des Leitstandes von EZsped an das Projekt EZrola soll der Begriff der Software-Qualität und deren Sicherung bilden. Die Norm EN ISO 8402 definiert Qualität wie folgt:

„Qualität ist die Gesamtheit von Merkmalen (und Merkmalswerten) einer Einheit bezüglich ihrer Eignung, festgelegte und vorausgesetzte Erfordernisse zu erfüllen“.

Im Falle dieser Arbeit liegt der Leitstand für autonomes Fahren als Einheit zugrunde und kann somit über eine Gesamtheit von Merkmalen zur Beschreibung einer Software-Architektur beurteilt werden. Sicherlich gibt es gerade im Gebiet der Softwaretechnik diverse Kriterien und Richtlinien, welche je nach Sichtweise eine präzise Beurteilung zulassen. In der vorliegenden Arbeit geht es zunächst darum ein Gesamtbild der Qualität des Leitstandes zu erstellen. Dabei werden ganz bewusst einige Merkmale herausgegriffen, die für zukünftige Projekte innerhalb der Forschungsgruppe von großer Bedeutung sein werden. Die wesentlichen Merkmale sind:

- Wiederverwendbarkeit:
  Die Wiederverwendbarkeit ist der Maßstab an eine Software[4], der beurteilt, in welchem Maße die Software an neue Anwendungsfälle angepasst werden kann und ob es Einschränkungen dabei gibt. Dies ist gerade in Bezug auf einen Allgemeinen Leitstand für autonomes Fahren von großer Bedeutung, denn mit steigender Abstraktion einer Software steigt auch deren Wiederverwendbarkeit. Dies bedeutet aber zugleich auch eine Steigerung des Implementierungsaufwandes, der vollzogen werden muss, um die Software später wieder an einen konkreten Anwendungsfall anzupassen. Daher ist es von besonderer Bedeutung, den richtigen Mittelweg zwischen hoher Abstraktion und anwendungsnaher Implementierung zu finden.

- Wartbarkeit:
  Unter dem Begriff der Wartbarkeit werden die drei folgenden Aspekte verstanden:

  - Verständlichkeit:
    Die Verständlichkeit beurteilt den Aufwand, den ein außen stehender Anwender aufbringen muss, um sich in die bestehende Software einzuarbeiten und diese zu verstehen. Sind komplexe Softwaremodule gut strukturiert, ausführlich kommentiert und ihre Funktionsweise schnell ersichtlich, so spricht dies für eine hohe Verständlichkeit.
  - Flexibilität:
    Die Flexibilität ist der Maßstab dafür, inwieweit die bestehende Software um neue

[4] hier der Leitstand

Module und Elemente ergänzt werden kann. Kann ein bestehendes System sehr schnell um neue Strukturen erweitert werden, so liegt ein hohes Maß an Flexibilität vor.

  - Validierbarkeit:
    Die Validierbarkeit bezieht sich auf die Qualitätssicherung einer Software. Sie drückt die Möglichkeit aus, in welcher das System erlaubt, die erreichten Ziele aufzuzeigen und festzuhalten.

- Zuverlässigkeit:
  Die Zuverlässigkeit beurteilt, inwieweit eine Software die angestrebten Ergebnisse unter den verschiedensten Randbedingungen garantiert. Diese ist besonders bei sicherheitskritischen Anwendungen von großer Bedeutung, da hier die Berechnung eines Programmes unter keinen Umständen von anderen parallel laufenden Programmen beeinflusst und verändert werden darf.

- Effizienz und Benutzerfreundlichkeit:
  Die Effizienz drückt aus, inwieweit die vorliegende Software die ihr zur Verfügung stehende Hardware belastet oder wie rechenaufwendig einzelne Module oder Klassen implementiert sind. Gerade in den Programmiersprachen C und C++ kann eine unachtsame Speicherverwaltung schnell zu hohen Leistungseinbußen führen.

  Die Benutzerfreundlichkeit ist das Maß dafür, wie einfach oder kompliziert die Software vom Endanwender verwendet werden kann. Sind alle möglichen Eingaben eines Anwenders berücksichtigt und reagiert die Software verständlich und intuitiv, so kann von einer hohen Benutzerfreundlichkeit gesprochen werden.

Die genannten Punkte sind einige der wesentlichen Qualitätseigenschaften einer Softwarearchitektur und somit von fundamentaler Bedeutung für den zukünftigen Einsatz des Leitstandes. Eine Beurteilung dient somit auch der Qualitätssicherung des gesamten Projektes, da mögliche Abweichungen vom angedachten Konzept ebenso analysiert und Verbesserungsvorschläge gemacht werden können. Gerade der Begriff der Wiederverwendbarkeit rückt im Bezug auf zukünftige Projekte der Forschungsgruppe in den Vordergrund, denn an ihr entscheidet sich, ob die aktuelle Architektur weiterhin angewendet werden kann oder für neue Anwendungen modifiziert werden muss.
Durch die Anpassung an ein konkretes Fallbeispiel sind aber weitaus mehr als nur diese Beurteilungen möglich. Es kann herausgearbeitet werden, welche Schritte bei einem gegebenen Fallbeispiel vollzogen werden müssen, um den vorhandenen Leitstand den Umständen entsprechend

zu modifizieren. So sollte es möglich sein, eine Mechanik zu entwickeln, welche die wesentlichen Anpassungsschritte enthält und somit eine weitere Wiederverwendung erleichtern kann.

Weiterhin kann dadurch ein Mehrwert gewonnen werden, der sich auf unterschiedliche Aspekte bezieht. So können z. B. Programmier- und Architekturfehler gefunden und behoben, Änderungsvorschläge eingebracht, oder weitere Strukturen und Module entworfen und implementiert werden, welche die Wiederverwendbarkeit ebenso steigern können.

Des Weiteren soll durch die genaue Analyse des vorliegenden Systems herausgearbeitet werden, was die Elemente des Allgemeinen Leitstandes sind und welche Änderungen und Anpassungen an die Fallbeispiele von EZsped und EZrola vorgenommen werden müssen. Dabei gilt es zu unterscheiden, ob die notwendigen Änderungen spezifisch für die Anwendungen getroffen werden müssen oder ob es sich um Erweiterungen der Funktionalität des Allgemeinen Leitstandes handelt.

1.2.2.2 Wirtschaftliche Aspekte

Wie in Abschnitt 1.3 noch genauer erläutert wird, genießt die Rollende Landstraße in der aktuellen Verkehrspolitik keinen besonders einfachen Status, da sie neben einer Reihe von Vorteilen auch diverse Nachteile beinhaltet. Lediglich das Land Österreich subventioniert die Betreiber der RoLa-Terminals und setzt ganz bewusst auf diese Förderung, um die Rollende Landstraße in Zukunft höher zu frequentieren und die Autobahnen somit vom Transitverkehr zu entlasten. Der Grundgedanke, die Lkw über große Distanzen auf Schienen zu befördern, hat sowohl ökologische als auch wirtschaftliche Vorteile. Diese kommen allerdings nur bei hoher Auslastung der RoLa zum Tragen. Daher soll ein neuer Ansatz geschaffen werden, der die Verwendung der Rollenden Landstraße für große Speditionen und kleinere Unternehmen attraktiver macht und bei dem die Vorteile den vorhandenen Nachteilen klar überwiegen. Doch ein neues Konzept alleine reicht für eine Meinungsneubildung eines der RoLa gegenüber bisher eher skeptisch eingestellten Unternehmens bei Weitem nicht aus. Die Vorteile müssen gezielt aufgezeigt und z. B. in einer Simulation dargestellt werden. Diese ist im Rahmen des Projektes EZrola entstanden und kann somit als Anschauungsobjekt für zukünftige Projekte dienen. Doch bietet diese Simulation nicht nur reine visuelle Gesichtspunkte, sondern kann auch die Basis einer wirtschaftlichen Studie über die Nutzung und Auslastung eines RoLa-Terminals bilden. Verschiedene Fragestellungen wie z. B. zur

Modellierung des Geländes, Realisierung der Übergabebereiche, zeitliche und räumliche Grenzen und Auslastungen, maximale Kapazitäten und ein möglicher Zugtakt können durch die Simulation beantwortet werden. Nahezu alle Parameter sind unabhängig vom System definierbar, womit verschiedene Testszenarios zügig erstellt und effizient ausgewertet werden können. Die gewonnenen Erkenntnisse können somit auf die aktuelle Nutzung eines RoLa-Terminals übertragen und Verbesserungsvorschläge gemacht werden. In dieser Arbeit wird sich auf die genannten Aspekte beschränkt und die weitere Auswertung einer möglichen wirtschaftlichen Studie überlassen.

1.3 EZroLa

Das Projekt EZrola beschäftigt sich mit der autonomen Umsetzung der Rollenden Landstraße, einer Form des kombinierten Güterverkehrs. Eine genaue Definition, sowie die zu deren Einsatz notwendige Infrastruktur, soll an dieser Stelle kurz aufgeführt werden. Ebenso werden einige Vor- und Nachteile erläutert und die Zielsetzung des Projektes EZrola im Zusammenhang mit dieser Arbeit analysiert.

1.3.1 Begriffsdefinition und Förderung

Das Projekt EZrola bildet den Rahmen der Dissertation von Diplom-Informatiker Philipp Wojke und beschäftigt sich mit der Automatisierung von Terminals der Rollenden Landstraße. Diese ist eine Form des kombinierten Güterverkehrs, der durch die Zusammenarbeit der Wirtschaftskommission für Europa der Vereinten Nationen (UN/ECE), der Europäischen Konferenz der Transportminister (ECMT) und der Europäischen Kommission (EC) wie folgt definiert wurde:

„Kombinierter Verkehr [ist] Intermodaler Verkehr, bei dem der überwiegende Teil der in Europa zurückgelegten Strecke mit der Eisenbahn, dem Binnen- oder Seeschiff bewältigt und der Vor- und Nachlauf auf der Straße so kurz wie möglich gehalten wird[5] *“.*

„Intermodaler Verkehr [ist der] Transport von Gütern in ein und derselben Ladeeinheit oder demselben Straßenfahrzeug mit zwei oder mehreren Verkehrsträgern, wobei ein Wechsel der La-

[5] vgl. [ECE] Seite 18

deeinheit, aber kein Umschlag der transportieren Güter selbst erfolgt[6] *“.*

Innerhalb des kombinierten Verkehrs werden die Ausprägungen *begleitet* und *unbegleitet* unterschieden. Bei der Rollenden Landstraße handelt es sich um die Form des begleiteten kombinierten Verkehrs Straße-Schiene, bei welcher der gesamte Lkw auf spezielle Niederflurwaggons der Deutschen Bahn verladen und transportiert wird. Der Fahrer trägt dabei die Verantwortung für das Auf- und Abfahren auf die Niederflurwaggons und die Sicherung des Lkw zum Transport. Er begleitet den Zug in einem Personenwagen der Deutschen Bahn.

Da das Verkehrsaufkommen des Straßenverkehrs jährlich um etwa 3% steigt[7], kann dieses Wachstum nicht mehr allein durch den Bau neuer Infrastrukturen bewältigt werden. Daher fördert die Europäische Kommission den kombinierten Verkehr durch Programme wie PACT und dessen Nachfolger Marco Polo oder das Projekt NEAT (Die Neue Eisenbahn Alpen-Transversale). Gerade Letzteres soll den Alpen überquerenden Verkehr fördern, z. B. durch den Bau zweier neuer Basistunnel zur Fahrzeitverkürzung sowie einer Erhöhung der Transportkapazitäten. Aber nicht nur die Europäische Union, sondern auch die betroffenen Transitländer wie z. B. Österreich befürworten und subventionieren einen Einsatz des kombinierten Verkehrs. Diese Förderung ist in den Grundsätzen der österreichischen Verkehrspolitik verankert und sieht sich in dem steigenden Sendungsaufkommen der RoLa seit ihrer Gründung im Jahre 1980 positiv bestätigt.

1.3.2 RoLa-Infrastruktur

Um den Betrieb der RoLa zu gewährleisten, braucht es im Grunde nur zwei Dinge:

- RoLa-Terminals
- RoLa-Züge

Die RoLa-Terminals sind Bahnhöfe, die speziell für die Verladung der Lkw auf die Niederflurwaggons ausgestattet sind. Dazu braucht es zunächst eine Waage und eine Messeinrichtung, um alle relevanten Daten der Lkw entgegen zu nehmen und die Einhaltung der Richtlinien zum

[6] vgl. [ECE] Seite 17
[7] vgl. [EUK] Kapitel 3.2

Transport auf der Schiene zu überwachen. Alle angenommen Lkw warten danach auf dem Gelände, bis sie auf den Zug verladen werden können, daher braucht es eine ausreichende Anzahl an Stellplätzen. Das Verladen kann über Kopfbahnhöfe stattfinden, an welchen die Lkw über die Stirnseite den Zug befahren. Ebenso ist es möglich dies über in den Boden eingelassene Gleise durchzuführen, sodass die Lkw darauf fahren und den Zug über eine angestellte Rampe befahren können. Bereits an dieser Stelle wird ersichtlich, wie einfach und effizient sich die Umrüstung eines Bahnhofes zu einem RoLa-Terminal darstellen kann, da sehr viele Bahnhöfe die genannten Anforderungen ohne große Umbaumaßnahmen erfüllen können.

Die RoLa-Züge bestehen aus einer Lok, speziellen Niederflurwaggons und einem Personenbegleitwaggon. Die Niederflurwaggons[8] haben eine Länge von 19 m, ein Gewicht von ca. 19 t und benötigen aufgrund der niedrigen Ladehöhe von 0,41 m besondere Drehgestelle mit kleinen Raddurchmessern. Alle nach StVZO zugelassenen Lastkraftzüge mit einer maximalen Breite von 2,5 m in Höhe der Räder, einem maximalen Gewicht von 45 t sowie einer maximalen Länge von 18,50 m können die Niederflurwaggons befahren und auf ihnen transportiert werden. Der RoLa-Zug wird üblicherweise in Einfachtraktion, d. h. mit nur einer Lok gefahren, daher darf das Gesamtgewicht des Zuges 1200 t nicht überschreiten. Hieraus ergibt sich die Beschränkung, je nach Beladung und Waggontyp, auf ca. 20 Niederflurwaggons. Inklusive Lok und Personenbegleitwaggon ergibt sich eine Gesamtlänge für den Zug von etwa 440 m.

1.3.3 Vor- und Nachteile

Neben einer ganzen Reihe von Vorteilen, welche die Verwendung der Rollenden Landstraße mit sich bringt, gibt es aber auch einige Kritikpunkte. Diese werden oftmals vorgetragen, wenn die Förderung und der Ausbau der Rollenden Landstraße Gegenstand politischer und wirtschaftlicher Diskussionen sind. Daher wurde zur genaueren Untersuchung im Jahr 2000 die Verkehrs- und Umweltpolitische Bedeutung der RoLa für Österreich in einer Studie genauer untersucht und verschiedene Vor- und Nachteile jeglicher Art herausgearbeitet. Besonderes Ziel dieser Studie war es, die Entlastungswirkung der RoLa für Umwelt, Infrastruktur sowie Volkswirtschaft aufzuzeigen und deren Auswirkungen zu quantifizieren. Einige der herausgearbeiteten Punkte sollen nun zur vollständigen Analyse aufgeführt, dazu aber keine wertende Stellung eingenommen werden, da dies nicht Gegenstand dieser Arbeit ist.

[8] Typ Saadkms nach UIC-Klassifikation für Güterwagen

Vorteile der RoLa:

- Entlastung der österreichischen Infrastruktur
- positive Umweltauswirkungen[9]
- Reduktion der Verkehrsunfälle
- Reduktion der Staus
- Treibstoffersparnis der auf dem Zug stehenden Lkw
- keine Mautgebühren
- Fahrer können im Begleitwaggon Ruhezeiten einhalten
- kein Fahrverbot an Sonn- und Feiertagen
- keine Abhängigkeit von witterungsbedingten Straßenverhältnissen
- Bahnhöfe sind einfach und kostengünstig umzurüsten
- Niederflurwaggons der Deutschen Bahn können genutzt werden
- ...

Neben den genannten Vorteilen existieren beispielsweise folgende Nachteile:

- lange Verladezeiten
- schlechter Zugtakt
- zu viel "Totlast", da Zugmaschinen mittransportiert werden müssen
- Fahrer müssen die Lkw begleiten
- mangelnder Service für die Fahrer im Begleitwaggon sowie an den RoLa-Bahnhöfen
- schwierige Koordination der unterschiedlichen Betreiber der RoLa-Terminals
- schlechter Ausbau des Streckennetzes der RoLa
- ...

[9] deutliche Reduktion der Abgas-Emission

1.3.4 Zielsetzung

Das Projekt EZrola hat die autonome Betreibung eines RoLa-Terminals zum Ziel. Hierzu sollen alle Abläufe auf dem Gelände von autonom fahrenden Terminaltraktoren oder dazu ausgerüsteten Kundenfahrzeugen durchgeführt werden. Dabei überwacht, kontrolliert und initiiert ein Leitrechner alle notwendigen Fahraufgaben. Durch die autonome Verwendung sollen die Terminals flexibel, sicher und wirtschaftlich betrieben werden können, sowie deren Auslastung gesteigert werden. Die Umrüstung der Terminals soll weitgehend schnell und einfach durch den Einsatz von vorhandenen Mitteln[10] umgesetzt werden. Dadurch soll sich die Möglichkeit ergeben, mehrere RoLa-Terminals zu einem Netzwerk zusammenschliessen zu können.

So schnell diese vermeintlich einfachen Ziele genannt werden können, so komplex stellt sich die Umsetzung insbesondere aus der Sichtweise der Informatik dar. Das hinter der autonomen Verwendung stehende Softwaresystem muss eine Vielzahl von Anforderungen erfüllen und daher sehr gut analysiert und präzise spezifiziert werden, um der Liste der zu erfüllenden Aufgaben[11] flexibel gewachsen zu sein.

Die Erstellung des Gesamtkonzeptes ist hierbei Teil der Dissertation von Diplom-Informatiker Philipp Wojke, in deren Rahmen sich die Themenstellung dieser Diplomarbeit bewegt und nach Vollendung eingegliedert wird.

1.4 EZauto

Das Projekt EZauto wurde 1998 ins Leben gerufen und beschäftigt sich *mit dem autonomen Fahren von Serienfahrzeugen im Gespann mit hoher Präzision vorwärts sowie rückwärts*[12]. Ziel war es eine Basis zu definieren, welche von hoher Flexibilität und Wiederverwendbarkeit zeugt, sodass folgende Arbeiten und Projekte darauf aufbauen können. Diese Grundlage wurde innerhalb weniger Jahre durch die Forschungsgruppe gelegt und durch Diplom-Informatiker Philipp Wojke in seinem wissenschaftlichen Beitrag *"EZauto - Softwarearchitektur für autonomes oder simuliertes Fahren"*[13] vorgestellt. Im Rahmen seiner Dissertation wird diese Bibliothek weiterhin

[10] Niederflurwaggons, Parkplätze und mobile Auffahrrampen
[11] Verwaltung, Überwachung, Kontrolle, Kommunikation etc.
[12] vgl. [Zöb], Folie 11
[13] vgl. [Woj-a]

durch ihn betreut. Schon zum jetzigen Zeitpunkt aber bestätigt sich der Erfolg des Konzeptes, da bereits aktuell zahlreiche Anwendungen der Forschungsgruppe auf dieser Bibliothek aufsetzen.

EZauto besteht aus einer Architektur und einer zugehörigen Bibliothek, welche diese Architektur in der Programmiersprache C++ implementiert. Durch sie werden grundlegende Strukturen und Sichten auf Fahrzeuge bereitgestellt, sowie eine Vielzahl von notwendigen Funktionen und Algorithmen im Zusammenhang mit dem autonomen Fahren implementiert. EZauto besteht aus drei Teilen: der Basis, dem Kern und den Anwendungen.

In der Basis werden grundlegende Funktionen definiert, die keinen direkten Zusammenhang zum Fahrzeug aufweisen. Ebenso finden sich hier programmiertechnische Hilfsmittel wieder, wie z. B. eine Objektverwaltung, eine Objektarchivierung als auch eine gesonderte Ereignisbehandlung. Die Objektverwaltung erleichtert dem Programmierer den Umgang mit erzeugten Objekten der Programmiersprache C++ mittels der Anwendung eines Smartpointer-Konzeptes. Durch die Objektarchivierung werden Datengrundlagen in Form von XML-Dateien verfügbar macht. Somit können nicht nur gekapselte Strukturen eingelesen und verwaltet, sondern diese ebenso zur Verarbeitung an andere Prozesse weitergereicht werden. Dieses Übersenden wird hierbei von Prozessen der Ereignisbehandlung übernommen, welche die Kommunikation zwischen den Komponenten, auch über Prozessgrenzen hinweg, gewährleistet.

Im Kern der Bibliothek EZauto befinden sich für das autonome Fahren domänenspezifische Komponenten, die ganz bewusst in einem hohen Abstraktionsgrad vorliegen. So können diese, losgelöst von einer konkreten Anwendung, eine Wiederverwendbarkeit für unterschiedliche Fragestellungen gewährleisten. So finden sich hier kinematische Fahrzeugbeschreibungen sowie Bewegungsmodelle, welche zur Simulation eines autonom fahrenden Fahrzeuges notwendig sind. Dabei bildet das Einspurmodell zunächst die Grundlage und besteht aus der statischen Beschreibung des Fahrzeuges[14], den dynamischen Fahrzeugdaten[15] als auch den Fahrzeugfunktionen[16]. Um das Verhalten von Gespannen zu simulieren, können einzelne kinematische Beschreibungen zu Ketten zusammengefügt werden, während die genaue Berechnung der Bewegungen auf Bewegungsmodelle[17] zurückgreift.

[14] z. B. die Abmessungen des Fahrzeuges, dessen Radstand etc.
[15] z. B. der aktuelle Lenkwinkel oder der Einknickwinkel des Anhängers in einem Gespann
[16] ähnlich zu den dynamischen Daten, hier aber als Funktionen der Zeit
[17] basierend auf den Fahrzeugbeschreibungen

Der dritte Teil der EZauto Bibliothek enthält nun letztlich die aus Basis und Kern entstandenen Anwendungen. Darunter finden sich z. B. Fahrassistenzen, die dem Fahrer das Rückwärtsfahren eines Pkw mit Anhänger erleichtern, der autonom fahrende Modell-Lkw oder die Software des Fahrsimulators im Labor der Forschungsgruppe[18].

1.5 EZsped

"Nach dem Befahren des Logistikhofes parkt der Fahrer seinen Lkw auf dem Gelände. Daraufhin meldet er sich bei einer zentralen Stelle an und empfängt dort weitere Instruktionen, an welcher Rampe er sein Fahrzeug wann be- und entladen lassen kann und wann er beispielsweise die Servicehalle zur Wartung seines Lkw nutzen darf. Im Anschluss begibt er sich zurück zu seinem Fahrzeug und steuert dies zu den ihm angesagten Uhrzeiten zu den notwendigen Hallen. Nach der Vollendung aller Aufgaben verläßt er schließlich wieder das Gelände" - so oder so ähnlich könnte der Ablauf auf dem Gelände eines Speditionshofes aussehen. Markant dabei ist, dass der Fahrer alle Abläufe koordinieren muss und somit z. B. keine Ruhezeiten nutzen kann. Um ihm diese zu ermöglichen und vor allem eine effizientere Nutzung des Speditionshofes zu gewährleisten, beschäftigt sich das Projekt EZsped der Forschungsgruppe Echtzeitsysteme in Zusammenarbeit mit dem Automobilkonzern Daimler-Chrysler mit der Automatisierung eines solchen Speditionshofes. Dabei könnten die automatisierten Abläufe wie folgt aussehen: Jeder Lkw verfügt über einen Boardcomputer, der alle relevanten Daten zum Fahrzeug[19] und dessen Beladung kennt. Des Weiteren besitzt dieser Boardcomputer eine Kommunikationsschnittstelle, über die er Daten senden und empfangen kann. Die Fahrzeuge selbst sind technisch so modifiziert, dass sie in der Lage sind, vorgegebene Wegstrecken vollkommen autonom abzufahren.
Kommt der Fahrer nun mit seinem Lkw an einem autonomen Logistikhof an, so verlässt er das Fahrzeug und schaltet es durch einen Knopfdruck in den autonomen Fahrmodus. Durch diese Umschaltung meldet sich das Fahrzeug selbstständig über seine Kommunikationsschnittstelle bei einem das Gelände überwachenden Leitrechner an. Dieser kontrolliert ab diesem Zeitpunkt alle weiteren Bewegungen auf dem Gelände. Nachdem der Fahrer nun dem Leitrechner die anstehenden Aufgaben in Bezug auf seinen Lkw über eine gesonderte Eingabemaske[20] angegeben hat (z.

[18] vgl. [Woj-a]
[19] z. B. Abmessungen, Gewicht etc.
[20] z. B. über einen Touchscreen-Monitor

B. Entladen - Beladen - Waschen - Betanken), bekommt er vom Leitrechner mitgeteilt, wann er sein Fahrzeug wieder entgegen nehmen kann und verlässt das Gelände. Somit kann er z. B. Ruhezeiten einhalten oder anderen Arbeitsaufgaben nachgehen. In der Zwischenzeit übernimmt der Leitrechner alle notwendigen Aufgaben seines Fahrzeuges auf dem Gelände (z. B. Fahren zu Rampe 2, Fahren zu Waschanlage 1 sowie Fahren zu Zapfsäule 4).

Bereits mit dieser beispielhaften Ablaufbeschreibung ist die Komplexität eines Systems, das die genannten Anforderungen erfüllt, unübersehbar. Auf die genauen Abläufe innerhalb des Leitsystems bei der Verplanung und Steuerung der einzelnen Aufgaben wird in Kapitel 4 genauer eingangen. Mit den Erkenntnissen des PProla in Bezug auf eine mögliche Software-Architektur, sowie diversen Studienarbeiten, wurde ein solches Leitsystem, fortan Leitstand (engl: control center) genannt, in der gemeinsamen Diplomarbeit von René M. Lotz und Vanessa Thewalt entworfen[21]. Es handelt sich dabei um eine Softwarearchitektur, welche das autonome Fahren unabhängig vom Kontext[22] in bestimmten Richtlinien ermöglichen soll. Zur funktionalen Beurteilung wurde diese Architektur beispielhaft für einen Speditionshof-Prototyp von Daimler-Chrysler auf einem Gelände in Weinstadt-Großheppach nahe Stuttgart implementiert.

Um den entstandenen Leitstand testen zu können, wurde dieser mit einer Fahrzeugsimulation verknüpft, welche die sich auf dem Gelände bewegenden Fahrzeuge simuliert. Dabei ist der Leitstand nicht in der Lage zu unterscheiden, ob es sich um simulierte oder reale Fahrzeuge handelt. Somit entstand eine Gesamtsimulation, die zu Demonstrationszwecken eines autonomen Speditionshofes herangezogen werden kann. Da das Gelände in Weinstadt-Großheppach bereits über ein ausgebautes WLAN verfügt und Daimler-Chrysler ebenfalls im Besitz von Versuchsfahrzeugen[23] ist, sollte der Schritt hin zum realen Einsatz, in dem sich ein Lkw autonom auf dem Gelände in Großheppach bewegt, nicht mehr lange auf sich warten lassen.

[21] vgl. [LuT]

[22] z. B. Gelände, Fahrzeuge und Anwendungen

[23] die ausgerüstet mit einem PowerTrainController und einem integrierten Antriebsstrang in der Lage sind autonom zu fahren

1.6 Visualisierung

Um die Abläufe auf dem Gelände des Leitstandes eines autonomen Speditionshofes[24] darstellen zu können, bot sich die Entwicklung einer 3D-Visualisierung an. Diese wurde von Jörg Sesterhenn in seiner Diplomarbeit[25] spezifiziert und implementiert, sodass alle Abläufe übersichtlich dargestellt werden können. Doch nicht nur die reine Darstellung sollte durch die Visualisierung gewährleistet sein, sondern es sollte vielmehr auch eine Interaktion des Benutzers mit dem Leitstand ermöglicht werden. So könnte die Visualisierung als Überwachungsinstrument eingesetzt werden, durch das der Benutzer gezielt einzelne Fahrzeuge anhalten oder Informationen zu deren Status erfragen kann. Bereits bei der Entwicklung des Leitstandes stellte sich die Visualisierung als effizientes und leistungsstarkes Tool zur anschaulichen Darstellung von Testfällen heraus. Aus diesem Grund wird sie auch innerhalb dieser Arbeit ihren Einsatz finden und gegebenenfalls an die Anforderungen der Rollenden Landstraße angepasst werden müssen. Wie auch der Leitstand, setzt die Visualisierung auf der EZauto-Bibliothek auf.

1.7 Ablauf der Arbeit

Um der Aufgabenstellung[26] gewachsen zu sein, wurde sich in dieser Arbeit bei der Entwicklung an einem einfachen Wasserfallmodell orientiert, das sich über verschiedene Phasen der Softwareentwicklung erstreckt. Dabei wurde auf ein inkrementelles und iteratives Vorgehen, wie es z. B. im *Rational Unified Process (RUP)* angewendet wird, verzichtet, da bei dieser Projektgrößenordnung mit einem einfacheren Stufenmodell die gleichen Ergebnisse zu erwarten waren. Ebenso war somit etwas mehr Planungsspielraum innerhalb der einzelnen Phasen vorhanden. Der gesamte Verlauf dieser Arbeit ist in die folgenden sieben Phasen zerlegt worden:

1. Einarbeitung:
   In der ersten Phase dieser Arbeit ging es zunächst darum, sich in das Themengebiet des autonomen Fahrens einzufinden. Dazu erfolgte die Einarbeitung in die EZauto-Bibliothek. Dies schuf zum einen eine fundierte Wissensbasis für den weiteren Verlauf der Arbeit, frischte aber auch zum anderen die Kenntnisse und den Umgang mit der Programmiersprache C++ auf. Ebenso ging es darum eine geeignete Entwicklungsumgebung und alle

[24] vgl. Abschnitt 1.5
[25] vgl. [Ses]
[26] vgl. Abschnitt1.2

notwendigen Tools, die im weiteren Verlauf der Arbeit eingesetzt worden sind, auszuwählen.

2. Analyse RoLa:

   Die zweite Phase beschäftigte sich mit den thematischen Hintergründen der Rollenden Landstraße. Da diese bereits im PProLa und dem Projekt EZrola betrachtet worden ist, galt es nun die daraus gewonnenen Erfahrungen auszuwerten und in diese Arbeit zu übernehmen. Dabei ging es zunächst um die generellen Abläufe auf einem Terminal der Rollenden Landstraße, welche als Grundlage für den späteren Anwendungsfall des Leitstandes dienten. Um die Analyse des Ablaufes so realistisch wie möglich zu gestalten, wurde am 27.06.2006 das Terminal in Regensburg besichtigt. Aus dieser Besichtigung und den Erkenntnissen des PProLa wurde eine Anforderungsliste erstellt, die alle notwendigen Abläufe für eine autonome Abwicklung eines RoLa Terminals enthielt.

3. Analyse Leistand:

   Erst mit der Fertigstellung des Leitstandes der vorangehenden Arbeiten begann die dritte Phase dieser Arbeit. Hier galt es zunächst, das Gesamtkonzept kennen zu lernen und die Arbeitsweise der einzelnen Module zu analysieren. Es sollte festgehalten werden, was die beispielhafte Implementierung des Speditionshofes an Möglichkeiten für das Projekt EZrola bot und welche Funktionalitäten noch fehlten und damit implementiert werden mussten. Ebenso galt es die vorliegende Spezialisierung des Leitstandes auf ein allgemeines Niveau zu abstrahieren, um den Anwendungsfall der Rollenden Landstraße von einem *Allgemeinen Leitstand* aus zu realisieren. Mit den Ergebnissen der Phasen zwei und drei konnte ein Soll-Ist Abgleich vorgenommen werden, um die weitere Vorgehensweise festzulegen.

4. Entwurf / Spezifikation:

   Die am Leitstand notwendigen Änderungen in Bezug auf diese Arbeit wurden in Phase vier nun konkret spezifiziert. Dabei ging es darum, die fehlenden Module zu entwerfen und sie in das Gesamtkonzept einzubinden. Ebenso mussten die notwendigen Änderungen, welche zur generellen Funktionsweise des Leitstandes zählten und noch nicht implementiert waren, lokalisiert und festgehalten werden. Verbesserungsvorschläge für einige bekannte Mängel wurden an dieser Stelle ebenso angebracht.

5. Implementierung:

   Die in Phase vier festgehaltenen Änderungen wurden im Anschluss in Phase fünf umgesetzt und im Quelltext implementiert. Dabei galt es zunächst einige Mängel zu beheben und

funktionale Änderungen vorzunehmen, um im Anschluss weitere Module und Klassen zu implementieren, die zusätzliche Funktionalitäten gewährleisteten. Parallel dazu wurden die Änderungen fortwährend getestet, daher können die Phasen fünf und sechs kaum voneinander unterschieden werden. Zum Abschluss der Implementierung wurde ein beispielhaftes Szenario erstellt, das einem ausführlichen Test unterzogen wurde.

6. Test:
   Alle Änderungen sowie neue Strukturen wurden fortwährend getestet. Daher kann Phase sechs nicht als eigenständige Phase angesehen, sondern muss mit in die Phase der Implementierung einbezogen werden. Abschließend, nachdem alle Änderungen und Strukturen implementiert waren, wurde das gesamte System in einem kompletten Testszenario ausführlich getestet.

7. Dokumentation:
   Bereits während allen Phasen sind die einzelnen Ergebnisse dokumentiert worden. Diese wurden in der abschließenden Phase sieben ausgewertet und in dieser Ausarbeitung zusammengefasst.

In Zeichnung 1.1 wird der Verlauf der vorliegenden Arbeit skizziert und die einzelnen Phasen den Entwicklungsmonaten zugeordnet.

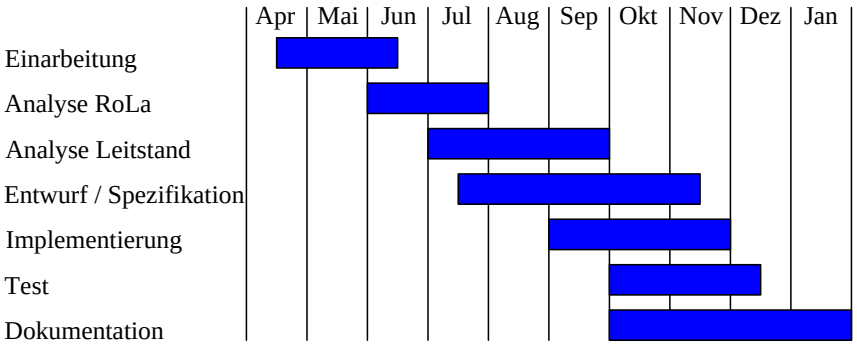


Abbildung 1.1: Ablauf der Arbeit

1.8 Aufbau der Ausarbeitung

Intension dieses ersten Kapitels war es die Hintergründe und Grundlagen der vorliegenden Arbeit zu erläutern. Dabei ging es sowohl um den thematischen Hintergrund und die Einordnung innerhalb der Forschungsgruppe Echtzeitsysteme, als auch die übersichtliche Darlegung aller Grundlagen, auf welchen die vorliegende Arbeit basiert. Anhand dieser wurden ebenso die genaue Aufgabenstellung aus der Sichtweise der Informatik, der Ablauf der Arbeit, sowie der dem Thema zugrunde liegende Anwendungsfall des Terminals der Rollenden Landstraße in Regensburg erläutert.

In Kapitel 2 wird Bezug auf das wissenschaftliche Umfeld der Fahrerlosen Transportsysteme genommen. Das Gesamtkonzept dieser Arbeit soll dort im Forschungsgebiet der Fahrerlosen Transportsysteme positioniert und ein Ausblick auf derzeit laufende Projekte gegeben werden. Ebenso werden zwei bereits umgesetzte Systeme als Beispiel genannt, ehe der spezielle Anwendungsfall dieser Arbeit behandelt wird.

Dieser wird anschließend in Kapitel 3 im Detail analysiert und die derzeitige Verwendung des Terminals mit allen Teilabläufen vorgestellt. Diese Analyse ist die Grundlage für den im Anschluss aufgezeigten Entwurf der autonomen Verwendung des Terminals, der in dieser Arbeit umgesetzt wird. Ebenso wird an dieser Stelle erklärt, wie die Datengrundlagen zur Darstellung des Geländes entstanden sind.

Das anschließendene Kapitel 4 beschäftigt sich zunächst mit der Grundlage dieser Arbeit: dem Leitstand für autonomes Fahren. Dabei soll zunächst der Allgemeine Leitstand vorgestellt und dessen Aufbau und mögliche Umgebung übersichtlich erklärt werden. Dieser Allgemeine Leitstand ist dabei die Grundlage der Spezialisierungen für die Anwendungsfälle von EZsped und EZrola, die beide ganz unterschiedliche Anforderungen mit sich bringen. Diese werden im Falle von EZsped übersichtlich und im Falle von EZrola detailliert erläutert. Diese Erläuterungen bilden die Grundlage für die Implementierungen des Leitstandes von EZrola. Ebenso sollen in Kapitel 4 einige Module des Allgemeinen Leitstandes genauer analysiert werden, da sie eine zentrale Bedeutung im Gesamtkonzept besitzen. Diese mussten im Rahmen dieser Arbeit teilweise geändert werden. Kapitel 4 stellt damit sozusagen die allgemeine Grundlage für Kapitel 5, welches sich ausschließlich mit dem Leitstand von EZrola beschäftigt. An dieser Stelle sollen alle

notwendigen Implementierungen vorgestellt werden, die zum einen den Allgemeinen Leitstand zur anwendungsfallspezifischen Spezialisierung heranreifen lassen, aber auch dessen bloße Funktionalität erweitern. Ebenso werden hier einige Verbesserungen aufgezeigt, welche im Rahmen dieser Arbeit am Allgemeinen Leitstand vorgenommen worden sind.

Das abschließende Kapitel 6 übernimmt die Bewertung der mit dieser Arbeit angestrebten Zielsetzung. An dieser Stelle soll ganz konkret Bezug auf einzelne Gesichtspunkte der Softwaretechnik genommen und das vorliegende Konzept des Allgemeinen Leitstandes bewertet werden. Es soll ein Ausblick gegeben werden, zum einen darauf, wie der Allgemeine Leitstand auf einen anderen Anwendungsfall angepasst kann kann, aber auch zum anderen, welche nachfolgenden Arbeiten geleistet werden können oder sogar geleistet werden müssen, um einen sinnvollen Einsatz überhaupt zu garantieren.

Sofern programmiertechnische Details in Bezug auf den Leitstand, sowie möglicherweise vorgenommene Änderungen, nicht in Kapitel 5 enthalten waren, finden sich diese abschließend in Kapitel 7, dem Anhang zu dieser Arbeit, wieder.

Kapitel 2

Positionierung im Bereich der Fahrerlosen Transportsysteme

Im folgenden Kapitel soll der dieser Arbeit zugrunde liegende Leitstand für autonomes Fahren im Gebiet der Fahrerlosen Transportsysteme (FTS) eingeordnet werden. Dazu wird zunächst eine allgemeine Definition eines FTS nach VDI-Norm betrachtet und im Anschluss mögliche Kategorien aufgezeigt, nach welchen sich bestehende Fahrerlose Transportsysteme markant voneinander unterscheiden können. Abschließend wird das zugrunde liegende Konzept in aktuellen Forschungsgebieten positioniert.

2.1 Definition: Fahrerlose Transportsysteme (FTS)

Definition[1]:

„Fahrerlose Transportsysteme (FTS) sind innerbetriebliche, flurgebundene Fördersysteme mit automatisch gesteuerten Fahrzeugen, deren primäre Aufgabe der Materialtransport, nicht aber der Personentransport ist“.

[1] vgl. [VDI] S. 6f

FTS werden innerhalb und außerhalb von Gebäuden eingesetzt und bestehen im Wesentlichen aus folgenden Komponenten:

- einer Leitsteuerung
- einem oder mehreren Fahrerlosen Transportfahrzeugen
- Einrichtungen zur Standortbestimmung und Lageerfassung
- Einrichtungen zur Datenübertragung
- Infrastruktur und periphere Einrichtungen

Definition[2]:

„Ein Fahrerloses Transportfahrzeug (Abk. FTF) oder Englisch Automated Guided Vehicle (AGV) ist ein flurgebundenes Fördermittel mit eigenem Fahrantrieb, das automatisch gesteuert und berührungslos geführt wird. Fahrerlose Transportfahrzeuge dienen dem Materialtransport, und zwar zum Ziehen oder Tragen von Fördergut mit aktiven oder passiven Lastaufnahmemitteln“.

Der dieser Arbeit zugrunde liegende Leitstand für autonomes Fahren erfüllt beide genannten Definitionen und weist alle angegeben notwendigen Komponenten auf. Anhand dieser Komponenten können bestehende Fahrerlose Transportsysteme unterschieden werden, da sie je nach Anwendungsfall in unterschiedlichen Ausprägungen vorliegen können.

2.2 Unterscheidungskriterien und allgemeine Beschreibung

So allgemein die Definition der Fahrerlosen Transportsysteme in Abschnitt 2.1 durch die VDI-Richtlinie formuliert ist, so unterschiedlich kann deren Ausprägung und Umsetzung je nach Anwendungsfall ausfallen. Das zentrale Element eines FTS ist dabei stets die Leitsteuerung, die alle notwendigen Berechnungen durchführen und die Einhaltung jeglicher Sicherheitsrichtlinien zur Laufzeit garantieren muss. Dies kann durch unterschiedliche Architekturen geschehen, deren

[2] vgl. [VDI] S. 6f

wesentlichen Merkmale eine Kategorisierung der eingesetzten Leitsteuerungen zulassen[3].

Im Fokus stehen dabei die folgenden vier Kriterien:

- die Durchführung der Berechnungen
- das zugrunde liegende Wegenetz
- die Art und Weise der Planung
- das vorliegende Sicherheitskonzept und dessen Strukturen

Werden die Berechnungen ausschließlich zentral durchgeführt, so nimmt die Leitsteuerung eine allwissende Position innerhalb des FTS ein. Sie übernimmt jegliche Planung und instruiert die ihr unterstellen Fahrzeuge. Diese führen lediglich Befehle aus und besitzen keine Kenntnis über das Gelände oder andere Fahrzeuge. In einer dezentralen Ausprägung hingegen, werden alle notwendigen Berechnungen auf die angemeldeten Fahrzeuge verteilt, sodass jedes Fahrzeug die für seine Bewegung notwendigen Berechnungen selbst durchführt. Dazu ist die Kenntnis über das Gelände, sowie über die sich ebenso auf diesem Gelände befindlichen anderen Fahrzeuge notwendig. Dies setzt somit eine erhöhte Kommunikation zwischen den Fahrzeugen voraus.

Um nun eine Route vom Start zum Zielpunkt einplanen zu können, benötigt die Leitsteuerung Angaben über die möglichen Fahrwege, die dem Fahrzeug zur Verfügung stehen. Dazu gibt es zwei mögliche Ansätze. Ein statisches Wegenetz gibt, unter der Berücksichtigung der von den Fahrzeugen gefahrenen Trajektorien, alle Fahrwege fest vor. Dieses Wegenetz wird vor dem Einsatz der Leitsteuerung entworfen und erschließt das gesamte Gelände mit möglichen Fahrwegen, die in einer Vorgänger-Nachfolger-Beziehung zueinanderstehen. Soll nun eine Route geplant werden, so muss lediglich ein Suchalgorithmus einen Weg vom Start- zum Zielpunkt finden. Ein dynamisches Wegenetz hingegen, sieht keinerlei feste Fahrwege vor. Hier erkennt die autonome Einheit die Umgebung selbstständig und plant anhand der vorliegenden Daten den Fahrweg ein. Weiterhin existieren Mischformen, die statische und dynamische Wegenetze miteinander vereinen.

Bevor jedoch ein Fahrzeug eine zugewiesene Route abfahren kann, muss diese zunächst im Plan der Leitsteuerung eingetragen und kollisionsfrei ausführbar sein. Wird dabei stets ein fester

[3] vgl. [LuT] Abschnitt 2.2

Plan vor der Ausführung erstellt und dieser Plan auch zur Laufzeit nicht mehr verändert, so handelt es sich um eine statische Planung. Jede Änderung an einem in der Ausführung befindlichen Plan würde eine komplette Neuplanung hervorrufen. Ist es hingegen möglich, Pläne zur Laufzeit zu modifizieren, ohne eine komplette Neuplanung hervorzurufen, so handelt es sich um eine dynamische Planung.

Analog dazu kann das zugrunde liegende Sicherheitskonzept anhand seiner Strukturen unterschieden werden. Die Überwachung der Fahrzeuge kann dabei durch statische aber auch durch dynamische Strukturen gewährleistet sein. Statische Strukturen sind dabei sehr oft an ein statisches Wegenetz geknüpft und werden auf der Grundlage der Fahrwege berechnet. Dazu wird die abzufahrende Strecke in sicherheitstechnische Sektoren unterteilt. Diesen Sektoren werden Zeitpunkte zugewiesen, zu denen sich das Fahrzeug auf seinem Weg vom Start- zum Zielpunkt in einem solchen Sektor befinden soll. Dabei kann ein Sektor immer nur von einem Fahrzeug befahren werden und wird in diesem Falle für alle anderen Fahrzeuge gesperrt[4].
Bei rein dynamischen Strukturen wird komplett auf die Zuweisung fester Sektoren verzichtet, da der Aufenthaltsort jedes Fahrzeuges fortwährend neu berechnet wird. Hierzu bedarf es präziser Berechnungen sowie Fahrzeugdaten, damit der vom Fahrzeug belegte Raum stets exakt berechnet werden kann.

Alle Kriterien vereinen in ihren unterschiedlichen Ausprägungen diverse Vor- und Nachteile, die je nach Anwendungsfall verschieden stark ins Gewicht fallen. Eine genaue Analyse dieser allgemeinen Gesichtspunkte wird an dieser Stelle nicht vorgenommen. Stattdessen wird auf Abschnitt 2.2 der Vorgängerarbeit in Quelle [LuT] verwiesen. Ebenso findet dort in Abschnitt 2.3 eine Einordnung des Allgemeinen Leitstandes in die genannten Kriterien statt, sodass auf deren Basis nun im folgenden Abschnitt eine Einordnung des Konzeptes in aktuelle Forschungsgebiete vollzogen werden kann.

[4] ähnlich einer Semaphore

2.3 Positionierung innerhalb der Forschungsgebiete

Fahrerlose Transportsysteme werden in den unterschiedlichsten Anwendungsgebieten eingesetzt, da sie in vielerlei Hinsicht konventionelle Methoden in ihrer Effizienz übertreffen. Dabei finden sie in kleineren Systemen, mit nur einem einzigen autonomen Fahrzeug, aber auch in großen Terminals, mit einer Vielzahl von autonomen Transporteinheiten, ihren Einsatz. Aufgrund der steigenden Anzahl an eingesetzten Fahrerlosen Transportsystemen ist die Entwicklung eines unabhängigen Referenz-Architekturmodells wünschenswert. Ein solches Modell sollte alle notwendigen Kernfunktionalitäten enthalten und diese einer strukturellen Ordnung unterwerfen. Neben dem dieser Arbeit zugrunde liegenden Modell existiert noch eine Vielzahl weiterer Architekturen. Diese sind zwar je nach Anwendungsfall unterschiedlich geprägt, weisen aber alle auf konzeptioneller Ebene ähnliche Strukturen auf.

Ein grober Überblick bisher entworfener Architekturen sowie eine eigene Modell-Empfehlung in deren wissenschaftlichem Kontext wurden von S.Hallé, F.Gilbert, J. Lauchmonier sowie B.Chaibdraa[5] vorgestellt. Alle geannten Architekturen weisen dabei, wie der in diesem Projekt eingesetzte Leitstand, meist eine Schichtenaufteilung mit jeweils einzelnen Modulen auf. Die Schichten stellen dabei unterschiedliche Abstraktionsebenen dar, welche über fest definierte Schnittstellen miteinander kommunizieren. Dadurch ist gewährleistet, dass die einzelnen Schichten unabhängig voneinander modifiziert oder komplett austauscht werden können. Diese Schichten setzen sich aus verschiedenen Modulen zusammen, die je nach Ausprägung unterschiedlich implementiert sind. Durch ihre große Abhängigkeit vom Anwendungsfall kommt dabei eine Vielzahl möglicher Umsetzungen in Frage, die je nach Kontext Gegenstand der Forschung sind.

Eine Zusammenfassung der einzelnen Thematiken liefert der in der Zeitschrift „European journal of operational research“ veröffentlichte Artikel von Tuan Le-Anh und M. B. M De Koster[6]. Dort werden unter anderem folgende Themen diskutiert: der Entwurf eines Leitlinienkonzeptes, das Scheduling und die Verwaltung untätiger Fahrzeuge, die Routenplanung und die Konfliktbehandlung. Alle diese Themen mussten auch im Konzept des Allgemeinen Leitstandes berücksichtigt werden. Besonders die Konfliktbehandlung sollte für anstehende Arbeiten im Anschluss von großer Bedeutung sein, da innerhalb dieses Projektes bisher noch kein eigenes Konzept entwickelt worden ist. Dabei kann von den Überlegungen vieler Veröffentlichungen profitiert werden.

[5] vgl. [Hal]
[6] vgl. [Anh]

Als Beispiel dafür sei hier der Artikel von Kap Hwan Kim, Su Min Jeon und Kwang Ryel Ryu[7] genannt, der eine Möglichkeit zur Deadlock-Verhinderung vorstellt. Diese basiert auf einer Sektoraufteilung des Geländes und sieht verschiedene Funktionen zu deren Verwaltung vor.

Kap Hwan Kim setzte sich ebenso mit dem Einsatz autonomer Transportsysteme im Bereich großer Container-Terminals auseinander und brachte im Kooperation mit Hans-Otto Günther eine Sammlung ausgewählter Artikel zu diesem Thema heraus[8]. Unter anderem werden Thematiken wie die verschiedenen einsetzbaren autonomen Fahrzeugtypen[9] oder eine Auftragszuweisung an die verfügbaren Fahrzeuge[10] beschrieben. Auch die Raum-Zeit-Planung des Leitstandes bedarf einer Überarbeitung und könnte sich am Artikel von Ling Qui, Wen-Jing Hsu, Shell-Ying Huang und Han Wang[11] orientieren, der eine Zusammenfassung über Scheduling und Routing Algorithmen enthält.

Zahlreiche Fahrerlose Transportsysteme befinden sich derzeit im Einsatz oder in der Entwicklung, lassen jedoch zumeist keinen Einblick in die zugrunde liegende Architektur zu, da sie industriell entwickelt und vertrieben werden. Eine frühzeitige Veröffentlichung könnte an dieser Stelle finanziellen Schaden verursachen. Daher ist es kaum möglich eine verwendete Architektur mit der des Leitstandes genauer zu vergleichen und es bleibt lediglich die Nennung einiger umgesetzter Beispiele. Zwei werden nun im Folgenden aufgeführt.

Für das Untergrund-Logistik System des Amsterdamer Flughafens Schiphol wurde ein Konzept für ein autonomes unterirdisches Warentransportsystem[12] entworfen, welches unter anderem auf dem Konzept von TRACES (Traffic Control Engineering System) aufbaut. Dabei handelt es sich um ein semaphorenbasiertes Leitsystem, das auch in einer Simulation des Delta Sealand Terminal (DSL) Maasvlakte eingesetzt worden ist[13].
Ein dem Leitstand-Konzept dieser Arbeit augenscheinlich näher stehendes Beispiel stellt das Container-Terminal Altenwerder in Hamburg dar. Dieses wird mit einer in Java implementierten autonomen Architektur betrieben, die das Terminal im Client-Server-Betrieb steuert[14]. Auch

[7] vgl. [Kim-b]
[8] vgl. [Kim-a]
[9] vgl. [Vis]
[10] vgl. [Lim]
[11] vgl. [Qui]
[12] vgl. [Ver]
[13] vgl. [Dui]
[14] vgl. [Kra]

hier finden sich dem Leitstand-Konzept ähnliche Architekturmerkmale wieder.

Abschließend kann gesagt werden, dass eine Einordnung des Leitstand-Konzeptes in die aktuelle Forschung problematisch ist, da z. B. für das Thema Fahrzeugsübergabe aus dem manuellen in den autonomen Betrieb keine Referenz zu finden ist. Des Weiteren arbeiten viele bestehende Systeme mit einer geringeren Anzahl an Fahrzeugen und tauchen deshalb nicht so tief in die Komplexität der Raum-Zeit-Planung ein. Dennoch weist der Leitstand mit den großen Container-Terminals diverser Hafenanlagen große Gemeinsamkeiten auf. Da in beiden Fällen viele Fahrzeuge und deren Aufträge koordiniert werden müssen, bewegen sie sich somit im gleichen wissenschaftlichen Kontext.

Kapitel 3

Rollende Landstraße Regensburg

In diesem Kapitel wird die Rollende Landstraße als thematischer Hintergrund dieser Arbeit betrachtet und deren Abläufe am Beispiel des Terminals in Regensburg genauer dargestellt. Dazu wird zunächst die aktuelle Nutzung analysiert und im Anschluss die mögliche autonome Verwendung eines solchen Terminals vorgestellt, die in dieser Arbeit umgesetzt wurde. Darüber hinaus wird skizziert, wie die Grundlagen zur Darstellung der 3D-Visualisierung entstanden und welche Programmstrukturen dazu verwendet worden sind.

3.1 Entstehung und Betreiber des Terminals

Das Terminal Regensburg ist Teil der RoLa-Verbindung Graz/Werndorf - Regensburg und wurde am 02. November 2005[1] mit der Inbetriebnahme eines Zugpaares pro Tag eröffnet. Beide Terminals wurden im Rahmen zweier PPP[2] - Projekte der SCHIG mbH[3] geplant und erstellt. Grundgedanke der PPP - Projekte ist die Einbringung von privatem Kapital und Know-how in den Ausbau und/oder den Betrieb von Infrastrukturanlagen in Form attraktiver Modelle, die damit nicht zuletzt eine Forcierung des Wettbewerbs erreichen sollen. So wurde im Rahmen eines PPP - Projektes ab 1997 ein Modell für das Terminal Graz/Werndorf erstellt und im Anschluss als Musterbeispiel für ein Terminal der RoLa umgesetzt. Um die bestehenden RoLa-Verbindungen zu entlasten, konnte durch die Errichtung des Terminals in Regensburg eine RoLa-Verbindung zum bestehenden Terminal Graz/Werndorf eingesetzt werden. Dabei fungiert Regensburg als

[1] vgl. [SCHIG-a]
[2] Public Private Partnership
[3] Schieneninfrastruktur-Dienstleistungsgesellschaft mbH

Bündelungspunkt[4] sowohl für die Verkehrsströme in Richtung Osten und Südosten, wie auch für die Transporte aus Ost- und Südeuropa, die in den Norden und Westen Europas weitergeleitet werden. Mit einem Investitionsvolumen von 2,5 Mio. Euro wurde hier die notwendige Infrastruktur geschaffen und die Umbaumaßnahmen am Terminal Regensburg vorgenommen. Partner im Rahmen des PPP - Projektes waren die Unternehmen Cargo Center Graz (CCG), Rail Cargo Austria AG, Ökombi, Bayernhafen Regensburg, die Länder Steiermark und Oberösterreich, das Bundesministerium für Verkehr, Innovation und Technologie (BMVIT) sowie die SCHIG mbH.

Die Verantwortungen der Betreibung des Terminals wurden nach Errichtung auf den Bayernhafen Regensburg als Infrastrukturbetreiber, sowie die Ökombi GmbH[5] als Operateur der RoLa übertragen. Letztere ist auch direkter Ansprechpartner für die Kunden der RoLa und übernimmt alle notwendigen Buchungen.

3.2 Aktuelle Verwendung: manuelle Fahrt

Um die aktuelle Nutzung und die Abläufe innerhalb eines Terminals der Rollenden Landstraße so realistisch wie möglich abbilden zu können, wurde das Terminal in Regensburg am 27. Juni 2006 besichtigt. Dabei ging es zunächst darum, einen tieferen Einblick zu gewinnen, aber auch mögliche Problemstellungen in Bezug auf eine autonome Nutzung zu erkennen und diese, wenn möglich, zu beheben.

Wie in Abschnitt 1.3 bereits erläutert, handelt es sich bei der aktuellen Verwendung der Rollenden Landstraße um einen kombinierten und begleiteten Güterverkehr, d. h. die Sattelzüge werden komplett von den Fahrern verladen. Diese begleiten im Weiteren ihren Sattelzug in einem Personenbegleitwaggon der Deutschen Bahn und setzen nach der Ankunft am Zielbahnhof die Reise fort.

Wie die einzelnen Abläufe beim Be- und Entladen der Züge aussehen, wird nun im Weiteren genauer betrachtet. Dazu wird der Lageplan vom 08.08.2003 des Vermessungsamtes in Regensburg verwendet, der freundlicherweise durch Mitarbeiter des Terminals in Regensburg zur Verfügung gestellt worden ist. Die Abbildung 3.1 auf Seite 42 skizziert die aktuelle Verwendung

[4] vgl. [SCHIG-b], Seite 6, Abschnitt 2

[5] Tochterunternehmen der Rail Cargo Austria AG

sowie die notwendigen Abläufe auf dem Terminal-Gelände. Diese werden anhand der einzelnen Stationen eines Lkw auf dem Terminal genauer erläutert. Doch bevor darauf eingegangen wird, sollen zunächst einige allgemeine Aspekte zum Terminal betrachtet werden.

3.2.1 Terminal Regensburg im Überblick

Das Terminal in Regensburg erstreckt sich über eine Fläche von etwa 24500 m^2, welche von den Betreibern des Bayernhafens Regensburg zur Verfügung gestellt worden ist. Maximal soll eine Kapazität von zwölf Zugpaaren am Tag realisiert werden können, wozu 44 Stellplätze für die ankommenden Lkw vorhanden sind. Alle Abläufe auf dem Gelände werden von insgesamt drei Angestellten, davon zwei des Betreibers und einer der Deutschen Bahn, betreut.

Zum Zeitpunkt der Besichtigung wurde das Terminal mit zwei Zugpaaren betrieben, eines am frühen Morgen, mit der Ausfahrt des Zuges gegen 7 Uhr, und eines am späteren Abend, mit einer Abfahrtszeit von etwa 21:45 Uhr[6]. Allerdings war das erste Zugpaar aus organisatorischen Gründen vom Betreiber ausgesetzt, sodass lediglich die Abläufe des Zugpaares am Abend besichtigt werden konnten.

Die verwendeten Züge bestanden aus der Lok, einem Begleitwaggon und 21 Niederflurwaggons zum Transport der Lkw. Nach der Einfahrt des Zuges, was am Tage der Besichtigung schon gegen 14 Uhr geschehen war, wurde die Lok und der Begleitwaggon vom restlichen Zug getrennt und auf einem separaten Gleis abgestellt. Daraufhin konnten die Lkw die Niederflurwaggons über eine angestellte Rampe verlassen und setzten ihre weitere Reise fort. Die auf dem Gleis stehenden Waggons waren somit wieder für den nächsten auslaufenden Zug bereit und wurden lediglich vom Wagenmeister der Deutschen Bahn einer Sichtprüfung unterzogen.

Mit den vorhandenen 21 Niederflurwaggons gibt es ebenso viele Stellplätze für Lkw auf dem Zug. Diese können über ein Buchungssystem des Betreibers im Internet eingesehen und telefonisch gebucht werden. Bei den Interessenten handelt es sich meist um große Speditionen, die mehrere Stellplätze buchen und diese intern an ihre Fahrer weiterleiten. So ist es z. B. auch möglich, dass im Falle des Ausfalls eines Fahrzeuges auf dem Weg zum Terminal, ein Fahrer-Kollege mit einem anderen Fahrzeug den Stellplatz auf dem Zug einnehmen kann. Buchungen einzelner Unternehmer bildeten mit lediglich zwei Stellplätzen an diesem Tag die Ausnahme, die restlichen

[6] aktuelle Fahrpläne unter www.oekombi.de

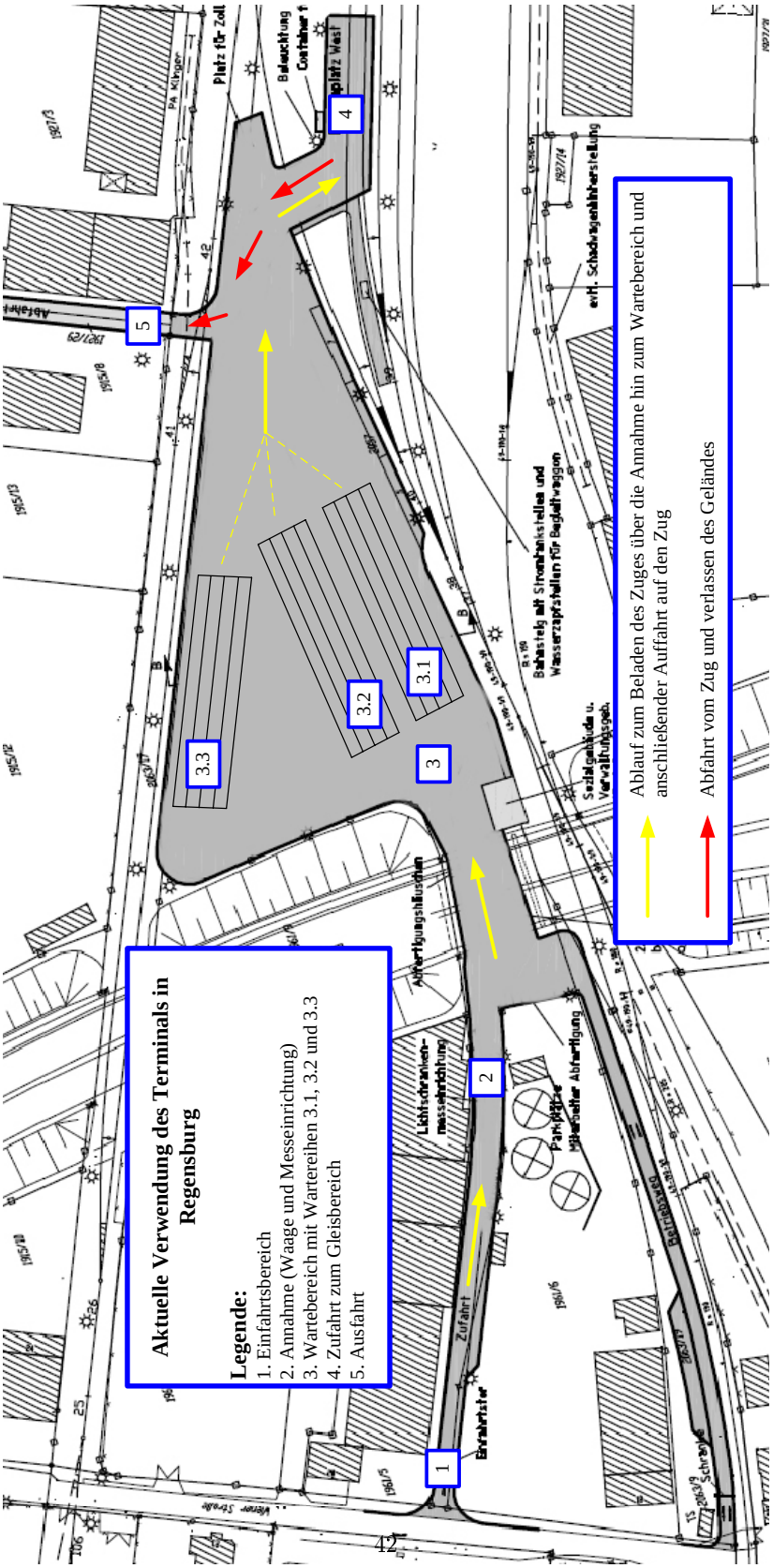


Abbildung 3.1: Aktuelle Verwendung des Terminals in Regensburg

wurden auf zwei Speditionen aufgeteilt.

Um die pünktliche Ausfahrt des Zuges zu gewährleisten, müssen alle Lkw das Terminal bis zu einer gewissen Uhrzeit befahren haben. Im Falle des Abendzuges mit der Ausfahrt um 21:45 Uhr war der Annahmeschluss um 19:00 Uhr, den alle Fahrzeuge an diesem Abend einhielten. Wäre dies nicht der Fall gewesen, so wäre dem Fahrer der Transport auf der Schiene an diesem Tag verweigert worden. Alle rechtzeitig angekommenen Fahrzeuge warteten von da an auf dem Gelände bis zur Auffahrt auf den Zug.

Etwa die Hälfte der Fahrzeuge traf am Tage der Besichtigung sogar schon vor 18 Uhr ein, die andere Hälfte innerhalb der Stunde zw. 18 und 19 Uhr. Zwei Fahrzeuge einer Spedition warteten schon seit dem frühen Nachmittag auf den Terminal, allerdings ohne die zugehörigen Fahrer. Diese hatten die Fahrzeuge lediglich abgestellt und die notwendigen Unterlagen an Kollegen der Spedition übergeben. So konnten später ankommende Fahrer ihr eigenes Fahrzeug, sowie das wartende Fahrzeug des Kollegen, auf den Zug auffahren. Am Zielbahnhof nahm dann ein anderer Fahrer der Spedition dieses Fahrzeug entgegen. Die Wartezeiten auf dem Terminal wurden von den Fahrern zur Verpflegung und Erfrischung in den bereitgestellten sanitären Anlagen genutzt.

Die Auffahrt der Lkw auf den Zug wurde nach der Annahme aller Fahrzeuge durch die Mitarbeiter des Terminals geleitet. Waren alle Lkw ordnungsgemäß verladen und gesichert, so wurden der Begleitwaggon und die Lok angekuppelt und der Zug verließ nach dem Einsteigen der Fahrer das Terminal.

In dieser groben Beschreibung wurde bisher nicht auf die genaue Dauer der einzelnen Vorgänge, z. B. der Annahme, der Abfahrt vom und der Auffahrt auf den Zug, eingegangen. In den folgenden Ablaufbeschreibungen wird auf diese Zeitspannen kurz eingegangen, für genauere Angaben sei jedoch auf die Diplomarbeit von Andreas Kern verwiesen, welcher sich ganz gezielt mit dieser Thematik auseinandergesetzt und verschiedene Verfahren und Optimierungen entwickelt hat.

Um nun die Abläufe genauer zu beschreiben, werden im Anschluss die einzelnen Stationen der Lkw auf dem Terminal anhand der Karte aus Abbildung 3.1 betrachtet.

3.2.2 Einfahrt und Anmeldung

Die Zufahrt zum Gelände befindet sich auf der Karte in Punkt 1 und führt direkt zur Anmeldung in Punkt 2. Diese besteht momentan aus einer Waage, einer Lichtschrankenmesseinrichtung sowie einem kleinen Büro für die Angestellten des Terminals. Die Lkw werden hier auf die Einhaltung der Richtlinien[7] zum Transport auf den Niederflurwaggons geprüft. Bei Mängeln wird dem Fahrer die Möglichkeit gegeben, diese zu beseitigen. Sollte dies nicht innerhalb kürzester Zeit möglich sein, so wird der Transport dieses Lkw auf der Schiene abgelehnt und der Fahrer muss das Gelände mit seinem Lkw über den Betriebsweg verlassen. Diese Ablehnung stellt allerdings die absolute Ausnahme dar und kommt, laut den Angaben eines Terminalangestellten, im Jahr maximal vier bis fünf Mal vor. Die Waage an dieser Annahmestelle ist in einer überhöhten Position zum Erdboden angebracht und nur über kleinere Betonrampen zu erreichen. Dies gewährleistet, dass nur Fahrzeuge das Terminal befahren, die diese Rampen überfahren können, da dies zum Auffahren auf den Zug erforderlich ist. Während der Vermessung kann der Fahrer dabei bequem aus seinem Führerhaus heraus alle zur Abwicklung notwendigen Daten, z. B. Angaben zum Fahrzeug, der Spedition, der Ladung etc. dem Terminalmitarbeiter durch das geöffnete Fenster des Büros weiterleiten. Sollte dies erfolgreich vollzogen sein, so setzt der Fahrer die Fahrt zum Wartebereich fort. Die Gesamtdauer des Vermessens, Wiegens und der Annahme an dieser Stelle beträgt etwa 5 min je Fahrzeug. Abbildung 3.2 zeigt diese Annahmestelle.

Abbildung 3.2: Annahmestelle in Regensburg

[7] zu detaillierten Hinweisen vgl. [Oek-b]

3.2.3 Wartebereich

Der Wartebereich beginnt in Position 3 der Abbildung 3.1 und unterteilt sich in die Wartebereiche A (Position 3.1), B (Position 3.2) und C (Position 3.3). Jeder Wartebereich besteht aus mehreren Wartespuren, in denen sich die Lkw hintereinander aufstellen können. Die Zuteilung der Wartebereiche und Spuren wird an der Annahme vorgenommen und dem Fahrer mitgeteilt, sodass er den ihm zugewiesenen Platz mit seinem Lkw anfahren kann. Dabei ist zu beachten, dass die Sattel- und Gliederzüge, aufgrund ihrer unterschiedlichen Längen, später abwechselnd auf den Zug auffahren müssen. Ein Gliederzug ist länger als ein Sattelzug, daher würde ein Teil des Gliederzuges über den Niederflurwaggon hinausstehen und damit verhindern, dass ein weiterer Gliederzug direkt dahinter korrekt platziert werden kann. Durch einen kürzeren Sattelzug wird dies ausgeglichen. Um daher die Beladung des Zuges so effizient und schnell wie möglich zu gestalten, werden Glieder- und Sattelzüge in unterschiedlichen Wartereihen getrennt voneinander platziert. Dies hat den Vorteil, dass beim Beladen des Zuges später im einfachen Reißverschlussverfahren die Mischung von Gliederzug - Sattelzug - Gliederzug sehr leicht gewährleistet werden kann. Dabei entscheidet ein Terminalmitarbeiter, welche Wartereihe von welchen Fahrzeugen befahren wird. Abbildung 3.3 zeigt Lkw in den Wartebereichen A und C.

Abbildung 3.3: Wartebereiche in Regensburg

3.2.4 Auffahrt auf den Zug

Steht nun auf dem Gleis ein Zug zum Beladen bereit, beginnt nach den Anweisungen eines Terminalmitarbeiters die Auffahrt auf die Niederflurwaggons. Dabei legt der Mitarbeiter die Reihenfolge der auffahrenden Lkw fest, woraufhin die Fahrer ihre Lkw in den Gleisbereich in

Position 4 der Abbildung 3.1 steuern. Dort können sie die Niederflurwaggons über eine angestellte Rampe befahren. Das Befahren dieser Rampe und die Fahrt auf den Waggons verlangt dabei vom Fahrer erhöhte Präzision, denn die Fahrspur ist nur etwa 15 bis 20 cm breiter als die Außenmaße des Lkw. Dabei ist ebenso zu bedenken, dass der erste auffahrende Lkw die gesamte Länge des Zuges bis zu seinem Stellplatz am anderen Ende überfahren muss. Aufgrund der beengten Platzverhältnisse geschieht das Auffahren in Schrittgeschwindigkeit. Die Abbildungen 3.4 und 3.5 zeigen das Befahren der Niederflurwaggons über die Rampe, in Abbildung 3.6 ist die Fahrt der Lkw über die Zuglänge hinweg zu sehen.

Abbildung 3.4: Auffahrt auf den Zug über die Rampe (1)

Abbildung 3.5: Auffahrt auf den Zug über die Rampe (2)

Abbildung 3.6: Fahrt der Lkw auf dem Zug

3.2.5 Positionierung und Sicherung auf dem Zug

Bevor der Zug nach dem Befahren der Lkw das Terminal verlassen kann, müssen alle Lkw die richtige Position auf dem jeweiligen Niederflurwaggon eingenommen haben. Dazu werden sie anhand vorhandener Markierungen, welche in Abbildung 3.7 zu sehen sind, vom Wagenmeister der Deutschen Bahn eingewiesen. Haben die Lkw ihre Endposition erreicht, kann der Fahrer mit dem Sichern des Lkw beginnen. Dies beinhaltet z. B. das Anklappen der Außenspiegel, das Ablassen der Luftfedern, das Unterlegen von Reifenkeilen und das Entfernen jeglicher Radioantennen. Die korrekte Sicherung wird ebenso durch den Wagenmeister überprüft und erst nach dessen Abnahme darf sich der Fahrer zum Begleitwaggon begeben. Abbildung 3.8 zeigt den Wagenmeister bei der Einweisung, nach der die Fahrer im Anschluss mit der Sicherung ihrer Lkw beginnen (vgl. Abbildung 3.9).

3.2.6 Ankunft und Abfahrt vom Terminal

Die bisher beschriebenen Abläufe beziehen sich rein auf die Ankunft der Lkw am Terminal über die Straße und deren Verladung. Kommt hingegen ein beladener Zug am Terminal an, so gestaltet sich der Ablauf sehr einfach. Die Fahrer entsichern ihre Fahrzeuge und bereiten diese zur Abfahrt vor. Sobald die Rampe vor dem Zug positioniert ist, beginnen die Fahrer mit dem Verlassen des Zuges. Da bei der Ankunft keinerlei Formalitäten zu erledigen sind, können die Fahrer das Terminal sofort über die Ausfahrt in Position 5 verlassen. Sobald der Zug komplett entladen ist, wird er vom Wagenmeister inspiziert und steht von diesem Moment an für das neue Beladen zur Verfügung.

Abbildung 3.7: Positionsmarkierungen auf den Niederflurwaggons

Abbildung 3.8: Einweisung durch den Wagenmeister

Abbildung 3.9: Fahrer bei der Sicherung der Lkw

3.3 Mögliche Verwendung: Prototyp zur autonomen Fahrt

Im folgenden Abschnitt soll nun der autonome Entwurf zur Verwendung des Terminals in Regensburg vorgestellt werden. Dazu werden zunächst die sich ergebenden Anforderungen an das Terminal festgehalten, die im Anschluss in einem konkreten Entwurf umgesetzt und ausführlich erklärt werden. Analog zur manuellen Verwendung werden hier alle Abläufe einzeln dargestellt. Abschließend wird kurz aufgezeigt, wie die Grundlagen zur 3D-Darstellung der Visualisierung entstanden sind.

3.3.1 Anforderungen

Der in Abschnitt 3.2 dargestellten manuellen Verwendung des Geländes in Regensburg soll nun eine mögliche autonome Verwendung gegenübergestellt werden. Der Grundgedanke hierbei ist, die wesentlichen Abläufe auf dem Gelände ohne den Fahrer, d. h. durch eine autonome Fahrt der Fahrzeuge[8], durchzuführen. So könnte z. B. ein Fahrer den Lkw lediglich am Start-Terminal übergeben und ein anderer diesen am Ziel-Terminal wieder entgegennehmen, ohne dass der besagte Lkw auf dem Weg von Start- zum Ziel-Terminal durch einen Fahrer begleitet werden muss. Die Rollende Landstraße würde somit den Schritt zum unbegleiteten kombinierten Güterverkehr vollziehen. Doch um dies zu verwirklichen, sind zunächst alle zusätzlichen Elemente und Anforderungen, die mit einer autonomen Fahrt an das Terminal gestellt werden, in einem konkreten Entwurf festzuhalten.

[8] die von einem Leitrechner überwacht wird

Um dies umzusetzen, müsste das gesamte Terminal von einem Leitrechner organisiert werden, der das autonome Fahren steuert. Dazu braucht es zum einen eine Übertragungstechnologie, um Befehle an die einzelnen Fahrzeuge zu senden, und zum anderen ein präzises Positionierungssystem. Über dieses muss jederzeit die Position jedes Fahrzeuges genau bestimmt werden können. Bei der Übertragungstechnologie könnte z. B. auf ein WLAN zurückgegriffen werden, das mit ausreichenden Sicherheitsmaßnahmen vor dem Zugriff Dritter geschützt wird. Bei einem passenden Positionierungssystem besteht derzeit noch Entwicklungsbedarf, da in einem solchen Falle aktuelle Systeme wie z. B. GPS keine ausreichende Genauigkeit liefern. Solche Systeme sind Gegenstand der aktuellen Forschung und sollten in absehbarer Zeit verfügbar sein. Neben den beiden bisher genannten Punkten kommt als dritter die Fähigkeit von Fahrzeugen hinzu, sich einer autonomen Fahrt unterordnen zu können.

Bereits an dieser Stelle existieren erste Einschränkungen, denn nicht alle Fahrzeuge von Kunden der Rollenden Landstraße können einfach und kostengünstig technisch so modifiziert werden, dass eine autonome Fahrt möglich wäre. Aus diesem Grund empfiehlt sich der Einsatz von autonomen Terminaltraktoren.

Ein Terminaltraktor ist eine modifizierte Zugmaschine, die in der Lage ist, Sattelauflieger sowie jegliche Art von Anhängern anzukuppeln und über das Gelände zu ziehen. Dazu sind weder eine Fahrerkabine noch weitere Aufbauten notwendig, was eine kostengünstige Produktion dieser Traktoren ermöglichen sollte. Ebenso sind diese in der Lage sich auf dem Gelände autonom zu bewegen und können somit alle notwendigen Fahrmanöver durchführen. Die Traktoren sind Eigentum des Terminal-Betreibers und dürfen sich nur in für die autonome Fahrt ausgezeichneten Bereichen bewegen. In diesen darf sich aus Sicherheitsgründen kein Personal während der autonomen Fahrt aufhalten.

Die Aufgabe eines Terminaltraktors ist es nun, die Sattelauflieger und Anhänger anzukuppeln, deren zugehörige Kundenzugmaschine nicht in der Lage ist eine autonome Fahrt durchzuführen. Der Traktor begleitet somit den Sattelauflieger oder Anhänger über das gesamte Terminal und wird schließlich mit auf dem Zug verladen. Erst wenn er die endgültige Position am Zielbahnhof erreicht hat, wieder er wieder vom jeweiligen Sattelauflieger oder Anhänger getrennt. Somit entsteht ein fortwährender Austausch von Traktoren zwischen dem Start- und Zielterminal. Bekommt ein Traktor zwischenzeitig keine Aufgabe zugeteilt, so wird er am Terminal geparkt und

wartet dort auf weitere Anweisungen. In dieser Zeit kann er ebenso betankt und gewartet werden.

Dieses Umsatteln von Kundenzugmaschinen auf Traktoren stellt eine weitere Anforderung an eine mögliche autonome Nutzung: die Einführung von Mischbereichen der autonomen und manuellen Fahrt. Dabei handelt es sich um bestimmte Gebiete des Terminals, die abwechselnd manuell, von den ankommenden Zugmaschinen der Kunden, sowie autonom, von den Terminaltraktoren, befahren werden können. Diese Bereiche dienen rein zur Übergabe der Sattelauflieger oder Anhänger. Dazu werden am Terminal zwei Bereiche unterschieden: *Übergabebereich 1* und *Übergabebereich 2*

In Bereich 1 findet die Übergabe vom Kunden an das Terminal statt. Hier werden die Sattelauflieger und Anhänger von Traktoren entgegen genommen. Die Anzahl der möglichen Übergabebereiche ist dabei stark von den räumlichen Verhältnissen, aber auch vom Zugtakt abhängig. Dieser gibt vor, wie viele Lkw in welcher Zeit entgegen genommen werden müssen, um eine bestimmte Auslastung des Zuges zu gewährleisten. Je nach Terminal und Auslastung ergeben sich unterschiedliche Gestaltungen des Übergabebereiches 1. Für genauere Angaben hierzu sei auf die Diplomarbeit von Andreas Kern verwiesen, der sich dieser Thematik gewidmet hat.
In Bereich 2 findet die Übergabe vom Terminal an den Kunden statt, d. h. hier werden die Sattelauflieger und Anhänger bis zur Abholung abgestellt. Zunächst müssen dazu alle Traktoren vom Zug ab und den Übergabebereich befahren. Dort kuppeln sie den Sattelauflieger oder Anhänger ab und verlassen im Anschluss den Übergabebereich. Haben alle Traktoren den Bereich verlassen, so kann dieser für die autonome Fahrt gesperrt und für die manuelle Fahrt freigegeben werden. Somit können die Kunden von außen das Gelände befahren und ihre Sattelauflieger und Anhänger abholen.

Aus dieser Beschreibung lässt sich schon ein wesentlicher Unterschied zwischen den Bereichen 1 und 2 ableiten. Während in Bereich 1 nur wenige Umkuppelprozesse zur gleichen Zeit ablaufen müssen, muss in Bereich 2 gewährleistet sein, dass alle auf dem Zug ankommenden Sattelauflieger und Anhänger abgestellt werden können. Daher muss Bereich 2 mindestens genauso viele Parkplätze wie maximal mögliche Stellplätze auf dem Zug enthalten.

Da alle ankommenden Traktoren den Übergabebereich 2 sofort verlassen, muss auch für diese eine Parkfläche im rein autonomen Bereich bereitgestellt werden, auf der sie bis zur nächsten

Aufgabe abgestellt sind.

Doch nicht nur die Anzahl der Stellplätze auf dem Zug beeinflusst die Zahl der notwendigen Parkplätze in Bereich 2. Die bisherigen Überlegungen bezogen sich ausschließlich auf den Fall, dass ein Ende-zu-Ende Transport der Traktoren mit Transporteinheiten vom Start- zum Zielterminal ohne Zwischenstationen vorliegt. Dabei wird die Transporteinheit am Ziel-Terminal zur Abholung bereitgestellt. Der zugehörige Traktor ist somit für einen weiteren Transport frei. Dies entspricht auch der derzeitigen Verwendung der Rollenden Landstraße, in der ausschließlich eine bestimmte Strecke, im vorliegenden Beispiel Graz/Werndorf - Regensburg, befahren wird.

Für die Zukunft könnte aber auch ein Netzwerk von Terminals der Rollenden Landstraße denkbar sein, sodass z. B. eine Verbindung zwischen den beiden Terminals A und C nur über das Terminal B erfolgen kann. In diesem Fall müssten Traktoren im Terminal A mit Sattelaufliegern oder Anhängern für das Terminal C ebenso in Terminal B abgeladen und geparkt werden. Daher muss der Mischbereich 2 auch für diese möglichen Fälle Parkplätze bereithalten. In diesem Fall werden Traktor und Sattelauflieger nicht voneinander getrennt. Im Anschluss reiht sich der Traktor schließlich wieder zur Auffahrt auf den Zug zu seinem Zielterminal in die Wartespuren ein. Dies kann sowohl der nächste ausfahrende, als auch ein wesentlich späterer Zug sein. Daher kann es auch vorkommen, dass ein Traktor zusammen mit seinem Sattelauflieger oder Anhänger mehrere Stunden im Mischbereich auf den Anschlusszug wartet.

Damit wurden nun die wesentlichen Veränderungen für die Umstellung des Terminals von der manuellen auf die autonome Verwendung zusammengefasst. Hinzu kommt ebenso die Neugestaltung der Abläufe innerhalb des Terminals. Diese galt es in einem Entwurf so zu integrieren, dass die autonome Nutzung eine wirtschaftlich sinnvolle Alternative zur manuellen bietet.

3.3.2 Entwurf

Bevor nun der konkrete Entwurf betrachtet wird, sollen nochmals kurz die im vorherigen Abschnitt gestellten Anforderungen an das Terminal aufgelistet werden. Um eine autonome Fahrt zu ermöglichen, muss das Terminal folgende Punkte gewährleisten:

- Unterteilung in die Bereiche autonomer und manueller Fahrt
- 2 Mischbereiche:

1. Mischbereich: Abgabe der Sattelaufliegers/Anhänger durch die Kunden, maximal zwei Stellplätze
2. Mischbereich: Abholung der ankommenden Sattelauflieger/Anhänger, mindestens so viele Parkplätze wie Stellplätze auf dem Zug möglich sind

- Wartebereiche für die Traktoren im autonomen Bereich
- Wartebereiche für die Traktoren mit Sattelaufliegern/Anhängern im autonomen Bereich zur Auffahrt auf den Zug
- ausreichend Platz zur Gewährleistung aller notwendigen Fahrmanöver auf dem Gelände

Hinzu kommen weitere Anforderungen, die ganz speziell auf die räumlichen Gegebenheiten in Regensburg angepasst worden sind. Dies soll nun anhand der in Abbildung 3.10 gezeigten Karte erläutert werden.

3.3.3 Anfahrt, Annahme und Übergabe

Die Einfahrt zum Terminal befindet sich im autonomen Entwurf weiterhin an Position 1 in Abbildung 3.10. Von dort aus gelangt der Kunde mit dem Lkw direkt zur Annahme in Position 2. Hier wird er, analog zur aktuellen Verwendung, gewogen und vermessen sowie auf die Einhaltung der Richtlinien für den Transport auf der Schiene kontrolliert. Derzeit befindet sich an dieser Stelle nur eine einzige Annahmestelle mit einer Messeinrichtung und Waage. Um aber die Abläufe zu optimieren, wird diese Annahmestelle im vorliegenden Entwurf um eine zweite Messeinrichtung und Waage erweitert, sodass zwei Fahrzeuge parallel angenommen werden können. Die aktuellen Platzverhältnisse lassen dies zu.

War die Annahme am Terminal erfolgreich, so befährt der Fahrer auf Anweisung[9] den ersten Mischbereich in Position 3. Dort kuppelt er seinen Sattelauflieger oder Anhänger ab und verlässt daraufhin das Gelände des Terminals über die Betriebszufahrt. Diese wird in der aktuellen Verwendung nur in Ausnahmefällen[10] befahren, bietet aber die notwendige Infrastruktur und sollte ohne Änderungen genutzt werden können. Damit ist der Prozess der Annahme für den Kunden bereits vollzogen. Wird ein technisch modifizierter Lkw am Terminal übergeben, der eine

[9] z. B. durch eine Lichtzeichenanlage gesteuert
[10] z. B. wenn einem Lkw der Transport auf der Schiene untersagt wird

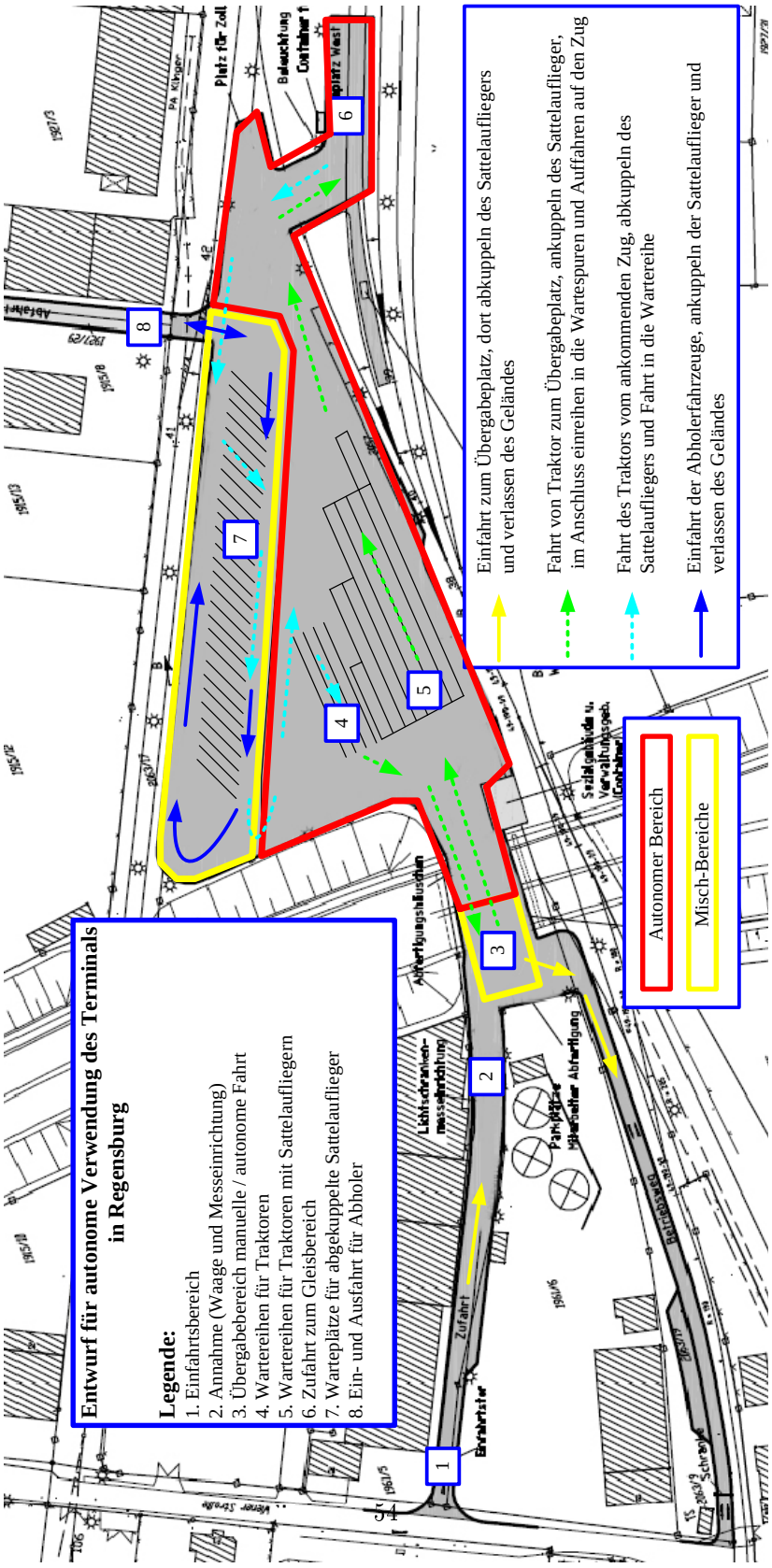


Abbildung 3.10: Entwurf der autonomen Nutzung des Terminals Regensburg

autonome Fahrt selbstständig durchführen kann, so parkt der Fahrer den Lkw lediglich im Übergabebereich. Ein Mitarbeiter meldet daraufhin dem Leitrechner, dass sich im Übergabebereich ein Lkw befindet, woraufhin der Leitstand alle weiteren autonomen Abläufe einleitet.

3.3.4 Fahrweg des Traktors auf den Zug

Im nächsten Schritt beauftragt der Leitrechner einen Terminaltraktor damit, den im Übergabebereich abgestellten Sattelauflieger oder Anhänger aufzunehmen. Wie bereits in den Anforderungen an die autonome Nutzung erwähnt, warten alle Traktoren auf speziell für sie vorgesehenen Stellplätzen im autonomen Bereich. Dabei handelt es sich ebenfalls um sogenannte Parkspuren, in denen die Traktoren dicht hintereinander abgestellt werden. Dies stellt aufgrund der beengten Platzverhältnisse in Regensburg die effizienteste Ausnutzung dar. Die Spuren werden von den Traktoren ausschließlich rückwärts befahren, da somit weitere notwendige Fahrmanöver auf dem Weg zum Übergabebereich entfallen. Innerhalb der Spuren kann immer nur der erste Traktor[11] mit einer Fahraufgabe beauftragt werden. Geschieht dies, so rücken alle weiteren Traktoren um einen Stellplatz auf, sodass sich stets weitere freie Traktoren von hinten in die Parkspuren einreihen können. Diese befinden sich in Position 4 der Karte in Abbildung 3.10. Insgesamt wurden im vorliegenden Entwurf drei Spuren eingeplant.
Wird nun einer der drei vordersten Traktoren mit dem Abholen eines Sattelaufliegers oder Anhängers im Übergabebereich beauftragt, so fährt dieser aus der Parkspur zum Übergabebereich. Diese Fahrt findet, wie bereits erwähnt, rückwärts statt, da der Traktor so den Sattelauflieger oder Anhänger sofort aufnehmen kann. Dazu fährt der Traktor mit seiner Kupplung die Position der Kupplung der Transporteinheit an. Die genaue Positionierung muss dabei vom Leitrechner vorgenommen werden. Alle weiteren notwendigen Abläufe beim Ankuppeln, z. B. das Herstellen der Verbindungen von Vorrats- und Bremsdruckleitung sowie des Stromanschlusses, müssen im Anschluss manuell durch einen Mitarbeiter vorgenommen werden. Derzeit befinden sich Kupplungssysteme in der Entwicklung, die diese Vorgänge vollautomatisch ermöglichen sollen, sodass keine weiteren Maßnahmen durch Angestellte notwendig sind.
Hat der Traktor den Sattelauflieger oder Anhänger korrekt angekuppelt, so verlässt er mit diesem den Übergabebereich und befährt seine Parkposition in der Wartespur, die er durch den Leitrechner mitgeteilt bekommt. Die Wartespuren befinden sich in Position 5. Hier kommt das gleiche Platz sparende Prinzip wie bei den Traktor-Parkspuren zum Einsatz. Alle Traktoren mit Sattelaufliegern oder Anhängern werden in Reihen direkt hintereinandergestellt. Dabei entschei-

[11] alle Weiteren sind durch die Aufreihung hintereinander blockiert

det der Leitrechner, welchen Platz der Traktor in welcher Spur einnehmen soll[12].
Wird dem Leitrechner signalisiert, dass der Zug zum Beladen bereitsteht, beauftragt er die einzelnen Traktoren in den Wartespuren diese zu verlassen und den Zug über den Gleisbereich zu befahren. Dabei liegt die Verantwortung der richtigen Reihenfolge bei Befahrung des Zuges erneut beim Leitrechner. Auch stellt das Auffahren auf den Zug besondere Anforderungen an das notwendige Positionierungssystem, da die Abweichungstoleranzen hier viel geringer sein müssen als auf dem freien Gelände. Dazu würde sich die Verwendung eines zusätzlichen Systems, z. B. die direkte Messung der Abstände des Lkw zu den Seiten der Niederflurwaggons, anbieten. Ein solches System würde dem Leitrechner weitere notwendige und gleichzeitig präzisere Daten liefern, die zur punktgenauen Fahrt auf den Niederflurwaggons unabdingbar sind.
Nachdem der Traktor seine Zielposition auf dem Zug erreicht hat, wird er von einem Mitarbeiter gesichert und deaktiviert, sodass ein Transport zum Zielbahnhof stattfinden kann. Ist dies für alle Traktoren geschehen, so steht der Zug zur Ausfahrt bereit.

3.3.5 Fahrweg des Traktors vom Zug

Nach der Einfahrt eines beladenen Zuges werden die Traktoren zunächst entsichert und zur autonomen Fahrt vorbereitet. Ist dies geschehen, aktiviert ein Mitarbeiter alle Traktoren, die sich mit ihrer Aktivierung automatisch beim Leitrechner anmelden. Dieser verteilt im Anschluss die notwendigen Fahraufgaben, um den Zug komplett zu entladen. Dazu beauftragt er die Traktoren vom Gleisbereich in den Übergabebereich 2 in Position 7 zu fahren. In diesem Bereich bekommt jeder Traktor einen festen Stellplatz zugewiesen. Diese Stellplätze werden zunächst nacheinander vergeben. Lediglich die Traktoren, die Sattelauflieger oder Anhänger für einen weiteren Transport auf der Schiene[13] angekuppelt haben, werden gesondert abgestellt und warten dort auf ihren Anschlusszug. Alle weiteren Traktoren kuppeln nach Erreichen ihres Stellplatzes die Sattelauflieger oder Anhänger dort ab und begeben sich zu den Traktor-Parkspuren in Position 4. Haben alle Traktoren den Mischbereich verlassen, so wird dieser für die autonome Fahrt gesperrt. Die Sattelauflieger oder Anhänger stehen nun zur Abholung durch den Kunden im Übergabebereich bereit.

[12] z. B. können die Fahrzeuge je nach Länge bestimmten Wartespuren zugewiesen werden
[13] vgl. Abschnitt 3.3.1 auf Seite 49

3.3.6 Abholung durch den Kunden

Nach der Sperrung des Mischbereiches für die autonome Fahrt, können die Kunden mit ihren Zugmaschinen das Terminal über die zweite Ein- und Ausfahrt in Position 8 befahren. Dabei ist es wichtig, dass innerhalb des Mischbereiches die Fahrtrichtung einer Einbahnstraße[14] eingehalten wird. Jeder Kunde begibt sich zu seinem Sattelauflieger oder Anhänger, kuppelt diesen an und kann das Terminal verlassen. Die Ein- und Ausfahrt ist für einen Gegenverkehrbetrieb zu schmal gestaltet, daher sollte das Ein- und Ausfahren mit einer Lichtzeichenanlage geregelt werden.

3.3.7 Lageplan, CAD-Entwurf und Grundlagen der Visualisierung

Während die beiden Karten der Abbildungen 3.1 auf Seite 42 und 3.10 auf Seite 54 lediglich Skizzen der jeweiligen Verwendung zeigen, galt es nun die angedachte autonome Variante in einem konkreten, detaillierten und maßstabsgetreuen Entwurf festzuhalten. Die Basis für diesen Entwurf bildete wiederum der Lageplan des Vermessungsamtes in Regensburg. Dieser Lageplan konnte mithilfe des Programmes AutoCad2006 bearbeitet und verschiedene mögliche Verwendungen entworfen werden. Dabei galt es die beiden folgenden Punkte einzuhalten und zu verwirklichen:

1. Einhaltung des Planungs-Maßstabes
2. Garantie der Befahrbarkeit der eingeplanten Bereiche[15]

Während Punkt 1 durch die Verwendung von AutoCad und maßstabsgetreuem Zeichnen leicht zu realisieren war, setzte die Gewährleistung von Punkt 2 Hintergrundwissen über den Platzbedarf von Fahrzeugen bei den verschiedensten Fahrmanövern voraus. Um sicher zu stellen, dass jedes nach StVZO zugelassene Fahrzeug auch auf dem Terminal jede Fahraufgabe ausführen kann, wurden Bauvorschriften für den Straßenbau als Orientierung herangezogen. Diesen wurden die notwendigen Abmessungen für Parkplätze, Zufahrten, Freiräume bei Kurvenfahrten etc. entnommen und im Entwurf umgesetzt. Bei den Abmessungen dienten die Angaben aus Quelle [Neu] als Orientierung. Der entstandene Entwurf mit den genauen Abmessungen findet sich im Anhang in Abschnitt 7.2 auf Seite 194.

Aus der nun vorhandenen 2D-Modellierung der autonomen Verwendung des Terminals folgte die Gestaltung einer 3D-Umgebung für die in Abschnitt 1.6 vorgestellte Visualisierung. Diese

[14] entsprechend der Pfeilrichtungen in Abbildung 3.10 auf Seite 54

[15] alle Bereiche (Parkspuren, Wartespuren etc.) müssen jederzeit von realen Fahrzeugen befahren werden können

verwendet zur dreidimensionalen Darstellung Karten der Open-Source-Software Blender, die nach einigen Konvertierungen problemlos in die Visualisierung geladen werden können. Daher wurde die Modellierung des Geländes mithilfe der Blender-Software vorgenommen. Diese basierte auf der Grundlage der Auto-CAD-Zeichnung, um die Einhaltung des Maßstabes zu gewährleisten. Es wurde bei der Modellierung des Geländes versucht, so nah wie möglich die reale Umgebung des Terminals in Regensburg abzubilden, da ein hoher Wiedererkennungswert erlangt werden sollte.

Kapitel 4

Der Leitstand

In diesem Kapitel soll zunächst die Grundlage der Projekte EZsped und EZrola betrachtet werden: der *Allgemeine Leitstand*. Die Zielsetzung ist, den Aufbau und die Funktionsweise kennen zu lernen und einen Überblick über die Leistungsmerkmale des Leitstandes zu gewinnen[1]. An dieser Stelle können nicht alle Gesichtspunkte und Merkmale detailliert aufgezeigt werden. Stattdessen wird der Fokus auf das Gesamtkonzept und dessen Schnittstellen nach außen hin gerichtet, da diese bei jeglicher Spezialisierung angesprochen werden müssen. Im Anschluss wird die Spezialisierung des Leitstandes aus dem Projekt EZsped[2] betrachtet und die Anforderungen, sowie die Umgebung des innovativen Logistikhofes aufgezeigt. Die Spezialisierung des Leitstandes von EZrola[3] rückt daraufhin in den Fokus. Hier werden alle Anforderungen an den Leitstand in dieser Spezialisierung erläutert und die Basis für die in Kapitel 5 vorgestellten Modifikationen gelegt. Zum Abschluss des Kapitels werden drei zentrale Elemente des Leitstandes herausgegriffen und analysiert[4], da sie ebenfalls eine Grundlage für das folgende Kapitel bilden.

4.1 Allgemeiner Leitstand

In diesem Abschnitt wird zunächst eine Definition des Allgemeinen Leitstandes gegeben, ehe die an diesen gestellten Anforderungen in einer zusammenhängenden Übersicht aufgezeigt und

[1] vgl. Abschnitt 4.1
[2] vgl. Abschnitt 4.2
[3] vgl. Abschnitt 4.3
[4] vgl. Abschnitt 4.4

die dazu notwendigen Datengrundlagen erläutert werden. Im Anschluss wird der genaue Aufbau betrachtet, die einzelnen Aufgaben der zugehörigen Module kurz vorgestellt und der Allgemeine Leitstand in seine mögliche Umgebung eingegliedert. Die dazu notwendigen Schnittstellen zur Umgebung werden dazu nochmals gesondert betrachtet.

4.1.1 Definition

An dieser Stelle soll die Frage beantwortet werden, wie der Allgemeine Leitstand definiert ist und welche Leistungsmerkmale diesen kennzeichnen. Umgangssprachlich müsste man eine Definition wie folgt formulieren:

„Der Allgemeine Leitstand für autonomes Fahrens ist die Menge an notwendigen Funktionalitäten, welche unabhängig von der Anwendung immer gewährleistet sein müssen, um ein autonomes Fahren überhaupt zu ermöglichen. Er bildet den Kern aller anwendungsfallbezogenen Spezialisierungen, die durch das Aufsetzen von spezifischen Erweiterungen diesen zu einem vollständigen System ergänzen.“

4.1.2 Allgemeine Anforderungen und Datengrundlagen

In diesem Abschnitt werden die Anforderungen an den Allgemeinen Leitstand, sowie dessen Datengrundlagen, im Zusammenhang aufgezeigt. Dabei soll ein Überblick entstehen, durch den die einzelnen Leistungsmerkmale und Module begründet und miteinander verbunden werden.

Um der Definition aus Abschnitt 4.1.1 gerecht zu werden, benötigt der Allgemeine Leitstand folgende elementare Datengrundlagen:

- Die Topologie des Geländes sowie die möglichen Fahrwege.
- Die auf dem Gelände zu erfüllenden Aufgaben.
- Die Kenntnis über die verwendeten Fahrzeuge.
- Die Interaktion mit der Umwelt über die dynamischen Abläufe auf dem Gelände.

Aus diesem Grund muss der Allgemeine Leitstand in der Lage sein, Daten sowohl aus Datenbanken einzulesen, als auch zur Laufzeit von außen empfangen zu können. Daher bedarf es einer *Kommunikationsschnittstelle* zu Modulen außerhalb des Leitstandes, um fortwährend Daten austauschen und auf diese reagieren zu können. Doch nicht nur externe Module, sondern auch die Fahrzeuge und interne Strukturen, müssen über definierte Schnittstellen zum Datenaustausch angesprochen werden können.

Um nun ein sich auf dem Gelände befindliches Fahrzeug einer autonomen Fahrt zu unterziehen, muss dem Leitstand dies über die externe Kommunikationsschnittstelle mitgeteilt werden. Anhand der zugrunde liegenden Informationen über die zu absolvierenden Aufgaben auf dem Gelände, zerlegt die *Auftragsverwaltung (OrderAdministration)* daraufhin den erhaltenen Auftrag in seine einzelnen Bestandteile. Dabei ist diese Zerlegung durch die Anwendung geprägt und kann von Fall zu Fall stark variieren. Ziel jedoch ist es, die notwendigen elementaren Fahraufgaben zu extrahieren und zur Einplanung weiterzuleiten.

Dazu werden diese im Anschluss an die *Terminplanung (Scheduler)* weitergereicht, die dafür zuständig ist, die Ausführung aller Aufgaben einzuleiten und zu überwachen. Doch bevor eine Aufgabe ausgeführt werden kann, müssen alle Teilaufgaben eingeplant und Routen für die Fahraufgaben bestimmt werden. Das Einplanen einer Fahraufgabe benötigt somit weitere Komponenten: die *Routenplanung (RoutePlanning)* und die *Detailplanung (VCPlanning)*. Zunächst gilt es sich anhand der möglichen Fahrwege auf dem Gelände die Route vom Start- zum Zielpunkt zu suchen, und diese je nach Fahrzeugtyp zu berechnen. Dies kann in mehreren Schritten geschehen, sodass eine grobe Route bestimmt und diese bis hin zu einer vom Fahrzeug gefahrenen Trajektorie verfeinert wird. Ist dies geschehen, so muss die Route in den für das Gelände vorliegenden Zeitplan der Detailplanung eingefügt werden. Dieser Zeitplan (auch *Raum-Zeit-Plan* genannt) ist das zentrale Planungselement der Detailplanung, da dieser alle auf dem Gelände ablaufenden Aufgaben enthält und den Aufenthaltsort aller Fahrzeuge zu jedem Zeitpunkt kennt. Die neue Route wird hier kollisionsfrei eingeplant, sodass Fahrzeuge sich auf dem Gelände nicht gegenseitig behindern können[5]. Im Anschluss wird die erfolgreiche Verplanung der Terminplanung mitgeteilt, damit diese die Teilaufgaben starten und beenden kann. Natürlich laufen die Abläufe in der Realität nie exakt so ab wie geplant, daher muss fortwährend ein Soll/Ist-Abgleich geschehen und bei Abweichungen entsprechend eingegriffen werden.

[5] z. B. in Engstellen

Wird die Fahraufgabe gestartet, so müssen die notwendigen Instruktionen, wie z. B. die abzufahrende Trajektorie, an das Fahrzeug gesendet werden. Dies übernimmt das *Fahrzeugmanagement (VehicleManagement)*. Es verfügt über alle notwendigen Daten, der sich auf dem Gelände bewegenden Fahrzeuge, und kann diese über eine eigene Kommunikationsschnittstelle ansprechen. Um auf Ausnahmen und Fehler auf dem Gelände zu reagieren und Unfälle zu vermeiden, bedarf es des Weiteren einer Sicherheitsschicht, der *Fahrzeugkontrolle (VehicleControl)*. Diese überwacht alle auf dem Gelände stattfindenden Fahraufgaben und kann bei drohenden Kollisionen in den Ablauf eingreifen.

Dies waren die elementaren Funnktionalitäten des Allgemeinen Leitstandes in einem groben Überblick zusammengefasst. Bevor nun auf den genauen Aufbau eingegangen wird, sollen die wesentlichen genannten Elemente dieses Abschnittes in Bezug auf den Kern des Systems nochmals festgehalten werden. Die Elemente des Kerns sind somit:

- Die Kommunikationsschnittstellen zum externen und internen Datenaustausch.
- Die Fähigkeit statische und dynamische Daten einlesen und verarbeiten zu können.
- Die Verarbeitung von elementaren Aufgaben in der Auftragsverwaltung (OrderAdministration).
- Die Routenplanung (RoutePlanning) der abzufahrenden Fahrwege.
- Die Terminplanung (Scheduler) sowie deren Detailplanung (VCPlanning).
- Das Fahrzeugmanagement (VehicleManagement) zur Ansteuerung der Fahrzeuge.
- Die Fahrzeugkontrolle (VehicleControl) zur Verhinderung von Kollisionen.

Im Folgenden wird das Konzept des Leitstandes betrachtet und auf dessen Grundlagen eingegangen werden.

4.1.3 Aufbau des Leitstandes

Abbildung 4.1 zeigt die Aufteilung der im vorherigen Abschnitt angesprochenen Module auf zwei Leitstand-Prozesse, die *Planungskomponente* und die *Überwachungskomponente*. Sie bilden, parallel ausgeführt, den Kern des Leitstandes. Um jegliche Kommunikation zwischen den Modulen

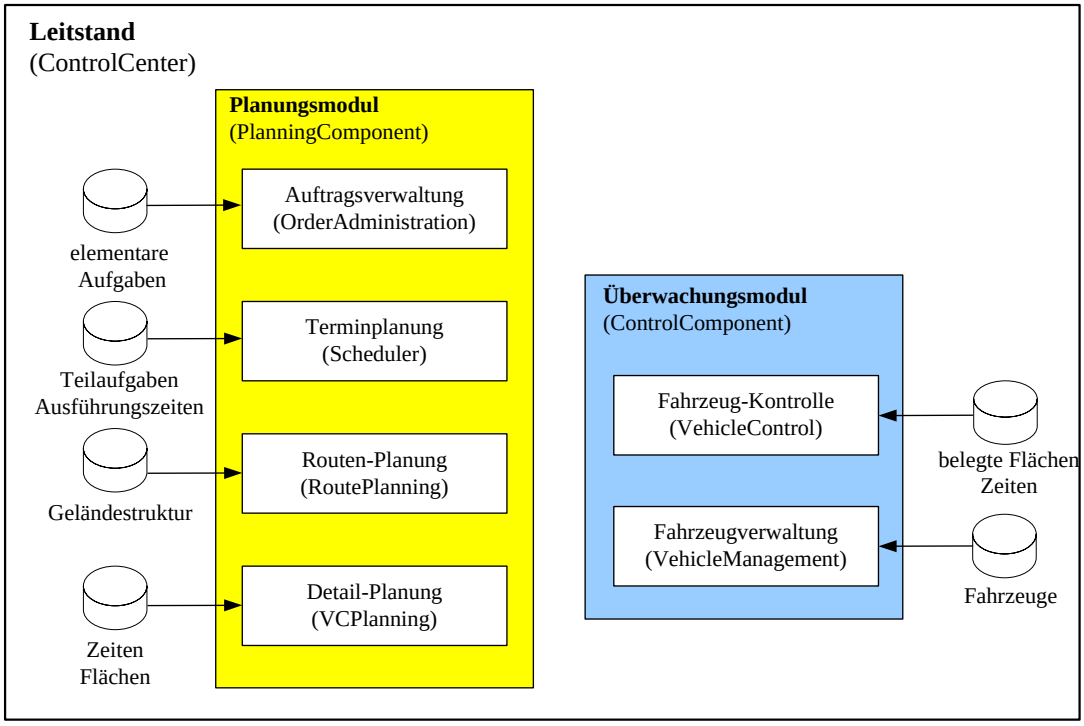


Abbildung 4.1: Interner Aufbau des Allgemeinen Leitstandes

und den beiden Prozessen zu gewährleisten, melden diese sich bei dem sogenannten Internen Ereignisverteiler an. Die Funktionsweise des Ereignisverteilers und dessen Rolle im Rahmen der EZauto-Bibliothek wird in Abschnitt 4.1.3.1 genauer erklärt. Durch die Registrierung an einem solchen Verteiler bekommt das jeweilige Modul nur die für sich relevanten internen Nachrichten zugestellt. Die Schnittstellen zur Nachrichtenübermittlung sind dabei so allgemein wie möglich gehalten und sollen keine für ein Modul spezifische Datenstrukturen enthalten. Nur so können einzelne Module problemlos ausgetauscht und verändert werden, ohne dass es weit reichende Konsequenzen mit sich führt.

Zu beachten ist dabei, dass Abbildung 4.1 bisher nur den Kern und die darin enthaltenen Module mit deren Datengrundlagen des Leitstandes zeigt, diesen aber noch nicht in seiner Umgebung eingliedert. Nur durch weitere (externe) Module ist der Leitstand in der Lage mit den Fahrzeugen zu kommunizieren oder Aufträge von außen entgegenzunehmen. Daher wird dieser in Abschnitt 4.1.4 sukzessive erweitert. Doch zunächst sollen die internen Module und deren Datengrundlagen und Funktionsweisen genauer betrachtet werden.

4.1.3.1 Kommunikation über Ereignisverteiler

Die Kommunikation zwischen allen Prozessen sowie zwischen Submodulen geschieht ereignisbasiert. Zu diesem Zweck melden sich alle Prozesse über ein TCP/IP-basiertes Netzwerk bei einem oder mehreren Ereignisverteilern an. Diese fungieren ähnlich wie ein Postverteilersystem. Sie nehmen Ereignisse von Prozessen oder Modulen an und reichen diese an den (oder die) Empfänger weiter. Den Kommunikationspartnern bleibt jedoch, im Gegensatz zum Postverteilersystem, komplett verborgen, mit wem sie kommunizieren. Auf diese Weise sind Module leichter anpassbar oder sogar austauschbar, da sie im übertragenen Sinn nicht an die Stelle des alten Moduls treten müssen, sondern nur für die gleichen Nachrichten registriert werden und die gleichen Nachrichten verschicken müssen, wie das alte Modul. Das Konzept der Ereignisverteiler entstammt der Softwarebibliothek EZauto.

Ein Ereignis hat immer einen bestimmten Typ, der als Entscheidungsgrundlage für die Weiterleitung dient. Dieser muss daher eindeutig sein. Darüber hinaus können in einem Ereignis noch weitere Attribute übertragen werden. Einige Standardtypen wie Integer oder Strings können ohne Weiteres hinzugefügt werden. Mit etwas Implementierungsaufwand kann jedoch grundsätzlich jedes beliebige Objekt mit einem Ereignis verschickt werden. Der Empfänger kann die Attribute dann wieder aus dem Ereignis auslesen. Um Ereignisse empfangen zu können, muss eine Klasse eine bestimmte Schnittstelle aus EZauto (*sEreignisEmpfaenger*) implementieren und entsprechende Methoden für den Empfang und das Verarbeiten von Ereignissen besitzen. Der Empfänger muss beim zuständigen Ereignisverteiler angemeldet werden und sich dort für die Nachrichten registrieren, die er empfangen will.

Für die Kommunikation über Prozessgrenzen hinweg wird ein Client-Server-Modell genutzt. Jeder Prozess muss dann über einen Client verfügen, der mit dem zuständigen Server über TCP/IP verbunden wird. Der Client teilt dem Server bei der Anmeldung mit, für welche Ereignisse Registrierungen bei ihm vorliegen. Der Server weiß somit, welche Ereignisse übertragen werden müssen und kann so unnötigen Netzwerkverkehr vermeiden. Der Client muss den Server jedoch über zusätzliche Registrierungen informieren, bzw. ihm mitteilen, wenn bestimmte Ereignisse nicht mehr gebraucht werden. Ebenso muss der Server den Client über die benötigten Ereignisse informieren. In Abbildung 4.2[6] ist die Kommunikation über Prozessgrenzen hinweg

[6] vgl. [LuT], Seite 74

dargestellt.

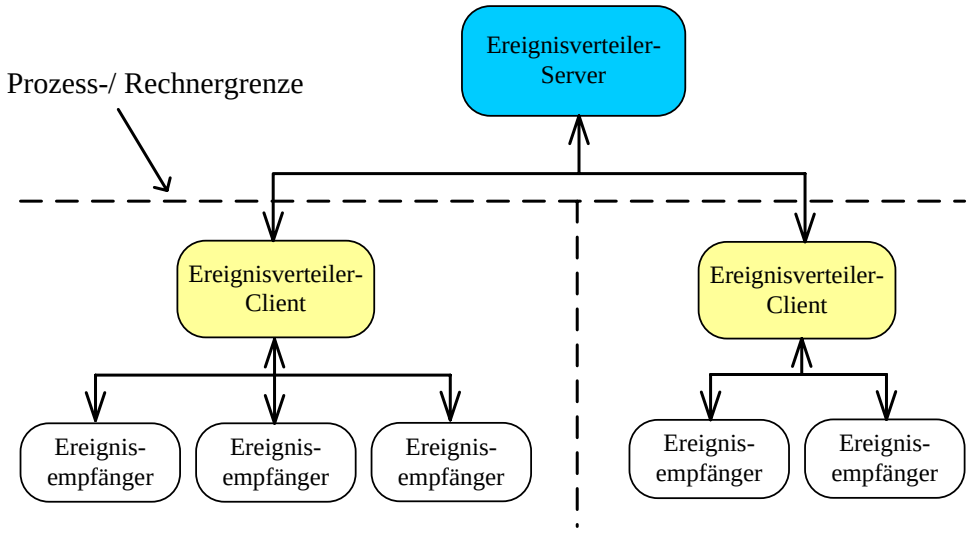


Abbildung 4.2: Kommunikation über Prozessgrenzen hinweg

Die Kommunikation mit den Fahrzeugen geschieht über die *VehicleCommunication*. Diese ist ausschließlich für die Koordination dieser Kommunikation konzipiert. Sie leitet eingehende Nachrichten an das jeweilige Fahrzeug weiter, bzw. in umgekehrter Richtung von den Fahrzeugen an den Leitstand. Diese Kommunikation wird von der restlichen Kommunikation getrennt behandelt und daher über einen eigenen Ereignisverteilerserver abgewickelt. Eine direkte Kommunikation mit den Fahrzeugen ist für Leitstandsmodule nicht möglich, dieser Aspekt bleibt vor ihnen jedoch verborgen.

Nach [LuT][7] wurde diese Trennung vorgenommen, damit die Fahrzeugkommunikation so geordnet und koordiniert werden kann, dass die in einem Funknetzwerk möglichen Kollisionen von Daten möglichst ausgeschlossen werden können. Dies ist wichtig, um den Zeitbedingungen eines Echtzeitsystems gerecht werden zu können. Denkbar wäre dann eine Kommunikation auf Grundlage von TDMA[8], bei dem, im synchronen Modus[9], jedes Fahrzeug zyklisch feste Zeitfenster zum Senden zugewiesen bekommt. Da die Kommunikation momentan nur mit simulierten Fahrzeugen

[7] vgl. Abschnitt 3.3.2

[8] *Time Division Multiple Access*

[9] möglich ist der synchrone und der asynchrone Modus. Beim synchronen Modus wird jedem Kommunikationspartner ein fester Zeitschlitz zugeteilt. Wird dieser nicht genutzt, so bleibt das Übertragungsmedium zu diesem Zeitpunkt ungenutzt. Im asynchronen Modus wird die Zeit bedarfsgerecht zugeteilt

stattfindet, wurde eine solche Lösung bislang nicht implementiert.

Ähnlich zur Kommunikation mit Fahrzeugen wird auch die Kommunikation mit externen Modulen getrennt behandelt. Zu diesem Zweck wurde die *ExternalCommunication* als Teil des Leitstandes implementiert. Auch hier ist für leitstandsinterne Module keine direkte Kommunikation möglich. Dadurch ist die Unabhängigkeit von der Kommunikationsmethode der externen Module, die mit anderen Übertragungstechniken oder Protokollen arbeiten könnten, gewährleistet. In jedem Fall müsste ausschließlich die *ExternalCommunication* angepasst werden.

Externe Module bekommen bei ihrer Anmeldung an den externen Ereignisverteiler von diesem eine eindeutige ID zugewiesen. Auf diese Weise kann das Kommunikationsmodul speichern, welches Modul für welche Ereignisse des Leitstandes registriert ist. Die Nachrichten, welche nach außen weitergeleitet werden, können dort eine ganz andere Gestalt und einen anderen Umfang haben als leitstandsintern. Dies spiegelt sich auch in der Namensgebung der Ereignisse wieder. Interne Nachrichten tragen meist ein „*I*“ am Anfang des Namens, während externe Ereignisse an dieser Stelle ein „*E*“ tragen. Bei der Überarbeitung der Ereignisse führt die *ExternalCommunication* deshalb auch diese Namensänderung durch. Eine beispielhafte Kommunikation ist in Abbildung 4.3 auf der nächsten Seite[10] zu sehen. Hier erfragt die Auftragsverwaltung den Auftrag für ein Fahrzeug beim externen Modul zur Auftragsannahme. Dort ist zu erkennen, dass die Auftragsverwaltung dem Namen zufolge eine interne Nachricht (mit beginnendem „*I*“) verschickt. Bei der Antwort, die sie erwartet, handelt es sich ebenfalls um eine interne Nachricht. Den leitstandinternen Modulen bleibt also komplett verborgen, ob sie mit externen oder internen Modulen kommunizieren.

Um Ereignisse überarbeiten zu können, benötigt die externe Kommunikation Zuordnungstabellen, welche ihr Auskunft darüber geben können, welche interne Nachricht zu welcher externen Nachricht gehört und umgekehrt. Will sich ein externes Modul für eine bestimmte Nachricht anmelden, so kann die externe Kommunikation die entsprechende interne Nachricht in der Tabelle finden und sich für diese Nachricht beim internen Ereignisverteiler registrieren. Empfängt sie dann diese interne Nachricht, so kann diese modifiziert und an das externe Modul weitergereicht werden. Nähere Informationen zur externen Kommunikation sind in [LuT] in Abschnitt 3.4.1 zu finden.

[10] vgl. [LuT], Seite 66

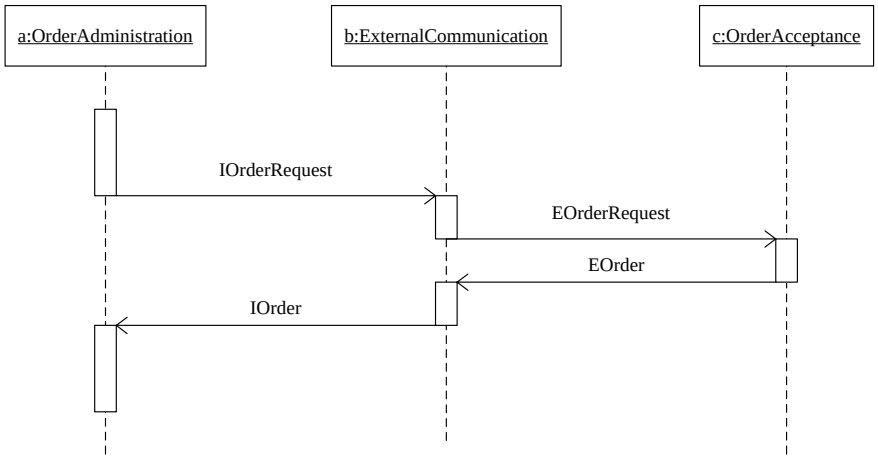


Abbildung 4.3: Beispiel für externe Kommunikation

4.1.3.2 Auftragsverwaltung

Die Aufgabe der Auftragsverwaltung ist, dem Namen entsprechend, die Verwaltung und Weiterverarbeitung aller Aufträge im System. Unter einem Auftrag wird die Gesamtheit der Aufgaben verstanden, die ein bestimmtes Fahrzeug auf dem Gelände erledigen soll. Jede Aufgabe des Auftrags verweist in die statische Aufgabenstruktur[11]. Die Weiterverarbeitung überführt die Aufträge in eine Struktur, die von der Terminplanung verstanden und von dieser weiter geplant werden kann. Die Datengrundlage dieses Moduls erstreckt sich daher auch nur über Aufgaben und Teilaufgaben der einzelnen Fahrzeuge, sowie einige wichtige Eigenschaften der Fahrzeuge, um deren Aufträge korrekt behandeln zu können.

Außerdem ist vorgesehen, dass Aufträge Regeln enthalten können. Mit diesen Regeln können komplexe Zusammenhänge zwischen verschiedenen Aufgaben dargestellt und deren Reihenfolge beeinflusst werden. Es kann festgelegt werden, dass bestimmte Aktivitäten nur zu bestimmten Zeitpunkten durchgeführt werden dürfen[12].

Bevor die Aufträge verarbeitet werden können, müssen sie zuerst beschafft werden. Das gehört ebenfalls zum Aufgabengebiet der Auftragsverwaltung. Sie kommuniziert daher mit der

[11] siehe Abschnitt 5.4

[12] Die Verwendung von Regeln ist konzeptionell vorgesehen, sie werden jedoch weder bei EZsped noch bei EZrola in der Implementation berücksichtigt

Auftragsannahme, um die nötigen Informationen zu bekommen. Nähere Informationen zur Auftragsverwaltung sind, soweit sie diese Arbeit betreffen, in Abschnitt 5.3 auf Seite 116 zu finden. Ansonsten sei auf [LuT], Abschnitt 4.4, verwiesen.

4.1.3.3 Terminplanung

Die Terminplanung erfüllt zwei wesentliche Aufgaben: das Einplanen sowie das Starten und Beenden eingeplanter Teilaufgaben. Bekommt die Terminplanung über die Schnittstelle zur Auftragsverwaltung eine Anfrage zur Einplanung verschiedener Aufgaben, so zerlegt sie diese in ihre elementaren Teilaufgaben[13] und erstellt für jedes Fahrzeug einen festen Plan. Dieser beinhaltet alle notwendigen Informationen, um die jeweiligen Aufgaben ausführen zu können. Handelt es sich nun um eine Fahraufgabe, so muss zunächst die Routenplanung zurate gezogen und die Route vom Start- zum Zielpunkt berechnet werden. Ist dies geschehen, so muss die Fahraufgabe mit dem Raum-Zeit-Plan der Detailplanung abgeglichen und in diesen eingetragen werden. Erst nach erfolgreicher Eintragung meldet die Detailplanung die möglichen Ausführungszeiten der Terminplanung. Diese übernimmt die Ausführungszeiten nach einer Prüfung und Bestätigung an andere Module in den Plan der jeweiligen Fahrzeuge. Für die Einplanung und die rechtzeitige Ausführung der einzelnen Teilaufgaben innerhalb der Terminplanung ist der Scheduler verantwortlich. Dieser überwacht alle Ausführungszeiten periodisch, startet anstehende Teilaufgaben und informiert alle weiteren Module über die anlaufenden Aufgaben. Die genaue Funktionsweise des Schedulers wird in Abschnitt 4.4.2 genauer erläutert.

4.1.3.4 Routenplanung

Die Datengrundlage der Routenplanung sind die auf dem Gelände verlegten Leitlinien, also eine Abfolge von Punkten der möglichen Fahrwege. Jede Leitlinie kann vier Komponenten enthalten, welche die Fahrtrichtung[14] und die Fahrweise[15] bestimmen. Alle Komponenten stehen in einer Vorgänger-Nachfolger-Relation zueinander, sodass sie zusammen über das gesamte Gelände einen Graphen aufspannen. Die Aufgabe der Routenplanung ist es nun, die Route für einen Fahrweg vom Start- zum Zielpunkt auf dem Gelände zu finden. Sie bekommt dies in einer Anfrage über die Schnittstelle zur Terminplanung mitgeteilt. Dabei sind lediglich die Koordinaten des Start-

[13] entsprechend den Teilaufgaben des Aufgabenbaumes der Auftragsverwaltung
[14] in Richtung der Leitlinie oder dagegen
[15] vorwärts oder rückwärts

und des Zielpunktes, die Ausrichtungen des Fahrzeuges in diesen Punkten, sowie der mögliche Referenzpunkt am Fahrzeug, mit welchem der Zielpunkt angefahren werden soll, Teil der Nachricht. Die erste Aufgabe der Routenplanung ist es, den Koordinaten von Start- und Zielpunkt die nächstgelegenen Leitlinien zuzuordnen. Dies geschieht über die unmittelbare Suche nach Leitlinien im Umkreis der Koordinaten. Dabei wird der Suchradius so lange vergrößert, bis eine Leitlinie gefunden wurde. Neben der Leitlinie müssen auch die Komponenten der Leitlinie passend ausgewählt werden. Dabei kann die Auswahl unter den vier Komponenten oftmals nicht eindeutig sein[16], sodass mehrere Möglichkeiten zur Auswahl bleiben. Diese werden im Anschluss dem Suchalgorithmus A* bereitgestellt, der den kürzesten Weg zwischen den beiden Komponenten bestimmt. An dieser Stelle wird nicht weiter auf die genaue Funktionsweise von A* eingegangen. Es sei statt dessen auf die Erläuterungen von [LuT] in Abschnitt 4.6.3 in Anlehnung an [Rus] S.97 - S.101 verwiesen.

Da die von A* gefundene Route sich bisher nur an den Leitlinien orientiert, handelt es sich noch nicht um die für ein Fahrzeug fahrbare Trajektorie. Diese muss erst noch in der Detailplanung für das jeweilige Fahrzeug berechnet und eingeplant werden. Daher übergibt die Routenplanung die grobe Route, die lediglich eine Abfolge von Punkten im Raum beinhaltet, an die Detailplanung. Sollte die gewählte Route aus verschiedenen Gründen nicht einplanbar sein, so kann die Detailplanung bei der Routenplanung weitere Routen anfragen. Genauere Informationen zur Routenplanung und Berechnung, sowie bestehenden und behobenen Problemen, werden in Abschnitt 4.4.3 erläutert.

4.1.3.5 Detailplanung

Erhält die Detailplanung über die Schnittstelle zur Routenplanung die für eine Fahraufgabe gefundene Route, so muss sie drei wesentliche Aufgaben erfüllen:

- Die Trakjektorienberechnung
- Die Korridorberechnung
- Die Raum-Zeit-Planung

[16] z. B., wenn ein Fahrzeug parallel zu einer Leitlinie steht, wäre es möglich diese Linie vorwärts in Linienzugrichtung oder rückwärts gegen die Linienzugrichtung zu befahren

Die Trajektorienberechnung passt die grobe Route an die jeweiligen Fahrzeuge an und erstellt daraus die fahrbaren Trajektorien. Beeinflussende Faktoren sind hierbei die Ausrichtung des Fahrzeuges im Zielpunkt und der Referenzpunkt am Fahrzeug, mit dem der Zielpunkt angefahren werden soll. Die genaue Berechnung der Trajektorie geschieht dabei in mehreren Schritten und basiert auf den Funktionen der EZauto-Bibliothek.

Aus der errechneten Trajektorie wird im Anschluss der Korridor der Fahraufgaben berechnet. Der Korridor ist der Bereich, den das Fahrzeug beim Abfahren der Trajektorie niemals verlassen darf. Bei der Berechnung des Korridors spielen die Abmaße des Fahrzeuges und die Trajektorie selbst eine große Rolle, da der Korridor im Bereich von Kurvenfahrten wesentlich größer angelegt sein muss, als bei gerader Fahrt. Der Korridor bildet eine der Grundlagen für die Raum-Zeit-Planung.

Anhand des Korridors und der genauen Berechnung der Fahrzeiten, kann für jedes Fahrzeug vorhergesagt werden, wo es sich zu welchem Zeitpunkt im Korridor befindet und damit welche Fläche es belegt. Diese Informationen für alle Fahrzeuge bilden zusammen den Raum-Zeit-Plan, der das zentrale Element der Detailplanung ist. Erst wenn eine Fahraufgabe erfolgreich in den Raum-Zeit-Plan eingeplant worden ist, kann sie zur Ausführung freigegeben werden. Durch die vorliegenden Informationen über alle Fahrzeuge kann es möglich sein, dass manche Fahraufgaben nur unter bestimmten Voraussetzungen, z. B. durch eine Verzögerung der Startzeiten oder einer Veränderung der Geschwindigkeit, eingeplant werden können. Dies erzeugt eine hohe Planungskomplexität und zahlreiche Abhängigkeiten unter den zeitlich eingeplanten Abläufen. Sollte ein Ablauf aus irgendeinem Grund aus dem Plan fallen, so müssen die Auswirkungen berechnet und alle davon abhängenden Aufgaben ebenfalls beeinflusst werden. Aus diesem Grund ist ein fortlaufender Soll/Ist-Abgleich unabdingbar. Genauere Beschreibungen zum Raum-Zeit-Plan und der Raum-Zeit-Planung sind in Abschnitt 4.4.1 dargestellt.

Aufgrund der zentralen Bedeutung der Raum-Zeit-Planung besitzt die Detailplanung die breiteste Schnittstelle zu allen anderen Schichten. Sie muss in der Lage sein, eine Vielzahl von Nachrichten auszuwerten und selbst die Planung zu beeinflussen.

4.1.3.6 Fahrzeugkontrolle

Die Hauptaufgaben der Fahrzeugkontrolle ist die Überwachung der auf dem Gelände befindlichen Fahrzeuge. Dazu gilt es folgende Teilaufgaben zu erfüllen:

- Erkennen und Verhindern von Kollisionen
- Spurkontrolle
- Kontrolle der Einhaltung des Raum-Zeit-Plans
- Erkennung, Vermeidung und Beseitigung von Deadlocks

Die Datengrundlage ist hierbei ähnlich der Detailplanung. Die Fahrzeugkontrolle verwaltet Flächenbelegungen jeglicher Art und berechnet diese auf unterschiedlichste Weisen. Während feste Hindernisse des Geländes durch feste Belegungspolygone symbolisiert werden, so müssen alle belegten Flächen von sich bewegenden Fahrzeugen fortwährend neu berechnet werden. Abgestellte Fahrzeuge und Container werden in diesem Fall wie starre Hindernisse behandelt. Während die Berechnung für feste Hindernisse einfach handzuhaben ist, muss bei der Berechnung von sich bewegenden Fahrzeugen sehr viel detaillierter vorgegangen werden. Dazu werden die Belegungspolygone in verschiedene Klassen unterschieden. Die beiden wichtigsten Klassen bilden die reinen Belegungspolygone und die Sicherheitspolygone.

Die Belegungspolygone sollen die zu einem Zeitpunkt von einem Fahrzeug belegte Fläche repräsentieren. Dazu wird jedes Fahrzeugteil in seiner aktuellen Ausrichtung und Position im Weltkoordinatensystem zur Berechnung herangezogen. Somit kann eine konvexe Hülle, die das Fahrzeug komplett umschließt, berechnet werden. Für alle weiteren Details dieser aufwendigen Berechnung sei auf [LuT] Abschnitt 4.8 verwiesen.

Die Sicherheitspolygone sind Prognosen für die Zukunft. Sie geben den Bereich an, in dem sich ein fahrendes Fahrzeug in Kürze befinden wird. Die Berechnung der Sicherheitspolygone hängt daher von vielen Faktoren ab, wie z. B. der aktuellen Position und Ausrichtung, den Abmessungen des Fahrzeuges, des aktuellen Lenkwinkels und der abzufahrenden Trajektorie. Die Sicherheitspolygone dienen im Weiteren zur Erkennung von möglichen Kollisionen und beeinflussen die Fahrgeschwindigkeit der Fahrzeuge. Berührt das Sicherheitspolygon eines Fahrzeuges ein anderes auf dem Gelände befindliches Polygon[17], so wird das fahrende Fahrzeug zunächst

[17] z. B. die Polygone von Hindernissen, Fahrzeugen etc

gebremst, in der Hoffnung eine drohende Kollision zu vermeiden. Diese Bremsung kann bis zu einem vollkommenen Stillstand fortgesetzt werden, damit das Fahrzeug über einen Nothalt eine Kollision verhindert.

Die Grundidee der Spurkontrolle ist die Überwachung des eingeplanten Fahrweges eines Fahrzeuges. Sie soll dafür Sorge tragen, dass ein Fahrzeug niemals den eigenen Korridor verlässt. Dazu ist zum einen die exakte Berechnung des Korridors notwendig und zum anderen ein fortwährender Soll/Ist-Abgleich der berechneten und der realen Fahrzeugposition vorzunehmen. Bei Abweichungen muss auch an dieser Stelle entsprechend reagiert werden können.

Die Einhaltung des Raum-Zeit-Planes ist eine weitere wichtige Aufgabe der Fahrzeugkontrolle. Auf der Datengrundlage der Detailplanung und dem erstellten Raum-Zeit-Plan, sowie Momentaufnahmen des Geländes, können Abweichungen vom Plan erkannt und ihnen entgegengewirkt werden. Wie bereits in Abschnitt 4.1.3.5 erwähnt, können sehr viele Fahraufgaben in Abhängigkeiten zueinanderstehen, sodass die Verzögerung einer Fahraufgabe gleich eine Verzögerung aller davon abhängenden Fahraufgaben bedeuten kann. Aus diesem Grund sollte stets versucht werden, den Raum-Zeit-Plan durch regulierende Maßnahmen einzuhalten. Dies kann z. B. über die Fahrgeschwindigkeit der Fahrzeuge geschehen, indem sie bei einer Verzögerung angehoben und bei einer zu frühen Ankunft gesenkt wird.

Aufgrund der zentralen Verplanung der Fahraufgaben über den Raum-Zeit-Plan, sowie der vorhandenen Kollisionserkennung und -verhinderung, sollte es augenscheinlich nicht möglich sein, dass Fahrzeuge in eine Verklemmung geraten und sich damit gegenseitig blockieren. Diese Aussage ist nur dann korrekt, wenn stets alle Abläufe auf dem Gelände den vorgesehenen Plan einhalten. In der Realität dagegen ist es denkbar, dass einzelne Fahrzeuge aus verschiedenen Gründen aus dem für sie berechneten Plan herausfallen. Diese Abweichungen können dann unter Umständen selbst durch eine höhere Fahrtgeschwindigkeit nicht mehr korrigiert werden. Somit ist es möglich, dass sich z. B. an Kreuzungspunkten zweier Fahrwege Verklemmungen bilden, aus denen nur eine Sonderbehandlung[18] herausführt. Dies ist die Aufgabe der Deadlockerkennung, -vermeidung und -beseitigung. Eine Verklemmung soll vom Leitstand erkannt, wenn möglich im Voraus vermieden und bei Eintritt beseitigt werden können. In der aktuellen Implementierung ist dies jedoch nur vorbereitet und nicht vollends funktional eingearbeitet.

[18] aller an der Verklemmung beteiligten Fahrzeuge

4.1.3.7 Fahrzeugmanagement

Die Fahrzeugverwaltung ist die Datenbank des Leitstandes für alle registrierten Fahrzeuge. In ihr werden alle Komponenten von Fahrzeugen wie Zugmaschinen, Sattelauflieger und Container in aktive und passive Fahrzeugteile untergliedert und deren Daten für alle anderen Module bereitgestellt. Meldet sich ein Fahrzeug am Leitstand an, so überträgt es alle Fahrzeug-relevanten Daten[19] an die Fahrzeugverwaltung. Jede Fahrzeugkomponente bekommt daraufhin eine eindeutige ID zugewiesen und wird über diese in allen weiteren Abläufen leitstandsintern verwaltet.
Aufgrund der Tatsache, dass alle Module verschiedene Informationen über die registrierten Fahrzeuge benötigen, besitzt die Fahrzeugverwaltung eine breite Schnittstelle zur Kommunikation. Sie ist in der Lage sehr viele unterschiedliche Anfragen aller Module zu beantworten.
Eine weitere Aufgabe der Fahrzeugverwaltung ist die Aufbereitung der dynamischen Fahrzeugdaten. Diese werden in regelmäßigen Abständen von allen Fahrzeugen gesendet und beinhalten Informationen über die aktuelle Position, Ausrichtung, Fahrgeschwindigkeit etc. Da nicht alle Module diese direkten Informationen verarbeiten können, bereitet die Fahrzeugverwaltung diese Daten den Modulen entsprechend auf und leitet sie an alle Interessenten weiter.

4.1.4 Umgebungsmodellierung

Die wesentlichen Umgebungselemente des Allgemeinen Leitstandes sind zum einen die zu steuernden autonomen Fahrzeuge, aber auch zum anderen alle Abläufe auf dem Gelände, über deren Status der Leitstand fortwährend Informationen benötigt. Während die Fahrzeuge in der Simulation Teil einer virtuellen Umgebung sind, können alle weiteren Abläufe über zusätzliche externe Module dargestellt werden. Beides wird im Folgenden kurz erläutert.

4.1.4.1 Externe Module

Wie in Abschnitt 4.1.1 erklärt, bedarf es weiterer Module, die außerhalb des Leitstandes diesen mit notwendigen Informationen versorgen. Um diese externen Informationen empfangen und verarbeiten zu können, verfügt der Leitstand über einen zusätzlichen externen Ereignisverteiler.

[19] Abmessungen, kinematische Daten, eindeutige Beschriftungen oder das Kennzeichen

An diesem können sich externe Module anmelden und mit dem Leitstand kommunizieren. Auch hier gilt das gleiche hohe Abstraktionsprinzip, sodass die versendeten Nachrichten weitgehend unabhängig von den Modulstrukturen zu implementieren sind. Somit können einzelne Module einfach modifiziert und angepasst werden. Auf die genaue Erläuterung der Funktionsweise des externen Ereignisverteilers sei an dieser Stelle verzichtet. Es sei statt dessen auf [LuT] Abschnitt 3.4.1 sowie den Abschnitt 4.1.3.1 dieser Arbeit verwiesen.

Zum Aufbau der Verbindung sendet das Modul eine Nachricht an den externen Ereignisverteiler und wird dort über eine feste ID bei diesem registriert. Nach der Registrierung kann sich das Modul für bestimmte Nachrichten eintragen und bekommt diese fortan vom Ereignisverteiler weitergeleitet. Dabei kann das Modul nicht nur Nachrichten empfangen, sondern ebenso an den Leitstand versenden. Somit hat es die Möglichkeit z. B. Daten abzufragen oder steuernd in den Ablauf einzugreifen. Über dieses Konzept lässt sich die Umgebung des Leitstandes einfach und effizient erweitern und sogar auf mehreren Rechnern verteilt ausführen, sofern die Rechner mit dem Leitstand über ein Netzwerk verbunden sind.

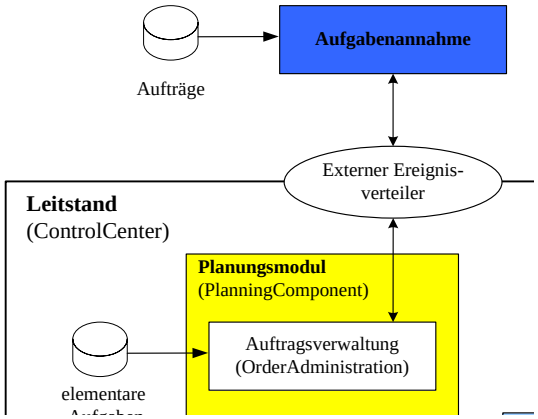


Abbildung 4.4: Aufgabenannahme als externes Modul

Das augenscheinlich notwendigste externe Modul ist die *Aufgabenannahme*, durch die die abzuarbeitenden Aufgaben auf dem Gelände erst in das System des Leitstandes gelangen. Die Aufgabenannahme ist die Schnittstelle zum Benutzer des Leitstandes und kann je nach Anwendungsfall die verschiedenen Aufgaben auf unterschiedlichen Wegen in das System weiterleiten. In Abbildung 4.4 ist die Aufgabenannahme als ein externes Modul dargestellt, das so beispielsweise

durch eine Benutzeroberfläche die notwendigen Daten vom Anwender abfragt. Die Eingabe dieser Daten muss allerdings nicht ausschließlich über ein einziges externes Modul erfolgen, sondern es könnten ebenso weitere Module am externen Ereignisverteiler angemeldet sein. So wäre es möglich, dass jeder Mitarbeiter des Terminals auf einem Notebook das für seine Arbeit relevante externe Modul ausführt[20] und darüber Daten direkt an den Leitstand weiterleiten kann.

Meldet sich ein Fahrzeug am Leitstand an, so fragt die Auftragsverwaltung über den externen Ereignisverteiler bei der Aufgabenannahme nach Aufgaben für dieses Fahrzeug. Der Aufgabenannahme liegt eine Auftragsdatenbank in XML-Form zugrunde, sodass sie dort die notwendigen Informationen einlesen und an den Leitstand weiterleiten kann. Diese Datenbasis muss je nach Anwendungsfall erzeugt und gegebenenfalls um weitere Aufgabenstrukturen erweitert werden.

Das zweite in dieser Arbeit eingesetzte externe Modul, ist die von Jörg Sesterhenn in seiner Diplomarbeit realisierte Visualisierung[21]. Diese meldet sich ebenso beim externen Ereignisverteiler des Leitstandes an und trägt sich für den Empfang der für sie relevanten Daten zur Darstellung der Abläufe ein. Diese sind z. B. die aktuellen Daten der Fahrzeuge, deren Trajektorien oder die von Fahrzeugen belegten Flächen. Doch nicht nur die bloße Darstellung der Abläufe, sondern auch die Kommunikation mit dem Leitstand ist über die Visualisierung möglich. So können Angaben zu den Fahraufgaben der einzelnen Fahrzeuge abgefragt oder in den Ablauf aktiv[22] eingegriffen werden.

4.1.4.2 Fahrzeuge und virtuelle Umgebung

Um dem Leitstand das Ansteuern der autonomen Fahrzeuge zur ermöglichen, bedarf es einer weiteren Schnittstelle nach außen, der sogenannten *Fahrzeugkommunikation*. Nur über diese können die Fahrzeuge mit dem Leitstand kommunizieren und ausschließlich über diese kann der Leitstand Nachrichten an die Fahrzeuge weiterleiten. Durch die präzise Definition dieser Schnittstelle kann garantiert werden, dass nicht versehentlich interne Nachrichten an die Fahrzeuge weitergeleitet werden.

[20] z. B. zum Überwachen von Fahrzeug-Transformationen, Mischbereichen etc.
[21] vgl. [Ses]
[22] z. B. durch das Stoppen von Fahrzeugen oder dem Auflösen von eingeleiteten Notbremsungen

Die Funktionsweise der Fahrzeugkommunikation ähnelt der eines Ereignisverteilers und wird in der momentanen Implementierung auch von einem solchen übernommen. Alle Nachrichten an die Fahrzeuge werden zunächst an die Fahrzeugkommunikation gesendet. Diese leitet sie, nach einer eventuellen Modifikation, an das jeweilige Fahrzeug weiter. Somit können alle Fahrzeuge, die die Schnittstelle der Fahrzeugkommunikation implementieren, mit dem Leitstand verbunden werden. Dabei merkt das System nicht, ob es sich um ein reales oder ein simuliertes Fahrzeug handelt. Bei beiden Arten erfragt der Leitstand die für ihn notwendigen Daten, wie die aktuelle Position und Ausrichtung, den Einknickwinkel bei Sattelaufliegern und Anhängern oder die derzeitige Geschwindigkeit. Bei einem realen Fahrzeug würde diese Anfrage z. B. per WLAN geschehen. Daraufhin würde die auf dem Fahrzeug befindliche Recheneinheit die aktuellen Fahrzeugdaten auslesen und dem Leitstand zusenden.

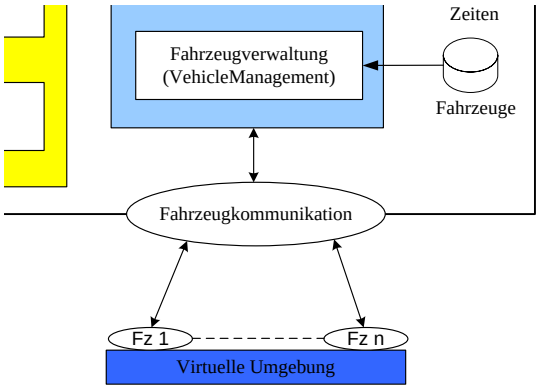


Abbildung 4.5: Kommunikation des Leitstandes mit den Fahrzeugen

Zu ausführlichen Tests im Labor empfiehlt sich der Einsatz von simulierten Fahrzeugen durch eine Fahrzeugsimulation. Dabei werden die Fahrzeuge über ein Basisreferenzpunktsystem und ein lokales Koordinatensystem beschrieben[23], in dem deren Abmessungen festgehalten werden. Alle weiteren notwendigen Angaben befinden sich in der Beschreibung der statischen Kinematikdaten. Um das Verhalten der Fahrzeuge so realistisch wie möglich zu modellieren, unterliegen sie alle einer Fahrzeugsimulation, die gewährleistet, dass die für den Leitstand notwendigen Daten immer rechtzeitig vorliegen.
Alle Fahrzeuge und die Fahrzeugsimulation sind Teil der virtuellen Umgebung (*VirtualEnvironment*) des Leitstandes. Sie bildet einen weiteren Prozess, der parallel zum Leitstand ausgeführt

[23] vgl.[LuT] Abschnitt 3.3

wird und diesen mit weiteren notwendigen Eingabedaten versorgt. Dabei können die virtuelle Umgebung und der Leitstand niemals direkt miteinander kommunizieren, sondern lediglich indirekt durch den Einfluss auf die Fahrzeuge.

Eine weitere Aufgabe der virtuellen Umgebung ist die Erzeugung der initialen Position der Fahrzeuge, noch bevor sie die Kommunikation mit dem Leitstand aufnehmen. Der Umgebung liegen Datengrundlagen im XML-Format vor, die angeben wann, welches Fahrzeug wo auf dem Gelände erscheinen und sich bei dem Leitstand anmelden soll. Dabei wird der Erscheinungsort in der Datei *EntryAreas.xml* festgehalten, der Zeitpunkt und die Art des Fahrzeuges in Datei *Configfile.xml* und die notwendigen genaueren Angaben zum Fahrzeug in Datei *ObjectDescription.xml*. Wurde ein Fahrzeug auf dem Gelände erzeugt, so meldet es sich mit einer bestimmten Nachricht am Leitstand an und wird von dort an durch diesen geleitet.

Die virtuelle Umgebung bietet neben den externen Modulen eine weitere Möglichkeit zusätzliche Funktionalitäten, die nicht direkt in den Leitstand eingreifen, zu implementieren. Dies ist besonders für das Erstellen ganzer Simulationsszenarios sehr hilfreich. Die in der derzeitigen Implementierung verwendeten simulierten Fahrzeuge, sowie die dahinter stehende Fahrzeugsimulation, sind Teil der EZauto-Bibliothek und konnten dank ihrer hohen Abstraktion und Flexibilität in die Entwicklung des Leitstandes einbezogen werden.

4.1.5 Schnittstellen zur Spezialisierung

Der folgende Abschnitt beschäftigt sich nochmals gezielt mit dem Konzept des Allgemeinen Leitstandes. An dieser Stelle werden die notwendigen Schnittstellen zur Anpassung an einen Anwendungsfall herausgestellt. Abbildung 4.6 zeigt den Allgemeinen Leitstand in die Spezialisierung eines Anwendungsfalls eingearbeitet.

Während die inneren Module des Allgemeinen Leitstandes unabhängig von der Anwendung sind[24], bildet das äußerste Modul, die Auftragsverwaltung, die breiteste Schnittstelle zur Anwendung. Die Aufgabe der Auftragsverwaltung ist es, Aufträge von außen an den Leitstand weiterzuleiten und entsprechend aufzubereiten. Je nach Anwendungsfall können diese Aufträge unterschiedlichste Merkmale aufweisen. So könnte z. B. ein Anwendungsfall mit relativ weni-

[24] sie können somit ohne Modifikationen übernommen werden

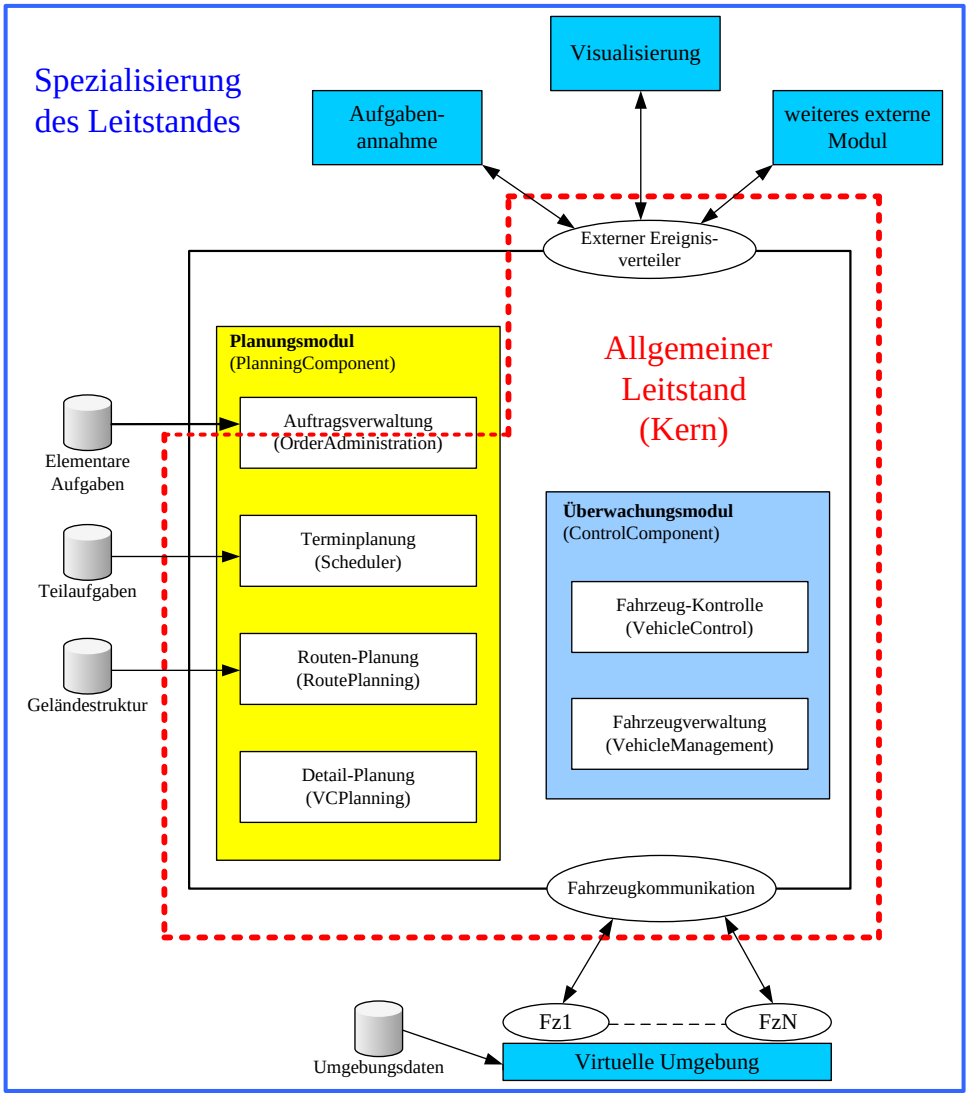


Abbildung 4.6: Kern und Spezialisierung des Leitstand

gen Aufträgen auskommen, während in einem anderen sehr viele Aufträge den Leitstand erreichen müssen. Ebenso kann die Art und Vielfalt der auf dem Gelände möglichen Aufgaben stark voneinander abweichen, je nach Größe des Geländes und den Variationsmöglichkeiten des Anwendungsfalles. Diese Tatsachen sind bei der Gestaltung der Auftragsverwaltung zu berücksichtigen. Da die Auftragsverwaltung sehr eng mit dem externen Modul der Aufgabenannahme zusammenarbeitet, gelten die Änderungen hier ebenso. Beide müssen aufeinander abgestimmt sein, um effizient und vollständig alle relevanten Daten an den Leitstand weiterleiten zu können.

Dazu sind alle auf dem Gelände und im Anwendungsfall möglichen Aufgaben zu erfassen und in Teilaufgaben zu zerlegen. Diese bilden die Schnittstelle zum Leitstand und können somit der Terminplanung zur Bearbeitung übergeben werden. Die Einhaltung dieser Schnittstelle ist dabei unumgänglich, da nur auf dieser Basis eine Bearbeitung der Aufgaben und die Unabhängigkeit des Allgemeinen Leitstandes von der Anwendung gewährleistet ist.

Eine weitere Schnittstelle zur Anwendung sind die verschiedenen Datengrundlagen der einzelnen Module. Auch diese sind an den Anwendungsfall anzupassen und müssen bestimmte Standards erfüllen. So muss z. B. das Gelände in Form von Leitlinien erschlossen sein und diese der Routenplanung in einer XML-Datenbank zugänglich gemacht werden.

Aufgrund der Tatsache, dass der Allgemeine Leitstand bisher nur beispielhaft implementiert wurde, kann es bei einer Spezialisierung auf die verschiedenen Anwendungsfälle auch zu notwendigen Änderungen am Kern kommen. Diese erweitern die Funktionalität des Kerns und zeichnen sich dadurch aus, dass sie unabhängig von der Anwendung auch in anderen Beispielen eingesetzt werden können. Änderungen am Kern ziehen oftmals zahlreiche Auswirkungen mit sich, daher müssen sie sehr gut geplant werden und dazu alle internen Abläufe verstanden sein.

Das Verhalten der Umgebung des Leitstandes kann je nach Anwendungsfall durch weitere externe Module dargestellt oder ebenso in die virtuelle Umgebung eingearbeitet werden. Handelt es sich bei der Anwendung um eine reine Simulation, so ist der Einsatz einer Fahrzeugsimulation unerlässlich. Auch diese kann an den Anwendungsfall angepasst und weitere Fahrzeuge eingearbeitet werden.

Auch die Visualisierung bedarf je nach Kontext einiger Anpassungen. So müssen auch hier die notwendigen Darstellungsgrundlagen, wie die Umgebungskarte, die Fahrzeugmodelle oder aber auch weitere Funktionalitäten implementiert werden.

4.2 Leitstand EZsped

In diesem Abschnitt soll kurz umrissen werden, in welcher Umgebung der Leitstand im Projekt EZsped eingesetzt worden ist. Die Zielumgebung ist der innovative Logistikhof. Die Anforderun-

gen, die an das System gestellt werden, sind jene eines Speditionshofes.

4.2.1 Anforderungen

Die grundsätzliche Aufgabe eines Speditionshofes geht schon aus dem Handelsgesetzbuch hervor. Darin verpflichtet sich ein Spediteur durch den Speditionsvertrag dazu, die Versendung des Gutes zu besorgen[25]. Der Transport von Gütern steht also im Vordergrund. Fahrzeuge kommen am Hof an und müssen dort von der manuellen Fahrt auf die autonome Fahrt umgestellt werden. Dabei müssen sie dem Leitstand bekannt gemacht, und unter dessen Kontrolle gestellt werden. Sind die Fahrzeuge angemeldet, so müssen sie dort autonom all jene Aufgaben erfüllen, die im manuellen Betrieb von Fahrern durchgeführt werden würden. Führen die Fahrzeuge Güter mit sich, so müssen diese entladen werden. Anschließend müssen wieder Waren auf das Fahrzeug aufgeladen werden. Für beide Vorgänge bedarf es einer exakten Anfahrt an die Laderampen mit dem Heck der Fahrzeuge. Die Lkw müssen dann so lange an den Rampen verweilen, bis ihre Güter ab- bzw. aufgeladen wurden. Erst dann können sie ihre weiteren Aufgaben erfüllen.

Durch Be- und Entladevorgänge wird der Aufbau der Fahrzeuge verändert - sie werden *transformiert*. Zu den sog. Transformationsaufgaben zählt auch das An- bzw. Abkuppeln einzelner Fahrzeugteile. Für Speditionshöfe sind solche Vorgänge von größter Bedeutung und sollten daher ebenfalls autonom durch den Leitstand durchgeführt werden können[26].

Zu jeder Zeit des Aufenthaltes auf dem Speditionshof kann es zu Verzögerungen kommen, in denen die Fahrzeuge geparkt werden müssen, um für andere Fahrzeuge kein Hindernis darzustellen. Der Leitstand muss also in der Lage sein, einen Parkplatz für das Fahrzeug zu finden und das Fahrzeug für eine Zeitspanne dort warten zu lassen. Die Länge dieser Wartephase ist unter Umständen nicht bekannt. Bevor das Fahrzeug das Gelände wieder verlässt, müssen möglicherweise noch Reparaturen an Fahrzeugteilen vorgenommen werden. Ebenso kann es erforderlich sein, das Fahrzeug zu waschen oder zu betanken. Soll das Fahrzeug betankt werden, so muss es mit seiner Tanköffnung exakt vor den Tankstutzen fahren. Die Aufgaben, welche ein Fahrzeug auf einem Speditionshof auszuführen in der Lage sein sollte, sind also:

- Be- und Entladen an Laderampen

[25] [HGB] §453 Abs. 1

[26] Transformationen werden von EZsped zwar vorgesehen, wurden jedoch erst in EZrola implementiert

- An- und Abkuppelvorgänge
- Parken
- eine Waschanlage anfahren
- eine Tankstelle anfahren
- eine Reparatur- oder Wartungshalle anfahren

4.2.2 Umgebung

Mit dem Begriff Umgebung sind alle Vorgänge gemeint, die nicht zum Funktionsumfang des Leitstandes gehören. Der Leitstand ist für alle Aufgaben zuständig, die von den Fahrzeugen selbst durchgeführt werden, jedoch nur von dem Zeitpunkt an, an dem die Fahrzeuge in den Machtbereich des Leitstandes übergeben werden und auch nur so lange, bis die Fahrzeuge aus diesem Machtbereich wieder gelöst werden. Dem Machtbereich selbst lässt sich auch eine räumliche Ausdehnung zuordnen, welche die Grenzen des Speditionshofes nicht überschreitet. Alles, was aus dem räumlichen bzw. dem funktionalen Rahmen herausfällt, fällt in den Aufgabenbereich der Umgebungssimulation.

4.2.2.1 Fahrzeugsimulation

Die Simulation der Fahrzeuge ist wohl das wichtigste Element der Umgebungssimulation. Solange keine realen Fahrzeuge oder Modelle existieren, lässt sich ohne eine solche Simulation keine Aussage über die Fähigkeiten oder auch die Probleme des Leitstandes treffen. Eine Validierung ist dann unmöglich. Für den Leitstand soll verborgen bleiben, ob die Daten, die dieser bekommt, von echten oder simulierten Fahrzeugen stammen. Selbst ein Mischbetrieb sollte möglich sein. Auf diese Weise kann ohne Änderungen am Leitstand eine Portierung in ein reelles System erfolgen, vorausgesetzt die Ausmaße des Geländes sind korrekt erfasst.

Die Fahrzeugsimulation muss kontinuierlich Positionsdaten und Ausrichtungen modifizieren, um die Fahrt auf der berechneten Trajektorie zu simulieren. Die berechneten Daten müssen dem Leitstand über dessen spezielle Schnitstelle bereit gestellt werden. Die Fahrzeugsimulation muss so nah wie möglich an den kinematischen Eigenschaften der realen Fahrzeuge angelehnt sein, um möglichst alle Probleme, die bei der Verwendung von echten Fahrzeugen auftreten können, schon vor der Portierung erkennen und beseitigen zu können.

4.2.2.2 Funktionale Einheiten

Einige der geländespezifischen Aufgaben[27] erfordern Funktionalitäten, die nicht in den Aufgabenbereich des Leitstandes fallen. Alle diese Aufgaben haben gemeinsam, dass sie an speziellen Einrichtungen auf dem Gelände des Speditionshofes durchgeführt werden, die durch die Fahrzeuge angefahren werden müssen. Solche Einrichtungen werden funktionale Einheiten genannt. Diese müssen getrennt vom Leitstand implementiert werden. Kommen die Fahrzeuge bei diesen Einheiten an, so übernehmen diese für einige Zeit gewissermaßen die Kontrolle über die Fahrzeuge. Ein tankendes Fahrzeug darf beispielsweise erst dann die Tankstelle verlassen, wenn der Tankstutzen entfernt und der Tank des Fahrzeuges wieder geschlossen wurde. Erst wenn es von der funktionalen Einheit freigegeben wurde, darf der Leitstand wieder einen Auftrag an das Fahrzeug vergeben.

Die jeweiligen Vorgänge in den funktionalen Einheiten müssen in den meisten Fällen nicht simuliert werden. So hat beispielsweise das Waschen eines Fahrzeuges für den Leitstand keine relevanten Folgen. Das Gleiche gilt für den Tankvorgang. Der Tankfüllstand sollte nicht durch den Leitstand, sondern durch die Fahrzeugelektronik geprüft werden. Im Falle einer Knappheit wäre eine Nachricht an den Leitstand denkbar. Dieser könnte dann automatisch reagieren und eine Tankaufgabe einfügen. Ebenso könnte mit Reparaturen verfahren werden, wenn die Bordelektronik ein Problem erkennt. Fährt das Fahrzeug die Wartungshalle an, so könnten diese Problemsignalisierungen im Fahrzeug gelöscht werden. Es würde die Wartungshalle ohne Defekte verlassen[28].

Vorerst ist es jedoch ausreichend, die Vorgänge an den meisten funktionalen Einheiten durch Wartezeiten zu simulieren. Die Fahrzeuge können dann, nach fest vorgeschriebener oder zufällig festgelegter Zeit, wieder freigegeben werden.

4.3 Leitstand EZrola

In diesem Abschnitt soll der im Kontext der Rollenden Landstraße eingesetzte Leitstand vorgestellt werden. Um den Schritt vom Allgemeinen Leitstand zur Spezialisierung des RoLa-Leitstandes zu vollziehen, bedarf es zunächst einer ausführlichen Analyse der an das System

[27] z. B. Be- und Entladen, Tanken, Waschen und Wartungs- und Reparaturaufgaben

[28] Diese Tiefe der Simulation wird momentan weder von der Umgebungssimulation, noch von der Leitstandssoftware erreicht, ließe sich aber (auch durch Änderungen, die durch die vorliegende Arbeit eingepflegt wurden) auf Leitstandseite durchaus verwirklichen

gestellten Anforderungen. Dabei werden die wesentlichen Punkte aufgeführt, die im Rahmen dieser Arbeit entwickelt und implementiert wurden. Neben diesen, auf den Leitstand bezogenen Anforderungen, gibt es weitere Änderungen an der Gesamtumgebung des Systems. Auch diese werden an dieser Stelle beschrieben und nachfolgend der Entwurf des RoLa-Leistandes aus einer funktionalen Sicht betrachtet. Die Feinheiten der Implementierungen und die internen Abläufe werden daraufhin in Kapitel 5 genauer erklärt.

4.3.1 Anforderungen

Die in den folgenden Abschnitten erläuterten Anforderungen sind zum Teil in Bezug auf das zu verstehen, was nach dem Abschluss des Projektes EZsped zur Verfügung stand. Es handelt sich um eine Art inkrementellen Ablauf, d. h. die Grundlage der Überlegungen waren nicht ausschließlich die Anforderungen von EZrola, sondern auch die Überlegungen, ob diese Anforderungen nicht durch das Ergebnis von EZsped schon erfüllt worden sind.

4.3.1.1 Parken

Die Anforderungen eines RoLa-Terminals machen das Parken zu einem zentralen Bestandteil des Leitstandes, in dessen ursprünglicher Form Parken nur ein Unterbaum der statischen Taskstruktur, bestehend aus Fahraufgaben, war. Es existierte keine gesonderte Parkplatzverwaltung. Sollte ein Fahrzeug auf dem Gelände parken, so musste für dieses Fahrzeug genau die Fahraufgabe aus der Aufgabenstruktur gewählt werden, die dem Parkplatz entsprach, auf welchem geparkt werden sollte. Wurde der übergeordnete Knoten[29] ausgewählt, so entschied sich der Leitstand grundsätzlich für den ersten Nachfahren. Sollten zwei Fahrzeuge auf diese Weise parken, so wurde versucht, diese für den gleichen Parkplatz zu verplanen. Für die Rollende Landstraße ist eine automatische Parkplatzvergabe jedoch zwingend notwendig, da sonst manuell jedem Fahrzeug ein Parkplatz zugeteilt werden müsste.

Die Notwendigkeit einer zentralen Parkplatzverwaltung zeigt sich besonders, wenn vor Augen geführt wird, welche verschiedenen Parkmöglichkeiten es auf dem Gelände gibt und welche Anforderungen an diese Typen jeweils gestellt werden:

1. „Normales“ Parken

[29] der übergeordnete Knoten entsprach der Aufgabe „Parken im Allgemeinen“

Der „normale“ Parkvorgang unterscheidet sich gegenüber dem Parken in der ursprünglichen Version des Leitstandes nur dadurch, dass der Parkplatz automatisch gewählt und belegt wird und, wenn alle Fahrzeugteile ihn wieder verlassen[30], freigegeben wird.
Dieser Parktyp wird für zwei verschiedene Aufgaben auf dem RoLa-Terminal verwandt: Für Auflieger, die vom Zug kommend für den Kunden zur Abholung auf dem Hof bereitgestellt werden und für Fahrzeuge, die vom Zug kommend auf dem Hof auf einen anderen Zug warten.

2. Warten auf den nächsten Zug
   Fahrzeuge, die auf dem Hof gesammelt werden, um auf den nächsten Zug aufzufahren, müssen möglichst platzsparend geparkt werden. Hierfür empfehlen sich Wartespuren, da die Fahrzeuge dort sehr dicht aufeinander auffahren können. Hierbei muss beachtet werden, dass sich die Länge der Lkw unterscheiden kann. Von besonderer Bedeutung ist eine Einschränkung, die durch die Beschaffenheit der Züge auferlegt wird. Überschreiten Lkw eine bestimmte Länge, so dürfen auf dem vorhergehenden und dem nachfolgenden Waggon nur kürzere Fahrzeuge stehen. Aus diesem Grund müssen die Lkw schon vor dem Auffahren sortiert werden. Kommen längere Fahrzeuge an, so müssen diese in gesonderte Wartespuren geleitet werden. Sollen die Fahrzeuge dann auf den Zug aufgefahren werden, können abwechselnd Fahrzeuge mit Überlänge und Fahrzeuge mit normaler Länge gestartet werden. Die Auswahl einer Spur und des Punktes auf dieser Spur soll vollautomatisch ablaufen. An dieser Stelle muss das Konzept der komplett statischen Aufgabenstruktur verlassen werden, da der Punkt, an dem das Fahrzeug zum Stehen kommen soll, erst zur Laufzeit berechnet werden kann[31].

3. Parken von Terminaltraktoren
   Ähnlich zu den Lkw, die auf den nächsten Zug warten, warten die Terminaltraktoren auf neue Aufgaben. Da alle Traktoren die gleichen funktionalen Eigenschaften besitzen, ist es nicht von Bedeutung, welcher Traktor für die Erfüllung einer Aufgabe ausgewählt wird. Daher können diese also ebenfalls in Spuren geparkt werden. Die Anforderungen unterscheiden sich hierbei von den Anforderungen der Wartespuren des zweiten Parktyps. Es werden

[30] Durch Transformationsaufgaben auf einem Parkplatz kann es dazu kommen, dass der Auflieger eines Lkw alleine auf dem Parkplatz zurückbleibt. Die Fahrzeugteile müssen daher von der Parkplatzverwaltung getrennt behandelt werden.

[31] Außer die Spuren werden in Abschnitte fester Größe unterteilt und die Haltepunkte schon beim Entwurf des Hofes festgelegt

keine verschiedenen Längen unterschieden. Die Traktoren fahren rückwärts in die Spuren, um das Abholen der Auflieger im Übergabenereich zu erleichtern. Bekommt ein Traktor neue Aufgaben, so entstehen Freiräume in den Spuren, die aufgefüllt werden müssen.

Genauere Informationen zur Implementierung des Parkens finden sich in Kapitel 5.5.

4.3.1.2 Transformationen

Bei der Analyse[32] der Abläufe eines autonom genutzten Terminals in Regensburg musste festgestellt werden, dass aufgrund der Verwendung von Terminaltraktoren die Sattelauflieger und/oder Anhänger sehr oft ihr Zugfahrzeug wechseln. Dies wäre bei der vorgestellten Verwendung zunächst bei fast jedem Kunden der Fall, da der Anteil der zur autonomen Fahrt umgerüsteten Kundenfahrzeuge zunächst als verschwindend gering anzusehen ist. Dieser würde erst mit zunehmender autonomer Nutzung steigen[33]. Aus diesem Grund, muss das verwendete System in der Lage sein, die Veränderungen von Fahrzeugen einzuleiten und verarbeiten zu können. Jegliche Modifizierung an unter der Kontrolle[34] des Leitstandes stehenden Fahrzeugen, wird von nun an *(Fahrzeug-)Transformation* genannt.

Obwohl das Transformieren der Fahrzeuge ein wesentlicher Bestandteil zur Umsetzung eines autonomen Terminals ist, ist deren Entwicklung und Implementierung nicht primär zu den für die RoLa notwendigen Anforderungen zu zählen. Sie stellt vielmehr eine Erweiterung des Allgemeinen Leitstandes dar. Das Modifizieren von Fahrzeugen kann in sehr vielen Anwendungsgebieten von großer Bedeutung sein und sollte damit zentraler Bestandteil der durch einen Allgemeinen Leitstand bereitgestellten Funktionalitäten sein.

Eine Transformation kann für das System Folgendes bedeuten:

- Das An- und Abkuppeln von Sattelaufliegern und Anhängern[35].
- Das Aufnehmen und Abstellen von Transportelementen wie z. B. Containern[36].

[32] vgl. Abschnitt 3.3
[33] aufgrund fallender Umrüstungskosten bei Serienproduktion oder Neukauf der Zugmaschine
[34] zur Laufzeit, d. h. die Fahrzeuge sind bereits am Leitstand angemeldet
[35] hier ist ein Betrieb eines Terminaltraktors mit mehreren angekuppelten Anhängern durchaus denkbar
[36] leitstandsintern als Body bezeichnet

Im vorliegenden Anwendungsfall wurde sich ausschließlich auf Punkt 1 der genannten Aufzählung bezogen, da ein Umschlagen von Containern im Kontext der RoLa keinerlei Rolle spielt. Dennoch wurde auch dies mit in die Analyse und spätere Spezifikation einbezogen, um die Einarbeitung der notwendigen Erweiterungen in späteren Arbeiten zu erleichtern. Alle weiteren Details zur Umsetzung und Implementierung vgl. Abschnitt 5.1.

4.3.1.3 Dynamische Strukturen

Viele Abläufe des ursprünglichen Leitstandes, beispielsweise die Berechnung der Trajektorien für eine Fahraufgabe, geschehen dynamisch. Wird im Konzept des Leitstandes die Auftragsverwaltung und deren Zusammenspiel mit externen Modulen betrachtet, so muss festgehalten werden, dass einige Prozesse noch als statisch zu bezeichnen sind.

So liegt jeder zu erfüllenden Aufgabe auf dem Hof ein Knoten des statischen Aufgabenbaums zugrunde. Alle Informationen, die zur Ausführung der Aufgabe nötig sind, müssen schon vor der Laufzeit festgelegt sein. Dies gilt z. B. für den Zielpunkt einer Fahraufgabe. Aus diesem Grund kann keine Fahraufgabe auf dem Hof an einem Punkt enden, der nicht im statischen Aufgabenbaum festgelegt ist. Für den Anwendungsfall der Rollenden Landstraße musste diese Statik durchbrochen werden. Es gibt gleich mehrere Fälle, in denen der Zielpunkt erst zur Laufzeit berechnet werden kann. Beim Parken in einer Wartespur werden die Fahrzeuge mit einem festen Abstand[37] hintereinander platziert. Da die Fahrzeuge unterschiedlich lang sind, lässt sich der Zielpunkt erst bestimmen, wenn bekannt ist, welche Fahrzeuge bereits in der Spur stehen[38]. Das gleiche Problem stellt sich beim Befahren des Zuges, da die Position der Waggons ebenfalls nicht vorhergesagt werden kann[39]. Auch bei Transformationen muss der Zielpunkt dynamisch berechnet werden können, da beispielsweise beim Ankuppeln eines Aufliegers dessen Position auf dem Hof erst erfragt werden muss[40].

Das in dieser Arbeit entwickelte Konzept für die Anwendung der Leitstand-Software auf die Rollende Landstraße bewirkt, dass über einen Traktor, der das Terminal mit dem Zug erreicht, zu Beginn keine Aussage über seine künftigen Fahraufgaben getroffen werden kann. Für den

[37] momentan mit einem Abstand von 1,5 m
[38] vgl. Abschnitt 5.5
[39] vgl. Abschnitt 5.3
[40] vgl. Abschnitt 5.1

Fall, dass der Auflieger, den er befördert, zum Abholen bereitgestellt werden soll, wird sich der Traktor, der nach dem Abkuppelvorgang frei ist, in eine Traktorspur begeben. Irgendwann wird er dann, nachdem er in der Spur unter Umständen einige Male aufgerückt ist, um weiteren Fahrzeugen Platz zu bieten, ausgewählt, um einen Auflieger abzuholen und diesen wieder auf den Zug zu fahren. Zu Beginn ist weder bekannt wann das sein wird, noch um welchen Auflieger es sich handeln wird. Die bisherige Struktur sah jedoch vor, dass zu jedem Fahrzeug zu Beginn genau feststeht, welche Aufgaben es zu erledigen hat. Der Leitstand muss deshalb so modifiziert werden, dass Fahrzeuge auch dynamisch ausgewählt werden können. So könnten dann Aufgaben zugewiesen werden, die nicht von Beginn an für diese Fahrzeuge geplant waren. Genauere Informationen dazu sind in Abschnitt 5.3 und in Abschnitt 5.5 zu finden. In diesen Abschnitten werden die Änderungen an der Auftragsverwaltung und die Implementation der Parkplatzverwaltung beschrieben.

Auch an der Umgebung des Leitstandes müssen statische Strukturen beseitigt werden. So konnten bisher nur Fahrzeuge erzeugt werden, die in der Konfigurationsdatei der virtuellen Umgebung explizit aufgeführt wurden. In den Testszenarien dieser Arbeit werden innerhalb einer Stunde etwa 70 aktive Fahrzeuge erzeugt. Mit komplett statischen Strukturen würde dies einen extrem hohen Zeitbedarf in der Planung der Szenarien bedeuten. Die Erzeugung der Fahrzeuge erfolgt daher im Zusammenspiel mehrerer Module und mithilfe einiger statischer Standardeinträge in der Konfigurationsdatei der virtuellen Umgebung[41].

Die gleiche Problematik betrifft die Aufträge für Fahrzeuge. Bisher musste auch hier für jedes Fahrzeug ein Eintrag in einer XML-Datei vorliegen. Davon abgesehen, dass diese Aufträge vor der Laufzeit noch nicht zugeordnet werden können, würde auch hier die Erzeugung der nötigen Strukturen einen sehr großen Zeitbedarf bedeuten. Deshalb wird auch hier mit einigen Standardeinträgen in den statischen Strukturen gearbeitet, welche dann aber dynamisch den Fahrzeugen zugeordnet werden[42].

4.3.1.4 Auftragsvergabe

Für EZrola ist eine ganz andere Auftragsstruktur nötig, als es noch bei EZsped der Fall war. Je nach Rolle des Fahrzeuges kommen verschiedene Aufträge in Frage. Für einen abgegebenen Auflieger müssen Aufgaben an einen Traktor vergeben werden, welche diesen anweisen zu dem

[41] vgl. die Abschnitte 5.2 und 5.7
[42] vgl. Abschnitt 5.8

Auflieger zu fahren, ihn anzukuppeln, um dann mit ihm in eine Wartespur zu fahren. Für Fahrzeuge, die mit dem Zug ankommen, gibt es mehrere Möglichkeiten:

- Ein Fahrzeug kann auf einen Parkplatz geleitet werden, wo dann der Auflieger abgekuppelt wird, um später vom Kunden abgeholt zu werden.
- Ein Fahrzeug kann geparkt werden, um auf einen späteren Zug zu warten.
- Ein Fahrzeug kann direkt vom Zug in eine Wartespur beordert werden, um auf denselben Zug wieder aufzufahren; das wäre nötig, wenn dies nur ein Zwischenhalt wäre; sind die Lkw unsortiert müssten alle den Zug verlassen.

Zur Wahl des richtigen Auftrages muss ein Weg geschaffen werden, möglichst komfortabel auf das Szenario einwirken zu können, jedoch ohne eine feste Zuordnung von Aufträgen zu Fahrzeugen zu verlangen, da sonst der Aufwand zur Erzeugung eines Szenarios zu hoch wäre.

Sind alle Fahrzeuge vom Zug abgeladen, so muss an die wartenden Fahrzeuge der Auftrag vergeben werden, auf den Zug aufzufahren. Auch die Fahrzeuge, die von einem anderen Zug kommend auf diesen Zug gewartet haben, gilt es dabei zu beachten.

4.3.2 Umgebung

Die Umgebung des Leitstandes von EZrola ist durch zwei wesentliche Elemente geprägt: die virtuelle Umgebung zur vollständigen Simulation aller Abläufe auf dem Gelände, sowie die Informationen über ein- und ausfahrende Züge. Dabei muss die virtuelle Umgebung eine Vielzahl von Aufgaben, je nach Zielsetzung der entstehenden Gesamtsimulation, übernehmen. Die Hauptaufgabe der Zugdarstellung ist es dagegen mit dem Leitstand zu kommunizieren und notwendige Informationen auszutauschen. Beides wird im Folgenden kurz erläutert.

4.3.2.1 Virtuelle Umgebung

Die Gestaltung der virtuellen Umgebung des EZrola Leitstandes hängt stark von der Zielsetzung der entstehenden Gesamtsimulation ab. Dabei ist zu unterscheiden, ob die Simulation nur die rein internen Abläufe des Leitstandes darstellen können soll, oder ob auch externe Einflüsse Teil dieser Simulation werden sollen.

Die internen Abläufe auf dem Terminal des EZrola Leitstandes betreffen alle Fahraufgaben, die sich innerhalb des autonomen Bereiches abspielen, z. B. das Befahren der Züge oder die Fahrten der Traktoren zu ihren Parkspuren und Wartereihen. Dabei bilden die Misch- oder Übergabebereiche des Terminals mit ihrer abwechselnden manuellen sowie autonomen Fahrt die Schnittstellen zu den externen Abläufen. Soll die Gesamtsimulation lediglich die internen Abläufe darstellen, so wird der zu transportierende Sattelauflieger oder Anhänger nur in der Zeitspanne seiner Registrierung am Leitstand selbst dargestellt. Dies würde für den Übergabebereich 1 des Terminals Regensburg bedeuten, dass die Sattelauflieger oder Anhänger am Übergabeplatz erscheinen, sobald die Kundenfahrzeuge diese dort abgekuppelt haben und sie am System registriert sind. Die Anhänger würden in der Simulation plötzlich dargestellt, ohne dass dem Beobachter alle damit verbundenen Abläufe zur Ankunft und Übergabe des Anhängers durch den Kunden gezeigt worden wären. Dies würde sich im Mischbereich 2, dem Bereich der Abholer, ebenso ereignen. Dort würden alle zur Abholung bereitgestellten Sattelauflieger oder Anhänger nach der Umschaltung in die manuelle Fahrt zu dem Zeitpunkt verschwinden, an dem sie die Kontrolle des Leitstandes verlassen.Dies wäre zum Zeitpunkt der Übernahme durch den Kunden der Fall. Auch hier würde der Beobachter weder die Ein- noch Ausfahrt der Kundenfahrzeuge dargestellt bekommen.

Wird die Simulation als rein überwachendes Organ zur Visualisierung der Abläufe auf dem Terminal betrachtet, so würde die Darstellung der internen Abläufe vollkommen ausreichen. Sie entspricht exakt der Sicht des Leitstandes auf die Abläufe. Wird der Simulation allerdings auch ein demonstrativer Hintergrund zugesprochen, sodass sie in Präsentationen und Vorträgen den Kontext der Verwendung eines Terminals der Rollenden Landstraße widerspiegeln soll, so ist die Darstellung der externen Abläufe unerlässlich. Aus diesem Grund müssen auch sie mit in die virtuelle Umgebung aufgenommen werden. Damit muss die den EZrola-Leitstand umgebende virtuelle Umgebung folgende Aspekte erfüllen:

- Simulation aller Fahrzeuge
- Erzeugung der Fahrzeuge auf dem Zug
- Erzeugung der Kundenfahrzeuge zur Übergabe
- Erzeugung der Kundenfahrzeuge zur Abholung

4.3.2.2 Zugabwicklung

Das wohl augenscheinlichste Element, das für die Rollende Landstraße umgesetzt werden muss, ist die Abwicklung aller Vorgänge in Bezug auf die beteiligten Züge. Diese müssen - aus Sicht des Leitstandes - irgendwann in die Welt des Leitstandes eindringen und diese Welt nach einer gewissen Zeit wieder verlassen. Jeder Zug bedeutet für den Leitstand, dass neue Lkw erzeugt und kontrolliert werden müssen. Dafür muss ein virtueller Zug mit Waggons auf dem Hof erzeugt werden. Für die Fahrzeuge müssen Aufträge auf dem Hof generiert werden. Sind alle Lkw abgeladen und die Fahrzeuge, die den Hof mit diesem Zug verlassen sollen, aufgeladen, so muss der Zug aus dem System entfernt werden und mit ihm alle auf ihm befindlichen Fahrzeugteile.

Es muss ein Fahrplan für diese Züge implementiert werden, in dem festgelegt wird, zu welchen Zeitpunkten ein Zug ein- und ausfährt und welche Eigenschaften dieser Zug hat. Dazu zählt beispielsweise die Anzahl der Waggons, aus denen er besteht. Dieser Fahrplan muss von außen, über eine Beschreibungsdatei, in das System eingegeben werden können.

4.4 Analyse des Leitstandes

In den folgenden Abschnitten sollen drei zentrale Elemente bzw. Module des Leitstandes näher untersucht werden: der Raum-Zeit-Plan, der Scheduler sowie die Routenplanung und Trajektorienberechnung. Diese Elemente sind von grundlegender Bedeutung für Erweiterungen, die im Zuge dieser Arbeit durchgeführt wurden. Dabei gehen die Betrachtungen meist nur so weit, wie es für diese Arbeit relevant ist. An einigen Stellen wird deshalb auf andere beschreibende Dokumente verwiesen.

4.4.1 Der Raum-Zeit-Plan

Die Raum-Zeit-Planung kann als ein Teilmodul verstanden werden, welches das kritische Betriebsmittel „Raum“ des Hofes verwaltet. Es sei direkt darauf hingewiesen, dass damit (zumindest momentan) nicht der dreidimensionale Raum gemeint ist, sondern die zweidimensionale Fläche. Dieses Betriebsmittel wird dabei nicht in Einheiten zerlegt, die immer komplett vergeben werden, sondern als ganze Fläche gesehen, aus der die benötigten Flächen je nach Bedarf herausgeschnitten werden. Auch die Zeit wird nicht in konkrete Zeiteinheiten diskretisiert[43], sondern nach

[43] Durch die der Zeit zugrunde liegenden Datenstruktur liegt eine Diskretisierung vor. Die Zeit wird so in Bereiche mit einer Sekunde Länge geteilt

Bedarf zugeschnitten. Das erschwert eine formale Definition, es lässt sich jedoch feststellen, dass die Raum-Zeit-Planung als Funktion verstanden werden kann, die Teilmengen des kritischen Betriebsmittels Raum, abhängig von der Zeit, Fahrzeugen zuteilt.

Der Raum-Zeit-Plan beinhaltet aus Programmsicht Listen von Tupeln der Form $(Zeitpunkt \times Zeitpunkt \times Objektbelegungspolygon)$[44], welche *PlanningAllocationpolygon* genannt werden. Die beiden Zeitpunkte ergeben zusammen einen Zeitbereich. Im Objektbelegungspolygon werden Objekte einem Polygon, also einer Fläche auf dem Hof, zugeordnet. Das Tupel lässt sich also beschreiben als $(Zeitabschnitt \times ObjektIDs \times Flaeche)$. Auf diese Weise kann jedem Fahrauftrag ein Korridor zugeordnet werden, der jedem Zeitpunkt des Ablaufs ein Polygon des Hofes zuordnet. Dies wäre eine Teilmenge des „globalen“ Raum-Zeit-Plans. Die Größe des Polygons spiegelt einige Sicherheitsaspekte wieder. Sie muss so gewählt werden, dass das Fahrzeug diesen Bereich unter keinen Umständen verlässt, bzw. so, dass das Fahrzeug unter gegebenen Randbedingungen zum Stehen gebracht werden kann, bevor es diesen Bereich verlassen würde.

Soll ein Fahrzeug eingeplant werden, wird versucht, für dessen Trajektorie[45] einen Raum-Zeit-Plan zu erzeugen. Alle Daten, die dafür nötig sind, werden in einem sog. *DrivingJobProgressionPlan*[46] zusammengefasst. Bildlich gesprochen wird damit versucht, in diesem Vorgang Zeiten an den Fahrweg zu heften. Es wird also versucht, im bisherigen Raum-Zeit-Plan einen kollisionsfreien Korridor für das Fahrzeug und dessen Trajektorie zu finden. Der Planungsbeginn wird durch den Auftrag vorgegeben, liegt aber frühestens fünf Sekunden in der Zukunft, um sicher zu gehen, dass der fertige Fahrauftrag dann auch noch rechtzeitig vom Scheduler eingeplant werden kann. Um testen zu können, ob es eine Kollision mit einem anderen Objekt zu einem bestimmten Zeitpunkt gibt, wird sich eines Schnappschusses bedient, des sog. *AllocationSnapshots*. Für diesen werden aus dem Raum-Zeit-Plan alle Einträge gesucht, die für diesen Zeitpunkt gültig sind. Der strukturelle Zusammenhang zwischen Raum-Zeit-Plan, den PlanningAllocationPolygons und den AllocationSnapshots kann Abbildung 4.7 entnommen werden. Liegt eine Kollision vor, so wird versucht, diese durch Verringern der Geschwindigkeit zu verhindern. Die Geschwindigkeit wird also kontinuierlich gesenkt und dabei auf eine mögliche Kollision getestet. Kann diese damit verhindert werden, so wird mit der Planung fortgefahren und die Geschwindigkeit wieder auf den ursprüngliche Wert erhöht.

[44] nähere Informationen zu Belegungspolygonen sind in [LuT] in Abschnitt 4.7.1.1 zu finden
[45] die schon vorher berechnet worden ist
[46] siehe [LuT] Abschnitt 4.7.1.2

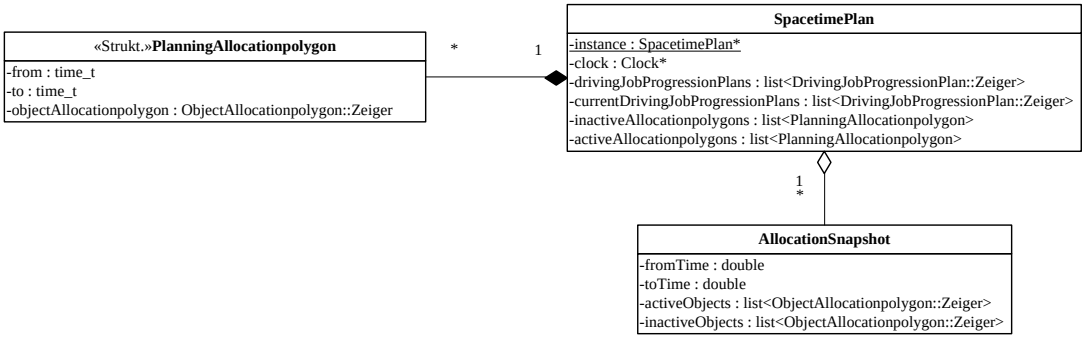


Abbildung 4.7: Klassendiagramm der Raum-Zeit-Planung

Kann die Kollision auf diese Weise nicht verhindert werden, so wird versucht, den Fahrauftrag später zu starten. Die maximale Anzahl dieser Versuche und die Zeit, um die der Start verzögert wird, sind festgelegt. Kann trotzdem kein Plan gefunden werden, so wird das Fahrzeug direkt gestartet und Kollisionen werden ausschließlich durch die *VehicleControl* verhindert. Für weitere Informationen zur Raum-Zeit-Planung sei auf [LuT] Kapitel 4.7, Abschnitt „*Detailplanung*“, verwiesen.

4.4.2 Der Scheduler

Der Scheduler übernimmt in der Planungskomponente des Leitstandes die Aufgabe der zeitlichen Planung der einzelnen Teilaufgaben. Er besitzt eine stark abstrahierte Sicht auf die Vorgänge in der realen Welt. Er soll ohne genaueres Wissen entscheiden, ob eine bestimmte Aufgabe zu einer bestimmten Zeit durchführbar ist und diese Aufgabe gegebenenfalls zu diesem Zeitpunkt einplanen. Der Scheduler ist dafür zuständig, Teilaufgaben zu den ihnen zugedachten Zeitpunkten zu starten und andere Module darüber, sowie über das Beenden einer Teilaufgabe, in Kenntnis zu setzen.

Die Planung eines Auftrags beginnt mit dem Empfang eines *TaskSchedulingQuery*-Ereignisses. Tritt dies ein, sucht der Scheduler nach dem zum Fahrzeug gehörenden Plan. Solch ein Plan speichert alle für den Scheduler wichtigen Informationen. So enthält er die *BasisTasks*, also die Tasks, die vom Scheduler eingeplant werden sollen, und eine sog. *Sequence*, welche die auszuführenden Teilaufgaben (*Subtasks*) in der richtigen Reihenfolge und einen Zeiger auf den momentan laufenden Subtask enthält. Zuerst prüft der Scheduler, ob der Plan momentan auszuführende Tasks

enthält, da in diesem Fall die ankommende TaskSchedulingQuery abgewiesen wird. Gibt es keine ausstehenden Aufgaben, kann die Anfrage bearbeitet werden. Dafür werden die Aufgaben, die als *InputTasks* ankommen, in *SBasisTasks* umgewandelt und im Plan gespeichert. Solche Umwandlungen werden an vielen Stellen durchgeführt, da die Module verschiedene Sichten auf Daten mit den gleichen Grundlagen besitzen. Als Nächstes wird für diese Tasks eine mögliche Sequenz berechnet. Es ist angedacht, dass beispielsweise zusätzlich übermittelte Regeln beachtet werden können, um die Reihenfolgen der einzelnen Aufgaben zu bestimmen. Momentan wird der Auftrag eines Fahrzeugs jedoch genau in der Form in die Sequenz eingefügt, wie er von der Auftragsverwaltung entgegen genommen wird. Mit dieser Grundlage wird anschließend der Planungsvorgang gestartet.

Im eigentlichen Planungsvorgang, der im Folgenden beschrieben wird, wird versucht, Teilaufgabe für Teilaufgabe zu verplanen. Dazu wird der erste noch nicht verplante Subtask gesucht. Der Scheduler unterscheidet nur drei verschiedene Aufgabentypen: *DrivingTask*, *TransformationTask* und *WaitingTask*. Jeder Typ hat verschiedene Attribute, die berechnet werden müssen. Sie enthalten dann alle Daten die für die Ausführung des jeweiligen Aufgabentyps unerlässlich sind.

Bei WaitingTasks handelt es sich um einfache Warteaufgaben. Das Fahrzeug steht dabei bewegungslos auf dem Hof und wartet auf ein Ereignis. Es sind nur der Startzeitpunkt und der Endzeitpunkt von Bedeutung. Als Startzeit wird der Endzeitpunkt, der in der Sequenz vorangehenden Teilaufgabe, herangezogen. Für die Endzeit gibt es drei Möglichkeiten:

1. Es wird auf eine sog. *WaitForMessage* gewartet. Trifft diese ein, wird der WaitingTask beendet. Die Planung kann in diesem Fall nicht weitergeführt werden, bevor diese Nachricht eintrifft.

2. Der Endzeitpunkt wird vorher festgelegt und kann direkt eingetragen werden. In diesem Fall kann mit der Planung beim nächsten Subtask fortgefahren werden.

3. Die Dauer wird angegeben und der Endzeitpunkt kann direkt berechnet werden. Auch in diesem Fall kann mit dem Planungsprozess fortgefahren werden.

WaitingTasks sind für die Umsetzung eines RoLa-Terminals in der dieser Diplomarbeit zugrunde liegenden Version nicht von Bedeutung, werden also nicht eingesetzt.

Bei einem TransformationTask muss festgelegt werden, welche Fahrzeugteile verbunden bzw. getrennt werden sollen. Für den weiteren Planungsprozess muss außerdem die im Plan gespeicherte Fahrzeugbeschreibungskette angepasst werden, da das Fahrzeug in den in der Sequenz folgenden Teilaufgaben seine Gestalt geändert haben wird. Eine Transformation endet mit dem Empfang einer bestimmten Nachricht, deshalb kann der Endzeitpunkt nicht bestimmt werden. Aus diesem Grund sollte an dieser Stelle nicht weiter geplant werden.

Am schwierigsten zu verplanen sind die Fahraufgaben. Es muss zuerst ermittelt werden, an welcher Position sich das Fahrzeug beim Start einer Fahraufgabe befindet. Ist diese nicht aus dem vorangehenden Subtask bekannt, so muss sie bei der Fahrzeugverwaltung angefragt werden (*IPositionRequest*). Da diese Anfrage über den Ereignisverteiler erfolgt, kann die Planung nicht fortgeführt werden. Dies geschieht erst beim Empfang des *IPositionAnswer*-Ereignisses. Mit der Startposition kann jetzt die Routenplanung initiiert werden. Nachdem die Anfrage (*IDrivingJob*) an die Routenplanung versandt wurde, wird der Planungsprozess vorläufig beendet und es wird auf eine Antwort gewartet. Bei dieser Antwort handelt es sich um eine sogenannte *ITimePossibility*, die Start- und Endzeit enthält. Diese werden in den DrivingTask übernommen. Die anderen Module werden über die Entscheidung für die betreffende Route benachrichtigt (*IRouteDecision*) und die Planung wird beim nächsten Subtask fortgesetzt.

Auf diese Weise wird die Aufgabensequenz sukzessive geplant. Es wird dabei versucht, so viele Teilaufgaben wie möglich zu bearbeiten, indem so lange fortgefahren wird, bis entweder eine weitere Planung nicht möglich ist, weil auf Daten gewartet werden muss, oder alle in der Sequenz enthaltenen Teilaufgaben eingeplant sind.

Der Scheduler prüft regelmäßig, ob die Startzeit von Subtasks überschritten wurde, und leitet den Start des jeweiligen Subtasks ein, sofern die vorhergehende Teilaufgabe beendet worden ist.

Das Beenden von Teilaufgaben wird durch verschiedene Ereignisse eingeleitet. Bei einer Fahraufgabe ist dies beispielsweise der Empfang des zugehörigen *IDrivingJobCompleted*-Ereignisses. Im Scheduler zieht das einige Reaktionen nach sich. Durch das Attribut *Endmessage* eines jeden Subtasks besteht die Möglichkeit, automatisch Nachrichten zu verschicken, wenn diese Aufgabe beendet wird. Ist dieses Attribut gesetzt, so wird vom Scheduler ein Ereignis versendet, welches als Namen den Wert trägt, der in ihm gespeichert ist. Danach wird in jedem Fall noch eine Standardnachricht über die Vollendung des Subtasks geschickt (*ISubtaskCompleted*). Sie trägt

die zughörige Subtask-ID als Attribut in sich. Der Unterschied zwischen dieser Nachricht und der vorher beschriebenen Ende-Nachricht ist, dass die Verwendung der Ende-Nachricht nicht die Kenntnis der Subtask-ID aufseiten des Empfängers erfordert. Außerdem muss der Empfänger sich ausschließlich für den Empfang der speziellen Endmessage eintragen und nicht alle Ereignisse des Typs ISubtaskCompleted verarbeiten, wenn er auf das Ende einer ganz bestimmten Teilaufgabe wartet. Als Letztes prüft der Scheduler, ob die Sequenz des Plans bereits komplett berechnet wurde. Sind noch Teilaufgaben vorhanden, die nicht den Status „geplant“ besitzen, so kann die Planung jetzt fortgesetzt werden. Wie weiter oben bereits erwähnt, kann bei einigen Teilaufgaben der Endzeitpunkt nicht im Voraus berechnet werden. Die Planung des nachfolgenden Subtasks kann dann nicht begonnen werden, weil für diesen der Anfangszeitpunkt nicht bekannt ist. Aus solchen Gründen löst das Ende einer Teilaufgabe immer eine mögliche Fortsetzung der Planung aus.

Tritt bei der Planung ein Fehler auf, wird der Ablauf für den betreffenden Plan gestoppt. Eine Fehlerbehandlung existiert nicht.

4.4.3 Routenplanung

In diesem Abschnitt wird der Ablauf der Routenplanung innerhalb des Leitstandes genauer betrachtet. Dabei wird in erster Linie auf eine Gesamtübersicht Wert gelegt, sodass der Zusammenhang der beteiligen Klassen und Module aufgezeigt und verständlich analysiert wird. Die Routenplanung nimmt hinter der Raum-Zeit-Planung und dem Scheduler die dritte zentrale Position im Gesamtkonzept des Leitstandes ein. Daher sollte sie bis ins Detail verstanden sein, ehe Verbesserungsvorschläge effizient umgesetzt werden können. Einige Elemente der Routenplanung mussten im Rahmen dieser Arbeit verbessert werden, sodass an dieser Stelle der angesprochene Überblick gegeben wird, ehe in Abschnitt 5.10.3 die Modifikationen einzelner Elemente kurz aufgezeigt werden.

Der Initiator zum Planen einer Route ist, wie bei allen elementaren Planungsschritten, stets der Scheduler, der die Aufgabe bekommt, eine Fahraufgabe für ein Fahrzeug einzuplanen. Dazu sendet er die Nachricht *IDrivingJob* mit allen notwendigen Fahrzeugdaten[47] und der einzuplanenden Route an die Routenplanung. Diese bereitet daraufhin die Planung vor, indem sie alle notwendigen Angaben ihrer Datenbank entnimmt und Fehlende erfragt. So muss die aktuelle Gesamtlänge des Fahrzeuges vor der Planung über die Nachricht *VDCLengthQuery* bei der

[47] die Start- und Endposition sowie die Ausrichtungen des Fahrzeuges in diesen Punkten

Fahrzeugverwaltung angefordert werden. Diese sendet die Antwort in der Nachricht *VDCLengthAnswer* zurück an die Routenplanung. Sind nun alle notwendigen Datensätze vorhanden, so kann die Planung der Route beginnen.

Dies geschieht in mehreren Schritten und wird von unterschiedlichen Klassen der Routenplanung geleistet. Zunächst gilt es, anhand der Leitlinien des Geländes einen passenden Weg vom Start- zum Zielpunkt zu finden. Dazu müssen zuerst die jeweils nächstgelegenen Leitlinien zum Start- sowie zum Zielpunkt über ein Suchverfahren bestimmt werden. Hierbei können natürlich in beiden Fällen mehrere Leitlinien in Betracht kommen. Somit muss die Auswahl anhand verschiedener Gesichtspunkte auf nur jeweils eine Linie in der Nähe des Start-, als auch eine Linie in der Nähe des Zielpunktes getroffen werden. Alle gefundenen Leitlinien, die nicht ausgewählt worden sind, könnten im Weiteren für alternative Routen genutzt werden, falls keine fahrbare Route für die getroffene Auswahl gefunden werden kann.
Wurden nun die Start- sowie die Zielleitlinie bestimmt, so gibt es für die mögliche Routenplanung theoretisch maximal 16 mögliche Kombinationen, diese Leitlinien in einer Route miteinander zu verbinden. Die Ursache hierfür ist im Aufbau der Leitlinien zu suchen. Jede Leitlinie ist ein Polygon bestehend aus mindestens zwei Punkten, welche die Leitlinie in Segmente unterteilen. Durch die Reihenfolge der Punkte, bei Angabe der Leitlinie, wird dieser eine Polygonzugrichtung zugewiesen[48], sodass unterschieden werden kann, ob sich nun in Richtung des Polygons oder dagegen bewegt wird. Zu diesen beiden Möglichkeiten kommen die beiden weiteren des Fahrzeuges, sich vorwärts oder rückwärts entlang der Leitlinie zu bewegen. Somit ergeben sich vier mögliche Fahrtrichtungen, nämlich in Richtung des Polygons oder dagegen, kombiniert mit der Fahrweise vorwärts oder rückwärts. Diese Varianten werden der Leitlinie über die zugehörigen Leitlinienkomponenten zugewiesen, d. h. eine Leitlinie muss mindestens eine Komponente besitzen, um befahren werden zu können, kann aber maximal nur vier Komponenten aufweisen. Abbildung 4.8 zeigt eine Leitlinie mit zwei zugehörigen Komponenten: in Polygonzug vorwärts sowie gegen Polygonzug rückwärts. Diese sind zur anschaulichen Darstellung parallel zur Leitlinie gezeichnet, besitzen jedoch die gleichen Koordinaten und sollten korrekterweise „auf“ der Leitlinie liegen.

Sind nun die Start- sowie Zielleitlinie und die dazugehörenden Komponenten bestimmt worden, so kommen diese meist nicht alle für eine mögliche Route in Frage, da die Fahrzeugausrichtung das Befahren mancher Komponenten ausschließt. Daher kann schon vor der Routenplanung

[48] der Polygonzug ist immer von Punkt 1 nach Punkt 2 nach Punkt 3 usw.

eine grobe Filterung unter den Komponenten vorgenommen werden, sodass nur die befahrbaren Komponenten in die Auswahl der Planung kommen. Der genaue Ablauf der Filterung wird in Abschnitt 5.10.3 erläutert.

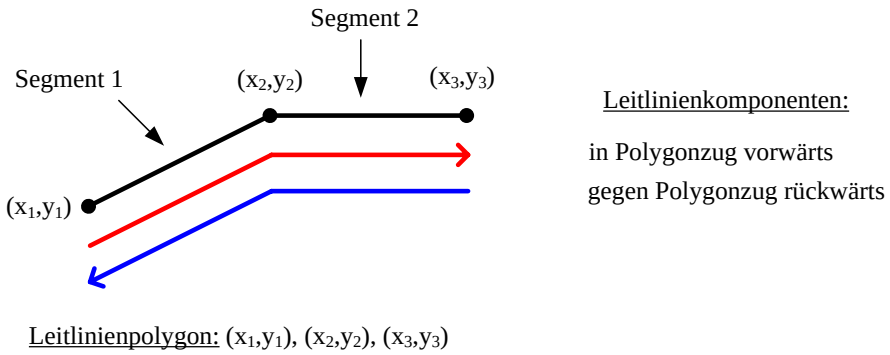


Abbildung 4.8: Aufbau einer Leitlinie

Alle zu einem Leitliniennetz gehörenden Komponenten stehen in einer Vorgänger-Nachfolger-Beziehung zueinander. So kann durch die Angabe einer möglichen Start- sowie Endkomponente eine Route zwischen diesen beiden über den Suchalgorithmus A* bestimmt werden, falls diese Route existiert. Die genaue Funktionsweise von A* wird an dieser Stelle nicht erläutert, es sei statt dessen auf [LuT] Abschnitt 4.6.3 auf Seite 89 verwiesen.
A* testet somit die vorliegenden Möglichkeiten und kann dabei zu unterschiedlichen Ergebnissen kommen. Findet A* keine mögliche Route, so muss die Auswahl der Leitlinienkomponeten in der Nähe vom Start- sowie dem Zielpunkt neu getroffen werden. A* testet daraufhin erneut die möglichen Kombinationen. Findet A* jedoch eine oder mehrere Routen, so kann die Planung fortgesetzt werden. Im Falle von mehreren Routenmöglichkeiten bietet sich an dieser Stelle zum einen die Auswahl der Route über eine Heuristik an. Zum anderen wird aber auch die Möglichkeit gegeben, die gewählte Route abzuändern, falls diese nicht eingeplant werden kann, weil sie z. B. durch andere Fahrzeuge befahren wird.

Die gefundenen groben Routen werden im Anschluss über die Nachricht *ISPQuery* an die Detailplanung gesendet, die die Aufgabe hat, aus dieser die Fahrzeugtrajektorie zu berechnen. Auch dies geschieht wiederum in mehreren Teilschritten. Bisher besteht die Route lediglich aus einer Abfolge von Punkten, die durch die Leitlinien vorgegeben werden. Nun gilt es zunächst, Start- sowie Endposition zu fixieren. Dazu kann es notwendig sein, dass die Leitlinien in der Nähe

dieser beiden Punkte beschnitten werden müssen, da sie unter Umständen nicht vollends befahren werden können. Ist vorgegeben, dass das Fahrzeug den Zielpunkt mit einem ganz bestimmten Referenzpunkt [49] anfahren soll, so könnte die Berechnung dazu auch an dieser Stelle durchgeführt werden. Dies wird in der aktuellen Implementierung jedoch nicht geleistet. Im Anschluss beginnt die eigentliche Berechnung der Fahrzeugtrajektorie. Dazu bedarf es der Kenntnis der statischen Kinematikdaten des Fahrzeuges, da sie in die Berechnung mit einfließen. Die Trajektorie wird über die Funktion *calculateTrajectories* der Detailplanung berechnet und basiert auf Basisfunktionen der EZauto-Bibliothek. Dabei wird die Abfolge der Routen-Punkte in Teilschritten zu einer fahrbaren Trajektorie verschoben und diese in mehreren Iterationen zu einem für das Fahrzeug fahrbaren Weg geformt. Alle beteiligten Funktionen sowie die genaue Berechnung finden sich in Abschnitt V auf Seite 164 von [LuT].

Die entstandene Trajektorie wird im Anschluss an die Raum-Zeit-Planung übergeben. Hier wird zunächst der Korridor des Fahrzeuges berechnet und damit die Grundlage für die Einplanung gelegt. Einzelne Details hierzu wurden bereits in Abschnitt 4.4.1 erläutert. War die Einplanung erfolgreich, so meldet die Raum-Zeit-Planung dies dem Scheduler über die Nachricht *ITimePossibility* und teilt diesem die möglichen Ausführungszeiten mit, nach denen dieser alle weiteren Abläufe einplanen kann. Abbildung 4.9 zeigt die Abläufe der Routenplanung in der Übersicht.

[49] z. B. hintere Kupplung, Tanköffnung etc.

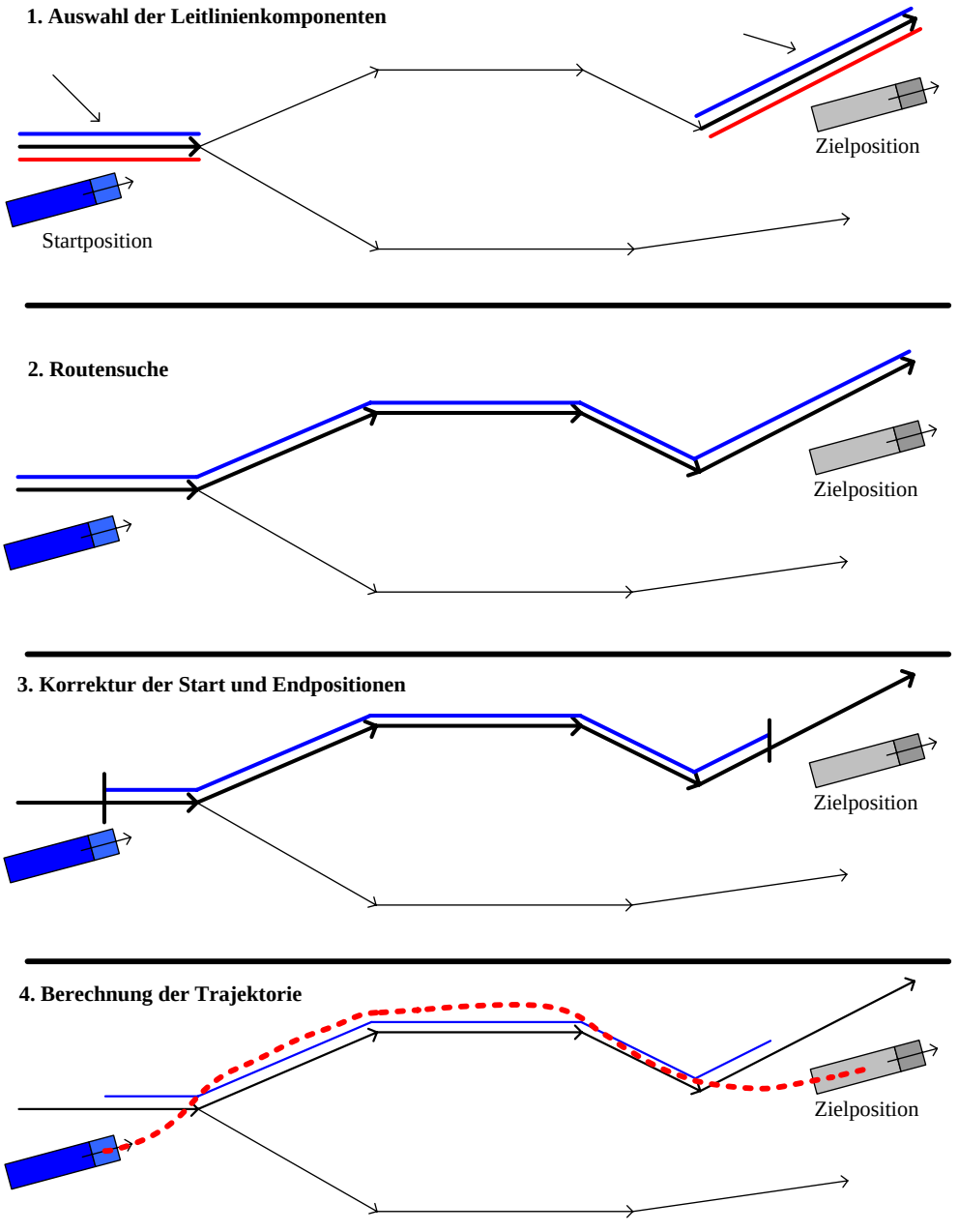


Abbildung 4.9: Routenplanung und Trajektorienberechnung

Kapitel 5

Die Erweiterungen und Konzepte zum Thema RoLa

Um aus dem Allgemeinen Leitstand eine Spezialisierung für den Anwendungsfall im Rahmen des Projektes EZrola zu erstellen, waren eine Reihe von Modifikationen notwendig, die in der Analyse in Kapitel 4 bereits angedeutet worden sind. Diese waren von unterschiedlichster Art und Ausprägung. So mussten z. B. die Funktionalitäten des Allgemeinen Leitstandes durch das Implementieren der Transformationen und einer Parkplatzverwaltung erweitert werden. Aber auch dessen Schnittstellen zur Umgebung, sowie die Umgebung des spezialisierten Leitstandes selbst, mussten neu entworfen und programmiert werden. Diese Umgebung beinhaltet sowohl die Simulation der auf dem Terminal einfahrenden Züge, als auch die zu einer Gesamtsimulation notwendigen Datengrundlagen. Darüber hinaus mussten an einigen Funktionen und Modulen ebenso Änderungen vorgenommen werden, die zum einen Fehler korrigierten und zum anderen deren Funktionalitäten erweiterten. Alle vorgenommenen Modifikationen werden in diesem Kapitel nacheinander ausführlich erklärt.

5.1 Transformationen

Bei den Transformationen handelt es sich um eine funktionale Erweiterung des Kernes des Allgemeinen Leitstandes. Diese sollen zunächst im Überblick betrachtet und die notwendigen Änderungen in Bezug auf den Leitstand ausführlich erklärt werden. Neben den Datengrundlagen

des Aufgabenbaumes mussten hierzu auch leitstandsinterne Abläufe integriert und aufeinander abgestimmt werden. Ebenso galt es, die Datenbanken der einzelnen Module zu modifizieren und Fahrzeugbeschreibungen zu aktualisieren. Abschließend werden die Auswirkungen der Transformationen auf das externe Modul der Visualisierung betrachtet.

5.1.1 Überblick und Zusammenhang

Um den Ablauf und die durch eine Transformation entstehenden Problemstellungen genauer erläutern zu können, sei folgendes Beispiel betrachtet: „Ein Terminaltraktor erreicht auf einem Zug stehend das Zielterminal und meldet sich beim Leitstand des Terminals an. Daraufhin bekommt er vom System den Auftrag, den angekuppelten Sattelauflieger in den Abholerbereich zu bringen und dort auf einem Parkplatz abzustellen. Nachdem er dies vollzogen hat, soll er den Abholerbereich verlassen und sich in die Parkspuren für Traktoren einreihen“.

Dieser zunächst einfach aussehende Ablauf bedeutet jedoch für den Leitstand zahlreiche Einzelheiten, auf die er während der Laufzeit reagieren muss. Zunächst meldet sich der Traktor zusammen mit dem Sattelauflieger als ein einziges Fahrzeug an. Diese Anmeldung wird als solches an alle Schichten weitergeleitet und jeder Prozess merkt sich die für ihn notwendige Datengrundlage. Wird an dieser Stelle die Sicherheitsüberwachung als Beispiel genommen, so speichert diese die vom Fahrzeug belegten Flächen als Polygone ab. Diese dienen ihr zur Kollisionserkennung. Ist die Anmeldung insgesamt vollzogen[1], so führt der Traktor die Fahraufgabe vom Zug zum Stellplatz aus. Dort angekommen, wird das Abkuppeln des Sattelaufliegers eingeleitet. Bei einem vollautomatischen Kuppelsystem muss der Vorgang nicht von einem Terminalangestellten überwacht werden und der Traktor kann dem System selbstständig mitteilen, wann alle Verbindungen zum Sattelauflieger getrennt sind und er somit seine Fahrt fortsetzen kann. Wird das Abkuppeln weiterhin manuell durchgeführt, so muss ein Terminalmitarbeiter dem Leitstand[2] mitteilen, wann der Vorgang vollständig durchgeführt worden ist.

Nach der Trennung von Traktor und Sattelauflieger beginnt die eigentliche Arbeit für das System, denn es gilt nun die entstandenen *Dateninkonsistenzen* der einzelnen Module, die Trak-

[1] alle Schichten sind informiert und besitzen die notwendigen Datengrundlagen

[2] z. B. durch den Einsatz eines Laptop mit WLAN, wodurch über die angebundene Visualisierung Nachrichten an den Leitstand verschickt werden können

tor und Sattelauflieger als ein einziges Fahrzeug ansehen, zu beseitigen. Dies zieht nach der Initiierung einer Transformation zahlreiche leitstandsinterne Nachrichten mit sich, die von jedem Prozess ausgewertet und verarbeitet werden müssen. Doch nicht nur intern, sondern auch allen extern angebundenen Modulen muss die Veränderung des Fahrzeuges in diesem Fall mitgeteilt werden. Dies gilt im Wesentlichen für die Fahrzeugsimulation und die Visualisierung, die beide über gesonderte Nachrichten informiert werden müssen. Dabei ist die Fahrsimulation hier als Ausnahme anzusehen, da sie lediglich zur vollständigen Simulation dient und die Benachrichtigung über eine Trennung an reale Fahrzeuge nicht notwendig ist[3].

Bei der Benachrichtigung der einzelnen Module reicht es oftmals nicht aus, die Informationen zu einer Transformation in nur einer einzigen Nachricht weiterzuleiten. Oftmals werden weitere Informationen über den Start- und Endzeitpunkt sowie weitere Fahraufgaben im Anschluss benötigt. Dies betrifft im Wesentlichen die Kollisionsüberwachung der Fahrzeugkontrolle, da sie die beiden durch eine Transformation entstandenen Fahrzeuge nicht unmittelbar nach dem Ende als vollständig getrennt ansehen darf. Der Traktor könnte z. B. noch unter dem Sattelauflieger stehen und sich somit die zur Überwachung dienenden Fahrzeugpolygone überdecken. Dies würde für den Leitstand eine Kollision bedeuteten und somit eine Ausnahme auslösen. Analog gilt dies beim Ankuppelvorgang, für den die Sicherheitsüberwachung des Sattelaufliegers oder Anhängers kurz vor dem Ankuppeln ausgeschaltet werden muss, damit eine Transformation überhaupt möglich ist.

Um die Planung einer Transformation für den Leitstand überhaupt zu ermöglichen, galt es zunächst, diese in den bestehenden Aufgabenbaum (die *StaticTaskstructure*) einzuarbeiten. Dieser wurde somit um einen Knoten erweitert, der selbst wieder zwei Söhne, für das Ankuppeln/-Aufnehmen und das Abkuppeln/Ablegen von Sattelaufliegern und Anhängern sowie Containern, besitzt. Um nun dem Leitstand das Ausführen einer solchen Aufgabe zu ermöglichen, bedarf es zunächst jedoch weiterer Informationen. Diese können dem System über das Anfügen von zusätzlichen Informationen, den *Additional Information*, mitgeteilt werden. Diese enthalten Angaben darüber, welcher Anhänger abgekuppelt oder welcher Container abgestellt werden soll oder geben die Position und den Anhänger für einen aufzunehmenden Container an. Diese Angaben konnten im Beispiel des Leitstandes von EZsped statisch von außen über die Auftragsannahme

[3] angenommen jedes Fahrzeug speichert sich nur seine eigenen Daten und nicht die der mit ihm verbundenen weiteren Fahrzeuge oder Transporteinheiten

dem Leitstand übermittelt werden, da sie lediglich die in der XML-Datei auszuführenden Aufgaben für die einzelnen Fahrzeuge auf dem Speditionshof erweiterten. Diese statische Angabe ist im Kontext von EZrola jedoch nicht empfehlenswert, da sehr viele Transformationen nach dem gleichen Schema auf dem Gelände stattfinden. Daher fiel die Entscheidung zugunsten einer dynamischen Erzeugung der zusätzlichen Informationen, die in der Auftragsannahme ausgeführt wird. Alle Abhängigkeiten und Details zu den jeweiligen Transformationsaufgaben werden in den folgenden beiden Abschnitten genauer betrachtet. Dabei wird sich in erster Linie auf das An- und Abkuppeln von Sattelaufliegern und Anhängern bezogen, da sie das Hauptaugenmerk im Beispiel des EZrola-Leitstandes bilden.

5.1.2 Transformationen: Teilaufgaben und interne Abläufe

Im Folgenden werden nun die leitstandsinternen Abläufe beim An- und Abkuppeln eines Sattelaufliegers oder Anhängers betrachtet. Dazu werden zunächst die einzelnen Teilaufgaben der Transformation im Aufgabenbaum analysiert und deren Notwendigkeit[4] begründet. Da das Transformieren von Fahrzeugen eine Modifikation der vorliegenden Fahrzeugbeschreibungen innerhalb der einzelnen Module bedeutet, wird ebenso erläutert[5], welche möglichen Sichtweisen die Module auf die registrierten Fahrzeuge haben und wie diese zu beeinflussen sind. Im Anschluss sollen die internen Abläufe von der Einplanung bis zur Ausführung einer Transformation[6] betrachtet werden.

5.1.2.1 Teilaufgaben des Abkuppelns

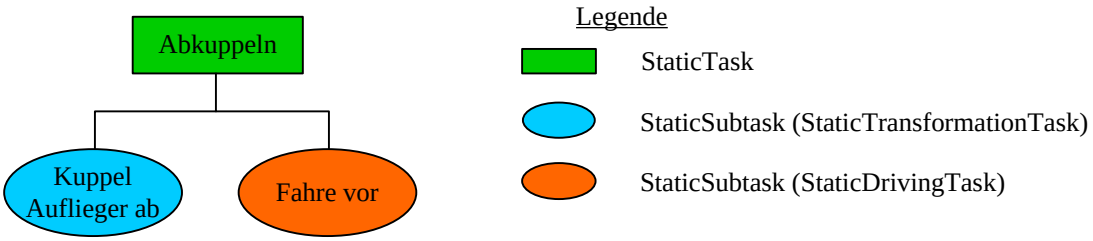
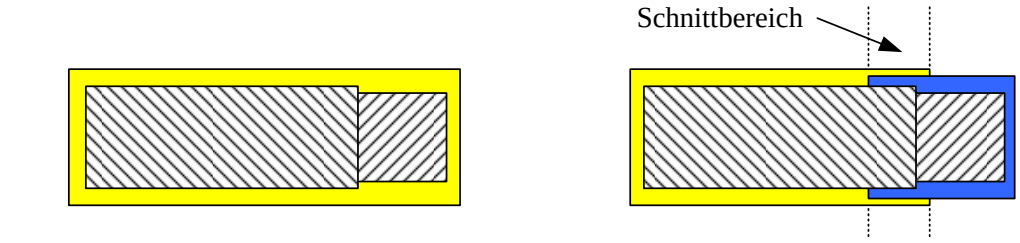


Abbildung 5.1: Teilaufgaben beim Abkuppeln

[4] vgl. die Abschnitte 5.1.2.1 und 5.1.2.2
[5] vgl. Abschnitt 5.1.2.3
[6] vgl. Abschnitt 5.1.2.5

Im statischen Aufgabenbaum beinhaltet das Abkuppeln zwei wesentliche Teilaufgaben: die eigentliche Transformationsaufgabe, sowie eine unmittelbar darauf folgende Fahraufgabe. Abbildung 5.1 zeigt die Zusammensetzung des Abkuppel-Knotens in der Baumstruktur. Der Grund für die eingefügte Fahraufgabe ist in der Fahrzeugsicherheit zu finden. Das Trennen eines Fahrzeuges bedeutet für den Leitstand, dass durch eine Transformation aus einem einzigen angemeldeten Fahrzeug zwei Fahrzeuge entstehen, die der Leitstand registrieren und überwachen muss. Wie in Abschnitt 5.1.1 erklärt wurde, besteht die Datengrundlage der Fahrzeugüberwachung aus den die Fahrzeuge umgebenden Polygonen, die auf der Grundlage verschiedenster Informationen über das jeweilige Fahrzeug berechnet werden. Kommt es nun zu einer Überschneidung zweier Fahrzeugpolygone, deutet der Leitstand dies als eine Kollision und leitet eine Notbremsung ein. Genau diese Situation ergibt sich unmittelbar nach der Transformation. Aus dem einen gesamten Fahrzeugpolygon für Zugmaschine und Sattelauflieger, wurden für die Fahrzeugüberwachung durch die Trennung zwei neue, sich überlappende, Polygone. Abbildung 5.2 verdeutlicht dies. Um dieses Problem zu beheben, wurde eine Fahraufgabe unmittelbar nach der Transformation eingefügt, die dazu dient, das Zugfahrzeug unter dem Auflieger hervor zu fahren. Somit können sofort definierte Zustände für die Fahrzeugüberwachung hergestellt werden. Dazu wird der Fahrzeugüberwachung zu Beginn der Transformation mitgeteilt, dass eine solche gerade gestartet wird und welche Fahrzeuge darin involviert sind. Diese Informationen wertet die Fahrzeugüberwachung aus, trennt im Falle des Abkuppelns die Fahrzeugteile voneinander und schaltet die Sicherheitsüberwachung nur für den beteiligen Auflieger aus. D. h. die Zugmaschine kann im Anschluss unter dem Auflieger hervor fahren, ohne dass eine Kollision gemeldet wird. Mit dem Ende der Fahraufgabe, die nur wenige Sekunden dauert, da die Zugmaschine lediglich 1,8 m unter dem Auflieger gerade hervorzieht, wird der Auflieger wieder in die vollständige Sicherheitsüberwachung aufgenommen.
Diese Handhabe bietet sich auch beim Abkuppeln einzelner Anhänger an, wobei hier die Distanz vom 1,8 m deutlich verkürzt werden kann, um den Anhänger schnellstmöglich in die vollwertige Sicherheitsüberwachung zu übernehmen.

Wie zu erkennen ist, geht dem Abkuppeln keine in die Transformation eingegliederte Fahraufgabe voraus. Dies bedeutet, dass sich die Transformation auf die aktuelle Position des Fahrzeuges bezieht und diese vorher durch eine einfache Fahraufgabe bestimmt werden muss. So bekommen z. B. die Kundenfahrzeuge, die in der Simulation das Terminal befahren, drei Aufgaben zugewiesen: Fahre zum Übergabeplatz, kuppele Anhänger ab, fahre zur Ausfahrt. Zusammen mit den beiden



(1) Gesamtpolygon für Zugmaschine und Sattelauflieger

(2) getrennte Polygone für Zugmaschine und Sattelauflieger

Abbildung 5.2: Sicherheitspolygone vor und nach dem Abkuppeln

flankierenden Fahraufgaben, führt das Fahrzeug also drei Fahraufgaben auf dem Grundstück aus.

Handelt es sich nicht um das Abkuppeln eines Aufliegers oder Anhängers, sondern um das Abstellen eines Containers, so kann unter Umständen ganz auf die Fahraufgabe verzichtet werden, je nach neuer Position des Containers neben dem Fahrzeug.

5.1.2.2 Teilaufgaben des Ankuppelns

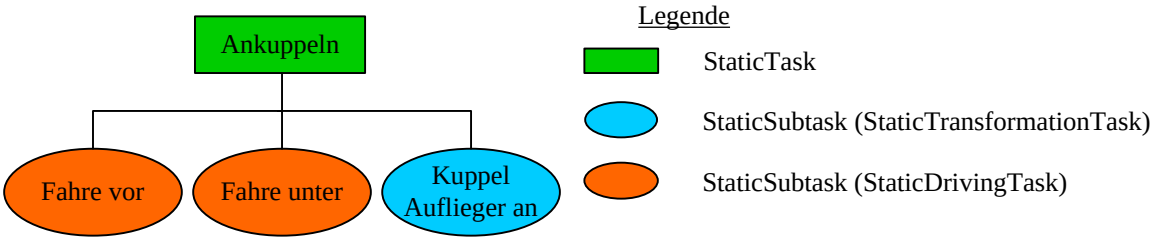


Abbildung 5.3: Teilaufgaben des Ankuppelns

Das Ankuppeln eines Sattelaufliegers oder Anhängers besteht im statischen Aufgabenbaum aus drei Teilaufgaben: Fahre vor den Auflieger, fahre unter den Auflieger, sowie kuppele den Auflieger an. Abbildung 5.3 verdeutlicht diese Einteilung. Anders als beim Abkuppeln, muss dem Ankuppeln keine gesonderte Fahraufgabe außerhalb der Transformationsaufgabe mehr vorausgehen. So kann ein Fahrzeug, von egal welcher Position auf dem Gelände, zu einer Ankuppel-Transformation beordert werden[7]. Ziel der ersten Fahraufgabe ist es, den Traktor oder die Zug-

[7] vorausgesetzt es existiert ein fahrbarer Weg

maschine in die richtige Position vor den Sattelauflieger zu fahren. Dabei spielt es wie gesagt keine Rolle, wo sich der Traktor zu diesem Zeitpunkt auf dem Gelände befindet.
Die zweite Fahraufgabe dient ähnlich dem Abkuppeln der Sicherheitsüberwachung der Fahrzeugkontrolle. Auch in diesem Fall muss die Sicherheitsüberwachung über den kurzen Kuppelvorgang informiert werden, da sie sonst das Unterfahren des Sattelaufliegers durch den Traktor als Kollision erkennen und dies somit durch eine Notbremsung verhindern würde. Daher dient Fahraufgabe 1 als Indikator für die anstehende Transformationsaufgabe und endet knapp vor dem Sattelauflieger. Mit der Beendigung dieser Fahraufgabe, setz die Fahrzeugkontrolle die Kollisionserkennung für den beteiligen Auflieger aus. Somit kann die folgende Fahraufgabe, vom Kupplungspunkt des Traktors auf den Kupplungspunkt des Anhängers, stattfinden. Ist dies geschehen, kann die Transformation ausgeführt werden und mit ihrem Abschluss die Fahrzeugkontrolle die Sicherheit für das gesamte Fahrzeug wieder herstellen.

5.1.2.3 Datenstruktur und Fahrzeugbeschreibungen

Wie in Abschnitt 5.1.1 aufgezeigt wurde, ist das Hauptproblem der Transformationen für den Leitstand die dadurch entstehende Dateninkonsistenz der Datenbanken der einzelnen Schichten. Diese gilt es nun, im Rahmen der Behandlung einer Transformation, zu beseitigen. Der dazu notwendige Berechnungsaufwand hängt ganz von der Struktur und dem Aufbau der einzelnen Datenbanken ab. Dabei wird an dieser Stelle nicht ins Detail gegangen, sondern es werden nur die wesentlichen Modifikationen genannt, um einen Einblick in die Problematik geben zu können. Dazu wird zunächst erläutert, wie die Registrierung eines Fahrzeuges am Leitstand vollzogen wird und welche Module dabei Daten über die Fahrzeuge speichern.

Meldet sich ein neues Fahrzeug über die Nachricht *VIRegisterNewVehicle* am Leitstand an, kommuniziert es zunächst ausschließlich mit dem Fahrzeugmanagement. Diesem teilt es alle relevanten Daten mit. Dort bekommt das Fahrzeug für jedes Fahrzeugteil eine eindeutige ID zugewiesen, sodass ein Sattelschlepper mit Auflieger z. B. die ID 1 für das Zugfahrzeug und die ID 2 für den Anhänger zugewiesen bekommt. Unter diesen IDs sind beide Fahrzeugteile nun fortwährend im Leitstand bekannt. Dies teilt das Fahrzeugmanagement den übrigen Modulen über die Nachricht *INewVehicle* mit und übersendet ihnen die sogenannte *VehicleDescriptionChain*. Sie enthält eine Kette von Fahrzeug-IDs, unter welchen die Fahrzeugteile von nun an leitstandsintern adressiert werden können. Diese Information ist jedoch für die meisten Module nicht

ausreichend, sodass sie weitere Anfragen[8] an das Fahrzeugmanagement stellen, um ihre Angaben über das neue Fahrzeug zu vervollständigen. Dabei besitzt jede Schicht eine andere Sichtweise und somit Datengrundlage für das neu angemeldete Fahrzeug. Während der Aufgabenplanung einfache Angaben über das Fahrzeug[9] vollkommen ausreichen, brauchen Module wie die Detailplanung präzisere Angaben zu den kinematischen Daten für die Trajektorienberechnung. So unterschiedlich die Anforderungen der einzelnen Module an das Wissen über die Fahrzeuge ist, so differenziert müssen diese auch nach einer Transformation aktualisiert werden. Dabei bildet die Wissensgrundlage der einzelnen Module stets die *VehicleDescriptionChain*. Je nach Sichtweise stehen hinter dieser Kette von Fahrzeug-IDs unterschiedliche Angaben. Während der Aufgabenplanung und der Auftragsverwaltung lediglich die bloße Zusammensetzung des Fahrzeuges als Leitstand-IDs vollständig ausreichen, verbergen sich in der Detailplanung hinter jeder ID (und damit jedem Fahrzeugteil) die kinematischen Daten der einzelnen Fahrzeuge.
Wird nun eine Transformation durchgeführt, so gilt es diese Kette, im Falle eines Abkuppelns, in zwei einzelne Ketten aufzubrechen und zwei unterschiedlichen Fahrzeugen zuzuordnen. Die beiden neu entstandenen Fahrzeuge werden daraufhin in der jeweiligen Datenbank abgelegt. Im Falle des Ankuppelns werden umgekehrt zwei Fahrzeugketten zu einer Neuen zusammengefügt. Sollte es bei dieser Modifikation zu Informationsverlusten kommen, so muss das jeweilige Modul über eine gesonderte Nachricht die Information bei der Fahrzeugverwaltung anfragen. Dies ist bei allen Modulen der Fall, die sich die Gesamtlänge eines Fahrzeuges in der Datenbank speichern. Sie kann beim Zusammenfügen einer Zugmaschine mit einem Sattelauflieger nicht durch die Addition der einzelnen Fahrzeuglängen errechnet werden. Die zentrale Position behält im Falle der Transformationen immer das Fahrzeugmanagement. Es verfügt über alle bekannten Daten eines Fahrzeuges und kann somit alle Anfragen der weiteren Module stets beantworten.

In nicht allen Datenbanken konnte ein Sattelauflieger, der ein passives Fahrzeug für das System darstellt, problemlos in die Datenbank eingefügt werden, da die Struktur stets die Angabe eines aktiven Fahrzeuges voraussetzte. Passive Fahrzeugteile wie Anhänger konnten lediglich über die direkte Verbindung zu einem aktiven Fahrzeugteil aufgenommen werden. Daher wurden die Datenbanken diesbezüglich angepasst, sodass alle Module die auf dem Gelände abgestellten passiven Fahrzeugteile verwalten können.

[8] z. B. über die Nachrichten *StaticObjectDataRequest, IVehicleInformationRequest* oder *GeneralVDCLength-Query*

[9] z. B. das Kennzeichen, der Besitzer und die ID

Die Fahrzeugbeschreibungen der Fahrzeugsimulation basierten auf den Vorgaben der EZauto-Bibliothek. Dort sind Fahrzeuge in einer Baumstruktur beschrieben. Auch hier konnten durch die Aufspaltung und das Zusammenfügen der Teilbäume die Transformationen verarbeitet werden.

5.1.2.4 Erweiterung des Fahrzeug-Referenzpunktmodells

Das Ankuppeln eines Sattelaufliegers setzt voraus, dass der Leitstand in der Lage ist, den Terminaltraktor mit seiner hinteren Kupplung an die Position der vorderen Kupplung des aufzunehmenden Anhängers zu manövrieren. Wie in Abschnitt 5.3.1.2 genauer erläutert wird, können die Fahrzeuge nur bestimmte, dem Fahrzeug zugeordnete, Referenzpunkte anfahren. Im Falle einer Transformation sind dies die eben genannten Kupplungspunkte. Um eine Transformation überhaupt zu ermöglichen, mussten diese Referenzpunkte in die Strukturen des Leitstandes aufgenommen und darauf basierende Berechnungen angepasst werden. Ebenso mussten die Beschreibungen der Fahrzeuge in den XML-Dateien um die genaue Angabe der Position der neuen Referenzpunkte erweitert werden, sodass die Kupplungspositionen berechnet und angefahren werden können.

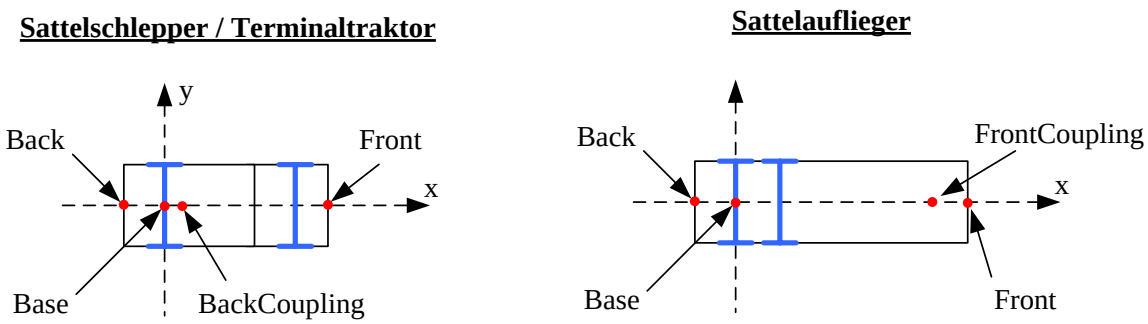


Abbildung 5.4: Erweiterung der Fahrzeug-Referenzpunkte

Abbildung 5.4 zeigt die neu eingefügten Referenzpunkte *BackCoupling* beim Sattelschlepper und Terminaltraktor, sowie *FrontCoupling* beim Sattelauflieger. Wie in der Zeichnung zu erkennen ist, liegt der Kupplungspunkt des Sattelschleppers sehr nahe am Basispunkt des Fahrzeuges in der Nähe der Hinterachse. In Zahlen ausgedrückt liegt der hintere Kupplungspunkt des Sattelschleppers 25 cm vor der Hinterachse und der vordere Kupplungspunkt des Sattelaufliegers 1

m hinter dem Referenzpunkt Front.

5.1.2.5 Interne Abläufe

Bevor nun das An- oder Abkuppeln eines Aufliegers vorgenommen werden kann, muss die Aufgabe über die Aufgabenplanung (*OrderAcceptance*) sowie die Auftragsverwaltung (*OrderAdministration*) zum *Scheduler* in der Terminplanung weitergeleitet werden. Dies kann, je nach Ausprägung des Anwendungsfalles, auf unterschiedliche Art und Weise geschehen.
Jede Transformation wird über zusätzliche Informationen (den sogenannten *AdditionalInformation*) genauer beschrieben. Diese beinhalten alle notwendigen Informationen zur anstehenden Transformation, z. B. die Art der Transformation, das Kennzeichen des an- oder abzukuppelnden Fahrzeugteiles, die vom Leitstand zugewiesene ID, etc. Genauere Angaben hierzu finden sich im Abschnitt 7.6.2.1 des Anhangs. Nun ist es die Aufgabe der Aufgabenplanung und der Auftragsverwaltung, diese Information vollständig der Terminplanung bereitzustellen, sodass diese die Transformation einplanen kann. Das kann auf verschiedene Weisen geschehen.
Im Beispiel von EZsped würde sich eine rein statische Angabe der notwendigen Informationen in der XML-Datengrundlage der Aufgabenplanung anbieten. Dort könnten alle relevanten Daten gespeichert und bei Bedarf sofort an die Auftragsverwaltung weitergeleitet werden. Dies empfiehlt sich im Beispiel von EZrola nicht, da sehr viele Aufträge das System erreichen müssen. Somit sollten diese zur Laufzeit dynamisch generiert werden. Aus diesem Grund übernimmt die Auftragsverwaltung das Erzeugen der AdditionalInformation komplett und leitet nur vollständige Angaben an die Terminplanung weiter.
Das Problem der Transformationsaufgaben für den Leitstand ist, dass deren Ende zeitlich nicht vorhergesagt werden kann, da das Ende lediglich als Information von außen dem System mitgeteilt wird. Dies ist bewusst so implementiert, da es in der Realität ähnlich verlaufen würde. Erst wenn ein Mitarbeiter des Terminals dem System die vollständige Transformation meldet, kann es weitere Abläufe einleiten. Wann diese Meldung den Leitstand erreicht, ist dabei jedoch ungewiss und kann nicht auf die Sekunde genau vorhergesagt werden. Ebenso ist das An- und Abkuppeln von der vorhergehenden Fahraufgabe abhängig, sodass die Planungen erst eingeleitet werden dürfen, wenn das Fahrzeug seine Fahraufgabe beendet hat. Diese Funktionalität übernimmt die Funktion *processOrder* der Auftragsverwaltung. Sie bereitet die von der Aufgabenplanung weitergeleiteten Aufgaben vor und trennt sie in einzelne Teilaufgaben, die sie dem Scheduler übersendet. Geht nun im Falle des Abkuppelns eine Fahraufgabe voraus, so wird diese

einzeln an den Scheduler weitergeleitet. Erst nach dem Ende dieser Aufgabe beginnt die Planung der eigentlichen Transformation.

Wie bereits erwähnt, erfordert das Einplanen der Transformationen vollständige *AdditionalInformation*. Da nicht alle notwendigen Angaben, wie z. B. die leitstandinternen Fahrzeug-IDs, der Aufgabenplanung im Voraus bekannt sein können, muss die Auftragsverwaltung diese Informationen über gesonderte Anfragen an andere Module erfragen. Die beiden wichtigsten Anfragen dabei sind die Frage nach der ID eines Fahrzeuges sowie der aktuellen Position. Dazu übersendet die Auftragsverwaltung die Nachrichten *IObjectIDRequest* sowie *ITargetTransformationPositionRequest* an die Fahrzeugverwaltung. Diese beantwortet in den äquivalenten Nachrichten *IObjectIDAnswer* sowie *ITargetTransformationPositionAnswer* die Anfrage mit den notwendigen Informationen. Aus diesen können nun die vollständigen AdditionalInformation bereitgestellt und an die Terminplanung übersendet werden. Dabei ist die aktuelle Position für die in den Transformationen enthaltenen Fahraufgaben von großer Bedeutung. Im Falle des Abkuppelns wird die aktuelle Position des Zugfahrzeuges erfragt, sodass die Position für die folgende Fahraufgabe, unter dem Auflieger hervor, berechnet werden kann (aktuelle Position + 1,8 m in Ausrichtung geradeaus).
Im Falle des Ankuppelns wird nicht die aktuelle Position des Zugfahrzeuges, sondern die des Sattelaufliegers beim Fahrzeugmanagement erfragt. Sie liefert die Informationen für die dem Ankuppeln vorausgehenden beiden Fahraufgaben. Aus diesen Angaben errechnet sich die erste Position, 1,8 m vor dem Sattelauflieger, und die zweite Position unter dem Sattelauflieger, für die Fahrt vom Kupplungspunkt der Zugmaschine auf den Kupplungspunkt des Sattelaufliegers. Ebenso an dieser Stelle wird, anders als beim Abkuppeln, bereits die Nachricht *SwitchSafetyOffForTransformation* an die Fahrzeugkontrolle versendet. In dieser Nachricht bekommt die Fahrzeugkontrolle alle Informationen über den geplanten Ankuppelvorgang mitgeteilt. Somit setzt sie nach dem Ende der ersten Fahraufgabe die Sicherheit für den beteiligten Sattelauflieger aus. Der Traktor kann nun in der zweiten Fahraufgabe unter den Sattelauflieger fahren und wird nicht von der Fahrzeugkontrolle davon abgehalten.

In der Terminplanung plant der Scheduler alle Teilaufgaben bis zur Transformation ein, d. h. im Falle des Ankuppelns beide Fahraufgaben sowie die Transformation, wogegen beim Abkuppeln lediglich die Transformation geplant werden kann. Alle weiteren Teilaufgaben können zu diesem Zeitpunkt noch nicht eingeplant werden. Zu deren Einplanung muss eine mögliche

Startzeit angegeben werden, über die der Scheduler zur Planungszeit aber noch nicht verfügen kann, da die Dauer der Transformation unbekannt ist.

Nach erfolgreicher Einplanung leitet der Scheduler daraufhin die Transformation mit der Nachricht *ITransformVehicle* mit allen notwendigen Angaben an das Fahrzeugmanagement, die Auftragsverwaltung sowie die externe Kommunikation ein. Nebenher aktualisiert er seine eigene Datenbank in Bezug auf die neu entstandenen Fahrzeuge, sodass er zukünftig darauf zugreifen kann, und trägt sich für die *WaitForMessage* der Beendigung der Transformationsaufgabe (*VITransformationComplete*) ein. Dies ist deshalb notwendig, da der Scheduler, wie schon erwähnt, erst nach dem Ende der Transformation weitere Aufgaben einplanen kann. Erst ab diesem Zeitpunkt sind ihm die weiteren Ausführungszeiten bekannt.
Auf die Nachricht *ITransformVehicle* hin aktualisiert die Auftragsverwaltung bereits ihre Datenbank und die externe Kommunikation wandelt die interne Nachricht *ITransformVehicle*, nach den Konventionen für externe Nachrichten, in *ETransformVehicle* um. Das Fahrzeugmanagement nimmt die Nachricht ebenso entgegen. An dieser Stelle darf es allerdings noch keine Aktualisierung der Fahrzeugdaten vornehmen, da die Transformation gerade im Verlauf ist und das Fahrzeugmanagement stets über aktuelle Daten verfügen muss. Daher nimmt es die Angaben zur Transformation lediglich entgegen und verarbeitet sie erst mit dem Erhalt der Nachricht *VITransformationComplete*. Diese wird nach dem Ende der Transformation vom Fahrzeug selbst versendet.
Das Fahrzeugmanagement informiert nun die virtuelle Umgebung sowie die Fahrzeugkontrolle durch die Nachricht *IModifiedVehicle* über die anstehende Transformation. Beide aktualisieren daraufhin ihre Datenbanken. Im Falle eines Abkuppelvorganges reagiert die Fahrzeugkontrolle selbstständig auf die anstehende Sicherheitsproblematik. Sie übernimmt alle notwendigen Daten, behält sich die Aufnahme des Aufliegers in die vollwertige Sicherheitsüberwachung jedoch bis zum Ende der nächsten Fahraufgabe der Zugmaschine vor. Dies wird ihr durch die Nachricht *IDrivingJobComplete* signalisiert. Im Falle des Ankuppelns wartet sie ebenso auf das Beenden der ersten Fahraufgabe kurz vor den Sattelauflieger, sodass sie für die folgende Fahraufgabe die Sicherheitsüberwachung für den Sattelauflieger aussetzen kann.

Das Fahrzeugmanagement muss ebenso das von der Transformation betroffene Fahrzeug benachrichtigen. Zu diesem Zweck sendet es die Nachricht *VITransformation* über die Fahrzeugkommunikation an das Fahrzeug. In dieser Nachricht ist die neue, nach der Transformation gültige,

Fahrzeugbeschreibung des Fahrzeuges enthalten. Dies ist ein Sonderfall rein für die Simulation, da ein reales Fahrzeug nicht über die anstehende Transformation informiert werden muss, da jedes Fahrzeugteil seine eigene Beschreibung selbst kennen sollte.
Ist dies geschehen, warten der Scheduler sowie das Fahrzeugmanagement auf die Beendigung der Transformation, die über die Nachricht *VITransformationComplete* signalisiert wird. Diese wird in der aktuellen Implementierung vom Fahrzeug nach 15 s Wartezeit versendet. Diese Zeitspanne ist für eine reale Trennung natürlich viel zu kurz gewählt, doch erschien sie für eine Simulation im Zeitraffer angebracht. Die Nachricht wird deshalb vom Fahrzeug versendet, da dies auch der Realität entsprechen könnte. So wäre es möglich, dass ein Terminalmitarbeiter den Ablauf der Trennung kontrolliert und nach Beendigung die Nachricht über einen Knopf am Fahrzeug sendet. Erreicht diese nun das Fahrzeugmanagement, so aktualisiert es seine schon vorbereiteten Daten und die Transformation ist damit beendet. Der Scheduler nutzt die Nachricht, um nun weitere anstehende Aufgaben für das transformierte Fahrzeug einzuplanen.

5.1.3 Visualisierung

Wie auch im Leitstand, so waren bei der von Jörg Sesterhenn entwickelten Visualisierung[10] keine Fahrzeug-Transformationen berücksichtigt. Diese galt es nun, analog zum Leitstand, in die Visualisierung zu integrieren.

Ebenso wie die virtuelle Umgebung, ist die Visualisierung als externes Modul implementiert, sodass sie lediglich über den externen Ereignisverteiler mit dem Leitstand kommunizieren kann. Aus diesem Grund mussten die zu einer Transformation notwendigen Nachrichten über diesen an die Visualisierung weitergeleitet werden. Dazu wurden die beiden internen Nachrichten *ITransformVehicle* und *VITransformationComplete* nach den Konventionen für externe Nachrichten in *ETransformVehicle* sowie *ETransformationComplete* umgesetzt. Beide werden von der Visualisierung empfangen und ausgewertet. Alle weiteren internen Nachrichten, in Bezug auf eine Fahrzeug-Transformation, waren für die Modifikation der Visualisierung nicht notwendig.
Die Verarbeitung der beiden Nachrichten innerhalb der Visualisierung verhält sich nun wie folgt. Der Beginn einer Transformation wird über die Nachricht *ETransformVehicle* registriert und alle notwendigen Daten, zu der gerade ablaufenden Transformation, entgegen genommen, jedoch noch nicht ausgewertet. Der Grund dafür sind die vom Leitstand empfangenen dynamischen

[10] vgl. [Ses]

Fahrzeugdaten jedes Fahrzeugteils.

Um die Fahrzeugbewegungen darstellen zu können, ist die Visualisierung auf periodische Fahrzeugdaten in festen Abständen angewiesen. So kann sie aus den Informationen, dass sich ein Fahrzeug zum Zeitpunkt t_1 an Position (x_1, y_1) und zum Zeitpunkt t_2 an Position (x_2, y_2) befindet, die Bewegung des Fahrzeuges zwischen diesen beiden Punkten interpolieren. Dazu empfängt die Visualisierung die vom Leitstand gesendeten Fahrzeugdaten aller aktiven sowie der damit verbunden passiven Fahrzeugteile. Rein passive Fahrzeugteile dagegen, wie z. B. ein abgestellter Sattelauflieger, versenden keine periodischen Fahrzeugdaten. Nun gilt es die Synchronität zwischen Leitstand und Visualisierung aufrecht zu erhalten, sodass die Trennung der Fahrzeuge innerhalb der Visualisierung zum gleichen Zeitpunkt vollzogen wird, wie es auch im Leitstand geschieht. Wäre dies nicht gewährleistet, so würde die Visualisierung dynamische Fahrzeugdaten empfangen, könnte diese aber nicht zuordnen, da sie für rein passive Fahrzeugteile keine solchen erwarten würde.

Aus diesem Grund wird erst mit dem Ende der Transformation, das durch die empfangene Nachricht *ETransformationComplete* signalisiert wird, die Transformation in die Datenbank der Visualisierung übernommen.

An diesem Beispiel wird deutlich, wie flexibel bestehende externe Module an einen Anwendungsfall angepasst, und um neue Funktionalitäten erweitert werden können. Durch die strikte Modularisierung, sowie die hohe Abstraktion der Nachrichten von rein internen Datenstrukturen, sollten dazu nur die folgenden Punkte berücksichtigt werden müssen:

- Erweiterung der Kommunikation durch neue Nachrichten
- Verarbeitung dieser Nachrichten innerhalb des externen Moduls

Alle weiteren Details zu den Modifikationen an der Visualisierung, sowie die in dieser Arbeit gemachten Erfahrungen im Umgang mit dieser, finden sich im Anhang in Abschnitt 7.7 auf Seite 226.

5.2 Modul zur Simulation des Gleises

In diesem Abschnitt wird das Modul, das die Abläufe im Gleisbereich des Terminals simuliert, beschrieben. Es handelt sich dabei um ein eigenständiges externes Modul, das über den externen Ereignisverteiler an den Leitstand angebunden ist. Aufgrund seines Aufgabengebietes trägt es den Namen *SimulatedTrack*.

Das RoLa-Terminal Regensburg verfügt derzeit nur über ein einziges Gleis, das für den Verkehr der Rollenden Landstraße genutzt wird. Aus Sicht dieses Moduls wird es über zwei Referenzpunkte auf dem Hof definiert: *FirstPoint* und *LastPoint*. Diese beiden Punkte werden (zusammen mit anderen Angaben) in einer Konfigurationsdatei (*config.xml*) an das Modul übergeben. Die Gerade durch diese beiden Punkte stellt das Gleis an sich dar. Der Zug kommt immer zwischen First- und LastPoint zum Stehen. Er kommt dabei aus Richtung des Punktes LastPoint, sodass Fahrzeuge, die auf den Zug auffahren, als Fahrtrichtung ($LastPoint - FirstPoint$) besitzen. Der genaue Haltepunkt wird für jeden Zug zufällig bestimmt, da auch in der Realität der genaue Punkt nicht vorherzusagen ist. Auf diese Weise soll der Eindruck der Dynamik erhöht werden. Anhand des Haltepunktes werden die Positionen der einzelnen Waggons berechnet. Die Ausmaße eines Waggons werden ebenfalls in der Konfigurationsdatei übergeben. Das simulierte Gleis initiiert außerdem die Erzeugung von Fahrzeugen auf dem Zug, im Einfahrtsbereich sowie die Erzeugung der Abholer. Letztere fällt logisch gesehen nicht in den Aufgabenbereich dieses Moduls, muss jedoch von einem externen Modul übernommen werden. Da dieser Vorgang abhängig von den Ankünften der Züge ist, schien dieses Modul am geeignetsten, diese Aufgabe zu übernehmen. Zu diesem Zweck befindet sich in der Konfigurationsdatei der Wert *secondsTillCollection*, der angibt, wie vielen Sekunden nach der Ankunft eines Zuges die Abholer erzeugt werden sollen.

Nach dem Start des Moduls werden die Konfiguration und der Zugfahrplan (*timetable.xml*) eingelesen. Der Zugfahrplan besteht aus einer Aneinanderreihung von Einträgen, die jeweils einen speziellen Zug betreffen. Für jeden Zug sind folgende Angaben zu machen:

- ID des Zuges
- Zeitpunkt der Ankunft
- Zeitpunkt der Abfahrt
- Anzahl der Waggons, aus denen der Zug besteht

- Anzahl der Fahrzeuge auf dem Zug
- Anzahl der Fahrzeuge, welche - von der Straße kommend - mit diesem Zug abfahren sollen

In regelmäßigen Abständen von etwa einer Sekunde wird geprüft, ob die Ankunfts- oder Abfahrtszeit eines Zuges oder der Zeitpunkt zum Erzeugen der Abholer überschritten wurde. Bei einer Ankunft müssen die Auftragsverwaltung und die virtuelle Umgebung informiert werden. Die Auftragsverwaltung benötigt folgende Daten:

- die Abmessungen des Zuges, um später die Zielposition der Fahrzeuge, die auf diesen Zug fahren sollen, berechnen zu können
- die Anzahl der Fahrzeuge auf dem Zug, um entscheiden zu können, wann alle Fahrzeuge den Gleisbereich verlassen haben; zu diesem Zeitpunkt können wieder Fahrzeuge auf den Zug auffahren
- die ID des nächsten ankommenden Zuges, um Fahrzeuge, die auf diesen Zug warten, schon vor dessen Ankunft in die Wartespuren zu ordern

Diese Daten werden in dem Ereignis *ETrainReady* gespeichert und versendet. Die externe Kommunikation wandelt dieses in das interne Ereignis *ITrainReady* um, das dann von der Auftragsverwaltung entgegengenommen wird. Die virtuelle Umgebung muss informiert werden, da sie für die Erzeugung der Fahrzeuge verantwortlich ist. Sie benötigt nur die Position der Waggons und die Anzahl an Fahrzeugen, die sich auf dem Zug befinden sollen. Diese Daten werden mit dem Ereignis *ECreateVehiclesOnTrain* übergeben. Zusätzlich wird hier auch der Zeitpunkt berechnet, zu dem später mit der Erzeugung der Kundenfahrzeuge zum Abholen der Auflieger, die mit diesem Zug angekommen sind, begonnen werden soll.

Wird die Abfahrtszeit des sich momentan auf dem Gleis befindlichen Zuges überschritten, sollte der Zug wieder aus dem System entfernt werden. Das darf jedoch erst passieren, wenn alle Fahrzeuge, die das Gelände mit dem Zug verlassen sollen, auf diesem angekommen sind. Hierüber wird das Modul von der Auftragsverwaltung mithilfe des Ereignisses *ETrainReadyForDeparture* (leitstandsintern *ITrainReadyForDeparture*) informiert. Die Abmessungen des nächsten Zuges werden berechnet[11]. Die virtuelle Umgebung wird anhand des Ereignisses *ECreateVehiclesOnStreet* angewiesen, so viele Fahrzeuge für den nächsten Zug in der Straßeneinfahrt zu erzeugen,

[11] wie bereits beschrieben anhand eines zufälligen Haltepunktes zwischen den beiden Referenzpunkten des Gleises

wie im Zugfahrplan angegeben. Die OrderAdministration wird mithilfe des Ereignisses *ETrainLeft* über die Abfahrt informiert, damit sie die Abmeldung aller auf dem Zug befindlichen Fahrzeuge einleiten kann.

Die Gleissimulation fängt keine Fehler im Fahrplan ab und ist deshalb auf dessen Konsistenz angewiesen. Es muss also z. B. immer darauf geachtet werden, dass keine Ankunft in einen Zeitabschnitt fällt, in dem sich noch ein Zug auf dem Gelände befindet. Ebenso muss in der Konfigurationsdatei darauf geachtet werden, dass die Wartezeit bis zum Abholen der Auflieger so groß gewählt wird, dass alle Lkw ihren Stellplatz erreicht haben und die Auflieger abgekuppelt sind, wenn die Abholer erzeugt werden. Ansonsten würde das System den Auflieger nicht finden und das Szenario müsste abgebrochen werden. Eine Übersicht über die Abfolge der Ereignisse in Zusammenhang mit diesem Modul inklusive einer Übersicht über die Kommunikation mit anderen Modulen ist im Anhang in Abschnitt 7.6.1 auf Seite 219 zu finden.

5.3 Auftragsverwaltung

Die Auftragsverwaltung (OrderAdministration) ist der Teil des Leitstandes, der die meisten anwendungsfallspezifischen Eigenschaften enthält. Soll der Leitstand an einen anderen Anwendungsfall angepasst werden, so müssen hier (neben den externen Modulen) die meisten Veränderungen vorgenommen werden. Oder anders gesagt: Der Allgemeine Leitstand unterscheidet sich vom spezifischen Leitstand am meisten in der OrderAdministration. Sie ist in einer hierarchischen Sicht auf den Leitstand das letzte Modul, das noch die Bedeutung der Aufträge versteht, die auf dem Gelände durchgeführt werden. Während der Verarbeitung durch den Scheduler gehen alle anwendungsfallspezifischen Informationen verloren und es wird nur noch zwischen Warte-, Transformations- und Fahraufgaben unterschieden.

5.3.1 Allgemeine Erweiterungen

Die Erweiterungen und Änderungen, die an der Auftragsverwaltung (*OrderAdministration*) vorgenommen wurden, lassen sich unterteilen in allgemeine und anwendungsfallspezifische Erweiterungen. Zu den allgemeinen Erweiterungen zählen jene, die sich dem allgemeinen Leitstand[12] zurechnen lassen, also für jeden denkbaren Anwendungsfall nötig oder einsetzbar sind.

[12] siehe Abschnitt 4.1 auf Seite 59

5.3.1.1 Umgang mit den Aufträgen

Die eigentliche Aufgabe der OrderAdministration ist es, wie der Name schon sagt, Aufträge zu verwalten. Sie speichert daher zu jedem aktiven Fahrzeug einen Überblick über alle von ihm durchzuführenden und bereits durchgeführten Aufträge. Dieser Gesamtüberblick wird im Folgenden mit *Order* abgekürzt. Zu Beginn dieser Arbeit war eine Order ein sehr starres Objekt. Es wurde erzeugt aus den Informationen, die die OrderAdministration bei der Anmeldung des betreffenden Fahrzeuges vom externen Modul *OrderAcceptance* erfragt hat und blieb von diesem Zeitpunkt an, bis auf kleine Änderungen, gleich. Diese Order wurde in abgeänderter Form, aber dafür komplett, dem Scheduler übergeben. Das Hinzufügen neuer Tasks war in der bestehenden Form nicht möglich. Im Zuge dieser Arbeit wurde das Konzept der OrderAdministration, genauer gesagt des Zusammenarbeitens der OrderAdministration mit dem Scheduler, leicht abgeändert.

Der Aufgabenbereich des Schedulers soll so klein wie möglich gehalten werden. Soweit es möglich ist, sollen alle Entscheidungen vor der Übergabe an ihn getroffen und alle nötigen Daten vorher gesammelt werden[13]. Aufseiten der OrderAdministration bedeutet das, dass nicht die komplette Order verarbeitet und an den Scheduler übergeben werden kann, sondern nur so viele Teilaufgaben, wie zu diesem Zeitpunkt komplett geplant werden können. Angenommen, ein Fahrzeug habe erst eine Fahraufgabe zu erledigen und soll anschließend einen bestimmten Auflieger ankuppeln, dessen Position sich noch ändern könnte. In diesem Fall behält die OrderAdministration den Ankuppelauftrag so lange zurück, bis sie die Position des Aufliegers bestimmten kann. Außerdem kann es ausdrücklich gewünscht sein, dass Tasks erst zu dem Moment geplant werden, wenn sie auch ausgeführt werden sollen. Das ist beispielsweise bei Parkaufgaben wichtig. Angenommen, ein Fahrzeug kommt mit dem Zug an. Der Auflieger soll für den Kunden zur Abholung auf den Abholerparkplätzen abgestellt werden. Danach soll sich der Traktor in eine Traktorspur begeben. Der Zeitraum, bis der Traktor die Fahrt zur Traktorspur beginnt, kann relativ groß sein. Währenddessen kann sich der Zustand in den Spuren ändern. Möglicherweise ist sogar erst dann ein Platz in einer Spur frei. Daher ist es sinnvoll, Parkaufgaben zeitlich möglichst nahe an deren Durchführung zu planen und deshalb mit den Berechnungen zu warten bis die vorherigen Aufgaben ausgeführt wurden. Die Planungsgrundlage bei Parkaufgaben ist immer der Ist-Zustand der jeweiligen Parkplätze, das bedeutet, dass keine Informationen über die zukünftigen Entwicklungen auf Parkflächen vorliegen.

[13] vgl. Abschnitt 5.6

Für solche Fälle wurde den Knoten der statischen Aufgabenstruktur[14] ein Attribut (*waitForCompletionOfPredecessor*) hinzugefügt. Dabei handelt es sich um einen Wahrheitswert, der angibt, ob diese Aufgabe verplant werden soll, bevor die vorangehende Aufgabe abgeschlossen ist. Eine Order behält sich daher Informationen zu Tasks, die noch nicht bearbeitet werden konnten und Informationen über Tasks, die bereits für den Scheduler vorbereitet sind, vor. Für jede Order gibt es mehrere definierte Zustände. Sie kann aktiv oder inaktiv sein[15], sie kann sich aber auch in einem Wartezustand befinden[16]. Eine inaktive Order muss angestoßen werden, wenn neue Aufgaben für sie ankommen, während eine aktive Order nach dem Hinzufügen möglicher neuer Aufgaben nicht weiter behandelt werden darf. Die neuen Tasks werden dann automatisch bei Beendigung der älteren Aufgaben bearbeitet. Eine Order, die sich gerade im Wartezustand befindet, wartet auf Daten, die zur Bearbeitung der nächsten Aufgaben nötig sind. Das ist beispielsweise der Fall, wenn die Position eines anzukuppelnden Aufliegers angefragt wird[17]. In diesem Zustand darf eine Order nicht weiter bearbeitet werden. Der Wartezustand muss erst beendet werden.

Zu Beginn werden so viele Tasks wie möglich an den Scheduler geschickt. Sind alle diese Aufgaben abgeschlossen, wird die OrderAdministration darüber informiert und sie versucht die nächsten Aufgaben zu verarbeiten. Der Scheduler versendet zu diesem Zweck die Nachricht mit dem Namen *IOrdersCompleted*.

Diese Änderung der Struktur ermöglichte auch das Hinzufügen neuer Aufgaben zu einer Order, was für den Anwendungsfall dieser Arbeit unumgänglich war. In diesem Fall werden die neuen Aufgaben einfach hinten an die Liste der Aufgaben angehängt, welche noch nicht verarbeitet werden konnten. Wurden alle alten Aufgaben bereits komplett ausgeführt, so muss zusätzlich noch die Verarbeitung der neuen Aufgaben ausgelöst werden, was ansonsten automatisch spätestens mit der Vollendung der letzten alten Aufgabe geschehen wäre.

Zusätzlich wurde noch eine Möglichkeit geschaffen, Tasks anzunehmen, die nicht zur Erfüllung der eigentlichen Aufgaben des Fahrzeuges dienen, sondern der Organisation des Hofes zuträglich sind. Solche Aufgaben werden nicht in der Order hinten an die noch nicht verarbeiteten Tasks

[14] nähere Informationen zur statischen Aufgabenstruktur sind in Abschnitt 5.4 zu finden
[15] angezeigt durch das Attribut *m_running*
[16] wenn das Attribut *m_waiting* den Wert *true* hat
[17] vgl. Abschnitt 5.1

angehängt, sondern sie werden direkt verarbeitet und an den Scheduler weitergeschickt. Sie werden zur Ausführung gebracht, wenn das Fahrzeug seine bisher übertragenen Tasks beendet hat, also vor den Tasks, die vielleicht noch unverarbeitet in der Order auf die Verarbeitung warten. Zu dieser Kategorie ist das Aufrücken in den Traktorspuren zu zählen. Angenommen, der Traktor hätte von Beginn an eine feste Auftragsstruktur. Diese könnte dann so aussehen, dass er zuerst einen Auflieger zur Abholung abstellt, dann in eine Traktorspur fährt und im Anschluss einen bestimmten Auflieger abholt, auf dessen Ankunft jedoch noch gewartet werden muss. Entsteht eine Lücke in der Spur, in der dieser Traktor steht (etwa weil ein vor ihm stehender Traktor einen neuen Auftrag erhalten hat), so muss die Fahraufgabe zum Aufrücken direkt durchgeführt werden, auch wenn noch andere offene Aufgaben anstehen. Im vorliegenden Projekt wird diese neu geschaffene Schnittstelle von der Parkplatzverwaltung[18] genutzt. Zu dieser Kategorie von Aufgaben würden aber auch dynamisch eingefügte Tank- oder Reparaturaufgaben zählen, die jedoch in der Spezialisierung EZrola bisher nicht vorgesehen sind. Damit gibt es also zwei Möglichkeiten, Aufträge für ein Fahrzeug zu übergeben, die sich in der Art der Aufträge unterscheiden. Soll ein Auftrag vergeben werden, der allen anderen hintenangestellt werden muss, so muss eine *IOrder* an die OrderAdministraion geschickt werden. Soll hingegen eine Aktion direkt ausgeführt werden, so muss ein sog. *DynamicOrderRequest* versendet werden. Bei möglichen Weiterentwicklungen des Leitstands müsste beim Empfang eines DynamicOrderRequests jedoch zuerst geprüft werden, ob die Ausführung der betreffenden Aktion momentan möglich ist. Eine Tank- oder Wartungsaufgabe ist beispielsweise unmöglich, während sich das Fahrzeug mitten in einer Parkspur befindet.

5.3.1.2 Fahrzeugdaten und Referenzpunktumrechnung

Wie viele andere Module auch, besitzt die OrderAdministration ihr eigenes Bild eines Fahrzeuges. So besteht ein Fahrzeug aus der Sicht der OrderAdministration unter anderem aus dessen Kennzeichen, der zugehörigen Order, dem Aufbau des Fahrzeuges und einigen weiteren Daten. Diese Daten müssen immer auf dem aktuellen Stand gehalten werden. Da im Verlauf dieser Arbeit das Durchführen von Transformationsaufgaben implementiert wurde, muss bei jeder Transformation die Fahrzeugbeschreibung aktualisiert werden. Auch musste das Feld *Länge* in die Beschreibung eines Fahrzeuges mit aufgenommen werden, da diese für das Parken[19] benötigt wird.

[18] vgl. Abschnitt 5.5
[19] vgl. Abschnitt 5.5

Des Weiteren sind einige Attribute wegen der verschiedenen Referenzpunkte der Fahrzeuge vorgesehen. Dies wurde jedoch nur als „Notlösung“ implementiert. Dabei geht es darum, dass ein Zielpunkt auf dem Hof mit verschiedenen Teilen am Fahrzeug angefahren werden kann. Die Wichtigsten sind *Front* (das Ziel wird mit dem vordersten Punkt der Längsachse angefahren) und *Back* (das Ziel wird mit dem hintersten Punkt der Längsachse angefahren). Standardmäßig werden Ziele immer mit *Base* angefahren, was dem Mittelpunkt der Hinterachse des Fahrzeuges entspricht. Im bisherigen Konzept sollte die Umrechnung auf einen anderen Bezugspunkt zu einem sehr späten Punkt der Planung durchgeführt werden. Diese Berechnung geschieht jetzt in der OrderAdministration, da dies hier vergleichsweise unkompliziert ist, vorausgesetzt die Lage des Referenzpunktes im Vergleich zum Basispunkt ist bekannt. In Abbildung 5.5 sind verschiedene Referenzpunkte in Bezug zum Basisreferenzpunkt dargestellt. Das Ziel ist es, in Modulen, die

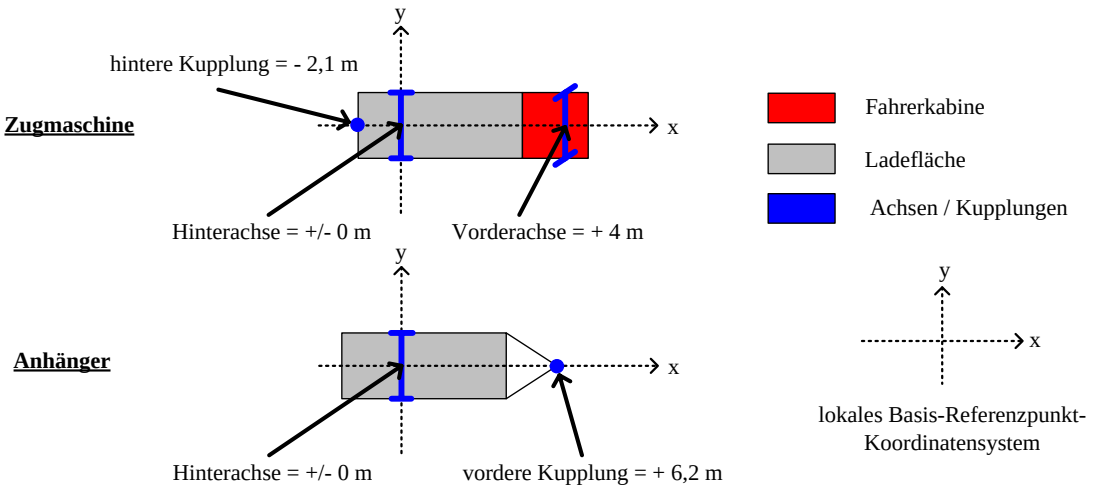


Abbildung 5.5: Referenzpunkte in Relation zum Basisreferenzpunkt

in der hierarchischen Sicht unter der OrderAdministration stehen, nicht mehr zwischen verschiedenen Referenzpunkten unterscheiden zu müssen. Wird eine Fahraufgabe vergeben, bei der der Zielpunkt mit einem anderen Referenzpunkt als *Base* erreicht werden soll, so wird mithilfe der in der Fahrzeugbeschreibung gespeicherten Referenzpunktabhängigkeiten berechnet, auf welchen Punkt das Fahrzeug mit *Base* fahren müsste, damit der gewählte Referenzpunkt den Zielpunkt erreicht. So erreichen nur Fahraufträge mit Referenzpunkt *Base* den Scheduler. Diese Lösung ist allerdings nur ansatzweise implementiert. Die Lage der Referenzpunkte wird nicht für jedes Fahrzeug getrennt angefragt und im Falle einer Transformation auch nicht aktualisiert. Für den Referenzpunkt *Front* ist ein Standardwert eingetragen, was in diesem Fall ausreichend ist, weil

momentan alle auf dem Hof befindlichen Fahrzeuge den gleichen Abstand zwischen *Front* und *Base* haben. Dasselbe gilt für *Back*. Hier ist zu beachten, dass momentan nur Traktoren Punkte mit *Back* anfahren können. Ein kompletter Lkw würde einen Punkt mit diesem Referenzpunkt nicht korrekt anfahren.

Zur Veranschaulichung der Umrechnung ist in Listing 5.1 auf der nächsten Seite ein Auszug aus der Funktion *correctReferencePoint(...)* der *OrderAdministration* zu finden. Dort werden die Berechnungen für den Referenzpunkt „*Front*“ durchgeführt. Zuerst muss ein neues Objekt für die Zielposition (*vehiclePos*) erzeugt werden, da die vorhandene Position von jedem Fahrzeug mit der gleichen Teilaufgabe referenziert wird. Ohne die Erzeugung des neuen Objektes würden die Berechnungen, die an dieser Stelle durchgeführt werden, Auswirkungen auf alle Fahraufgaben haben, denen die gleiche statische Teilaufgabe zugrunde liegt. Der Referenzpunkt wird auf „*Base*“ gesetzt, den einzigen Referenzpunkt, den die Routenplanung in dieser Implementation verstehen muss. Die Weiterleitung des Referenzpunktes wäre so nicht weiter notwendig und könnte aus den Strukturen der unter der Auftragsverwaltung liegenden Module gelöscht werden. Die Fahrzeugausrichtung kann aus dem ursprünglichen Auftrag kopiert werden. Lediglich die Zielposition muss neu berechnet werden. Diese liegt jedoch auf der Geraden, die durch die Fahrzeugausrichtung festgelegt wird, und ist abhängig vom Abstand zwischen der Hinterachse des Fahrzeuges und dessen Vorderkante[20]. Aus diesen Daten wird ein „*offset*“ berechnet. Für die neue Zielposition wird dieser Wert von der alten Zielposition abgezogen (vgl. Abbildung 5.6). Ein

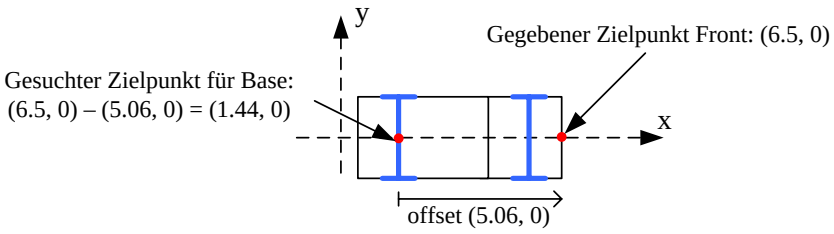


Abbildung 5.6: Referenzpunktberechnung

Zeiger auf die neue Fahrzeugposition wird in der Fahraufgabe (*subtask*) gesetzt. Die Berechnung ist damit abgeschlossen. Für andere Referenzpunkte sieht die Berechnung ähnlich aus. Liegt der gewünschte Referenzpunkt nicht auf der Mittelachse des Fahrzeuges, muss der alte Zielpunkt auf zwei Achsen verschoben werden. Die einzige Voraussetzung für diese Methode ist, wie oben bereits beschrieben, die genaue Kenntnis der Lage der Referenzpunkte. Hierfür müssen neue

[20] Da *Base* auf der Hinterachse und *Front* auf der Vorderkante des Fahrzeugs liegt

Nachrichten eingeführt werden, welche die OrderAdministration über diese Daten informiert.

Listing 5.1: Umrechnung der Zielkoordinaten für Referenzpunkt „Front“

```
[...]
else if(subtask->getTargetPosition()->getReferencePoint()
 == ReferencePoint::Front)
{
        vehiclePos = new VehiclePosition();
        vehiclePos->referenzEntfernen();
        vehiclePos->setReferencePoint(ReferencePoint::Base);
        vehiclePos->setOrientation(subtask->getTargetPosition()->
         getOrientation().normalize());
        Point2D offset = vehiclePos->getOrientation() *
         vehicle->getOffsetFront();
        vehiclePos->setPosition(subtask->getTargetPosition()->
         getPosition() - offset);
        subtask->setTargetPosition(vehiclePos);
}
[...]
```

5.3.2 Anwendungsfallspezifische Erweiterungen

Die in Abschnitt 5.3.1 beschriebenen Erweiterungen betreffen den Allgemeinen Leitstand. In den folgenden Abschnitten werden die Erweiterungen der OrderAdministration beschrieben, die dem besonderen Anwendungsfall des „RoLa-Terminals“ zuzuordnen sind und z. B. für EZsped keinen Nutzen gehabt hätten. Einige Änderungen dieser Kategorie betreffen die Umgebung des Leitstandes und gehören deshalb eigentlich nicht in die Auftragsverwaltung. Bei einer Portierung in eine reale Umgebung oder einer Erweiterung der Umgebungssimulation derart, dass die betroffenen Elemente durch externe Module übernommen werden, müssen diese wieder aus dem Modul OrderAdministration entfernt werden. Für ein besseres Verständnis der Abfolge der Ereignisse und der Kommunikation zwischen den Modulen OrderAdministration, SimulatedTrack und VirtualEnvironment befinden sich im Anhang einige Erläuterungen[21].

[21] siehe Abschnitt 7.6.1

5.3.2.1 Auffahren auf den Zug

Wie bereits erwähnt, betreffen einige Erweiterungen Aufgaben, die in der Realität nicht Aufgabe des Leitstandes wären. Dazu gehören vor allem Entscheidungen, wann bestimmte Vorgänge auszulösen sind. So müssen zu einem bestimmten Zeitpunkt die Fahrzeuge, die sich in den Wartespuren befinden, auf den Zug beordert werden. Zuerst ist zu überlegen, wie dies in der Realität funktionieren würde. Es wäre möglich, dass es zum Zeitplan gehört, der für jeden Zug existieren muss, damit würde also die Zeit vorher feststehen, oder aber ein Mitarbeiter entscheidet wann die Fahrzeuge tatsächlich losfahren sollen. Es wäre auch möglich, das Auffahren durch andere Ereignisse auf dem Hof anzustoßen. In diesem Fall ist zu entscheiden, welches Ereignis dafür geeignet ist. Wichtig ist, dass der Gleisbereich frei ist. Dies ist gegeben, wenn alle Fahrzeuge vom Zug ihre erste Fahraufgabe auf dem Hof erfüllt haben. Die Entscheidung fiel zu Gunsten dieser letztgenannten Variante, auch deshalb, weil sie zeitsparend ist. Soll das Auffahren anders angestoßen werden, so lässt sich die OrderAdministration ohne große Änderungen anpassen[22]. Da zu Testzwecken die Vorgänge in den Szenarien zeitlich besonders eng angeordnet sein sollen, wurde nicht gewartet, bis jeder Lkw seine erste eigentliche Zielposition erreicht hat. Statt dessen wurde für jeden vom Zug kommenden Lkw eine Extraaufgabe eingeführt, die natürlichsprachlich *Verlasse den Zug* lauten könnte. Sie führt die Fahrzeuge auf einen Punkt außerhalb, aber in der Nähe, des Gleisbereiches.

Haben alle Fahrzeuge diesen Punkt passiert, so ist das Gleis frei und die Fahrzeuge aus der Wartespur können auffahren. Zu diesem Zweck werden, den äußeren Bedingungen zufolge, abwechselnd normale Fahrzeuge und Fahrzeuge mit Überlänge verladen. Dazu werden jeweils Schnittstellen genutzt, welche die Parkplatzverwaltung zur Verfügung stellt. Dabei handelt es sich um die beiden Methoden *getWaitingVehicleOverlength()* und *getWaitingVehicleNormal()*. Über diese bekommt die OrderAdministration IDs von Fahrzeugen, die dann mit den entsprechenden Fahraufträgen versehen werden. Ist kein derartiges Fahrzeug mehr vorhanden, so wird ein Fehlercode (*-1*) zurückgegeben. Der Zielpunkt der Fahraufträge wird durch die Lage der einzelnen Waggons bestimmt. Diese Daten erhält die OrderAdministration von jedem Zug, wenn dieser eingefahren ist. Absender ist das externe Modul, welches das Gleis simuliert[23]. Sind keine Fahrzeuge mit Überlänge mehr vorhanden, so können die restlichen Lkw nacheinander die folgenden Waggons einnehmen. Sind dagegen keine normalen Fahrzeuge übrig, sondern nur noch

[22] Die Funktion *leadVehiclesOnTrain()* muss nur durch ein anderes Ereignis aufgerufen werden

[23] Die Nachricht heißt intern *ITrainReady*. Nähere Information sind in Abschnitt 5.2 zu finden

solche mit Überlänge, so muss zwischen diesen immer ein Waggon frei bleiben. Die Anzahl der Fahrzeuge, die verladen werden können, ist bei einer großen Anzahl an Lkw mit Überlänge demnach bis zu 50 Prozent geringer, als bei gleichmäßiger Auslastung oder Überzahl von normalen Fahrzeugen. Stehen nicht ausreichend Waggons zur Verfügung, so bricht die Methode unter Ausgabe eines Fehlers ab.

Die OrderAdministration merkt sich die ID der Fahraufgabe des letzten Fahrzeuges aus der Wartespur, um sagen zu können, wann der letzte Lkw seinen Platz auf dem Zug erreicht hat. Bei Vollendung dieser Fahraufgabe wird eine Nachricht (*ITrainReadyForDeparture*) an die Simulation des Gleises geschickt, die besagt, dass der Zug aus der Sicht der OrderAdministration jetzt bereit zur Ausfahrt ist. Sobald auch die planmäßige Abfahrtszeit des Zuges erreicht ist, wird dieser das Gelände verlassen.

Verlässt der Zug das Gelände, so wird vom zuständigen Modul eine Nachricht verschickt, die bei der OrderAdministration als *ITrainLeft* eingeht. In diesem Moment wird die Löschung aller Fahrzeuge veranlasst, die auf den Zug aufgefahren sind. Zu diesem Zweck wird für jedes Fahrzeug die Nachricht *IVehicleReadyToDisconnect* verschickt. Dafür hat sich die OrderAdministration bereits vorher die IDs aller Fahrzeuge gemerkt, die auf den Zug beordert worden sind.

5.3.2.2 Fahrzeuge für einen späteren Zug parken

Sind alle Fahrzeuge auf den Zug aufgefahren, so können die Fahrzeuge, die auf den nächsten planmäßigen Zug warten und daher momentan auf fest dafür vorgesehen Parkplätzen stehen, in die Wartespuren fahren. Auf diese Weise werden Parkplätze für die Lkw, die mit dem nächsten Zug kommen, frei und es muss keine Gasse frei gehalten werden, auf der die Fahrzeuge von den Parkplätzen aus direkt in den Gleisbereich fahren können. Im gleichen Moment, in dem der Zug von der OrderAdministration zur Abfahrt freigegeben wird, kann die Vergabe dieser Fahraufträge durch den Aufruf der Methode *leadParkingVehiclesOnWaitingLane()* eingeleitet werden.

Fahrzeuge, die auf einen späteren Zug warten sollen, müssen als Auftrag einen Verweis auf die statische Aufgabe bekommen, die dem dafür vorgesehenen Parkbereich entspricht. Dem Auftrag müssen außerdem noch zusätzliche Informationen, also Informationen, die nicht in der statischen Aufgabenstruktur enthalten sind, beigefügt werden. Diese geben Auskunft darüber, dass auf einen anderen Zug gewartet werden soll und um welchen Zug es sich dabei handelt. Für die

Übermittlung zusätzlicher Informationen sind Objekte des Typs *ITaskInformation* vorgesehen. Jede dem System übergebene Aufgabe kann beliebig viele dieser Informationen enthalten. Für die hier benötigte Information ist eine extra Klasse *ITaskTrainInfo* abgeleitet. Sie wird auch benutzt, wenn ein Fahrzeug für den Kunden zur Abholung bereitgestellt werden soll. Die Struktur ist in Abbildung 5.7 zu sehen. Wird solch eine zusätzliche Information einem Auftrag eines Fahrzeuges hinzugefügt und besagt diese, dass auf einen Zug gewartet werden soll, so speichert die Auftragsverwaltung eine Abbildung, die dem betroffenen Zug dieses Fahrzeug zuordnet. Derartige Abbildungen werden in der Datenverwaltung des Moduls OrderAdministration (*OADataManagement*) vorgehalten. Sie werden im Attribut *m_mapTrainVehicles* gespeichert.

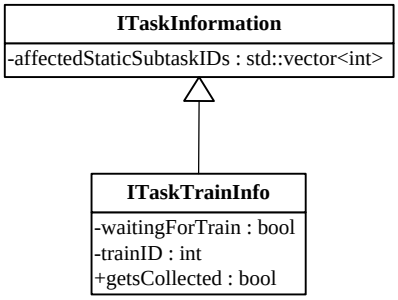


Abbildung 5.7: ITaskTrainInfo

Die Auftragsverwaltung wird, wenn ein Zug einfährt, darüber informiert, welcher Zug als nächstes kommen wird. Diese Information ist Teil der Nachricht ITrainReady. Ist der aktuelle Zug also fertig beladen, so kann die Auftragsverwaltung anhand der eben beschriebenen Abbildung von Zug-ID auf eine Menge von Fahrzeugen feststellen, welche Fahrzeuge für den Folgezug in die Wartespur geleitet werden müssen. Die jeweiligen Abbildungen werden daraufhin gelöscht.

5.3.2.3 Auflieger für Abholer bereitstellen und Abholung der Auflieger

Besagt eine zusätzliche Information im Auftrag eines Fahrzeuges, dass es vom Kunden abgeholt werden soll, so speichert die Auftragsverwaltung die ID des Aufliegers in einer Liste. Es stellte sich die Frage, zu welchem Zeitpunkt die Abholer kommen sollten. Wie schon bei der Entscheidung, zu welchem Zeitpunkt die Fahrzeuge in der Wartespur ihre Fahraufgaben auf den Zug erhalten sollten, galt es zu überlegen, welches Ereignis herangezogen werden konnte, um die Erzeugung der Abholer auszulösen. Dabei musste beachtet werden, dass der Auflieger dem System erst ab

dem Zeitpunkt für einen Ankuppelvorgang bekannt ist, wenn er vom Terminaltraktor abgekuppelt wurde. Versucht der Leitstand den Ankuppelvorgang vorher zu planen, so wird ein Fehler ausgelöst. Die Beendigung des Abkuppelvorgangs des letzten Fahrzeuges vom Zug schien also geeignet, die Erzeugung der Abholer einzuleiten. Die Strukturen des Leitstandes ließen es jedoch nicht zu, sich die ID des Abkuppelvorgangs des letzten Fahrzeuges zu merken, ohne größere Änderungen vorzunehmen, die für eine reale Umgebung unnötig wären. Aus diesem Grund wurde entschieden, das Abholen der Auflieger zu einem bestimmten Zeitpunkt auszulösen. Dieser wird in der Konfigurationsdatei des externen Moduls *SimulatedTrack* als Zeit-Offset angegeben. Der Wert gibt an, wie lange nach Ankunft eines Zuges mit dem Erzeugen der Abholer gewartet werden soll. Der Wert ist fest vorgegeben und für alle Züge gleich. Er muss somit groß genug gewählt sein, um auch bei Zügen mit vielen Fahrzeugen zu garantieren, dass alle Auflieger abgekuppelt wurden, bevor der erste Abholer seine Aufgaben zugewiesen bekommt. Ist dieser Zeitpunkt erreicht, so schickt das externe Modul eine Nachricht an die Auftragsverwaltung. Es handelt sich um das Ereignis mit dem Namen *ICollectTrailers*. Die Auftragsverwaltung reagiert, indem sie ihrerseits eine Nachricht an das Modul *VirtualEnvironment* schickt, um dieses anzuweisen, genau die richtige Anzahl an Abholern zu erzeugen. Die virtuelle Umgebung kann nicht direkt durch die Gleissimulation informiert werden, weil diesem Modul unbekannt ist, wie viele der geladenen Fahrzeuge für Kunden zur Abholung bereitgestellt werden müssen. Dies ist abhängig von den Aufträgen der einzelnen Lkw.

Das Modul zur Auftragsannahme hat keine Informationen darüber, welche Auflieger momentan zur Abholung bereitstehen, diese müssen aber den erzeugten Fahrzeugen zugeordnet werden. Soll ein Kuppelvorgang erfolgen, so muss dieser Aufgabe ebenfalls eine zusätzliche Information beigefügt werden. In diesem Fall heißt sie *ITaskTransformationInfo* und enthält beispielsweise das Kennzeichen des anzukuppelnden Fahrzeugteils. Diese Information wird durch das Modul *OrderAcceptance* jedoch frei gelassen. Empfängt die OrderAdministration einen Auftrag für ein Fahrzeug, so prüft sie anhand des Kennzeichens, ob es sich um einen Abholer handelt. Das Kennzeichen beginnt dann mit der Zeichenfolge „Collector“. Die Information über das anzukuppelnde Objekt wird automatisch eingefügt. Die Auflieger werden entgegensetzt zu der Reihenfolge, wie sie auf dem Zug standen, vergeben. In der Realität wäre der Leitstand für die Abholung der Auflieger überhaupt nicht zuständig. Der gerade beschriebene Teil müsste im Falle einer Portierung in eine reale Umgebung also aus der Auftragsverwaltung entfernt werden. Momentan beginnt die Erzeugung der Abholer zehn Minuten nach Einfahrt des Zuges.

5.3.2.4 Behandlung der vom Kunden abgegebenen Auflieger

Der Leitstand ist so konzipiert, dass er nur Aufträge für aktive Fahrzeugteile annehmen kann. Andere Fahrzeugteile sind nicht in der Lage Aktionen irgendeiner Art durchzuführen. Und doch musste diese Einschränkung beseitigt werden. Bringt ein Kunde seinen Auflieger auf den Hof, kuppelt ihn ab und verlässt den Hof wieder, so muss der Leitstand darüber informiert werden, was mit diesem Fahrzeugteil weiter geschehen soll. Dies geschieht zu einem Zeitpunkt, zu dem noch kein Traktor es abgeholt hat. Die Aufträge müssen also, zumindest vorerst, dem Auflieger zugeteilt werden.

Um dieses Problem zu lösen mussten die Auftragsannahme (*OrderAcceptance*)[24] und die Auftragsverwaltung überarbeitet werden. Die Auftragsannahme informiert die Auftragsverwaltung mittels einer Nachricht (*IOrder*) über Aufgaben eines Fahrzeuges. Dieser Nachricht wurde ein Attribut angehängt, welches Aussage darüber trifft, ob die Aufgaben zu einem aktiven oder einem passiven Fahrzeugteil gehören. Die OrderAdministration prüft beim Erhalt solch einer Nachricht, ob sie zu einem Auflieger gehört. Ist dies der Fall, können diese Aufträge nicht einfach an das Fahrzeug weiter gereicht werden. Es muss erst ein Traktor gefunden werden, der den Auflieger annimmt. Wie in Abschnitt 5.5 beschrieben wird, werden Traktoren in Spuren geparkt, wo sie darauf warten neue Aufträge zugewiesen zu bekommen. Dafür ist das sog. *ParkingLotManagement* zuständig, ein neues Teilmodul der OrderAdministration. Dieses Modul verwaltet auch die Traktoren und ist deshalb dafür zuständig, einen freien Traktor zu finden, der den Auflieger ankuppelt und dann dessen Aufträge erfüllt. Wurde ein freier Traktor gefunden (durch Aufruf der Methode *getATractor()* der Parkplatzverwaltung), so wird verfahren, als ob der Auftrag, den das Modul OrderAcceptance für den Auflieger vergeben hat, ihm galt. Der Ankuppelvorgang ist Teil der Aufgaben.

5.4 Statische Aufgabenstruktur

Jeder Aufgabe, die auf dem Hof ausgeführt werden kann, muss ein Knoten im sog. statischen Aufgabenbaum zugrunde liegen. Die Knoten dieses gerichteten Graphen sind vom Typ *TaskNode*.

[24] vgl. Abschnitt 5.8

Dieser kann entweder vom Typ *TaskNodeList* oder vom Typ *StaticTask* sein (vgl. Abbildung 5.8).

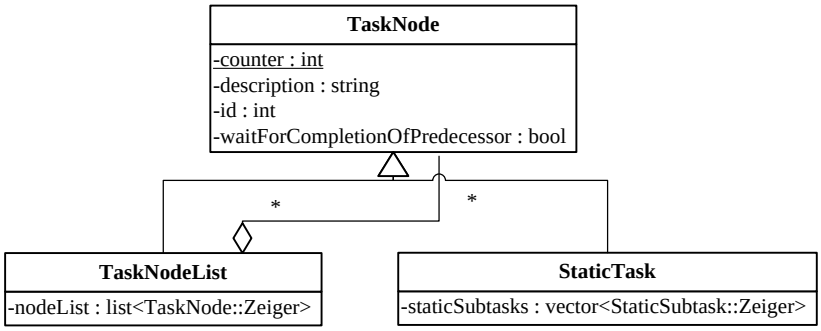


Abbildung 5.8: Struktur eines Aufgabenknotens

Die Blätter des Graphen müssen vom Typ StaticTask sein. Der Graph sollte so modelliert werden, dass die reale Aufgabe eines jeden TaskNodes durch die Ausführung eines seiner Nachfolger erfüllt werden kann. Einen Auszug des statischen Aufgabenbaums für EZrola zeigt Abbildung 5.9. Es ist ersichtlich, dass die Wahl mancher Knoten als Aufgabe für ein Fahrzeug keinen Sinn ergibt, da für deren Erfüllung Informationen fehlen, oder deren Nachfolger nicht gleichwertig sind. Wird einem Fahrzeug beispielsweise der Knoten für „*Transformieren*“ zugeteilt, so sind dessen Nachfolger, „*Ankuppeln*“ und „*Abkuppeln*“, unterschiedlicher Natur. Auch die Wahl der Wurzel ergibt wenig Sinn. Deren Bedeutung wäre in etwa „*Erfülle eine beliebige Aufgabe*“. Wird hingegen beispielsweise der Knoten „*Parken für Abholer*“ ausgewählt, so kann der Leitstand frei zwischen dessen Nachfolgern, also „*Parken auf Parkplatz 1*“ bis „*Parken auf Parkplatz n*“, wählen. Mit jedem dieser Knoten gilt der Grundauftrag als erfüllt.

Die Graphstrukur muss nicht unbedingt baumartig gewählt werden. In manchen Fällen würde es Sinn ergeben, dass mehrere Knoten die gleichen Nachfolger haben. Die Aufgabenstruktur entspricht also eigentlich einem gerichteten azyklischen Graphen. Ein Beispiel ist in Abbildung 5.10 auf der nächsten Seite zu sehen. Dort haben „*Parken für Abholer*“ und „*Parken für Folgezug*“ die gleichen Nachfolger. Das hätte zur Folge, dass beide Typen auf die gleichen Stellflächen zurückgreifen und diese dadurch optimal genutzt werden können. Für das RoLa-Terminal Regensburg wird von dieser Möglichkeit jedoch kein Gebrauch gemacht.

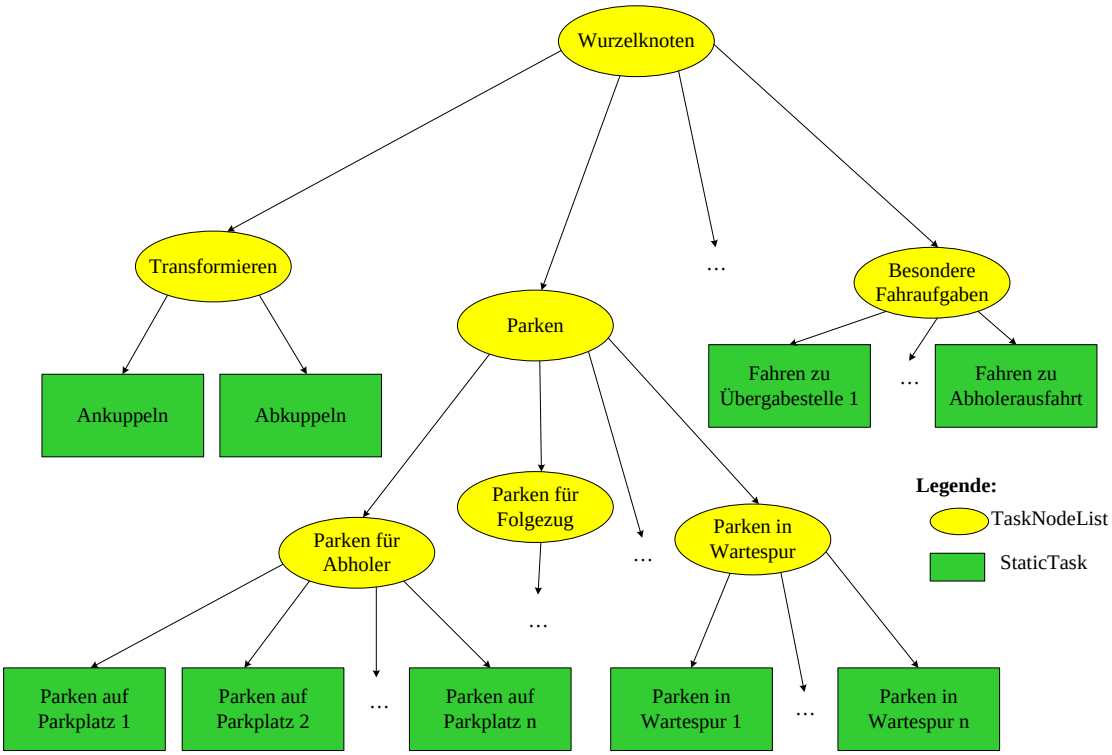


Abbildung 5.9: Auszug aus dem statischen Aufgabenbaum von EZrola

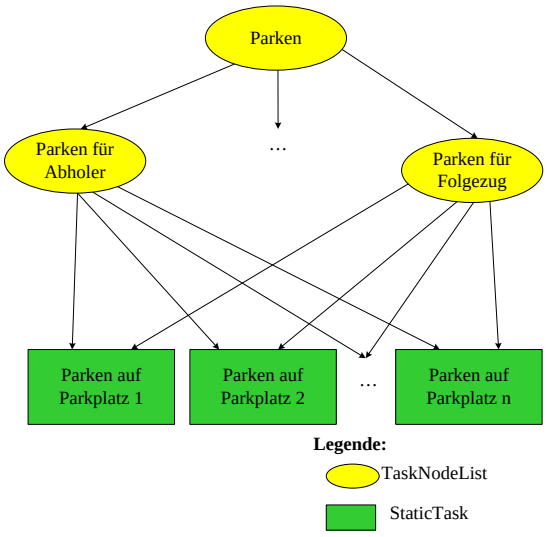


Abbildung 5.10: Beispiel für Knoten mit gleichen Nachfolgern

Der Vorteil dieser Struktur liegt darin, dass die Instanz, welche die Aufträge vergibt, keine Kenntnis über die gesamte Taskstruktur benötigt, sondern nur die wichtigsten Kategorien kennen muss. Ihr muss z. B. nicht bekannt sein, wie viele Stellplätze für Auflieger bereitstehen, die von Kunden abgeholt werden sollen. Außerdem kann so die Wahl des Tasks dem Leitstand überlassen werden. Dieser hat, wieder im Beispiel des Parkens, als einziger Kenntnis darüber, welche Parkplätze gerade frei sind. Der Leitstand muss jedoch über die Fähigkeit verfügen, eine qualifizierte Auswahl zwischen den Nachfolgern des gewählten Knotens zu treffen. Wie dies beim Parken erreicht wird, ist in Abschnitt 5.5 beschrieben. Für die weiteren Aufgabenkategorien ist keine automatische Auswahl durch den Leitstand vorgesehen.

Für EZrola wurden einige Aufgaben aus dem statischen Aufgabengraphen entfernt. Im Gegensatz zu EZsped sind in Fall von EZrola keine sog. *Funktionalen Einheiten*[25], wie z. B. Waschanlagen, Rampen, Tankanlagen und Wartungsstationen vorhanden, daher sind entsprechende Wasch-, Belade-, Tank- oder Wartungsaufgaben momentan überflüssig.

Jede statische Aufgabe (also jeder *StaticTask*) besteht aus einem oder mehreren atomaren Teilaufgaben, den sog. *StaticSubtasks*. Es gibt drei verschiedene Spezialisierungen dieser Teilaufgaben:

1. Fahraufgaben (*StaticDrivingTask*)
2. Transformationsaufgaben (*StaticTransformationTask*)
3. Warteaufgaben (*StaticWaitingTask*)

Soll beispielsweise ein Terminaltraktor einen Auflieger aufnehmen, so muss er zuerst vor ihn fahren. Anschließend, nachdem die Fahrzeugsicherheit ausgeschaltet wurde[26], wird die nächste Fahraufgabe ausgeführt. Diese führt den Traktor mit seinem Kuppelpunkt auf den Kuppelpunkt des Anhängers. Anschließend wird eine Transformationsaufgabe ausgeführt. Der Aufbau einer Transformationsaufgabe ist in Abbildung 5.12 dargestellt.

Warteaufgaben werden bei EZrola nicht ausgeführt. Bei EZsped werden diese z. B. bei funktionalen Einheiten gebraucht. Soll ein Fahrzeug an einer Rampe beladen werden, so ist unbekannt,

[25] Funktionale Einheiten werden in [LuT] Abschnitt 3.4.3 definiert

[26] Würde die Fahrzeugsicherheit für diese Fahrzeugteile nicht ausgeschaltet werden, so würde der Leitstand eine Kollision erkennen, wenn das Zugfahrzeug unter den Auflieger fährt. Vgl. Abschnitt 5.1

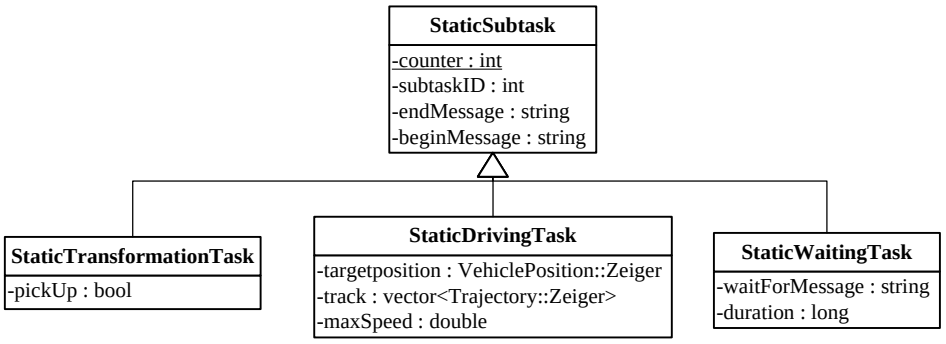


Abbildung 5.11: Struktur der atomaren Teilaufgaben

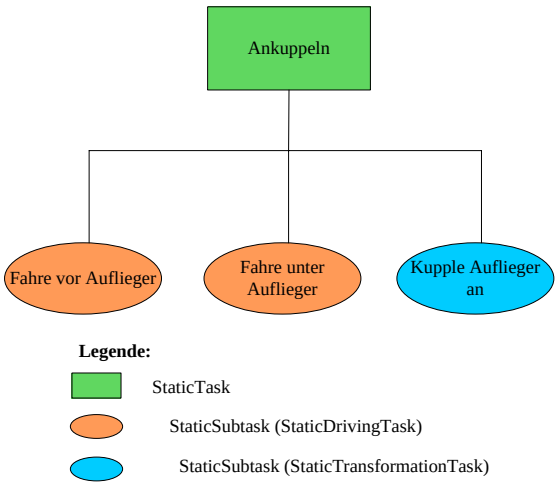


Abbildung 5.12: Aufbau einer Transformationsaufgabe

wie lange das Fahrzeug an der Rampe warten muss. Es führt dort daher eine Warteaufgabe aus, die durch den Empfang einer bestimmten Nachricht beendet wird. Nähere Informationen zu den verschiedenen Teilaufgaben sind in [LuT], Abschnitt 3.1.3, zu finden. Informationen zum Aufbau der XML-Beschreibungsdatei der statischen Auftragsstruktur sind im Anhang in Abschnitt 7.4 erläutert.

5.5 Parken

Ein Großteil der auf dem RoLa-Terminal ausgeführten Aufgaben fällt unter die Kategorie „Parken“[27]. Aber nicht nur in diesem Anwendungsfall spielt das Parken eine entscheidende Rolle, vielmehr ist es in allen denkbaren Anwendungsgebieten nötig, momentan nicht benötigte Fahrzeuge platzsparend an geeigneten Orten abzustellen. Es handelt sich bei dem in diesem Abschnitt beschriebenen Hilfsmodul also auch um eine Erweiterung des Allgemeinen Leitstands, die jedoch von Anwendungsfall zu Anwendungsfall im Detail unterschiedlich ausgeprägt sein kann. Bei EZ-rola wird zwischen den folgenden Typen des Parkens unterschieden:

- Parken für Abholer
- Vom Zug kommend parken, um auf einen folgenden Zug zu warten
- In der *Wartespur* parken, um auf den aktuellen/nächsten Zug auffahren zu können
- Traktoren in *Traktorspuren* parken, um sie auf den nächsten Auftrag warten zu lassen

Diese Aufgaben besitzen zum Teil sehr unterschiedliche Eigenschaften und stellen daher unterschiedliche Anforderungen an den Leitstand. Alle Berechnungen, die zu Parkaufgaben gehören, sind der Planung zuzurechnen und daher in der Planungskomponente unterzubringen. Der Hauptteil der Berechnungen wird bei der Verarbeitung der Aufgaben eines Fahrzeuges durchgeführt. Diese Berechnungen fallen also in das Aufgabengebiet der Auftragsverwaltung. Um diese jedoch so allgemein wie möglich zu halten, wird ihr ein Zusatzmodul zur Seite gestellt um diese Aufgaben zu bewältigen: das *ParkingLotManagement* (PLM). Die statische Struktur dieser Verknüpfung ist in Abbildung 5.13 auf der nächsten Seite zu finden. Soll die Auftragsverwaltung eine Parkaufgabe verarbeiten, so leitet sie die Anfrage direkt weiter an das PLM. Verarbeiten bedeutet in diesem Fall, dass aus den Aufgaben, wie sie von außen an das System übergeben werden, sog. *InputTasks* erzeugt werden. Diese enthalten alle Informationen, die nötig sind, um die darin enthaltenen Aufgaben durchführen zu können und werden vom Scheduler bei Planungsanfragen erwartet. Die OrderAdministration kann beim Verarbeiten aller Aufträge Parkaufgaben erkennen, da diese alle von einem allgemeinen Knoten „Parken“[28] abgeleitet sind. Ein Zeiger auf den Knoten der statischen Aufgabenstruktur ist Teil jeder Aufgabe eines Auftrags. Die Unterscheidung zwischen den einzelnen Typen wird erst durch das PLM vorgenommen.

[27] vgl. Abschnitt 4.3.1.1

[28] nähere Informationen zu den Taskstrukturen sind in Abschnitt 5.4 erläutert

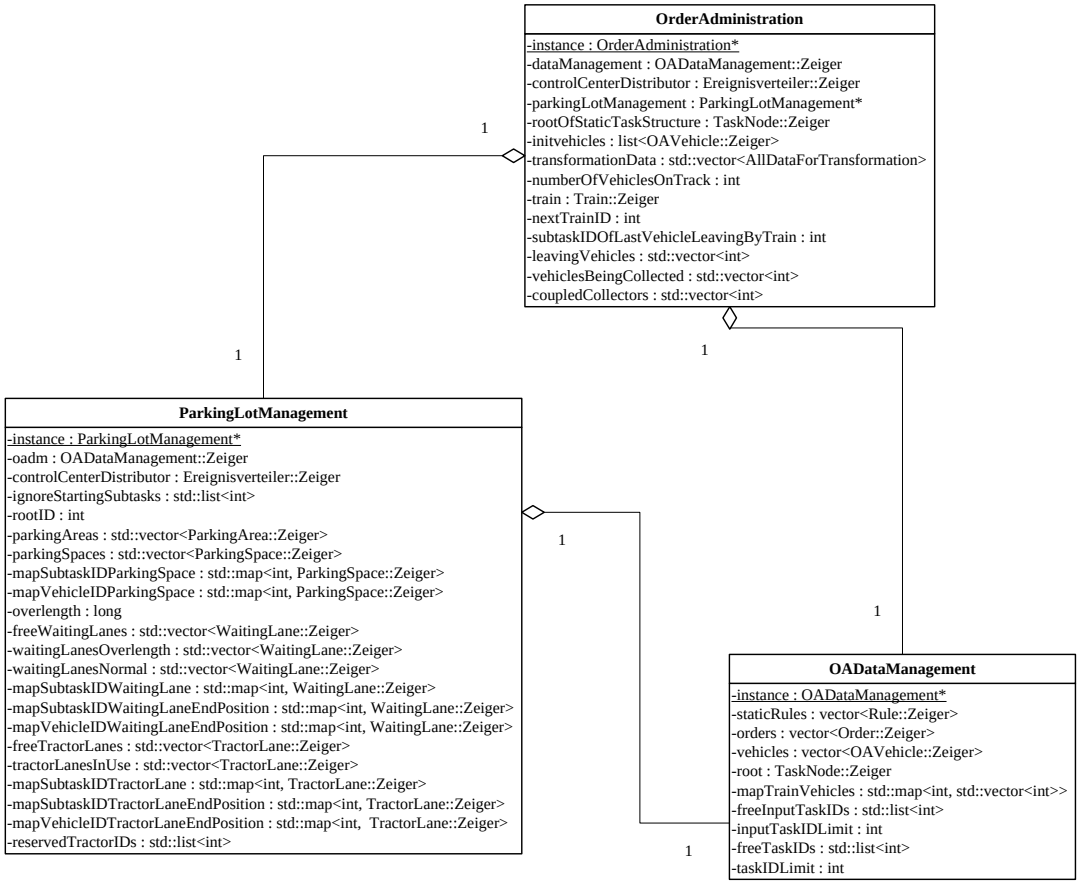


Abbildung 5.13: Einbettung der Parkplatzverwaltung

Das PLM wird gleich nach dem Scheduler und der OrderAdministration initialisiert. Dabei liest es alle nötigen Informationen ein und meldet sich für den Empfang einiger Nachrichten vom Ereignisverteiler an. Die Informationen, welche eingelesen werden, betreffen die Strukturen der Parkplätze, bzw. der verschiedenen Wartespuren. Da das Konzept der statischen Aufgabenstruktur beibehalten werden sollte, werden die nötigen Informationen daraus eingelesen. Hierzu bedient sich das PLM einer Referenz zur Datenverwaltung der OrderAdministration, der die Aufgabenstruktur vorliegt. Diese Datenverwaltung stellt einige Funktionen zum Durchlaufen des Baums zur Verfügung. Die Wurzel aller Parkaufgaben wird angefordert und dann dessen Nachfolger untersucht. Die verschiedenen Typen werden über das Attribut *Description* unterschieden, welches bei jedem Knoten vorhanden ist. Das Einlesen der Parkraum-Strukturen wird in der Funktion *extractParkingDataFromStaticTasks(...)* vorgenommen.

Das PLM benötigt immer aktuelle Daten über die Zusammensetzung von Fahrzeugen. Dazu bedient es sich ebenfalls der Datenverwaltung der OrderAdministration, deren Sicht auf ein Fahrzeug[29] um das Attribut *Länge* erweitert werden musste.

5.5.1 Parken auf üblichen Parkplätzen

Parkbereiche, auf denen Auflieger für Abholer bereitgestellt werden und solche auf denen Fahrzeuge auf andere Züge warten, sind strukturell nicht zu unterscheiden. Sie werden im statischen Aufgabenbaum durch eine Menge einzelner Parkplätze repräsentiert[30], die ihrerseits aus jeweils nur einer statischen Teilaufgabe (*StaticSubtask*) bestehen. Dieser StaticSubtask ist vom Typ „*StaticDrivingTask*“ und besitzt somit als wichtigstes Attribut eine sog. „*VehiclePosition*“. Diese besteht aus einem Punkt auf dem Hof und einer Fahrzeugausrichtung, die an diesem Punkt eingenommen werden soll. Das PLM liest diese Parkbereiche aus dem statischen Aufgabenbaum ein und übernimmt sowohl für diese Bereiche, als auch für die einzelnen Parkplätze, jeweils die ID vom jeweiligen Knoten des Baumes. Zu jedem Parkplatz wird ein Zeiger auf den dazu gehörigen *StaticDrivingTask* gespeichert. Außerdem besitzt jeder Parkplatz ein Attribut, das anzeigt, ob der Parkplatz frei oder besetzt ist und ein Attribut, das alle auf dem Parkplatz befindlichen Fahrzeugteile speichert. Eine leere Liste von Fahrzeugen bedeutet nicht, dass der Parkplatz frei ist, da dieser reserviert, aber noch leer sein könnte. Die Struktur eines Parkplatzes ist in Abbildung 5.14 zu sehen.

Die Schnittstelle des PLM, die die Auftragsverwaltung nutzt, um Parkaufgaben jeder Art vervollständigen zu lassen, ist die Funktion *completeOrder(...)*. Dort wird zwischen den verschiedenen Typen von Parken unterschieden. Der Auftrag wird an spezielle Funktionen delegiert. Soll ein Auflieger für einen Kunden zum Abholen bereitgestellt oder ein Fahrzeug für einen späteren Zug abgestellt werden, so muss im Auftrag entweder die ID des Knotens für den ganzen Parkbereich oder genau die ID des Knotens für den bestimmten Stellplatz angegeben werden. Die Funktion sucht daraufhin entweder exakt diesen oder einen freien Parkplatz im Parkbereich, wenn die ID eines Parkbereiches übergeben wurde. Für das Wählen eines freien Parkplatzes ist eine Funktion der Klasse zuständig, die einen Parkbereich repräsentiert. Es handelt sich dabei um die Funktion *getFreeParkingSpace(...)* der Klasse *ParkingArea*. Die Parkplätze auf dem RoLa-

[29] Es handelt sich dabei um Objekte des Typs *OAVehicle*

[30] vgl. Abbildung 5.9

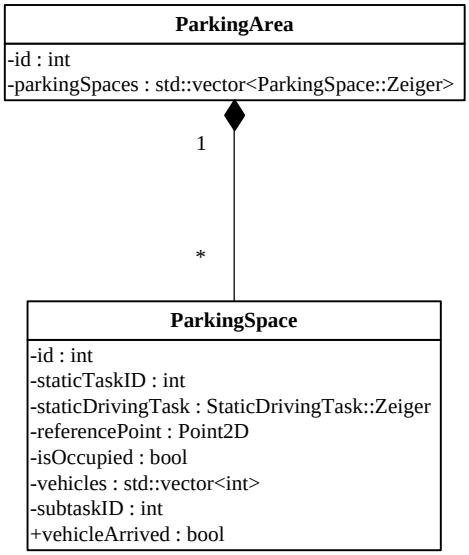


Abbildung 5.14: Struktur eines Parkplatzes

Terminal Regensburg sollten immer von links nach rechts angefahren werden[31], um das Anfahren zu erleichtern. Deshalb durchläuft die Suchfunktion die Parkplätze in der Reihenfolge, wie sie aus dem statischen Aufgabenbaum eingelesen wurden. Aus diesem Grund müssen die Parkplätze dort in der Reihenfolge aufgeführt werden, wie sie später ausgewählt werden sollen. Für eine Wahl aufgrund anderer Faktoren muss die Suchfunktion entsprechend angepasst werden. Denkbar wäre hier z. B. eine gleichmäßige Auslastung aller Parkplätze. Unter Umständen müssen auch Entscheidungskriterien in Form von Attributen bis zu dieser Suchfunktion durchgereicht werden. Dies wäre beispielsweise dann der Fall, wenn Eigenschaften des Fahrzeuges mit in die Parkplatzwahl[32] einfließen sollen, jedoch würde es sich in diesem Fall um Parkplätze komplexerer Struktur handeln, welche nicht mehr aus dem statischen Aufgabenbaum übernommen werden könnten.

Wurde ein freier Parkplatz gefunden, so wird dessen Status auf *besetzt* gesetzt. Es wird ein *Task* und der zugehörige *Subtask* (*DrivingTask*) erzeugt. Die ID des Subtasks[33] wird gespeichert. Der fertige Task wird - verpackt in einen *InputTask* - an die OrderAdministration zurückgege-

[31] vgl. Karte des Terminals in Abbildung 3.10
[32] z. B. die Länge des Fahrzeuges
[33] Subtask-IDs sind eindeutig gewählt

ben und kann weiter verarbeitet werden. Empfängt das PLM die Nachricht, dass der Subtask vollendet wurde[34], so werden alle Fahrzeugteile, die zu diesem Subtask gehören, in die Liste der Fahrzeugteile aufgenommen, die sich momentan auf dem Parkplatz befinden. Allen Fahrzeugteilen wird innerhalb einer Datenstruktur des PLM der Parkplatz zugeordnet. Ab jetzt wird beim Start eines jeden DrivingTasks geprüft, ob daran beteiligte Fahrzeugteile einem Parkplatz zugeordnet sind. Diese werden dann von diesem Parkplatz entfernt. Bei diesem Vorgang wird überprüft, ob der Parkplatz dabei frei geworden ist, was dann zur Folge hat, dass der Zustand *besetzt* wieder zurückgezogen werden kann.

5.5.2 Parken in Spuren

Neben der ersten Kategorie des Parkens auf normalen Parkplätzen, gibt es noch das Parken in Spuren. Diese Variante ermöglicht es sehr platzsparend zu parken, mit der Einschränkung, dass nicht jedes beliebige sich in der Spur befindliche Fahrzeug eine Fahraufgabe starten kann. Dazu sind nur das jeweils erste und letzte Fahrzeug in der Lage. Für Fahrzeuge, die auf den nächsten Zug auffahren sollen gilt, dass zwischen dem Parken und Auffahren unter normalen Umständen keine weiteren Aufgaben auszuführen sind. Wenn die Fahrzeuge in den Wartespuren schon die richtige Reihenfolge haben, lässt sich ein Fahrzeug nach dem anderen - von der Spitze an - auf den Zug leiten. Die Reihenfolge der Lkw ist bei der Rollenden Landstraße grundsätzlich beliebig. Es gilt nur zu beachten, dass keine zwei Lkw mit Überlänge hintereinander stehen dürfen[35]. Mit Wartespuren lässt sich dies ohne Probleme verwirklichen. Für Lkw mit Überlänge werden komplette Spuren herangezogen, falls es nötig ist. Auch in der begleiteten Rollenden Landstraße wird so vorgegangen.

Auf jedem RoLa-Terminal müssen ausreichend viele Terminaltraktoren zur Verfügung stehen, um Auflieger abzuholen, die vom Kunden zum Versand abgegeben werden. Diese Traktoren sollen alle vollkommen gleichwertig sein. Daraus folgt, dass es irrelevant ist, welcher freie Traktor gewählt wird, um einen Auflieger abzuholen. Auch für diesen Anwendungsfall lassen sich Wartespuren ohne Einschränkungen heranziehen.

[34] die Nachricht *ISubtaskCompleted* wird vom Scheduler bei Vollendung jeder Teilaufgabe verschickt und trägt als Attribut die ID des betreffenden Subtasks

[35] vgl. Abschnitt 3.2.3

Die beiden Typen von Wartespuren haben einige gleichartige Eigenschaften und sind deshalb von der gleichen Klasse abgeleitet, wie in Abbildung 5.15 dargestellt wird. In beiden Fällen müssen mehrere Fahrzeuge Positionen in der Spur hintereinander einnehmen und müssen, da sie keine Möglichkeit haben aneinander vorbei zu kommen, auch in der richtigen Reihenfolge an ihrem Zielpunkt in der Spur ankommen. Die erste Implementationsversion sah vor, dass die Fahrzeuge zuerst an den Anfang der Spur fahren und wenn sie dort angekommen sind, ihren finalen Punkt in der Spur erhalten. Auf diese Weise wäre garantiert, dass die Reihenfolge der Fahrzeuge in der Spur korrekt ist, da sie erst festgelegt wird, wenn ein Fahrzeug bereit zur Einfahrt ist. Außerdem wäre diese Version sehr effizient, da keine Anfahrtszeiten oder Ähnliches berechnet werden müssten. Jedes Fahrzeug, welches diese Wartespur benutzen soll, könnte direkt und ohne Zeitverzug starten, das schnellste bekommt den nächsten Platz. Ein bis dahin unbemerkt gebliebenes Problem an der Raum-Zeit-Planung[36] machte diese Variante jedoch unmöglich. Im Raum-Zeit-Plan würde ein Fahrzeug nach der Planung seiner ersten der beiden Fahraufgaben, nämlich der an den Anfangspunkt der Spur, diese so lange blockieren[37], bis sein Folgefahrauftrag eingeplant ist. Dieser könnte aber erst geplant werden, wenn der Zielpunkt des Fahrauftrags bekannt ist. Jener wiederum wäre erst bekannt, wenn das Fahrzeug am Anfangspunkt der Wartespur angekommen ist. Das hat zur Folge, dass alle folgenden Planungsanfragen abgewiesen werden würden, bis das Fahrzeug den Anfangspunkt erreicht hätte.

Aus diesem Grund wird einem Fahrzeug sein Zielpunkt innerhalb der Spur direkt mitgeteilt. Das erste Fahrzeug, das sich für eine Spur anmeldet, erhält auch den ersten Platz in der Spur. Diese Version lässt Anfahrtszeiten außer acht, erstellt also in zeitlicher Hinsicht keinen optimalen Plan. Erhält ein Fahrzeug mit sehr großer Entfernung zur Wartespur einen Platz in dieser, so müssen Fahrzeuge, die eine geringe Entfernung zur Spur haben und erst später eine Anfrage stellen, trotzdem so lange verzögert werden, bis sie sich hinter dem anderen Fahrzeug einreihen können. Beim RoLa-Terminal Regensburg käme diese Problematik zum Tragen, wenn etwa zur gleichen Zeit ein Fahrzeug, das mit dem Zug angekommen ist, und ein Fahrzeug, das über die Straßeneinfahrt angekommen ist, in die Wartespur fahren sollen, das Fahrzeug vom Zug jedoch zuerst verarbeitet wird. Um diese Problematik zu lösen, müsste entweder die Raum-Zeit-Planung so überarbeitet werden, dass die oben beschriebene Variante möglich wird oder es müsste eine

[36] vgl. Abschnitt 5.9

[37] Mit blockieren ist gemeint, dass im Raum-Zeit-Plan eingetragen wird, dass sich das betreffende Fahrzeug vom Zeitpunkt seiner Ankunft an diesem Punkt bis ins Unendliche dort befindet. Dieser Eintrag bleibt bestehen, bis bekannt ist, wie weiter mit dem Fahrzeug verfahren wird.

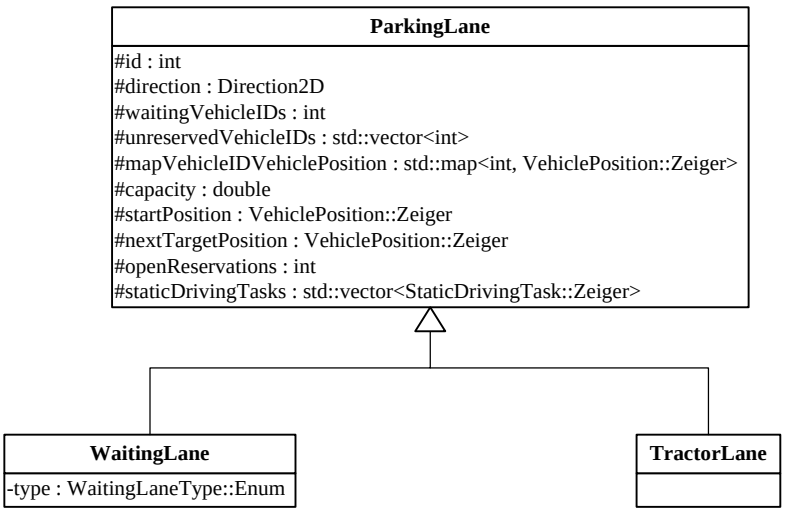


Abbildung 5.15: Struktur der Parkspuren

Möglichkeit eingefügt werden, um in den bestehenden Plan vom Scheduler eingreifen zu können. Der spätere Eingriff in den Plan ist für eine allgemeine Fehlerbehandlung unumgänglich und müsste in ähnlichen oder folgenden Projekten auf jeden Fall implementiert werden.

Auch jetzt besteht das Fahren in eine Wartespur - wie bei der ersten Variante - aus zwei einzelnen Fahraufgaben. Der erste Fahrauftrag führt immer noch an den Anfangspunkt der Spur. Dort angekommen wird geprüft, ob die Reihenfolge noch eingehalten wird. Es wird also geprüft, ob das Fahrzeug, welches dort seine erste Aufgabe vollendet hat, auch das nächste Fahrzeug ist, welches in der Spur erwartet wird. Auf diese Weise kann noch vor dem Start der zweiten Fahraufgabe auf einen Fehler reagiert werden. Dies wird jedoch wegen der Komplexität eines Eingriffs in den bestehenden Plan momentan nicht getan. Die Aufteilung in zwei getrennte Fahraufgaben hat zusätzlich den Vorteil, dass auf diese Weise auch für Wartespuren keine weiteren Informationen nötig sind, als jene, die im statischen Aufgabenbaum angegeben werden. Es müssen also, wie bei den üblichen Parkplätzen auch, keine weiteren Geländebeschreibungsdateien angelegt und eingelesen werden. Eine Wartespur wird als ungekrümmte Strecke angesehen, die durch zwei Punkte auf dem Hof definiert wird. Diese beiden Punkte sind die Zielpunkte der beiden Fahraufträge aus der externen Beschreibungsdatei des statischen Aufgabenbaums.

Für die Bestimmung des Zielpunktes in der Spur gab es mehrere Möglichkeiten. Am einfachs-

ten wäre es, die Spur in feste Abschnitte einzuteilen, mit festen Zielpunkten. Würde ein Fahrzeug einen Platz in der Spur anfordern, so würde es den nächsten freien Abschnitt erhalten. Auf diese Weise würden keine Informationen über das Fahrzeug benötigt und zur Laufzeit müssten keine weiteren Berechnungen - den Zielpunkt betreffend - mehr durchgeführt werden. Jedoch müsste für die Länge der Fahrzeuge der maximal mögliche Wert angenommen werden, wobei viel Platz verschwendet werden könnte. Die Alternative dazu ist, jedem Fahrzeug so viel Platz in der Spur zuzuordnen, wie es tatsächlich braucht, zuzüglich eines festen Sicherheitsabstandes. Der Zielpunkt muss auf diese Weise zur Laufzeit bestimmt werden. Aufgabe des PLM ist es, eine zur Aufgabe passende Spur zu finden, die genug freien Platz für das Fahrzeug birgt und anschließend einen Bereich in dieser Spur zu reservieren. Wie das PLM dabei vorgeht wird im Folgenden beschrieben.

Soll ein Fahrzeug auf den nächsten Zug auffahren, so bekommt es den Auftrag in eine Wartespur zu fahren. Die Auftragsverwaltung übergibt diesen Auftrag an das PLM, welches ihn anhand eines Attributes von anderen Parkaufgaben unterscheidet und die Anfrage wieder an eine spezielle Funktion[38] deligiert. In dieser muss eine zur Anfrage passende Spur gesucht werden. Für die Anfrage gibt es zwei Möglichkeiten. Es kann der allgemeine Auftrag „Fahre in eine Wartespur“, oder ein speziellerer Auftrag der Art „Fahre in Wartespur n“ gewählt werden. Von entscheidender Bedeutung bei der Wahl der Spur ist die Länge des Fahrzeuges. Eine Spur kann einen von drei verschiedenen Zuständen haben:

1. frei
2. reserviert für Fahrzeuge normaler Länge
3. reserviert für Fahrzeuge mit Überlänge

Ab welcher Länge ein Fahrzeug Überlänge hat, muss vor der Laufzeit festgelegt werden. Ist der Auftrag allgemein gewählt, so sucht das PLM mithilfe der Methode *reserveWaitingLane(...)* zuerst in den der Länge des Fahrzeuges entsprechenden reservierten Wartespuren nach einer Spur mit ausreichender Restkapazität. Ist keine Spur reserviert oder in keiner Spur ausreichend viel Platz, so prüft das PLM, ob noch freie Spuren zur Verfügung stehen, wählt im positiven Fall die Erste aus und überträgt sie in die Liste der reservierten Spuren für den jeweiligen Typ (normale Länge oder Überlänge). Bei einer Anfrage für eine spezielle Spur muss geprüft werden, ob diese entweder frei oder dem richtigen Typ zugeordnet ist und, ob in ihr noch ausreichend Platz zur Verfügung steht.

[38] *completeOrderWaiting(...)*

Wurde eine passende Spur gefunden, so muss ein Platz in ihr reserviert werden (mit der Methode *reserveASlot(...)*) und ein Task mit den beiden entsprechenden Subtasks erzeugt werden. Die IDs der Subtasks werden gespeichert, um deren Beendigung feststellen zu können. Nach Beendigung des ersten Subtasks wird die Reihenfolge in der Spur geprüft. Ein Fehler zieht jedoch momentan keine Konsequenz nach sich. Kommt das Fahrzeug an seiner Endposition in der Spur an, so speichert das PLM eine Abbildung vom Fahrzeug auf die Wartespur. Startet das Fahrzeug später eine Fahraufgabe, so kann das PLM durch diese Abbildung feststellen, in welcher Spur sich das Fahrzeug befindet und es aus dieser Spur löschen. Danach wird geprüft, ob die Spur leer ist, genauer gesagt, ob es noch Reservierungen für diese Spur gibt. Sind keine vorhanden, so kann die Spur reinitialisiert werden und muss daraufhin auch wieder in die Liste der freien Spuren übernommen werden.

Ist ein Zug bereit, die wartenden Lkw auffahren zu lassen, so muss das PLM die IDs der wartenden Fahrzeuge in sinnvoller Reihenfolge zur Verfügung stellen können. Zu diesem Zweck verfügt es über zwei Funktionen, von denen eine ein wartendes Fahrzeug mit Überlänge (*getWaitingVehicleOverlength()*) und die andere ein wartendes Fahrzeug mit normaler Länge zurückgibt (*getWaitingVehicleNormal()*). Diese Funktionen werden von der Auftragsverwaltung abwechselnd aufgerufen[39]. Die Fahrzeuge werden über Tiefensuche ermittelt, das heißt, dass zuerst alle Fahrzeuge der ersten Wartespur (des jeweiligen Typs) zurückgegeben werden und erst wenn dort kein Fahrzeug mehr verfügbar ist, das erste der nächsten Spur gewählt wird. Jedes gewählte Fahrzeug wird reserviert, um nicht noch einmal gewählt zu werden. Das ist notwendig, da zwischen Auswahl des Fahrzeuges und dem Starten der Fahraufgabe unabsehbar viel Zeit vergehen kann.

Obwohl sich Traktorspuren strukturell kaum von Wartespuren unterscheiden, besitzen sie doch Eigenschaften, die eine getrennte Behandlung nötig machen. Da Traktoren per Definition gleichwertig sind, wird unter ihnen nicht unterschieden. Es gibt für sie keine Überlänge, wie es bei den Wartespuren der Fall ist. Deshalb kann eine Traktorspur auch nur entweder den Zustand *frei* oder den Zustand *in Benutzung* haben. Im Grunde ist sogar, wie sich später noch zeigt, überhaupt keine Unterscheidung von Zuständen erforderlich. Traktorspuren werden im Vergleich zu Wartespuren in entgegengesetzter Richtung (also rückwärts) angefahren. Das hat geländespezifische Gründe und bedeutet keinen Unterschied für das PLM. Das rückwärtige Anfahren wird durch einen geeigneten Entwurf des Leitliniennetzes erreicht.

[39] vgl. Abschnitt 5.3

Da die Traktoren nicht unterschieden werden können, werden diese frei auf die vorhandenen Spuren verteilt, d. h. es muss nicht eine Spur nach der anderen gefüllt werden. Eine möglichst gleichmäßige Auslastung der Spuren ist sogar erwünscht. Das Gleiche gilt für die Terminaltraktoren. Ihre Standzeit soll möglichst gleich ausfallen. Um die gleichmäßige Auslastung der Spuren zu gewährleisten, könnte ein sich anmeldendes Fahrzeug immer in die Spur geschickt werden, die momentan die wenigsten Fahrzeuge beinhaltet. Bei den Traktoren müsste der gewählt werden, dessen Standzeit am höchsten ist. Für beide Fälle wurde ein weitaus einfacheren Weg gewählt. Wird ein Platz in einer Spur reserviert (mit *reserveTractorLane(...)*) oder verlässt ein Fahrzeug eine Spur, so wird die Liste der Traktorspuren rotiert. Zugriffe auf Spuren erfolgen immer in der Reihenfolge dieser Liste, daher wird so, mit sehr geringem Aufwand, eine einigermaßen gleichmäßige Auslastung der Spuren und der Terminaltraktoren erreicht.

Traktorspuren sind die komplette Zeit über in Benutzung und werden nicht, wie Wartespuren, zu bestimmten Zeitpunkten komplett geleert. Durch das Herausfahren von Traktoren entstehen an den Ausfahrten der Spuren Freiräume. Diese Freiräume können nicht durch „neue“ Traktoren gefüllt werden, müssen aber, aufgrund des geringen Stellplatzes für Traktoren, dennoch beseitigt werden. Das PLM stellt extra zu diesem Zweck eine Funktion bereit. Verlässt ein Traktor eine der Spuren, so wird berechnet, wie groß der Freiraum an der Ausfahrt der Spur ist. Überschreitet der Wert eine vorher festgelegte Schwelle, so wird die Funktion *fillATractorLaneGap(...)* des PLM aufgerufen. Sie sorgt dafür, dass die Traktorspur die neuen Positionen selbst berechnet[40] und diese in einer Datenstruktur zurückgibt, welche sie den betroffenen Fahrzeugen zuordnet. Ab diesem Zeitpunkt ist in der Spur theoretisch wieder Platz, auch wenn die Spur vorher voll war und die Fahrzeuge die an der Ausfahrt entstandene Lücke noch nicht aufgefüllt haben. Im Anschluss wird mithilfe der genannten Rückgabe für jedes in der Traktorspur befindliche oder dort angemeldete Fahrzeug in der Reihenfolge ihrer Position in der Spur, ein neuer Fahrauftrag erzeugt. Die IDs der zugehörigen Fahraufgaben müssen gespeichert werden, um zu verhindern, dass das PLM das Starten dieser Fahrzeuge vermeintlich für ein Verlassen der Spur hält. Der Start dieser Aufgaben wird vom PLM daraufhin ignoriert. Die Fahraufträge werden der Auftragsverwaltung über den Ereignisverteiler übergeben. Sie werden als *DynamicOrderRequests* versandt.

Werden neue Auflieger abgegeben, so muss ein freier Traktor ausgewählt werden, der diesen Auflieger abholt. Die Auswahl dieses Traktors fällt ebenfalls in den Aufgabenbereich des PLM

[40] Durch Aufruf der Methode *fillTheGap()* der Klasse *TractorLane*

und wird in der Methode *getATractor()* durchgeführt. Die Zeit, die es dauert, bis ein ausgewählter Traktor seinen Fahrauftrag erhält, kann so groß sein, dass in dieser Zeit schon der nächste Traktor angefordert werden könnte. Aus diesem Grund muss ein ausgewählter Traktor reserviert werden, bis er die Spur tatsächlich verlässt. Die Auswahl geschieht über eine Art Breitensuche über der Menge der Traktorspuren. Zuerst werden die vordersten Traktoren geprüft, sind diese reserviert, so wird mit den Nächsten fortgefahren, solange bis die Tiefe die Menge der Traktoren in der Spur mit den meisten Fahrzeugen überschritten hat. Wird kein freier Traktor gefunden, so wird ein Fehler ausgegeben. Eine Behandlung ist nicht implementiert.

5.6 Scheduler

Der Scheduler war in seiner ursprünglichen Form nicht dazu in der Lage, mehrere Planungsanfragen für ein Fahrzeug zu behandeln. Nachfolgende Anfragen wurden verworfen. Die Vorgänge auf einem RoLa-Terminal machen es jedoch zwingend notwendig, voneinander unabhängige Anfragen für ein einzelnes Fahrzeug stellen zu können. Das einfachste Beispiel dafür ist ein Terminaltraktor, bei dem, wenn er vom Zug kommt und seinen Auflieger abstellt, noch gar nicht abzusehen ist, welche Aufgaben er weiter zu erfüllen hat. Seine Aufgaben können also nicht in einer einzelnen Planungsanfrage (*TaskSchedulingQuery*) übergeben werden.

Der Plan eines jeden Fahrzeuges wurde daher um die Fähigkeit erweitert, weitere TaskSchedulingQueries zu speichern, sofern aktuell eine ältere Anfrage bearbeitet wird. Nach Abschluss aller aktuellen Aufgaben wird der Plan reinitialisiert und die Planung der nächsten Anfrage kann gestartet werden.

Außerdem sollte der Scheduler konzeptionell weiter von der Auftragsverwaltung entfernt werden. Abhängigkeiten zwischen verschiedenen Aufgaben sollen deshalb vor dem Scheduler verborgen bleiben und vorher gelöst werden. Dazu wurde die Kommunikationsschnittstelle zur Auftragsverwaltung um das Ereignis *IOrdersCompleted* erweitert. Dieses wird verschickt, wenn alle TaskSchedulingQueries abgearbeitet - also verplant und durchgeführt - sind. Es gibt der Auftragsverwaltung die Möglichkeit, eine weiterführende Anfrage zu erstellen und zu verschicken[41].

Wie jedes Modul ruft auch der Scheduler immer wieder die empfangenen Ereignisse ab und wertet diese aus. In der Zeit, in der diese Ereignisse abgearbeitet werden, kann der Scheduler nicht

[41] vgl. Abschnitt 5.3

den Plan überprüfen, also auch keine Aufgaben beginnen. Um größere Verzögerungen in Hochlastzeiten zu vermeiden, sollte der Scheduler in der Lage sein, zwischen der Ereignisverwertung seinen Plan zu prüfen.

Die meiste Zeit wird bei der Berechnung des Raum-Zeit-Plans verbraucht. Um bei der gleichzeitigen Ankunft mehrerer Fahrzeuge auf dem Gelände die Rechenlast besser zu verteilen, speichert der Scheduler jetzt TaskSchedulingQueries in einem Vektor und arbeitet bei jedem Aufruf seiner Hauptroutine eines dieser Ereignisse ab.

Tritt bei der Planung ein Fehler auf, wird der Ablauf für den betreffenden Plan gestoppt. Eine Fehlerbehandlung existiert auch in dieser Version nicht.

5.7 Virtuelle Umgebung

Die Änderungen der virtuellen Umgebung haben hauptsächlich mit dem Erzeugen von Fahrzeugen zu tun. Normalerweise werden alle Fahrzeuge, die sich auf dem Hof bewegen, durch dieses Modul bekannt gemacht. Dazu werden sie von der Fahrzeugsimulation initialisiert und melden sich dann selbstständig beim Leitstand an. Wie schon in Abschnitt 4.3.1.3 beschrieben wurde, muss für jedes Fahrzeug ein Eintrag in einer Datei, die zu Beginn der Laufzeit eingelesen wird, existieren. In der Simulation dieser Arbeit muss jedoch eine sehr große Anzahl an Fahrzeugen erzeugt werden. Ort und Zeit, wann diese Fahrzeuge erzeugt werden, hängt stark vom Szenario ab. Kleine Änderungen im Zugfahrplan ziehen die Notwendigkeit sehr vieler Änderungen in der Konfigurationsdatei nach sich. Die Berechnung der Zeitpunkte, zu denen die Lkw erzeugt werden müssen, würde außerdem sehr viel Zeit in Anspruch nehmen. Daher musste die Möglichkeit geschaffen werden, Fahrzeuge dynamisch bzw. auf Anfrage zu erzeugen. Das Modul musste daher zuerst um die Fähigkeit, Nachrichten vom Leitstand zu empfangen, erweitert werden. Bisher war das Modul VirtualEnvironment nur Versender von Ereignissen. Es müssen drei verschiedene Typen von Fahrzeugen erzeugt werden:

1. Kunden, die ihre Auflieger abgeben. Sie fahren über den Einfahrtsbereich des Terminals ein.
2. Fahrzeuge, die den Hof mit dem Zug erreichen.
3. Kunden, die ihre Auflieger abholen. Sie fahren über die Abholereinfahrt ein.

Für die Übergabe von Aufliegern gibt es zwei vorgeschriebene Bereiche. Für jeden Zug ist im Zugfahrplan angegeben, wie viele Auflieger für diesen abgegeben werden müssen[42]. Empfängt die virtuelle Umgebung das Ereignis *ECreateVehiclesOnStreet*, so müssen immer zwei Fahrzeuge (eines für jeden Übergabebereich) in der Einfahrt erzeugt werden. Die Gesamtanzahl der zu erzeugenden Fahrzeuge wird als Parameter übergeben. Solange noch nicht alle Fahrzeuge erzeugt wurden, werden alle 300 s hintereinander zwei Fahrzeuge erzeugt. Die Positionen, auf denen die Fahrzeuge erzeugt werden, sind fest in einer Datei vorgegeben. Der Aufbau der Fahrzeuge ist ebenfalls festgelegt. Es handelt sich dabei um ein Standardfahrzeug, d. h. jedes Fahrzeug, das im Zuge dieser Funktionalität erzeugt wird, hat dieselbe Zusammensetzung. Zur Identifikation dieser Fahrzeuge besitzen die Kennzeichen eine feste Struktur. In solch einem Kennzeichen ist gespeichert, dass es sich um ein Fahrzeug handelt, das im Einfahrtsbereich erzeugt wurde. Dabei wird zwischen der Zugmaschine und dem Auflieger unterschieden, die ID des jeweiligen Zuges wird eingefügt und ein Laufindex ist vorhanden, um die Einzigartigkeit jedes Kennzeichens zu gewährleisten. Diese Identifikation ist notwendig, da das Modul zur Vergabe der Aufträge mit Standardaufträgen arbeitet, also nicht für jedes Kennzeichen Daten in seiner Datenbank vorliegen hat[43]. Es prüft, ob das Kennzeichen in diesem Fall zu dem eben beschriebenen Muster passt, und vergibt dann einen dieser Standardaufträge.

Empfängt die virtuelle Umgebung ein Ereignis vom Typ *ECreateVehiclesOnTrain*, so müssen Fahrzeuge auf einem gerade eingefahrenen Zug erzeugt werden. Als Attribute besitzt dieses Ereignis Informationen darüber, wie viele Fahrzeuge erzeugt werden sollen und wie der Zug aufgebaut ist, genauer gesagt, an welchen Positionen sich die Waggons befinden. Aus diesen Angaben wird berechnet, an welcher Stelle die Fahrzeuge erzeugt werden sollen. Momentan werden die Fahrzeuge mit ihrem Basis-Referenzpunkt auf dem Mittelpunkt des vorderen Endes des dafür vorgesehenen Waggons (aus der Sicht des Lkw) erzeugt. Eigentlich sollte jeder Lkw mit seinem Mittelpunkt auf dem Mittelpunkt des jeweiligen Waggons stehen, dies sollte daher für spätere Anwendungen noch angepasst werden. Die Fahrzeuge werden ohne zeitliche Pause hintereinander erzeugt und erhalten wieder codierte Kennzeichen. Die Struktur ist dabei ähnlich den Fahrzeugen, die über die Haupteinfahrt einfahren.

[42] vgl. Abschnitt 5.2
[43] vgl. Abschnitt 5.8

Die Erzeugung der dritten Art von Fahrzeugen wird über die Nachricht *ECreateCollectors* initiiert. Auch dieses Ereignis hat ein Attribut, welches angibt, wie viele Fahrzeuge erzeugt werden sollen. Die Positionen für die Abholer müssen berechnet werden. Als Grundlage dient ein Punkt, der fest vorgegeben wird. Auf diesem Punkt wird der erste der Abholer erzeugt, die Restlichen werden hintereinander aufgereiht. Probleme in der Raum-Zeit-Planung, vor allem in Verbindung mit Rückwärtsfahrten[44] können zu Deadlocks führen, wenn Fahrzeuge zu dicht hintereinander fahren. Um diese Problematik ein wenig zu entspannen, werden die Abholer nicht direkt nacheinander erzeugt, sondern im Abstand von 20 s. Auch hier ist der Aufbau der Fahrzeuge in einem Standardfahrzeug gespeichert, weshalb auch die Abholer alle das gleiche Aussehen haben.

An der virtuellen Umgebung wurden zusätzlich Änderungen vorgenommen, welche die Transformation von Fahrzeugen anbelangt, also das Ändern des Aufbaus eines Fahrzeuges, wie z. B. dem Abkuppeln des Aufliegers. Informationen zu diesen Änderungen sind Abschnitt 5.1 zu entnehmen.

5.8 Auftragsannahme

Die einzige Aufgabe der Auftragsannahme (*OrderAcceptance*) ist es, dem Leitstand mitzuteilen, welche Aufträge für ein Fahrzeug vorliegen. Allein darauf beziehen sich auch die Änderungen, die im Zuge der Anpassung des Allgemeinen Leitstandes an den Anwendungsfall der unbegleiteten Rollenden Landstraße vorgenommen worden sind. Durch die gravierende Zunahme der Anzahl an Fahrzeugen musste ein Konzept überlegt werden, das nicht jedem einzelnen Fahrzeug eine Menge an Aufgaben zuordnet. Ein wichtiger Aspekt ist die geringe Vielseitigkeit der Aufträge. Ein Abholer wird immer eine Fahraufgabe, eine Transformationsaufgabe und dann wieder eine Fahraufgabe erhalten. Ein Kunde, der seinen Auflieger am Terminal abgibt, bekommt eine analoge Auftragsstruktur zugewiesen. Beide Fahrzeuge werden nach Erledigung ihrer Aufgaben aus dem System gelöscht. Zu diesem Zweck erhalten sie einen festen Fahrauftrag, der sie jeweils an die Stelle leitet, an der sie das System verlassen. Diesem Auftrag ist eine sog. *Ende-Nachricht* beigefügt, eine Nachricht, die verschickt wird, sobald das Fahrzeug den betreffenden Task beendet hat. In diesem Fall bewirkt diese Nachricht, dass sich das Fahrzeug vom System abmeldet. Auch die Aufgaben, die von den Traktoren ausgeführt werden müssen, welche die Auflieger abholen, sind strukturell die gleichen. Einzig die Fahraufträge der vom Zug kommenden Fahrzeuge

[44] vgl. Abschnitt 6.4

können große Unterschiede aufweisen.

Aus der Erkenntnis heraus, dass die meisten Aufträge auf einem RoLa-Terminal je nach Anwendungsfall[45] immer wieder die gleiche Struktur besitzen, fiel die Entscheidung zugunsten eines Konzeptes, welches auf Standardaufträgen basiert, aber dennoch die Möglichkeit bietet, Aufträge explizit anzugeben. Die OrderAcceptance verschickt Aufträge als Reaktion auf eine Anfrage aus dem Leitstand. Diese kommen meist (zu Beginn dieser Arbeit immer) von der Auftragsverwaltung. In manchen Fällen wird sie jedoch auch von anderen Stellen ausgelöst. Sie enthält das Kennzeichen des Fahrzeuges, dessen Aufträge der Auftragsverwaltung übergeben werden sollen. Dieses Kennzeichen ist Grundlage der Operationen, die durchgeführt werden. Kommt also eine Nachricht des Typs *IOrderRequest* bei der Auftragsannahme an, so wird zuerst geprüft, ob sich unter den Aufträgen, die aus einer Datei eingelesen werden, einer für das in der Nachricht als Attribut übergebene Kennzeichen befindet. In diesem Fall kann der Vorgang abgebrochen werden und dieser Eintrag wird übergeben. Entspricht keiner der Einträge dem Kennzeichen, so muss die Entscheidung auf einer anderen Grundlage getroffen werden. Das einzige zur Verfügung stehende Kriterium ist jedoch das Kennzeichen. Dieses muss also so codiert werden, dass die OrderAcceptance die verschiedenen Anwendungsfälle voneinander unterscheiden kann. Die Kennzeichen werden vom Modul *VirtualEnvironment* vergeben. Jedes Kennzeichen beginnt mit einer Phrase, welche es für den betreffenden Anwendungsfall identifiziert. So beginnen alle Kennzeichen von Fahrzeugen, die via Zug das Gelände erreichen, mit „SimulatedTrack“. Des Weiteren müssen auch Anhänger und Zugmaschine unterschieden werden, weshalb in jedem Kennzeichen eine Zahl (z. B. *1* für Zugmaschine oder *2* für Auflieger) enthalten ist. Als Letztes muss dafür gesorgt werden, dass jedes Kennzeichen einzigartig ist. Dafür wird beispielsweise bei den Fahrzeugen vom Zug dessen ID eingefügt und danach ein Zähler, welcher jedes Fahrzeug von diesem Zug unterscheidet. Ein Fahrzeug, das den Hof als fünftes Fahrzeug auf dem Zug mit der ID *3* erreicht, hat dann also das Kennzeichen *SimulatedTrack3-05-1* (Zugfahrzeug), bzw. *SimulatedTrack3-05-2* (Auflieger).

Die Unterscheidung zwischen Zugfahrzeug und Auflieger ist vor allem bei den Fahrzeugen von großer Bedeutung, die den Hof über die Haupteinfahrt erreichen. Hier muss die OrderAcceptance Aufträge für beide Fahrzeugteile bereithalten und sie daher unterscheiden können. Der Nachricht, die an den Leitstand geschickt wird, um diesem die Aufträge mitzuteilen (*EOrder)*, wurde das Attribut *isTrailer* hinzugefügt, welches anzeigt, ob dieser Auftrag zu einem Auflieger gehört. Auf

[45] z. B. Abholer, Fahrzeuge vom Zug, etc.

diese Weise erkennt die OrderAdministration[46], ob sie erst einen freien Traktor für die Erledigung dieses Auftrags anfordern muss oder nicht.

5.9 Raum-Zeit-Plan

Wie in Abschnitt 4.4.1 beschrieben wurde, wird beim Versuch einen Plan für ein Fahrzeug zu finden in gewissen Zeitabständen immer wieder ein sog. *AllocationSnapshot* genommen und mit der für das Fahrzeug zu diesem Zeitpunkt berechneten Position geschnitten, um eine mögliche Kollision feststellen zu können. Kommt es zu einer Kollision, so wird, wie in oben genanntem Kapitel beschrieben, versucht das Fahrzeug zu verzögern, um diese Kollision zu verhindern. Wenn dies nicht möglich ist, wird ein neuer Planungsversuch gestartet, dessen Startzeitpunkt im Vergleich zum vorherigen Versuch in der Zukunft liegt. Wie viele solche Versuche es gibt und um welchen Zeitabschnitt diese in die Zukunft versetzt werden, wird durch zwei Werte festgelegt. Mit ihnen kann bestimmt werden, bis zu welchem Zeitpunkt in der Zukunft versucht wird, den Fahrauftrag einzuplanen. Je weiter für die Planung in die Zukunft gegangen wird, desto länger dauert der Planungsvorgang. Dies bedeutet momentan, dass die komplette Planungskomponente lahmgelegt ist. In dieser Zeit bekommt der Scheduler keine Rechenzeit zugewiesen, was bedeutet, dass Tasks nicht gestartet werden können, was wiederum bedeutet, dass diese verzögerten Tasks aus ihrem Plan fallen. Wird jedoch zu kurz in die Zukunft geplant, so wird die Planung unter Umständen zu früh aufgegeben. Kann ein Fahrzeug nicht verplant werden, so wird damit sehr wahrscheinlich das ganze Szenario lahmgelegt, da es keine Fehlerbehandlung gibt und somit dieses Fahrzeug einfach stehen bleibt. Der Aufwand der gesamten Planung kann jedoch durch Zusammenspiel der beiden oben genannten Werte[47] beeinflusst werden. Der Zeitaufschlag kann als Granularität verstanden werden. Wird die Granularität sehr niedrig gewählt, so muss die Anzahl der Versuche sehr hoch sein. Auf diese Weise kann ein dichterer Plan entstehen, was zu Zeitersparnissen bei den Vorgängen auf dem Hof führen kann. Wird der Zeitaufschlag dagegen sehr hoch angesetzt, müssen weit weniger Planungsvorgänge durchgeführt werden. Allerdings werden auf diese Weise Fahrzeuge direkt vom Start an stark verzögert und auch Lücken im Plan, die für ein zu verplanendes Fahrzeug genutzt werden könnten, werden unter Umständen nicht gefunden. Nach einigen Tests wurde die Entscheidung für einen Zeitaufschlag von 15 Sekunden mit 40 möglichen Wiederholungen getroffen. Das entspricht einer möglichen Verschiebung des

[46] vgl. Abschnitt 5.3

[47] die Anzahl der Versuche und der Zeitaufschlag

Starts um 10 Minuten. Die hohe Startverzögerung kommt kaum zum Tragen, da viele Kollisionen schon durch die Verringerung der Geschwindigkeit verhindert werden können.

Ein Problem stellte das Endpolygon eines Fahrauftrages dar. Dieses ist das Polygon, welches für den Zielpunkt in den Raum-Zeit-Plan eingetragen wird und als Endzeitpunkt den Wert *unendlich* trug[48]. Dieses Belegungspolygon bleibt so lange erhalten, bis ein möglicher Folgefahrauftrag geplant wird. Sollen beispielsweise mehrere Fahrzeuge nacheinander die gleiche Route mit Zwischenstopp fahren[49], dann kann durch die Reihenfolge der Planungsvorgänge möglicherweise nur das erste Fahrzeug verplant werden. Alle anderen könnten abgewiesen werden, sofern ihre Planungsanfragen gestellt werden, bevor der zweite Fahrauftrag des Vorgängerfahrzeugs eingeplant ist. Da das Konzept des Raum-Zeit-Plans noch nicht ausgereift ist und deshalb noch überarbeitet werden muss (was nicht Teil dieser Diplomarbeit ist), wurde entschieden, den Endzeitpunkt der Endpolygone nur zehn Sekunden in die Zukunft zu legen, wohl wissend, dass dies nicht die Endlösung des Problems darstellt.

Das Vorgehen bei erfolgloser Planung führte für den Anwendungsfall der Rollenden Landstraße ebenso zu Problemen. Statt auf den Fehler zu reagieren und diesen entsprechend seines Grundes zu behandeln, wurde ein Plan errechnet, welcher direkt startet und Kollisionen im Raum-Zeit-Plan ignoriert. Das hat zur Folge, dass sich das Fahrzeug - ausschließlich durch die Sicherheitsüberwachung kontrolliert - in mögliche Lücken drängt und damit andere Fahrzeuge zum Verlassen ihres Planes zwingt und Reihenfolgen durcheinanderbringt. Falls ein Fahrzeug nicht eingeplant werden kann, so erhält es jetzt auch keinen Fahrauftrag. Beide Varianten stellen keine Lösung des Problems dar, jedoch fällt es in der bisherigen Variante schwer zu erkennen, dass überhaupt ein Problem vorliegt. Dies äußert sich dann nur durch andere Probleme, deren Ursachen daraufhin erst gefunden werden müssen. Diese Änderung stellt also keinen Mehrwert dar, sondern erleichtert das Testen und Analysieren des Programms.

Zu größeren Schwierigkeiten führte ein Fehler, bzw. eine Lücke, im Konzept der Raum-Zeit-Planung. Diese betrifft auch den Zielpunkt einer Fahraufgabe. Bei der Erzeugung des Plans wird für jeden Punkt der Trajektorie geprüft, ob das Fahrzeug ein Hindernis schneidet. Es wird

[48] Das bedeutet, dass der Zielpunkt der Fahraufgabe vom betreffenden Fahrzeug, von dessen Ankunft an bis ins Unendliche, belegt wird, sodass kein anderes Fahrzeug diesen Punkt mehr passieren kann

[49] dazu ist die Einplanung zweier Fahraufgaben notwendig

jedoch nicht geprüft, ob das Fahrzeug an seiner Endposition einem anderen Fahrzeug, das bereits eingeplant ist, im Weg steht.

Angenommen zwei Fahrzeuge (A und B) wollen in die gleiche Wartespur. A wurde vor B geplant und hat deshalb auch seinen Haltepunkt vor B. B hat jedoch den kürzeren Weg. Im Raum-Zeit-Plan wird also (obwohl A bereits geplant ist) ein Zeitkorridor gefunden, in dem B zu seiner Endposition findet, ohne dass es zu einer Kollision kommt. Dieser Korridor würde dann mit der ursprünglichen Version des Leitstandes auch ausgewählt werden, obwohl er Fahrzeug A seinen Plan zunichtemacht, denn es gibt für A keine Möglichkeit innerhalb der Spur an B vorbei zu kommen.

Dieses Problem konnte ohne größeren Aufwand gelöst werden, indem nach dem Fund eines kollisionsfreien Plans die Endposition des Fahrzeuges gegen Kollisionen in der Zukunft geprüft werden. Ausgehend davon, dass Fahraufgaben nicht sehr weit in die Zukunft geplant werden, wird nur auf Kollisionen bis zu 20 Minuten nach Beendigung des Tasks geprüft. Kommt es zur Kollision, wird ebenso wie im ersten Planungsschritt die Startzeit erhöht und die Planung komplett neu versucht. Die Berechnungen zum Finden eines kollisionsfreien Korridors befinden sich in der Methode *calculateSpacetimePlan(..)* der Klasse *SpacetimePlanning*.

Ein weiteres Problem, das erst sehr spät zu beobachten war, betrifft die Geschwindigkeit der Raum-Zeit-Planung. Bei einem Szenario von etwa einer Stunde Länge, sollten nach einiger Zeit mehrere (knapp 15) Lkw auf den Zug aufgefahren werden. Die Fahraufträge werden direkt nacheinander an den Scheduler verschickt. Es war zu beobachten, dass ab dem Zeitpunkt der Planungsanfrage mehrere Minuten keine Tasks (weder solche, die mit dem Auffahren, noch solche, die mit diesem Vorgang nichts zu tun hatten) mehr gestartet wurden. Einige Zeit später wurden dann alle Tasks gleichzeitig gestartet - mit großer Verspätung. Bei der Analyse des Problems wurde festgestellt, dass der Speicherbedarf der Planungskomponente mit der Zeit stark zunimmt (5 MB zu Beginn, mehr als 90 MB nach einer Stunde). Beide Erscheinungen wiesen darauf hin, dass bei der Planung Objekte erzeugt, aber nicht gelöscht wurden. Bei diesen Objekten handelte es sich um die PlanningAllocationPolygons. Dadurch erhöhte sich die Dauer des Erzeugens und Durchsuchens von Schnappschüssen mit der Zeit so weit, dass bei den gegebenen Systemen das Finden eines Korridors teilweise einige (mehr als zehn) Sekunden dauerte.

Beim Erzeugen eines Schnappschusses in der Methode *getAllocationSnapshot(...)* der Klasse *SpacetimePlan* werden alle PlanningAllocationPolygons durchlaufen. An dieser Stelle lässt sich sehr einfach überprüfen, ob die Einträge noch von Belang sind oder nicht. Liegt der End-

zeitpunkt des Eintrags in der Vergangenheit, so kann der Eintrag gelöscht werden. Durch diese Änderung konnte in den folgenden Tests der Speicherbedarf im Schnitt konstant gehalten werden.

Die Änderungen an der Raum-Zeit-Planung im Zuge der Anpassung des Konzepts für Fahrerlose Transportsysteme an die Anforderungen der Rollenden Landstraße, dienten im Großen und Ganzen nur dazu, Fehler zu beheben oder Probleme zu beseitigen, nicht aber der Verbesserung des Konzeptes der Raum-Zeit-Planung. Sowohl für dieses, als auch für andere Anwendungsgebiete in denen die hier beschriebene Software zum Einsatz gebracht werden soll, sollte dieses Konzept deshalb überarbeitet und verbessert werden[50].

5.10 Modifikationen und Fehlerkorrekturen

Im Zuge der Anpassungen des Allgemeinen Leitstandes an den Anwendungsfall der Rollenden Landstraße mussten einige Änderungen an den Modulen vorgenommen werden. Diese erweitern zum einen deren Funktionalitäten aber korrigierten zum anderen auch vorliegende Fehler in den Implementierungen. Manche Verbesserungen konnten nicht immer vollständig durchgeführt werden, sondern bilden teilweise nur Übergangslösungen, die in folgenden Projekten nochmals aufgearbeitet werden müssen. Die durchgeführten Modifikationen werden nun im Folgenden erklärt.

5.10.1 Anpassung der Sicherheitspolygone

Bei der in diesem Abschnitt beschriebenen Änderung, handelt es sich nicht um die Behandlung eines Fehlers im Leitstand, sondern um eine Anpassung, die erst durch die Anforderungen auf dem, dieser Arbeit zugrunde liegenden, Gelände nötig wurden. Auf dem RoLa-Terminal Regensburg steht sehr wenig Platz zur Verfügung, weshalb die Fahrzeuge sehr dicht aneinander vorbei fahren, bzw. sehr dicht aneinander geparkt werden müssen.

Um Kollisionen zu verhindern, arbeitet die Fahrzeugsicherheit mit den sog. Sicherheitspolygonen, also mit Kontrollflächen, die bei fahrenden Fahrzeugen von der Front (bzw. bei rückwärtiger Fahrt vom Heck) in Fahrtrichtung erzeugt werden. Je schneller das Fahrzeug fährt, desto größer ist diese Fläche. Schneidet diese Fläche ein Hindernis, so wird das Fahrzeug gebremst, bzw. angehalten, wenn die Kollision anders nicht mehr zu verhindern ist. Diese Kontrollfläche ist kein dem

[50] vgl. Abschnitt 6.4

Fahrzeug entsprechendes Rechteck, sondern wird in Trapezform berechnet. Die Breite der Fläche nimmt mit dem Abstand zum Fahrzeug zu. In Abbildung 5.16[51] ist das Sicherheitspolygon in rot eingezeichnet.

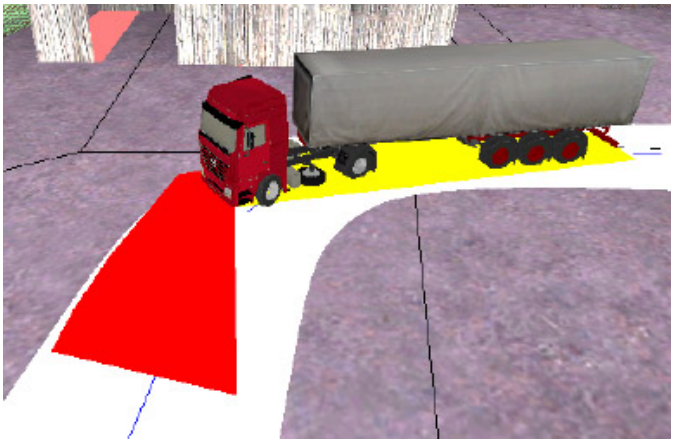


Abbildung 5.16: Belegungspolygone eines Fahrzeuges

Durch die Trapezform wird das Fahrzeug auch dann verzögert, wenn es knapp seitlich an Hindernissen vorbei fährt. Problematisch wird dies bei den Wartespuren. Diese sind direkt nebeneinander angeordnet, mit geringem Sicherheitsabstand zu den Seiten. Fährt ein Fahrzeug in einer Wartespur, so ragt das Sicherheitspolygon über diese Spur hinaus, bis in die direkt benachbarten Spuren hinein, wie in Abbildung 5.17 zu sehen. Befinden sich dort Fahrzeuge, so wird das betrachtete Fahrzeug abgebremst, bis dessen Polygon die Belegungs- (gelbe Fläche in Abbildung 5.16) und Sicherheitspolygone der anderen Fahrzeuge nicht mehr schneidet. Diese Verzögerung führt dazu, dass sich das Fahrzeug nicht mehr in seinem Zeitkorridor befindet.

Fährt das Fahrzeug gerade in die Spur ein, so wird dies kaum zu einem Problem führen. Anders verhält es sich jedoch, wenn das Fahrzeug aus der Spur ausfährt. Soll ein abfahrender Zug mit Lkw beladen werden, so werden annähernd gleichzeitig alle wartenden Fahrzeuge verplant, sodass sich diese in enger Abfolge in Bewegung setzen. Dabei wird[52] Spur für Spur geleert. Direkt nach dem letzten Fahrzeug der ersten Spur verlässt dann das erste Fahrzeug der zweiten Spur seinen Stehplatz. Das letzte Fahrzeug der ersten Spur muss jedoch einen langen Weg mit Hindernissen neben sich abfahren und wird dadurch einige Zeit verzögert. Da auf das Verlassen des Raum-Zeit-Plans durch ein Fahrzeug in der aktuellen Version des Leitstandes noch nicht

[51] vgl. [LuT], Abschnitt 2.3, Seite 40

[52] von der Trennung der Überlängefahrzeugen von den normalen Fahrzeugen abgesehen

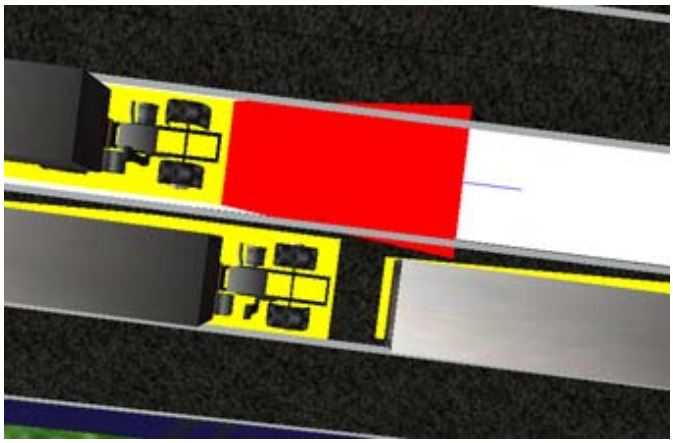


Abbildung 5.17: Sicherheitspolygon mit altem Trapezfaktor

reagiert wird, kann es passieren, dass das erste Fahrzeug der zweiten Spur startet, bevor das andere Fahrzeug es passiert hat. Kurz nach den Wartespuren werden die Fahrzeuge aller Spuren auf die gleiche Leitlinie geführt. Es gibt dann keine Möglichkeit mehr, die korrekte Reihenfolge wiederherzustellen.

Es gibt mehrere Möglichkeiten dieses Problem zu behandeln. Eine wäre es, die maximale Geschwindigkeit an solchen Engstellen herabzusetzen. Dies wäre auch eine sinnvolle Lösung, allerdings bewegen sich die Fahrzeuge bisher überall nur mit *10 km/h*, also nur knapp über Schrittgeschwindigkeit. Ein weiteres Herabsetzen der Geschwindigkeit schien auf Grundlage der vorhandenen Abstände nicht notwendig. Aus diesem Grund wurde entschieden das Problem vorerst zu lösen, indem der Faktor verringert wird, um den das Sicherheitspolygon aufgefächert wird. Dieser Wert wird im Attribut *m_safetyTrapezoidFactor* der Klasse *VehicleReferencePointCalculation* gespeichert. Er wird im Konstruktor mit einem Standardwert vorbelegt. Das auf dem geänderten Trapezfaktor basierende Sicherheitspolygon ist in Abbildung 5.18 zu sehen. Die Grundüberlegung der Trapezform ist sicherlich richtig, da die Fahrzeuge nicht zu schnell durch Engstellen fahren sollten, jedoch sollte der richtige Trapezfaktor gefunden werden. In Falle dieser Arbeit wurde er durch Tests ermittelt. Die Möglichkeit, dass dieser aus Sicherheitsgründen jetzt zu niedrig ist, ist jedoch gegeben.

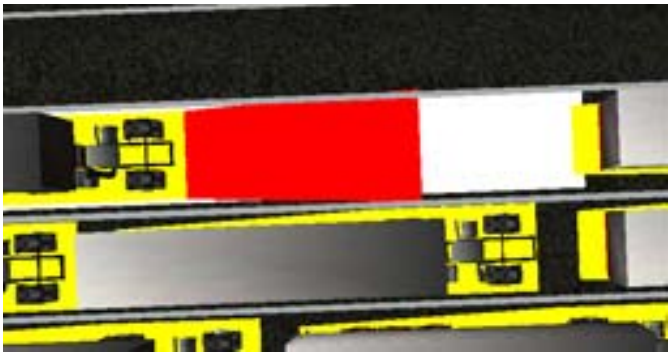


Abbildung 5.18: Sicherheitspolygon mit geändertem Trapezfaktor

5.10.2 Geschwindigkeitsanpassung bei Rückwärtsfahrt im Falle eines Hindernisses

Im Laufe einiger Tests konnte wiederholt beobachtet werden, dass Fahrzeuge in Rückwärtsfahrt bei einer drohenden Kollision[53] durch die Fahrzeugsicherheit nicht rechtzeitig zum Stehen gebracht werden konnten. Es schien sogar so, als ob sie, nachdem die drohende Kollision bemerkt wurde, kurzzeitig beschleunigten. Dieser Effekt trat oft zutage, wenn Traktoren in Traktorspuren eingefahren sind, und blieb bisher bei EZsped wahrscheinlich verborgen, da dichtes rückwärtiges Fahren dort eher unüblich war. Der Fehler lag in der Klasse *CollisionPrevention* der Kontrollkomponente. Dort wurde aufgrund eines Vorzeichenfehlers in der Funktion *computeSpeed(...)*, die dafür zuständig ist, die neue Geschwindigkeit eines Fahrzeuges zu berechnen, das sich einem Hindernis nähert, die Kollision zu „spät“ wahrgenommen. In Listing 5.2 auf der nächsten Seite wird auf Grundlage des Betrages der aktuellen Geschwindigkeit (Zeile *2*) in Zeilen *8ff* ein Sicherheitspolygon für das Fahrzeug erzeugt, dessen nächste Geschwindigkeit ermittelt werden soll. Dieses Sicherheitspolygon zeigt jedoch in die falsche Richtung, da es eine positive Geschwindigkeit als Grundlage besitzt. Deshalb wird im folgenden Test (Zeile *14*) keine Kollision erkannt. Die Geschwindigkeit wird vorerst heraufgesetzt und erst durch die Abfrage der Richtung in Zeile *23* wird für den nächsten Schleifendurchlauf ein Sicherheitspolygon errechnet (Zeile *25*), das in die richtige Richtung zeigt.

[53] Eine drohende Kollision wird erkannt, wenn sich das Sicherheitspolygon eines Fahrzeuges mit einem Polygon eines anderen Objektes schneidet

Listing 5.2: Auszug aus *computeSpeed(...)* der Klasse *CollisionPrevention*

```
[...]
double calculatedSpeed  = fabs(v->getCurrentSpeed());
// Make sure, the calculated speed lies between the two borders
if(calculatedSpeed > tempCeilingSpeed)
        calculatedSpeed = tempCeilingSpeed;
Allocationpolygon::Zeiger safety = new Allocationpolygon();
safety->referenzEntfernen();
safety->setVertices(
 m_vehiclereferencepointcalculation->
 computeSafetyVertices(v, calculatedSpeed));
while((fabs(tempCeilingSpeed - calculatedSpeed) > 0.02)
 || (fabs(tempFloorSpeed - calculatedSpeed) > 0.02))
{
        if( m_collisiontester->testForCollision(safety, otherVehicle) )
        {
                [decrease speed]
        }
        else
        {
                [increase speed]
        }
        double testspeed = calculatedSpeed;
        if(!forward)
                testspeed *= -1.0;
        safety->setVertices(
         m_vehiclereferencepointcalculation->
         computeSafetyVertices(v, testspeed));
}
[...]
```

In Listing 5.3 auf der nächsten Seite ist ein Ausschnitt der verbesserten Version zu sehen. Dort wurde die Variable *testspeed* aus der *while*-Schleife herausgezogen und (mit dem richtigen Vorzeichen versehen (Zeile *8*)) als Grundlage für die Berechnung des Sicherheitspolygons herangezogen.

Listing 5.3: Auszug aus der verbesserten Version von *computeSpeed(...)*

```
1 [...]
  double calculatedSpeed  = fabs(v->getCurrentSpeed());
3 // Make sure, the calculated speed lays between the two borders
  if(calculatedSpeed > tempCeilingSpeed)
5         calculatedSpeed = tempCeilingSpeed;
  double testspeed = calculatedSpeed;
7 if(!forward)
          testspeed *= -1.0;
9 Allocationpolygon::Zeiger safety = new Allocationpolygon();
  safety->referenzEntfernen();
11 safety->setVertices(
   m_vehiclereferencepointcalculation->
13  computeSafetyVertices(v, testspeed) );
  [...]
```

5.10.3 Routenplanung

Die Routenplanung stellte gleich von Beginn an eine Reihe von Problemen dar, die im Rahmen dieser Arbeit unbedingt gelöst werden mussten, um einen sinnvollen Einsatz auf dem Terminalgelände überhaupt zu ermöglich. So stellte sich bei den ersten Simulationen heraus, dass die Auswahl der Leitlinienkomponenten in der Nähe der Start- sowie der Endposition fehlerhaft war. Alle gefundenen Komponenten für eine mögliche Route wurden ungefiltert übernommen. Dies führte dann zu Problemen, wenn zwischen Start- und Zielposition eine Route mit zwei möglichen Fahrtrichtungen von A* gefunden wurde, zum einen in Polygonzug vorwärts und zum anderen in Polygonzug rückwärts. Da die Routenplanung analog zu allen anderen Modulen beispielhaft implementiert worden ist, wurde im Anschluss stets die erste gefundene Route gewählt, falls es mehrere Alternativen gab. Da, wie bereits gesagt, keine Filterung der Komponenten vorgenommen wurde, konnte nun die erste von A* gefundene Route aber eine Route sein, die das Fahrzeug aufgrund seiner Ausrichtung nicht befahren kann, da es dazu auf der Stelle hätte drehen müssen. Abbildung 5.19 verdeutlicht dieses Problem. Das Fahrzeug soll hier von seiner Startposition rückwärts entlang der Leitlinien zu seiner Zielposition fahren. Die Leitlinien dürfen sowohl in Polygonzug vorwärts (blaue Komponenten), als auch in Polygonzug rückwärts (rote Komponenten) befahren werden. Da keinerlei Filterung stattfindet, findet A* zunächst die Route mit der

Fahrtrichtung vorwärts, danach die Route mit der Fahrtrichtung rückwärts. Die Ursache liegt in der Reihenfolge der Kombinationen begründet, die A* zum Test vorgegeben werden. Hier wird stets mit den Komponenten begonnen, die in Fahrtrichtung vorwärts befahren werden. Daher findet A* die für das Fahrzeug falsche Route zuerst, die im Anschluss ausgewählt wird, um eine mögliche Trajektorie zu berechnen. Diese Berechnung scheitert, da das Fahrzeug dazu zweimal auf der Stelle hätte drehen müssen und die gesamte Planung wird mit einem Fehler beendet. Da es keinerlei Fehlerbehandlung gibt, die in einem solchen Falle eine alternative Route auswählt, wird die gesamte Planung der Fahraufgabe abgebrochen.

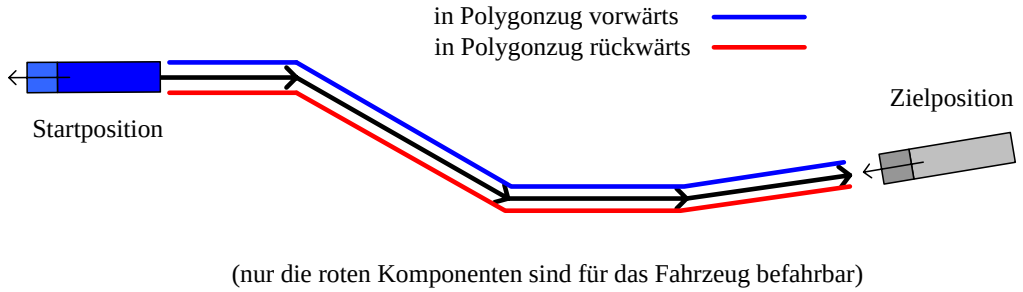


Abbildung 5.19: Auswahl der Leitlinienkomponenten

Um dies zu verhindern, wurden die gefundenen Leitlinienkomponenten gefiltert, noch bevor diese an A* zu einer möglichen Routenplanung weitergeleitet werden. Dazu wurde die aktuelle Ausrichtung des Fahrzeuges mit in den Filter einbezogen, sodass anhand dessen die Auswahl der möglichen Komponenten auf die wirklich fahrbaren beschränkt werden konnte. Abbildung 5.20 zeigt die Vorgehensweise. Das Fahrzeug steht hier in der Nähe einer Leitlinie mit den vier möglichen Komponenten in Polygonzug vorwärts, in Polygonzug rückwärts, gegen Polygonzug vorwärts sowie gegen Polygonzug rückwärts. Anhand der Ausrichtung des Fahrzeuges ergeben allerdings nur die Komponenten in Polygonzug vorwärts sowie gegen Polygonzug rückwärts Sinn, da zum Befahren der beiden übrigen das Fahrzeug seine aktuelle Position um 180 Grad verändern müsste. Dies ist einer der einfachen Fälle einer möglichen Filterung. Komplizierter wird es, wenn Start- sowie Zielposition der gleichen Leitlinienkomponenten zugeordnet werden, z. B. dann, wenn das Fahrzeug lediglich wenige Meter vorfahren oder zurücksetzen soll. Auch an dieser Stelle kann über die oben genannte Filterung zunächst die Auswahl der Komponenten auf zwei reduziert werden. Dennoch findet A* in beiden Fällen eine mögliche Route, von denen jedoch nur eine für das Fahrzeug befahrbar ist. Auch an dieser Stelle musste eine gesonderte Filterung über

die Fahrzeugausrichtung vorgenommen werden. Diese entscheidet, ob das Ziel vor oder hinter dem Fahrzeug liegt und beschränkt somit die Auswahl auf nur eine einzige Komponente. Beide Filter sind nun Teil der Routenplanung und in der Klasse *RouteDetermination* implementiert.

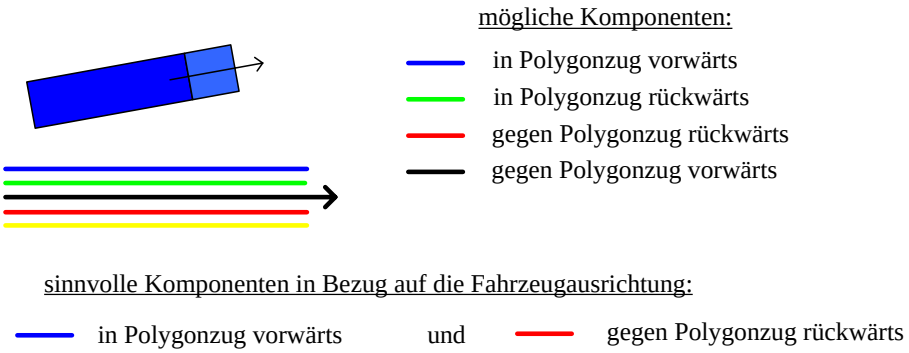


Abbildung 5.20: Filterung der Komponenten über die Fahrzeugausrichtung

Nachdem die Filter realisiert wurden, traten weitere Probleme auf, die sich ergaben, wenn aus der gefundenen Route eine fahrbare Trajektorie berechnet werden sollte. Lag der Fahrweg auf lediglich einer Leitlinienkomponente, z. B. wenn das Fahrzeug wenige Meter vorwärtsfahren sollte, so wurde die Trajektorie stets bis zum Ende der Leitlinie fortgesetzt. Dies lag an einer fehlerhaften Verkürzung der Leitlinie in Bezug auf die Endposition, die lediglich dann vorgenommen wurde, wenn der Weg aus mehreren und nicht aus nur einer einzigen Komponente bestand. Dies wurde an dieser Stelle korrigiert.

Auch die eigentliche Trajektorienberechnung musste im Laufe der Arbeit optimiert werden. So kam es zu dem Problem, dass die Trajektorien in unmittelbarer Nähe zum Start- sowie Zielpunkt, wie in Abbildung 5.21 dargestellt, oszillierten. Dies zog zahlreiche Probleme mit sich, da die Trajektorien auch die Grundlage der Korridore bilden und diese somit fehlerhaft berechnet werden. Ebenso konnten die Fahrzeuge diese Trajektorie unmöglich abfahren. Die Korrektur wurde an dieser Stelle dankenswerterweise von Philipp Wojke durch ausführliches Testen und Simulieren übernommen.

Wie bereits in Abschnitt 4.4.3 erwähnt, sieht die Routenplanung keinerlei Auswahl einer Route durch eine Heuristik vor und reagiert im Fehlerfall mit der Beendung der Planung. Dies sind notwendige Potenziale, die im Rahmen dieser Arbeit nicht geleistet werden konnten, aber für

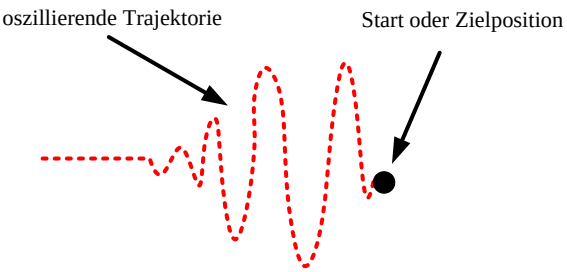


Abbildung 5.21: Oszillation der Trajektorie

zukünftige Projekte durchaus relevant werden könnten und somit in weiteren Arbeiten realisiert werden sollten.

Kapitel 6

Beurteilungen

Die Grundlagen dieser Arbeit sollen nun im folgenden Kapitel anhand verschiedener softwaretechnischer Gesichtspunkte ausführlich bewertet werden, sodass für weitere Projekte mit der gleichen Basis ein hoher Mehrwert gewonnen werden kann. Im Rahmen dieser Arbeit wurde ebenso eine Mechanik entwickelt, die als Richtlinie für die Modifikation des Leitstandes an weitere Anwendungsfälle dienen kann. Diese soll kurz vorgestellt werden. Abschließend soll der mögliche Einsatz des Leitstandes für autonomes Fahren im Fallbeispiel des Terminals Regensburg beurteilt, sowie ein kurzer Überblick über die in Zukunft anstehenden Arbeiten gegeben werden.

6.1 Softwaretechnische Gesichtspunkte

In diesem Abschnitt soll Bezug genommen werden auf die in Abschnitt 1.2.2.1 genannten softwaretechnischen Qualitätseigenschaften. Dabei wird die Leitstandssoftware in der Form, wie sie am Ende dieser Arbeit vorliegt, aber auch die zugrunde liegende Bibliothek EZauto zum Teil nach allen in diesem Kapitel genannten Kriterien beurteilt. Die Grundlage dieser Beurteilungen sind die Erfahrungen, die während der kompletten Arbeit am Projekt EZrola gemacht worden sind. Aus diesem Grund ist eine Beurteilung von EZauto in einigen Punkten nur bedingt oder gar überhaupt nicht möglich. Es wurden keine Tests oder Mechanismen durchgeführt, die rein auf diese Beurteilung abzielten. Dabei wird auch darauf hinweisen, dass nur beschränkte Mittel zum Testen der Software zur Verfügung standen, sodass vor allem die Beurteilung der Effizienz schwer fällt. Auch fehlt an dieser Stelle ein Vergleich mit ähnlicher Software, sodass in diesem Punkt nur

auf Grundlage persönlicher Einschätzungen und Erwartungen eine Bewertung erfolgen konnte. Die in den folgenden Abschnitten angebrachte Kritik, soll nicht davon freisprechen, dass ähnliche softwaretechnische Fehler nicht auch im Rahmen dieser Arbeit begangen worden sein könnten. Am Ende dieses Abschnitts wird auf Grundlage der einzelnen Beurteilungen eine Gesamtbewertung vorgenommen, die eine persönliche und qualitative Einschätzung der Leitstandssoftware widerspiegelt.

6.1.1 Wiederverwendbarkeit

Jedes Softwarekonzept oder jede Softwarebibliothek steht oder fällt mit dem Schlagwort der Wiederverwendbarkeit, denn nur durch diese ist der Einsatz in weiteren Projekten möglich. Daher muss die Beurteilung der Wiederverwendbarkeit von Software mit höchster Konzentration sowie Präzision geschehen, um ein vorschnelles Urteil zu vermeiden und keine Details, die oftmals großen Ausschlag geben können, zu übersehen.
Die Bewertung der Wiederverwendbarkeit der dieser Arbeit zugrunde liegenden Software wird im Folgenden in zwei Einzelbewertungen aufgeteilt, zum einen wird die Wiederverwendbarkeit der EZauto-Bibliothek und zum anderen die des Allgemeinen Leitstandes betrachtet.

6.1.1.1 EZauto-Bibliothek

Die EZauto-Bibliothek stellt in vielen Bereichen die Grundlagen dieser, aber auch weiterer Arbeiten innerhalb der Forschungsgruppe, dar. Zentrale Elemente konnten der Bibliothek entnommen und ohne große Modifikationen eingesetzt werden. Sie liefert viele Module und Funktionen, die uneingeschränkt in den unterschiedlichsten Projekten eingesetzt werden können. An dieser Stelle kann nur ein sehr kleiner Ausschnitt der gesamten Bibliothek bewertet werden, da bei weiten nicht alle Elemente in dieser Arbeit zum Einsatz kommen mussten. Zur Bewertung liegt der Fokus dabei nur auf den wesentlichen verwendeten Funktionalitäten, um nicht zu sehr ins Detail abzurutschen.

Zentrale Bedeutung übernimmt der Einsatz des Ereignisverteilerkonzeptes, welcher zur Einhaltung der Trennung der Belange zwischen den einzelnen Leitstandmodulen enorm beiträgt, da er die Schnittstelle zur Datenkommunikation darstellt. Alle Module können ausschließlich über diesen miteinander kommunizieren. Dies gewährleistet, dass jedes Modul ohne weit reichende Än-

derungen der anderen Module ausgetauscht und ersetzt werden kann. Damit zeugt dieses Konzept nicht nur von einer hohen Wiederverwendbarkeit, sondern trägt auch zur formalen Einhaltung von Programmierrichtlinien bei.

Für jede Simulation ist die Darstellung der Fahrzeuge von großer Bedeutung. Auch diese konnte der EZauto-Bibliothek entnommen und in der Umgebung des Leitstandes ohne große Modifikationen eingesetzt werden. Dabei ist sowohl die Simulation als auch die Beschreibung der Fahrzeuge über XML-Datenbanken in den unterschiedlichsten Anwendungen einsetzbar. Lediglich anwendungsfallspezifische Änderungen mussten an dieser Stelle vorgenommen werden. Zum einen betrafen sie die Fahrzeugtransformationen, die bisher nicht in der Simulation integriert waren, und zum anderen verschiedene Parameter bezüglich der Fahrpräzision der simulierten Fahrzeuge. Die Erzeugung des nicht vorhandenen Terminaltraktors verlief dagegen problemlos.

Ein weiteres aus der EZauto-Bibliothek eingesetztes Hilfsmittel war die Speicherverwaltung mittels eines Smartpointer-Konzeptes. Auch dieses konnte ohne Probleme verwendet werden und sollte auch in weiteren Projekten die Implementierungen unterstützen. Neben diesen zentralen Elementen, konnten der Bibliothek aber auch sehr viele Funktionen zur Berechnung unterschiedlichster Anforderungen entnommen werden. Hier sei die Trajektorienberechnung nur als eines von vielen Beispielen genannt. Diese konnte im Projekt EZsped ohne Probleme eingesetzt werden, musste dagegen im Projekt EZrola an einigen Stellen optimiert werden, was deren Wiederverwendbarkeit aber nur steigerte.

6.1.1.2 Allgemeiner Leitstand

Das theoretische Grundkonzept des Allgemeinen Leitstandes, wie es in Abschnitt 4.1 vorgestellt wurde, zeugt von einer hohen Wiederverwendbarkeit. Die gewählte Architektur der einzelnen Module, verbunden über das Ereignisverteilerkonzept, sollte genügend Spielraum bieten, die Software an unterschiedliche Anwendungsfälle anpassen zu können. Dabei spielt die Möglichkeit des Austauschens ganzer Module eine große Rolle, da somit einzelne Optimierungen vorgenommen werden können oder ein vollkommen neues Modul integriert werden kann. Auch die vorgesehenen Funktionalitäten der einzelnen Module sollten die Architektur für eine Vielzahl von Anwendungsfällen in den Bereichen des autonomen Fahrens einsetzbar machen. Dabei lässt sich die Umgebung des Leitstandes durch den Einsatz externer Module und deren Anbindung über den externen Er-

eignisverteiler realisieren.

Wie in dieser Arbeit gezeigt worden ist, konnte das Konzept des Allgemeinen Leitstandes des Projektes EZsped durchaus im Projekt EZrola praktisch wieder verwendet werden. Dabei stellen jedoch der aktuelle Grad der Implementierung sowie die dazu vorliegenden Dokumentationen[1] einige Einschränkung dar, die nur durch einen hohen Arbeitsaufwand kompensiert werden konnten. Viele Funktionen der einzelnen Module sind nur ansatzweise implementiert, sodass theoretisch zugeordnete Funktionalitäten zu diesem Zeitpunkt praktisch nicht geleistet werden können und in weiteren Arbeiten integriert werden müssen. Dies ist bei einem Projekt mit diesem Umfang durchaus verständlich und so konnten die Defizite selbst in dieser Arbeit nur in Bezug auf die für diesen Anwendungsfall notwendigen Funktionalitäten beseitigt werden. Daher werden bei der weiteren Verwendung zunächst stets fehlende Funktionalitäten am Kern des Leitstandes implementiert werden müssen, sofern der jeweilige Anwendungsfall diese vom System verlangt. Um dies Umsetzen zu können, bedarf es ausführlicher Kenntnisse aller Abläufe innerhalb des Leitstandes, die nur durch eine intensive Einarbeitungsphase zu erlangen sind. Neben den eventuellen Modifikationen am Kern, kann der Implementierungsaufwand in allen anwendungsnahen Modulen ebenso recht hoch sein. Alle externen Module sowie die Auftragsverwaltung, welche die Schnittstelle des Leitstandes zum Anwendungsfall darstellt, müssen teilweise komplett neu konzipiert und implementiert werden.

Das Gesamtkonzept des Allgemeinen Leitstandes stößt dann an seine Grenzen, wenn es um die Modellierung dreidimensionaler Gelände geht. Derzeit wird das zugrunde liegende Gelände ausschließlich als eine Ebene aufgefasst und als solche von allen Modulen verwaltet. Damit würden Anwendungsfälle, die z. B. Parkdecks oder Schiffe enthalten, bei denen auf mehreren Etagen gefahren werden kann, ausgeschlossen. Die Erfassung der fehlenden dritten Dimension würde einen enormen Implementierungsaufwand erfordern.
Ebenso bietet der Allgemeine Leitstand momentan keine Kommunikationsmöglichkeit für ein mögliches Netzwerk von mehreren Leitständen und ist damit immer nur eine Einzellösung für einen bestimmten Anwendungsfall.

Aufgrund aller genannten Tatsachen ist eine Wiederverwendbarkeit des Konzeptes des Allgemeinen Leitstandes in der Theorie und in den genannten Grenzen durchaus gegeben, welche

[1] vgl. Abschnitt 6.1.2.1

lediglich Einschränkungen in der praktischen Umsetzung hinnehmen muss. Diese können aber durch die kontinuierliche Weiterentwicklung und sukzessive Optimierung der einzelnen Module beseitigt werden. Nur so kann der notwendige Implementierungsaufwand in Grenzen gehalten werden.

6.1.2 Wartbarkeit

Die Wartbarkeit von Software ist als Maß zu verstehen, welches Auskunft darüber gibt, wie viel Aufwand notwendig ist, um Änderungen oder Erweiterungen an der betreffenden Software vorzunehmen. Wie Abschnitt 1.2.2.1 zu entnehmen ist, wird dieses Kriterium in der Softwaretechnik in die Unterkriterien Verständlichkeit, Flexibilität und Validierbarkeit untergliedert. Diese drei Kriterien werden im Folgenden bewertet.

6.1.2.1 Verständlichkeit

Verständlichkeit ist der Umfang, in dem es ein System anderen Personen als den Entwicklern erlaubt, seine Struktur, Annahmen und Einschränkungen zu erfahren. Im Laufe dieser Arbeit musste der Entwicklungsprozess vom reinen Anwender hin zum Systementwickler vollzogen werden, welcher nur durch eine intensive Einarbeitungsphase möglich war. Während diesem standen verschiedene Dokumente zur Verfügung, die in diesem Abschnitt getrennt betrachtet werden soll.

Zuerst sollen die begleitenden Dokumente analysiert werden. Dazu zählen Handbücher, Diagramme und die Abschlussdokumente vorangehender Arbeiten. Handbücher, in denen ausschließlich die Funktionsweise der Software beschrieben wird, standen nicht zur Verfügung. Sie wären oft von Vorteil gewesen, jedoch ist das Erzeugen und Pflegen von Handbüchern mit sehr viel Zeitaufwand verbunden, weshalb auch nach Abschluss dieser Arbeit keine Handbücher vorliegen. Auch was Diagramme angeht, kann kaum ein besseres Urteil gefällt werden. Klassendiagramme liegen zwar vor, deren Pflege ist jedoch ebenfalls sehr zeitaufwendig. Sie müssten immer parallel zur Softwareentwicklung gepflegt werden, da ansonsten sehr schnell der Überblick verloren geht. Zu einem späteren Zeitpunkt müsste dann die komplette Klassenstruktur auf Konsistenz überprüft werden. Sonstige eigenständige Diagramme, wie beispielsweise Sequenzdiagramme, die dem Verständnis von Abläufen dienen, liegen nicht vor. An begleitenden Dokumenten bleiben noch die Abschlussdokumente der Arbeiten, auf die diese Arbeit aufsetzt. Dazu ist vor allem die Arbeit

von René M. Lotz und Vanessa Thewalt[2] zu zählen, da sie Grundlage der vorliegenden Arbeit ist. Dieses Dokument liefert einen guten Überblick über die Gesamtstruktur des Leitstandes. Darin sind auch einige Diagramme enthalten, welche die Zusammenhänge und Abläufe zum Teil sehr gut veranschaulichen. Trotzdem kann solch ein Dokument kein Handbuch ersetzen, da es von der eigentlichen Software zu weit abstrahiert. Außerdem lag diese Arbeit erst zur Einsicht vor, nachdem die grobe Einarbeitungsphase durchlaufen war und konnte somit bei späteren Problemen im Zuge der Anpassung der Leitstandssoftware an die Rollende Landstraße nicht mehr weiterhelfen. Auch die vorliegende Arbeit gibt bestenfalls einen groben Überblick über die Zusammenhänge und vermittelt nur an einigen ausgewählten Stellen tiefes Detailwissen. Für Folgearbeiten wäre eine komplette Beschreibung des Leitstandes wünschenswert, da diese eine größere Detailtiefe aufweisen könnte und es wäre nicht nötig verschiedene Abschlussdokumente der Vorgängerarbeiten zu studieren.

Als Nächstes soll der Quelltext selbst in Bezug auf die Verständlichkeit bewertet werden. Dazu zählen der Programmcode sowie dessen Kommentierung. Auch der Programmcode soll wiederum unter verschiedenen Gesichtspunkten beurteilt werden, angefangen bei der Struktur. Im gesamten Projekt gibt es einige strukturelle Konventionen, was Einrückungen und ähnliche Dinge angeht. Diese Konventionen erhöhen die Lesbarkeit und erleichtern damit das Einarbeiten in den Programmcode. Sie werden in EZauto sehr strikt eingehalten und auch in der Leitstandssoftware lässt sich wenig Kritik anbringen. Das Entwurfsprinzip „Trennung der Belange“ ist auf Klassenebene gut durchgehalten. Bei Methoden ist es an einigen Stellen verbesserungswürdig. Die Aufteilung von großen funktionalen Blöcken in kleinere Methoden (nach dem Motto „teile und herrsche“) erhöht jedoch gerade auf dieser Ebene enorm die Lesbarkeit.

Einige Methoden, vor allem Getter- und Settermethoden, werden direkt in der Headerdatei implementiert, was das Auffinden teilweise verzögert. Dieser Bruch erspart jedoch bei Implementierungen Zeit. Das Auffinden von Methoden wird eher durch die mangelnde Ordnung der Methoden in den Dateien erschwert. Jedoch ist es fraglich, wie diese Ordnung aussehen soll. Es gäbe die Möglichkeit der alphabetischen Ordnung. Ebenso könnten die Methoden auch nach Sinnzusammenhang sortiert werden, wobei es jedoch zu kompliziert sein könnte klare Kriterien zu definieren, wann ein Sinnzusammenhang besteht. Auch muss dazu gesagt werden, dass das Auffinden von Funktionen durch alle höheren Entwicklungsumgebungen unterstützt wird.

Insgesamt wurde, soweit dies möglich war, der Eindruck gewonnen, dass die EZauto-Bibliothek

[2] [LuT]

strukturell sehr übersichtlich implementiert worden ist, die Leitstandssoftware jedoch einige Mängel aufweist, die aber keine negative Gesamtbeurteilung rechtfertigen sollten.

Als Nächstes soll der Programmcode hinsichtlich seiner Intuition beurteilt werden. Die Frage ist also, ob die Aufeinanderfolge der Anweisungen des Programms so gestaltet ist, wie es zu erwarten gewesen wäre. Hier eine Wertung anzubringen ist aus mehreren Gründen schwierig. Bei einem Projekt dieser Größenordnung kann beispielsweise keine für alle Teile stimmige Aussage getroffen werden. Außerdem steht sich die Verständlichkeit oft mit der Effizienz gegenüber, weshalb es an einigen Stellen von Vorteil ist, weniger intuitiv zu programmieren. Auch hier gibt es keinen Grund Kritik anzubringen.

Ein weiteres Kriterium, welches sich auf den Programmcode anwenden lässt, ist die Namensgebung von Klassen, Attributen, Methoden und Variablen. Auch hier sollte beim Lesen klar sein, welche Bedeutung ein Objekt hat, bzw. welche Funktionalität von einer Methode erwartet werden kann, auch wenn dies längere Bezeichner bedeutet. In diesem Fall lässt sich ein positives Urteil aussprechen, jedoch gibt es Unstimmigkeiten in der Sprachwahl. EZauto ist komplett in Deutsch gehalten, während die Leitstandssoftware englische Bezeichner verwendet. Die Visualisierung[3] vermischt die Sprachen gar, was auf jeden Fall vermieden werden sollte. Eine Übersetzung der EZauto-Bibliothek in die englische Sprache wäre daher eine Überlegung wert, um für einen möglichen internationalen Gebrauch gerüstet zu sein. Der Aufwand dabei ist sicherlich enorm hoch, da auch alle darauf aufbauenden Softwareprojekte angepasst werden müssten, falls diese von Änderungen an EZauto profitieren sollen.

Der andere Teil des Quelltextes, der ebenso bewertet werden soll, sind die Kommentare. Sie sind dazu geeignet, in hohem Maße die Verständlichkeit von Programmen zu erhöhen. Zu den Kommentaren zählen auch die *Doxygen*[4]-Kommentare, obwohl diese ebenfalls bei den ergänzenden Dokumenten hätten eingeordnet werden können. Sie ersetzen jedoch in einigen Fällen die üblichen Quelltextkommentare, weshalb sie an dieser Stelle behandelt werden. Zuerst werden jedoch die Kommentare betrachtet, welche die Funktionsweise des Programmcodes innerhalb von

[3] [Ses]

[4] Bei Doxygen handelt es sich um ein frei zugängliches Software-Dokumentationswerkzeug, mit dessen Hilfe Kommentare im Quelltext zu einer externen Dokumentation extrahiert werden können. Es wurde von Dimitri van Heesch entwickelt und kann von *http://www.stack.nl/~dimitri/doxygen/download.html#latestsrc* in der Version 1.5.1 (Stand 06.01.2007) heruntergeladen werden.

Methoden beschreiben. Sie sollten knapp und sachlich, aber dennoch verständlich und möglichst vollständig formuliert werden, vor allem bei komplexen Programmgebilden oder Methoden, die nicht einfach intuitiv verstanden werden können. So lagen zwar einige Kommentare vor, wären aber an manchen Stellen in zahlreicherer Form wünschenswert gewesen. Auch musste festgestellt werden, dass in der Leitstandssoftware, die wie oben beschrieben in Englisch gehalten ist, noch deutsche Kommentare vorkommen. Diese sollten langfristig auf jeden Fall entfernt werden.

Doxygen-Kommentare bedeuten vor allem bei umfangreichen Softwaresystemen eine enorme Zeitersparnis. Gerade bei einem fehlenden Handbuch sind sie der letzte Weg, um herauszufinden, welche Funktion eine Klasse, Methode oder ein Attribut hat, ohne sich näher mit dem Code auseinanderzusetzen. Sie sollten verständlich und nicht umgangssprachlich verfasst werden. Vor allem aber sollten sie vollständig sein. Gerade an dieser Stelle ist die Leitstandssoftware, im Gegensatz zu EZauto, zu kritisieren. In der Version des Leitstandes, die für diese Arbeit übernommen worden ist, sind beispielsweise mehr als 80 % der Klassen ohne eine Beschreibung über ihre Bedeutung oder Verwendung. Oft kann dann nur über Betrachtung der Attribute eine Aussage über die Funktionalität getroffen werden, ohne den betreffenden Programmcode nachvollziehen zu müssen. Das Untersuchen des Quelltextes bedeutet ein Vielfaches des Zeitaufwands im Vergleich zu Fällen, in denen Doxygen-Kommentare vorliegen. Doch auch wenn die Doxygen-Kommentare vorhanden sind, sind sie in vielen Fällen unzureichend und können wichtige Fragen nicht beantworten. Über die Güte der Doxygen-Kommentare in EZauto kann nur schwer eine Aussage getroffen werden, da sich mit dieser Bibliothek nicht in dem Maße auseinandergesetzt worden ist, wie es mit der Leitstandssoftware geschehen ist. Einen Grund zur Kritik gibt es an dieser Stelle jedoch nicht.

Doxygen stellt ein hervorragendes Mittel dar, Software zu dokumentieren, sofern dies von den Programmierern durchgehalten wird. Eine Weiterführung und Verbesserung der vorhandenen Kommentare wäre daher sehr zu empfehlen. Auch wenn dies viel Zeit kostet, erhöht es doch die Wartbarkeit und Qualität der Software. Außerdem könnte insgesamt die Zeitersparnis bei Folgeprojekten die Kosten an Zeit für das Verfassen der Kommentare deutlich übersteigen.

6.1.2.2 Flexibilität

Unter Flexibilität wird in der Softwaretechnik der Umfang verstanden, in dem Veränderungen eines Systems erleichtert werden. An dieser Stelle wurden unterschiedliche Erfahrungen gemacht. Je tiefer die Veränderungen in das System eingegriffen haben, desto komplexer wurde es, was jedoch auch zu erwarten war. Der Grund dafür war in den meisten Fällen nicht, dass das System

diese Änderungen nur schwer zugelassen hätte, sondern dass durch fehlende Dokumentationen das Auffinden der zu ändernden Stellen erheblich erschwert wurde. In manchen Fällen kam jedoch hinzu, dass Änderungen nicht vorherzusehende Seiteneffekte hatten. Dies führte oftmals zu groben Fehleinschätzungen, was den bevorstehenden Aufwand betraf.

Was oberflächlichere Veränderungen angeht, sehen die gemachten Erfahrungen ganz anders aus. Gerade durch die Konzepte aus EZauto war es ohne Probleme und vor allem auch ohne großen Aufwand möglich das System sogar um ganze Module zu erweitern. An dieser Stelle war vor allem das Ereignisverteiler-Konzept sehr hilfreich. Zusammenfassend lässt sich sagen, dass die Leitstandssoftware und die zugrunde liegende Bibliothek im Großen und Ganzen sehr flexibel sind, sofern bekannt ist, an welchen Stellen Änderungen vorgenommen werden müssen, um den gewünschten Effekt zu erzielen. Fehlt dieses Wissen, kann das Vorhaben in einem hohen Zeitaufwand resultieren.

6.1.2.3 Validierbarkeit

Validierbarkeit ist in der Softwaretechnik als der Umfang zu verstehen, in dem ein System festzustellen erlaubt, ob es die Anforderungen erfüllt. Dieser Definition ist zu entnehmen, dass die Grundlage für eine Bewertung eine detaillierte Anforderungsdefinition ist. Eine Anforderungsdefinition wurde beispielsweise in Bezug auf EZrola in Abschnitt 4.3.1 vorgenommen. Diese ist jedoch sehr grob und abstrahiert, sodass es nicht möglich ist, eine Aussage zu treffen, inwieweit diese Anforderungen erfüllt werden und falls sie dennoch getroffen wird, ist sie nicht sehr aussagekräftig. Die groben Anforderungen müssten in speziellere Anforderungen immer weiter untergliedert werden, sodass ein Baum entsteht, welcher von den Blättern bis zur Wurzel ausgewertet werden muss.

Angenommen eine detaillierte Anforderungsdefinition läge vor, so müssten Testumgebungen geschaffen werden, um einzelne Elemente zu validieren. Momentan existieren solche Umgebungen nicht, was als einzige Möglichkeit einen Test in der normalen Umgebung des Leitstandes zulässt. Um so die richtigen Umgebungsparamter für den Test zu schaffen, muss ein passendes Szenario erzeugt werden. Hierbei ist jedoch zu beachten, dass die Zusammenhänge innerhalb des Leitstandes teilweise so kompliziert sind, dass nicht ohne weiteres vorhergesagt werden kann, welche Parameter, in einem bestimmten Szenario, bei einem Modul oder einer Funktion ankommen, da die Eingabedaten schon viele Instanzen durchlaufen haben könnten, in denen sie eventuell manipuliert worden sind.

Eine Validierung ist also nur sehr oberflächlich möglich, sofern nicht die Arbeit geleistet wird, die Testumgebungen für einzelne Module zu erschaffen. Doch gerade die oberflächlichen Validierungen besitzen die Einschränkung, dass sich das System in einer simulierten Umgebung unter Umständen anders verhält als in einer Realen. Aus all diesen Gründen ist eine Validierung der Leitstandssoftware daher nur unter großen Einschränkungen möglich.

6.1.2.4 Gesamtbetrachtung

Unter Betrachtung der einzelnen Unterkategorien lässt sich abschließend über die Wartbarkeit des Leitstandes kein positives Urteil abgeben, was jedoch hauptsächlich am hohen Aufwand liegt, sich in das System einzuarbeiten und die Stellen zu finden, die im Falle eines Problems korrigiert oder im Falle einer Änderung angepasst oder erweitert werden müssen. Ebenso sind Tests mit einem hohen Zeitbedarf verbunden. Als positiv stellte sich jedoch die Flexibilität heraus.

6.1.3 Zuverlässigkeit

Laut Definition in Abschnitt 1.2.2.1 beurteilt die Zuverlässigkeit, inwieweit eine Software die angestrebten Ergebnisse unter den verschiedensten Randbedingungen liefert. Da es sich bei einigen Vorgängen des Leitstandes um in höchstem Maße zeitkritische Abläufe handelt, ist dieses Kriterium von besonderer Bedeutung. Um ausreichende Ausfallsicherheit zusagen zu können, sind ausführliche Tests unter verschiedenen äußeren Bedingungen notwendig. Für diese Tests fehlten zum einen die Zeit und zum anderen die notwendige Ausrüstung. Die wenigen Tests, welche durchgeführt worden sind, zeigten jedoch, dass es aufgrund der zu hohen Auslastung der verwendeten Rechner in einigen Fällen zu Kollisionen kam. Der Grund dafür war, dass der Kontrollkomponente nicht ausreichend viel Rechenzeit zur Verfügung stand. Sie sollte entweder alleine auf einem System laufen oder zumindest mit höchster Priorität. Doch selbst unter diesen Voraussetzungen könnte es unter sehr ungünstigen Bedingungen, z. B. wegen zu hoher Netzwerklast, noch zu Situationen kommen, in denen Fahrzeuge nicht rechtzeitig angehalten werden können. In einigen Fällen kann also zumindest von einem zeitweisen Ausfall der Fahrzeugsicherheit gesprochen werden, was den schlimmsten anzunehmenden Fehler nach sich ziehen kann: eine Fahrzeugkollision.

Doch dabei handelt es sich nicht um das einzige Problem, welches den Leitstand unzuverlässig

macht. Er reagiert nicht auf Fehler oder Ausnahmen. Geschehen unvorhergesehene Dinge, wie das Verlassen eines Zeitkorridors durch ein Fahrzeug, wird darauf kaum reagiert. Das Fahrzeug versucht zwar, seinen Plan wieder einzuholen, es wird jedoch nicht geprüft, ob dadurch andere Abläufe beeinflusst werden. In einigen Fällen führte dieses Verlassen des Plans zu einem Deadlock, da das betroffene Fahrzeug z. B. eine Wegstrecke rückwärts befahren musste, die von einem nachfolgenden Fahrzeug jedoch bereits besetzt wurde. Mittelfristig führt dies zum Stillstand des Hofes. Der komplette Stillstand ist die Folge fast aller Ausnahmen auf dem Hof, was den Leitstand sehr unrobust macht.

Die Ausfallsicherheit ist sicher ein Kriterium, was in anderen Umgebungen noch einmal genauer überprüft werden muss. Sie scheint aber nicht in ausreichendem Maße gewährleistet zu sein. An der Fehler- und Ausnahmebehandlung muss jedoch auf jeden Fall gearbeitet werden. Ohne ein umfassendes Konzept in diesem Bereich bleibt der Leitstand unrobust. Insgesamt ist daher der Leitstand momentan als unzuverlässig einzustufen.

6.1.4 Effizienz und Benutzerfreundlichkeit

Die Effizienz bezieht sich auf den Umgang des Programms mit den Betriebsmitteln des Systems. Eine Beurteilung fällt auch in diesem Punkt schwer, da Vergleiche mit ähnlichen Softwareprojekten fehlen, um aussagen zu können, ob sich der Gesamtbedarf an Ressourcen in einem vertretbaren Rahmen bewegt. Daher kann an dieser Stelle nur ein Eindruck vermittelt werden, der auf dem persönlichen Wissen der Autoren über den Programmierstil beruht.

Die Softwarebibliothek EZauto bietet dem Programmierer ein Objektverwaltungskonzept mit automatischer Speicherfreigabe, welches dem Garbagecollector von Java ähnelt. Es entbindet den Programmierer von der Last Objekte zu löschen, wenn diese nicht mehr gebraucht werden. Zu diesem Zweck merkt sich das Objekt, wie viele Referenzen auf es verweisen. Erreicht der Zähler den Wert Null, so kann das Objekt aus dem Speicher entfernt werden. Um diesen Vorgang noch weiter zu vereinfachen, wurde auf die Möglichkeit von Smartpointern zurückgegriffen. Auf diese Weise erlaubt C++ die Verwendung von Objekten, die sich wie Zeiger verhalten. Sie stellen die Referenzzähler und andere Funktionalitäten zur Verfügung. Dieses Konzept ermöglicht ohne große Mühe einen sparsamen Umgang mit Speicher, zumindest derart, dass erzeugte Objekte auch wieder gelöscht werden, wenn alle Referenzen, die auf sie verweisen, entfernt werden. Grundsätzlich wird vonseiten des Leitstandes dieses Konzept aus EZauto durchweg eingesetzt.

Es bleibt jedoch zu klären, ob nicht mehr Objekte als nötig erzeugt werden und ob die Referenzen bei Objekten, die nicht mehr benötigt werden, entfernt werden.

Nur in einem Fall wurde eine Stelle gefunden, an der Objekte vorgehalten wurden, die veraltet waren. Dieser Fehler führte dazu, dass der Speicherbedarf mit der Zeit enorm zunahm. Dieses Phänomen ist in Abschnitt 5.9 beschrieben und betrifft die Raum-Zeit-Planung. Es handelt sich dabei sicher nicht um den einzigen Fehler dieser Art, dieser jedoch war leicht zu bemerken, da er augenscheinliche Probleme nach sich zog, welche Kollisionen zwischen Fahrzeugen beinhalteten.

Da es sich bei der Leitstandssoftware um ein relativ großes Projekt handelt, das noch vergleichsweise jung ist, werden an vielen Stellen Algorithmen benutzt, welche zwar ihre Aufgaben erfüllen, die jedoch, was ihren Bedarf an Betriebsmitteln angeht, nicht optimiert worden sind. Auch muss erst ein guter Kompromiss zwischen Geschwindigkeit und Speicherbedarf gefunden werden. Während im Überwachungsmodul Geschwindigkeit das wichtigste Kriterium ist, da Echtzeitbedingungen zu erfüllen sind, kann in der Planung betriebsmittelschonend programmiert werden, sofern dies nur auf Kosten der Berechnungsdauer geht.

Zum Testen standen lediglich eigene Rechner, sowie die Rechner des Labors der Forschungsgruppe an der Universität Koblenz, zur Verfügung. Somit kann keine Aussage getroffen werden, ob die Effizienz für eine mögliche Zielumgebung ausreichend ist. Dennoch hat es den Anschein, dass sich viele Abläufe beschleunigen lassen würden und auch der Speicherbedarf wirkt vergleichsweise hoch. Bei der Entwicklung des Leitstandes stand merklich die Implementation als solche im Vordergrund, die Effizienz war eher zweitrangig. Aus diesem Grund muss das System vor einer Portierung in eine reale Umgebung überprüft und bei Bedarf überarbeitet werden, da die Abläufe innerhalb des Leitstandes nicht die nötigen Bedingungen für eine echtzeitkritische Anwendung erfüllen.

Ob eine Bewertung des Leitstandes unter dem Gesichtspunkt der Benutzerfreundlichkeit Sinn ergibt, ist fraglich. Laut Definition ist sie das Maß dafür, wie einfach oder kompliziert die Software vom Endanwender verwendet werden kann. Sie beschäftigt sich außerdem damit, ob alle möglichen Eingaben des Anwenders berücksichtigt werden und ob die Software so reagiert, wie es von ihr erwartet wird. Der Leitstand befindet sich momentan in einem Zustand, in dem an eine Auslieferung noch nicht zu denken ist. Ihn mit den Augen eines Endanwenders zu betrachten, ist daher im Grunde nicht zulässig. Wird dies dennoch versuch, so muss der Software eine

sehr schlechte Note ausgestellt werden. Die Eingabe von Daten stellt sich als sehr umständlich heraus, da zu diesem Zweck XML-Beschreibungsdateien manipuliert werden müssen. Läuft das System, ist kaum Interaktion möglich. Es muss jedoch auch gesagt werden, dass das System auf die wenigen bzw. komplizierten Eingabemöglichkeiten durchaus intuitiv reagiert. An dieser Stelle sei auf Abschnitt 6.4.4 verwiesen, in dem es um eine mögliche Erweiterung des Leitstandes geht - eine Bedienerschnittstelle. Eine Bewertung der Benutzerfreundlichkeit sollte erst vorgenommen werden, wenn ein solches Modul entwickelt worden ist.

6.1.5 Gesamtbewertung und Fazit

An dieser Stelle soll nun kurz resümiert werden, was in den vorangehenden Abschnitten einzeln analysiert worden ist. Wie bereits einleitend erklärt, spiegeln die abgegebenen Beurteilungen oftmals die persönlichen Meinungen sowie die mit dem Leitstand gemachten Erfahrungen der Autoren wider - dies wird in diesem Abschnitt nicht anders sein. Demnach birgt das theoretische Konzept und der Aufbau des Allgemeinen Leitstandes großes Potenzial, um eine Wiederverwendbarkeit in den unterschiedlichsten Anwendungsfällen zu garantieren. Dazu trägt die Basis des Leitstandes, die EZauto-Bibliothek, enorm bei.

Aufgrund der Tatsache, dass der Leitstand aktuell nur beispielhaft implementiert vorliegt, gibt es in der praktischen Umsetzung jedoch diverse Einschränkungen, die den Einsatz des Leitstandes in der Simulation erschweren und in einer realen Umsetzung noch unmöglich machen. Diese Defizite sollten aber durchaus, durch die kontinuierliche Weiterentwicklung und fortwährende Optimierung aller Module, beseitigt werden können. Viele einzelne „Baustellen“, die im Abschnitt der anstehenden Arbeiten[5] aufgezeigt werden, ergeben sich im Anschluss an dieses Projekt und werden teils unumgänglich sein, um das Gesamtkonzept zu vervollständigen und alle diesem theoretisch zugesprochenen Funktionalitäten auch in die Praxis umzusetzen.
Es ist durchaus bewusst, dass möglichen Nachfolgern im Rahmen dieser Arbeit, zunächst eine nicht unbedingt einfache Einarbeitungsphase in das Konzept des Leitstandes bevorsteht, so wie sie auch in dieser Arbeit durchlaufen werden musste. Dennoch kann abschließend gesagt werden, dass sich dieser Aufwand durchaus als lohnend erwiesen hat, da so ein vielversprechendes Konzept kennengelernt und mit beeinflusst werden konnte.

[5] vgl. Abschnitt 6.4

6.2 Mechanik zur Portierung auf andere Anwendungen

Eine der Zielsetzungen dieser Arbeit war die Entwicklung einer Vorgehensweise, nach welcher der Allgemeinen Leitstand an weitere Anwendungsfälle des autonomen Fahrens angepasst werden kann. Eine solche soll nun in den folgenden Abschnitten vorgestellt werden. Die drei wesentlichen Schritte zur Anpassung an einen neuen Anwendungsfall sind:

1. Analyse und Erzeugung der Datengrundlagen
2. Spezialisierung des Leitstandes
3. Modellierung und Anpassung der Umgebung

Dabei ist zu beachten, dass die Punkte 2 und 3 eine gewisse parallele Entwicklung durchlaufen sollten, da in vielen Anwendungsfällen die gesamte Simulation nicht ohne die Umgebung getestet werden kann, da diese notwendige Datengrundlagen bildet.

6.2.1 Analyse und Erzeugung der Datengrundlage

Zunächst gilt es den neuen Anwendungsfall so ausführlich wie möglich zu analysieren, um im weiteren Verlauf die Abläufe so realistisch wie möglich darstellen zu können. In Schritt 1 der Vorgehensweise geht es dabei zunächst darum, alle notwendigen Datengrundlagen für die Simulation zu erzeugen. Dazu wird die Kenntnis über folgende Gegebenheiten benötigt:

- Gelände
- Abläufe
- Fahrzeuge

Das für die Nutzung des Anwendungsfalles bereitstehende Gelände muss erfasst und ausgewertet werden. Hilfreich sind hierzu maßstabsgetreue Karten von Vermessungsämtern im CAD-Format. Diese Karten lassen sich mit verschiedenen Hilfsmitteln genauestens auswerten, sodass mögliche Problemstellungen, z. B. Engstellen oder Einbahnstraßen, frühzeitig erkannt werden können. Ebenso sind die Grundstücksgrenzen, sowie auf dem Grundstück befindliche feste Hindernisse wie Gebäude, Zäune, Bordsteine etc. von großer Bedeutung.

Als Nächstes erfolgt die Auswertung der auf dem Gelände möglichen autonomen Abläufe. Alle notwendigen Fahrmanöver müssen analysiert und festgehalten werden. Dabei sollte jeder Ablauf, wenn möglich in seine elementaren Teilaufgaben zerlegt werden können, da diese später in den Aufgabenbaum für das Gelände eingegliedert werden. Zusammen mit der Auswertung des Geländes ergibt sich nun ein autonomer Entwurf, der mögliche Fahrwege vorsieht. Anhand dieses Entwurfes gilt es nun die Fahrwege durch das legen der Leitlinien über das Gelände hinweg festzuhalten. Wie in Abschnitt 4.4.3 bereits erklärt, kann jede Leitlinie maximal vier Komponenten besitzen, welche die Fahrtrichtung und die Fahrweise auf dieser Leitlinie angeben. Die Leitlinien stehen in einer Vorgänger-Nachfolger-Relationen zueinander, sodass sie zusammengenommen einen Graphen über das ganze Gelände aufspannen. Beim Legen der Leitlinien ist auf die Platzverhältnisse und mögliche Engstellen zu achten und die Leitlinie dementsprechend zu positionieren bzw. die auf ihr maximal mögliche Geschwindigkeit anzupassen. Bei großen Geländen mit vielen Leitlinien würde sich die Entwicklung eines Scriptes zur automatischen Generierung aus einer grafischen Benutzeroberfläche[6] heraus durchaus lohnen, da das manuelle erzeugen der XML-Datengrundlage der Leitlinien hohe Konzentration erfordert und sehr fehleranfällig ist. Das erstellte Leitliniennetz (die sogenannte *StaticAreaStructure.xml*) bildet die Datengrundlage für die Routenplanung. Doch nicht nur die Leitlinien, auch mögliche Übergangsbereiche, sofern notwendig, müssen eingeplant und in den Entwurf integriert werden.

Wie bereits eben erwähnt, müssen auch alle auf dem Gelände möglichen Aufgaben in ihre Teilaufgaben zerlegt und in den Aufgabenbaum eingegliedert werden. Je nach Anwendungsfall kann dabei der Aufgabenbaum unterschiedliche Gestalt annehmen. Für die Erstellung des Baumes sind nicht nur die Abläufe sondern auch z.B. feste Positionen auf dem Gelände, wie Parkplätze, Rampenplätze, Anfahrtspunkte von Tankstellen etc. von Bedeutung, da sie mit in die elementaren Aufgaben aufgenommen werden. Der gesamte Aufgabenbaum (die sogenannte *StaticTaskStructure.xml*) bildet daraufhin die Datengrundlage für die Auftragverwaltung und die Terminplanung.

Nach der Erschließung des Geländes und der möglichen Aufgaben, müssen die für die Fahrzeuge notwendigen Beschreibungen erzeugt werden, da sie die Datengrundlage der Fahrzeugsimulation bilden. In den Projekten EZsped und EZrola bildeten hierzu die Modelle der EZauto-Bibliothek die Grundlage, die über die zugehörige Fahrzeugsimulation simuliert werden konnten. Neben Modellen von Sattelschleppern kann hier ebenso auf einen simulierten Gliederzug zurück-

[6] wie z. B. Blender

gegriffen werden. Bedingt durch die hohe Abstraktion der EZauto-Bibliothek, sollte es jederzeit möglich sein, weitere Fahrzeuge nach den vorgegebenen Richtlinien zu konstruieren. Dies wurde im Projekt EZrola durch die Erzeugung des Terminaltraktors bereits durchgeführt.

6.2.2 Spezialisierung des Leitstandes

Nachdem die Datengrundlagen der einzelnen Schichten gelegt sind, gilt es im Weiteren die Aufträge von außerhalb dem Leitstand zu übermitteln. Wie diese Übermittlung aussieht und um welche Art von Aufträgen es sich dabei handelt, hängt ganz vom Anwendungsfall ab. Wie am Beispiel der Spedition gezeigt werden konnte, war es in diesem Fall vollkommen ausreichend, die Aufträge über das externe Modul der Aufgabenplanung für jedes Fahrzeug entgegen zu nehmen. Es lag also eine vollkommen statische Auftragsverwaltung vonseiten des Leitstandes zugrunde, da er die auszuführenden Aufgaben lediglich entgegen nehmen und weiterleiten musste. Dass diese Struktur nicht immer eingesetzt werden kann, zeigte dagegen das Projekt EZrola. Hier galt es eine Vielzahl ähnlicher Aufträge abzuarbeiten, sodass sich eine dynamische Erzeugung dieser in der Auftragsverwaltung eher anbot, als jeden Auftrag einzeln bei der Aufgabenplanung anzufragen.

Wie bereits ersichtlich, ist die Entwicklung und Ausprägung der Aufgabenplanung und Auftragsverwaltung stark vom Anwendungsfall abhängig. Daher kann an dieser Stelle keine genaue Vorgehensweise festgemacht, sondern lediglich eine Empfehlung ausgesprochen werden. Diese orientiert sich an Anzahl und Variationen der möglichen Aufträge. Es wird eine rein statische Struktur empfohlen, wenn der Leitstand nur sehr wenige Aufträgen zur Bearbeitung übersendet bekommt oder aber die abzuarbeitenden Aufträge stark voneinander abweichen. Dagegen wird eine dynamische Erzeugung von Aufträgen empfohlen, wenn der Leitstand mit einer Vielzahl von gleichen Aufträgen umgehen können muss. Ebenso empfiehlt sich diese Vorgehensweise für die Erzeugung von rein internen Aufträgen, wie z. B. das Nachrücken der Traktoren in den Wartespuren im Projekt EZrola.

Bisher wurden die notwendigen Modifikationen zur Spezialisierung des Leitstandes immer außerhalb des Allgemeinen Leitstandes angesiedelt. Dass dies nicht unbedingt der Fall sein muss, zeigte das Projekt EZrola, welches auch den Kern um Funktionalitäten erweitern musste. Da der Kern in der bisherigen Implementierung an vielen Stellen noch Lücken aufweist, wird die

stetige Erweiterung in den Folgeprojekten unabdingbar sein. Für Änderungen am Allgemeinen Leitstand an sich, empfiehlt sich zunächst die ausführliche Analyse der internen Abläufe, sodass Erweiterungen vollständig und effizient eingearbeitet werden können.

6.2.3 Modellierung und Anpassung der Umgebung

Neben dem Leitstand, der Aufgabenplanung und der Auftragsverwaltung, sowie den Fahrzeugen, gibt es aber noch weitere Elemente, welche die Umgebung des Leitstandes darstellen. Diese können entweder über die virtuelle Umgebung oder aber weitere externe Module implementiert werden.

Externe Module empfehlen sich immer dann, wenn der Leitstand weitere Daten übermittelt bekommt oder mit bestimmten Einheiten seiner Umgebung kommunizieren können soll. So empfahl sich im Projekt EZrola die Simulation des Zuges über ein externes Modul, wogegen im Projekt EZsped die Funktionalen Einheiten über ein solches simuliert worden sind. Die Ausprägung und Komplexität der externen Module und der möglichen Kommunikation zum Leitstand hängt dabei ebenfalls sehr vom Anwendungsfall ab und sollte nur nach einer ausführlichen Analyse spezifiziert werden. Dabei sollte nicht nur die Architektur des externen Moduls, sondern auch die Kommunikation und die dadurch entstehende Einflussnahme auf den Leitstand genauestens untersucht werden.

Soll das Verhalten von den Leitstand umgebenden Strukturen lediglich simuliert werden und ist keine Kommunikation notwendig, so können diese auch durch die virtuelle Umgebung realisiert werden. In der derzeitigen Verwendung übernimmt diese lediglich die Funktion der Fahrzeugsimulation, könnte aber jederzeit um weitere Funktionen erweitert werden.

6.3 Umsetzung der autonomen Verwendung in Regensburg

An dieser Stelle soll nun kurz die mögliche autonome Verwendung des Terminals der Rollenden Landstraße in Regensburg betrachtet werden, so wie sie im Rahmen dieser Arbeit konzipiert worden ist. Um das Terminal durch den Leitstand autonom betreiben zu können, müssten folgende Punkte umgesetzt werden:

- Bauliche Veränderungen am Gelände:
  Das Gelände müsste dem Entwurf entsprechend angepasst, d. h. eine zweite Waage und Messeinrichtung installiert, sowie die Abtrennung zwischen den rein autonomen und den Mischbereichen geschaffen werden. Das Befahren der Mischbereiche sowie die Ausfahrt des Abholerbereiches sollte über eine Ampelanlage geregelt, sowie alle Stellplätze über Bodenmarkierungen eingezeichnet werden.

- Aufbau eines Funknetzwerkes:
  Das gesamte Gelände müsste zur Kommunikation der Fahrzeuge mit dem Leitstand mit einem Funknetz ausgestattet werden. Bei der Auswahl des Netzwerkes sollte dabei sichergestellt werden, dass kein Zugriff Dritter möglich ist.

- Installation eines Positionierungssystems:
  Die sich auf dem Gelände befindlichen autonomen Fahrzeuge müssen ihre aktuelle Position fortwährend über das installierte Positionierungssystem bestimmen können. Dabei muss das System gerade beim Befahren der Niederflurwaggons exakte Positionsdaten liefern, die vielleicht durch eine Kombination mehrerer Systeme gemessen werden könnten.

- Entwicklung und Einsatz von Terminaltraktoren:
  Die in dieser Arbeit zur Umsetzung verwendeten Terminaltraktoren müssten mit den notwendigen Fähigkeiten, autonom zu fahren sowie Sattelauflieger oder Anhänger ankuppeln zu können, entwickelt und in ausreichender Stückzahl angeschafft werden. Ebenso müsste auf dem Gelände eine Wartungshalle sowie eine Tankstelle für die Traktoren vorgesehen werden.

- Umrüstung des Terminals Graz/Werndorf zur autonomen Verwendung:
  Damit die autonome Verwendung über den Austausch von Terminaltraktoren zwischen zwei Terminals überhaupt realisiert werden kann, bedarf es der Umrüstung mindestens eines weiteren Terminals. Dies wäre im Anwendungsfall dieser Arbeit das Terminal Graz/Werndorf, für welches die gleiche Analyse und Konzeption, wie im Rahmen dieser Arbeit für Regensburg geschehen ist, durchgeführt werden müsste.

Nun stellt sich die Frage, ob eine autonome Verwendung der RoLa-Strecke Graz/Werndorf - Regensburg überhaupt wirtschaftliche Vorteile mit sich führen würde, die eine solche Umrüstung rechtfertigen könnten. Dies ist gerade am Fallbeispiel Regensburg nicht eindeutig zu belegen.

Wie im Rahmen dieser Arbeit gezeigt wurde, wäre eine autonome Nutzung durchaus realisierbar, ist aber mit einigen Einschränkungen verbunden. In der derzeitigen Auslastung, mit zwei Zugpaaren am Tag und 21 Waggonstellplätzen, stellt der autonome Betrieb durchaus einer Alternative zur manuellen Verwendung dar, da die Züge wesentlich schneller be- und entladen werden könnten. Auch das Einsetzen eines dritten Zugpaares sollte durchaus umgesetzt werden können. Jede weitere Steigerung der Auslastung kann dagegen nur unter ganz bestimmten Richtlinien geschehen. Die Ursache hierfür liegt in der Anzahl der Stellplätze auf dem Gelände begründet. Nach dem Abladen eines gesamten Zuges sind nahezu alle Stellplätze im Abholerbereich belegt, sodass diese bis zur Ankunft des nächsten Zuges von den Kunden komplett geräumt werden müssten. Dazu müssen sie das Gelände befahren, ihren Sattelauflieger oder Anhänger abkuppeln und das Gelände wieder verlassen. Das Zeitfenster, in welchem sie dies durchführen können, wird dabei vom Zugtakt vorgegeben, d. h. mit jeder Steigerung der Auslastung wird die Zeitspanne zur Abholung kleiner. Daher muss gewährleistet sein, dass alle Kunden in besagtem Zeitfenster ihren Sattelauflieger oder Anhänger abholen, da er sonst einen Parkplatz für den nächsten ankommenden Sattelauflieger blockiert. Im aktuellen Entwurf gibt es maximal vier Reserveparkplätze. Diese Problematik kann nur über eine Reduzierung der Stellplätze auf dem ankommenden Zug entschärft werden, sodass weniger Parkplätze durch einen Zug belegt werden. Ein weiterer begrenzender Faktor der Auslastung ist der maximale Durchsatz der Annahme, der vorgibt, in welcher Zeit wie viele Sattelauflieger und Anhänger von Kunden am Terminal abgegeben werden können. In groben Berechnungen wird die Zahl auf maximal zehn Übergaben pro Stunde beziffert. Zu genauen Angaben wird hierzu auf die Diplomarbeit von Andreas Kern verwiesen, der sich mit dieser Thematik auch im Anwendungsfall des Terminals in Regensburg genauer beschäftigt hat.

Aufgrund der genannten Faktoren kann eine autonome Verwendung des Terminals in Regensburg im Rahmen der aktuellen Auslastung als durchaus realisierbar angesehen werden, sofern die oben genannten Punkte umgesetzt werden können. Lediglich die beengten Platzverhältnisse lassen eine Steigerung der Auslastung nur in bestimmten Grenzen und Richtlinien zu, sodass dies bei einem neuen Entwurf für weitere Terminals der Rollenden Landstraße unbedingt berücksichtigt werden sollte. Nur so kann eine sinnvolle autonome Verwendung gewährleistet werden und deren Vorteile, gerade bei der sehr hohen Auslastung eines Terminals, zum Tragen kommen.

6.4 Ausblick: anstehende Arbeiten

In diesem Abschnitt soll auf einige Elemente der Leitstandssoftware hingewiesen werden, die einer weiteren Betrachtung und Überarbeitung in zukünftigen Arbeiten bedürfen. Einige sind von essenzieller Bedeutung, da sie die Grundfunktionalitäten des Allgemeinen Leitstandes betreffen. Dazu ist beispielsweise die Raum-Zeit-Planung zu zählen. Ferner geht es um Elemente, die zwar zum Gesamtprojekt gehören, jedoch im Rahmen dieser Arbeit nicht mehr implementiert werden konnten, oder um Elemente, welche die Umgebung des Leitstandes betreffen und die für eine Simulation in einer virtuellen Umgebung unabdingbar sind.

Es soll hier jedoch nicht nur um Dinge gehen, die in jedem Fall umgesetzt werden müssen, um die korrekte Funktionalität der Leitstandssoftware zu gewährleisten, sondern auch um Erweiterungen, die das Gesamterscheinungsbild des Leitstandes und der Umgebungssimulation für eventuelle Vorführungen abrunden oder die Bedienung und das Testen der Software erleichtern sollten.

6.4.1 Realisierung von „Mischbereichen“

Unter Mischbereichen werden Bereiche des Geländes verstanden, welche nicht die ganze Zeit unter der Kontrolle des Leitstandes liegen. Im Falle von EZrola sind das die Übergabebereiche, in denen abwechselnd autonom bzw. manuell gefahren werden darf. Für die korrekte Simulation müsste dafür gesorgt sein, dass zu bestimmten Zeitpunkten[7] kein autonom gesteuertes Fahrzeug in einen solchen Bereich einfahren kann. Momentan muss durch geeigneten Entwurf des Szenarios dafür gesorgt werden, dass es nicht zu Überschneidungen kommt.

Es müssen also neue Strukturen implementiert werden, die verhindern, dass sich autonom gesteuerte Fahrzeuge in Mischbereichen aufhalten, wenn diese für die manuelle Fahrt reserviert sind. Das rechtzeitige Sperren der jeweiligen Leitlinien wäre dafür nicht ausreichend, da die Fahraufträge inklusive der nötigen Trajektorien schon vergeben sein könnten. Es wäre jedoch möglich, um den Bereich einen Kreis zu ziehen und entweder die Kreislinie oder die Kreisfläche als Hindernis einzutragen. Das würde verhindern, dass ein Fahrzeug in den Bereich einfährt. Die Fahrzeuge, die dieses Gebiet befahren sollen, würden an der Grenze zum Stehen kommen und aus

[7] z. B. wenn Abholer kommen, um die Auflieger abzuholen

ihrem Plan fallen. Dieser „Fehler“ müsste wiederum behandelt werden[8]. Es wäre auch möglich, alle laufenden Fahraufgaben dahingehend zu überprüfen, ob ihre Korridore die gesperrte Fläche schneiden. In diesem Fall müssten diese Aufgaben neu eingeplant werden.

Andere Fahraufträge können aufgrund dieses Hindernisses nicht verplant werden, außer bei der Planung wäre bekannt, wie lange die Sperrung dieses Bereiches aufrecht erhalten wird. Ein Fehler bei der Planung müsste in diesem Fall auch behandelt werden.

Es bliebe dann jedoch immer noch zu klären, wie mit Fahrzeugen umgegangen werden soll, die sich gerade genau in diesem Bereich befinden, der für autonome Fahrt gesperrt werden soll. Diese müssten aller Voraussicht nach von diesem Gebiet entfernt werden. Nur welcher Regel soll dieses Entfernen folgen? Manche Fahrzeuge müssen vielleicht nur umgeleitet werden, während andere möglicherweise ihren Zielpunkt in diesem Bereich hatten.

6.4.2 Umgebungssimulation

Ein Ziel des Gesamtprojektes EZrola ist die Erzeugung einer vorführbaren Simulation der Abläufe auf einem Terminal der „Unbegleiteten Rollenden Landstraße“. Die Forderungen, die dabei an den Leitstand selbst gestellt werden, also die Fähigkeit alle anwendungsfallspezifischen Aufgaben auf dem Gelände zu erfüllen, wurde mit dieser Arbeit weitestgehend erfüllt. Die Abläufe um das Gelände herum werden jedoch entweder gar nicht oder nur ansatzweise simuliert.

Eine vorführbare Simulation ist für das Gesamtprojekt von großer Bedeutung, da wohl nur mit ihrer Hilfe Aufmerksamkeit und Akzeptanz bei möglichen Kooperationspartnern in Wirtschaft und Politik geschaffen werden kann. Diese Simulation sollte der Realität so nahe wie möglich sein, damit von ihr auf eine tatsächliche Nutzung geschlossen werden kann. In ihr müssen im Idealfall alle Aufgaben und Probleme der realen Nutzung darstellbar sein.

Im Zentrum dieser Simulation wird immer die Visualisierung stehen. Wie in den folgenden Abschnitten zu sehen ist, wird es unerlässlich sein, diese so zu modifizieren, dass sie in der Lage ist, mit mehreren Leitständen gleichzeitig zu kommunizieren.

[8] vgl. Abschnitt 6.4.3

6.4.2.1 Simulation der Züge

Zur Umgebung zählt beispielsweise alles, was mit den Zügen an sich zu tun hat. Die Züge existieren momentan nur als Menge von Polygonen, die von einigen Modulen für die Dauer von mehreren Minuten zwischengespeichert werden. Sie werden zu bestimmten Zeitpunkten erzeugt und sind absolut unbeweglich. Für eine Gesamtsimulation sollten die Züge von außen an das System herangeführt werden. Sie sollten sich dem Gelände tatsächlich nähern und auf diesem zum Stehen kommen. Die darauf befindlichen Fahrzeuge sollten am Leitstand angemeldet werden, sobald der Zug seinen Endpunkt auf dem Gelände erreicht hat.

Mit diesen Anforderungen sind direkt mehrere Probleme verbunden, die sich zum Teil erst bei genauerer Kenntnis des Gesamtsystems erschließen. Das augenscheinlichste Problem ist die Notwendigkeit des Entwurfs eines Zugmodells für die Visualisierung. Ist solch ein Modell vorhanden, so können die Züge dargestellt werden. Um diese zusätzlich zu bewegen, sind kontinuierliche Positionsdaten für den Zug notwendig. Da Züge nicht zum Gesamtkontext des Leitstandes gehören, müssen diese Positionsdaten von einem externen Modul versendet werden. Grundsätzlich könnte dies vom Modul *VirtualEnvironment* übernommen werden. Züge würden dann von der Visualisierung ähnlich wie Lkw behandelt werden.

Dieses externe Modul benötigt jedoch auch genaue Kenntnisse darüber, welche Lkw sich auf dem jeweiligen Zug befinden. Der Zug muss dann für die Visualisierung entweder als ein Gesamtobjekt dargestellt werden, welches nur abhängig von den Positionsdaten des Zuges ist, oder die darauf befindlichen Lkw müssten der Visualisierung als eigenständige Objekte bekannt gemacht werden. In diesem Fall müssten auch für sie ständig Positionsdaten versendet werden. Die Änderung der Position darf jedoch keine Drehung der Räder bewirken.

Ist der Zug auf dem Gelände angekommen, so muss eine Kommunikation zwischen dem Leitstand und diesem Modul stattfinden, um die Fahrzeuge in den Machtbereich des Leitstandes zu übergeben. Die Übergabe selbst sollte grundsätzlich kein größeres Problem darstellen, da alle nötigen Daten vorhanden sind. Ein größeres Problem entsteht dadurch, dass die Fahrzeuge auf dem Zug, gegenüber dem Hof, erhöht stehen. Weder der Leitstand noch die Visualisierung sind für Bewegungen im dreidimensionalen Raum ausgelegt. Dieses Problem könnte rein aufseiten der Visualisierung angegangen werden. Dem Leitstand könnte dann die dritte Dimension komplett verborgen bleiben. Für eine korrekte Darstellung müsste das Fahrzeug dann jedoch, wenn es über

die Rampe fährt, geneigt werden. Das gilt auch für die einzelnen Fahrzeugteile untereinander. Wird der Leitstand dabei außer Acht gelassen, so stimmen jedoch die Werte für die Geschwindigkeit nicht mehr, da das Fahrzeug durch die Änderung der Höhe schneller bewegt werden muss. Diese Einschränkung wäre aber, je nach Neigung der Rampe, zu vernachlässigen.

Eine einfachere Lösung wäre es, das Fahrzeug ab einem gewissen Punkt (am besten, wenn es komplett vom Zug abgefahren ist) auf das Niveau des Hofes abzusenken. Diese Lösung ist jedoch für eine mögliche Vorführung nicht elegant.

6.4.2.2 Simulation der Kundenfahrzeuge

Außer der Simulation der Züge müssen auch die Fahrzeuge der Kunden simuliert werden, wenn diese ihre Auflieger abgeben, bzw. abholen. Momentan wird diese Simulation durch den Leitstand übernommen. Dafür wurden einige Erweiterungen, beispielsweise an der *OrderAdministration*[9], vorgenommen. Diese Erweiterungen müssen vollständig entfernt werden. Der Leitstand soll die Fahrzeuge in einem dafür vorgesehenen Bereich in definierter Weise übergeben bekommen. Für diese Simulation könnte ein eigenständiger weiterer Leitstand eingerichtet werden, der parallel zum eigentlichen Leitstand läuft. Die Gesamtwelt der Simulation lässt sich dann aufteilen in das Gelände des Anwendungsfalls, für welches der eigentliche Leitstand verantwortlich ist (im Fall von EZrola das Gelände des Terminals), und den Rest der Welt, für den der zweite Leitstand zuständig ist. Die beiden Gebiete haben eine Schnittmenge, die dazu dient, Fahrzeuge von einem Verantwortungsbereich in den anderen zu überführen.

Es muss dann geklärt werden, wie in diesen Schnittbereichen dafür gesorgt werden soll, dass es keine Kollisionen zwischen den Fahrzeugen der verschiedenen Leitstände gibt. Eine mögliche Lösung wäre es, diese Bereiche in zeitlicher Abhängigkeit zwischen den beiden Instanzen zu übergeben. Auf diese Weise hätten zu keinem Zeitpunkt beide Leitstände die Kontrolle über diese Gebiete, womit Kollisionen ausgeschlossen wären.

Eine andere Möglichkeit wäre es, die Gebiete so zu gestalten, dass Leitstand A nur Fahrzeuge in dieses Gebiet einfahren lassen kann und Leitstand B nur Fahrzeuge von diesem Gebiet entfernen kann. In diesem Fall müsste jedoch auch sichergestellt werden, dass Leitstand B das Fahrzeug bereits entfernt hat, bevor Leitstand A ein weiteres Fahrzeug einfahren lässt. Zu diesem Zweck könnte B an A eine Nachricht der Form „*Übergabebereich n ist frei*“ schicken. Diese Lösung würde dann jedoch mit der vorherigen Lösungsvariante zusammenfallen, da die Verantwortung

[9] vgl. Abschnitt 5.3.2

in diesem Moment wieder an Leitstand A übergeht.

Alle denkbaren Möglichkeiten haben gemeinsam, dass ein Kommunikationsmodell zwischen den beiden Leitständen entworfen werden muss, das einerseits die Übergabe von Fahrzeugen und andererseits den Austausch von Informationen über die Nutzung der Schnittbereiche des Gesamtgeländes beinhaltet.

Außerdem muss es möglich sein, zwischen den Leitständen auch passive Fahrzeugteile übergeben zu können. Momentan meldet sich immer das aktive Fahrzeugteil beim Leitstand an und übersendet alle Informationen zu den zu ihm gehörenden weiteren Teilen.

6.4.3 Raum-Zeit-Planung

Wie schon in Abschnitt 5.9 angedeutet, sollte das Konzept der Raum-Zeit-Planung sowohl für das Anwendungsgebiet der Rollenden Landstraße, als auch für zukünftige Anwendungsgebiete analysiert und überarbeitet werden. Sie steht im Zentrum der Planung und ist Grundlage für alle Fahraufgaben, die auf dem Hof verplant werden sollen. Trotz der Änderungen, die in dieser Diplomarbeit vorgenommen wurden, blieben einige Probleme und Fragen offen.

Fahrzeuge, die neu erzeugt werden, besitzen keinen Eintrag in der Liste der *PlanningAllocationPolygons*. Aus diesem Grund sind sie der Raum-Zeit-Planung unbekannt und stellen für andere Fahrzeuge bei der Planung kein Hindernis dar, obwohl sie in Realität möglicherweise den Weg versperren. Zwar werden im Falle des RoLa-Terminals Regensburg keine neuen Fahrzeuge mitten auf dem Hof positioniert, aber es werden beispielsweise viele Fahrzeuge direkt hintereinander bei der Ankunft eines Zuges erzeugt. Werden diese Fahrzeuge nicht in der richtigen Reihenfolge verplant, so kann es vorkommen, dass Fahrzeuge innerhalb der Reihe Fahraufgaben zugewiesen bekommen, die mitten durch noch stehende andere Fahrzeuge hindurchführen. Zwar kann durch die Sicherheitsüberwachung eine Kollision verhindert werden, jedoch fällt das besagte Fahrzeug dadurch aus seinem Plan heraus, was ebenso für die Fahrzeuge gilt, die hinter diesem Fahrzeug starten.

Das Einfügen eines Eintrages für neue Fahrzeuge stellt an sich kein Problem dar, jedoch müsste dieser Eintrag ein Belegungspolygon mit unbekanntem Ende enthalten. Dies wiederum würde die Planung der Fahrzeuge weiter hinten in der Schlange solange unmöglich machen, wie der Eintrag des Polygons Bestand hat. Generell führt dies zur Fragestellung, wie im allgemeinen mit Hindernissen umgegangen werden soll, die aufgrund der angesprochenen Problematik eine

erfolgreiche Planung verhindern.

Kann die Raum-Zeit-Planung keinen Korridor für ein Fahrzeug finden, so ist entweder die Auslastung des Hofes zu hoch oder es befindet sich ein momentan unbewegtes Hindernis auf dem Fahrweg. Ist die Auslastung die Ursache, so wird sich an der Ausgangslage auch in näherer Zukunft nichts ändern[10]. War der Grund jedoch ein unbewegtes Hindernis, so könnte sich in der Zukunft etwas an der Ausgangslage geändert haben. Zwischen dem ersten, fehlgeschlagenen, und einem weiteren Planungsversuch könnte ein vorher unverplantes Fahrzeug, das das angesprochene Hindernis dargestellt hat, verplant worden sein. Dadurch wäre dessen Belegungspolygon, das vorher bis ins Unendliche eingetragen war, aus dem Plan gelöscht. Dieses Hintergrundwissen sollte bei der weiteren Verplanung unbedingt berücksichtig werden. Welche Vorgehensweise dabei die effizienteste ist, muss noch herausgearbeitet werden.

Bei der Planung lässt sich bestimmen, welches Objekt eine Kollision verursachen würde und damit die erfolgreiche Planung verhindert. Lägen genügend Informationen über alle Objekte vor, ließe sich unter Umständen eine Aussage darüber treffen, ob sich in Zukunft etwas ändern könnte. Dafür müsste jedoch z. B. bekannt sein, ob für dieses Objekt (wenn es sich um ein Fahrzeug handelt) ein Folgeauftrag in Sicht ist. In diesem Fall könnte der Leitstand so modifiziert werden, dass die Planung eines Fahrauftrags für das Hindernis einen erneuten Planungsversuch für das Fahrzeug auslöst, dessen Planung vorher durch dieses Hindernis verhindert wurde. Doch es kann durchaus sein, dass solche Informationen nicht vorhanden sind, da z. B. entweder momentan kein weiterer Auftrag für das Hindernis geplant ist (was aber nicht bedeutet, dass dies nicht in näherer Zukunft noch eintreten könnte) oder ein Fahrzeug gerade erst im System angemeldet wurde. Somit sind seine Aufgaben noch nicht an den Leitstand übergeben oder von diesem geprüft worden.

Liegen keine Informationen über das Hindernis vor, ist zu überlegen, ob nicht neue Planungsversuche in der Zukunft Erfolg versprechend sein könnten. In diesem Fall muss entschieden werden, wie lange gewartet wird und ab welchem Zeitpunkt die Planung aufgegeben werden soll.

[10] Damit ist nicht gemeint, dass die Auslastung auch in Zukunft nicht sinken wird, sondern dass eine Fahraufgabe, die geplant wurde, nicht ohne Weiteres wieder aus dem Plan verschwindet. Damit würde der Plan an einer bestimmten Stelle zu einem bestimmten Zeitpunkt beim zweiten Versuch der Planung keinen Unterschied zum ersten Versuch aufweisen.

Bevor ein Fahrzeug bei der Planung zu stark verzögert wird, sollte geprüft werden, ob es nicht einen anderen Fahrweg gibt, mit dem die Problemstelle umgangen werden kann. Hierfür sollten der Raum-Zeit-Planung verschiedene Routen zur Verfügung gestellt werden. Dies erfordert ein sehr ausgefeiltes Leitliniennetz, was stark von den jeweiligen örtlichen Begebenheiten abhängig ist. Auf einem kleinen Gelände, wie dem des RoLa-Terminals Regensburg, sind Alternativrouten fast undenkbar.

Momentan wird ein Fahrzeug nach dem anderen in den Plan eingefügt. Das erste Fahrzeug wird genau nach den Vorgaben seines Auftrages verplant. Das nächste Fahrzeug muss sich dann nach dem Ersten richten, da dessen Einträge nicht mehr zu ändern sind. So wird für jedes weitere Fahrzeug die Planung schwieriger. Es können Situationen entstehen, in denen Fahrzeuge nicht mehr in den Plan einzufügen sind, obwohl dies unter Betrachtung aller Aufgaben möglich gewesen wäre. Angenommen, das Gelände besitze eine längere Engstelle, die in zwei Richtungen befahren werden kann, aber keinen Platz für zwei Fahrzeuge nebeneinander bietet. Ein Beispiel hierfür wäre die Einfahrt für die Abholer des RoLa-Terminals Regensburg. Nacheinander sollen zwei Fahrzeuge A und B eingeplant werden. Beide müssen diese Engstelle passieren, jedoch in entgegengesetzter Richtung. A muss seinen Fahrauftrag im Zeitintervall $[t_1, t_2]$, B im Zeitintervall $[t_1, t_3]$ abarbeiten, mit $t_1 < t_3 << t_2$. Die Durchfahrt habe eine Dauer von Δt mit $2 * \Delta t < t_3 - t_1$ (es können also nicht beide Fahrzeuge im Zeitbereich $[t_1, t_3]$ die Engstelle passieren). Da A zuerst eingeplant wird, wird es bei t_1 gestartet und lässt somit B, welches erst später geplant wird, keinen ausreichenden Zeitbereich, um seinen Auftrag zu erfüllen, obwohl A auch hätte später eingeplant werden können.

Um solche Situationen zu vermeiden, sollte der Raum-Zeit-Plan immer auf der Gesamtheit der gerade vorliegenden Daten beruhen. Das bedeutet, dass beim Bekanntwerden eines neuen Fahrauftrags mindestens alle noch nicht gestarteten Fahraufträge einer erneuten Planung unterzogen werden sollten. Dies verhindert nicht nur die Unplanbarkeit eigentlich planbarer Aufträge (wie in obigem Beispiel), sondern kann auch die Effizienz erhöhen.

Ist der Raum-Zeit-Plan erstellt, so muss überprüft werden, ob dieser auch eingehalten wird. Dazu wird die tatsächliche (Ist) Position des Fahrzeuges mit der Position, die das Fahrzeug laut Plan haben sollte (Soll), verglichen. Kleine Diskrepanzen können durch Beschleunigen oder Bremsen beseitigt werden. Ab einer bestimmten Abweichung muss das Verlassen des Plans festgestellt werden. In diesem Fall ist eine umfassende Überprüfung des gesamten Plans notwendig.

In manchen Fällen kann es ausreichend sein, den Plan des betroffenen Fahrzeuges anzupassen. In anderen Fällen dagegen könnten die Folgen so weit reichend sein, dass sämtliche Fahrzeuge neu geplant werden müssten. So könnten durch diese Abweichung andere Fahrzeuge verzögert werden, Reihenfolgen zerstört oder Deadlocks erzeugt werden.

Da die Behandlung von Deadlocks sehr kompliziert ist, sollte der Raum-Zeit-Plan so konzipiert sein, dass diese so früh wie möglich ausgeschlossen werden können. Es kann beispielsweise schnell zu einem Deadlock kommen, wenn eine Wegstrecke in verschiedene Richtungen befahren werden soll. In diesem Fall muss verhindert werden, dass beide Fahrzeuge das betreffende Wegstück befahren.

Zusammenfassend lässt sich sagen, dass die Raum-Zeit-Planung sehr viel Raum für Verbesserungen lässt. Es gibt dabei noch mehr Aspekte, als die oben genannten, zu beachten, da in dieser Betrachtung sicher einige tiefer gehende Problematiken und Fragestellung übersehen worden sind, die sich erst bei bestimmten Konstellationen oder Abläufen in einer Anwendung erschließen. Die Raum-Zeit-Planung steht im Zentrum des Leitstandkonzeptes und sollte daher umfassend analysiert und verbessert werden. Die Praxistauglichkeit dieses Systems ist in höchstem Maße abhängig von diesem Modul.

6.4.4 Bedienerschnittstelle

Eine Bedienerschnittstelle ist ein Zusatzmodul, das es einem Bediener vereinfacht, in ein laufendes System einzugreifen. Die Rolle des Bedieners kann unterschiedlich sein. Es kann sich um den Endanwender handeln, der Auftragsdaten in das System eingeben oder die Vorgänge auf dem Hof überwachen will. Es kann sich jedoch auch um einen Software-Entwickler handeln, der Tests erzeugen und durchführen möchte.

Die Bedienerschnittstelle sollte über eine grafische Oberfläche verfügen. Es sollte möglich sein, sich zu jedem Fahrzeug alle verfügbaren Informationen anzeigen lassen zu können. Dazu gehören die aktuelle Position und Ausrichtung, die momentan auszuführende Aufgabe, die bisher ausgeführten Aufgaben sowie die Aufgaben, die in Zukunft ausgeführt werden sollen. Bei bereits geplanten Aufgaben sollten alle Eckdaten, wie voraussichtliche Start- und Endzeiten, abrufbar sein. Alle Daten, bei denen es Sinn ergibt, sollten auch manipulierbar sein. Dazu gehören Aufgaben der Zukunft, sowie möglicherweise die jeweils aktuelle Aufgabe. Fahrzeuge sollten auch entfernt werden können.

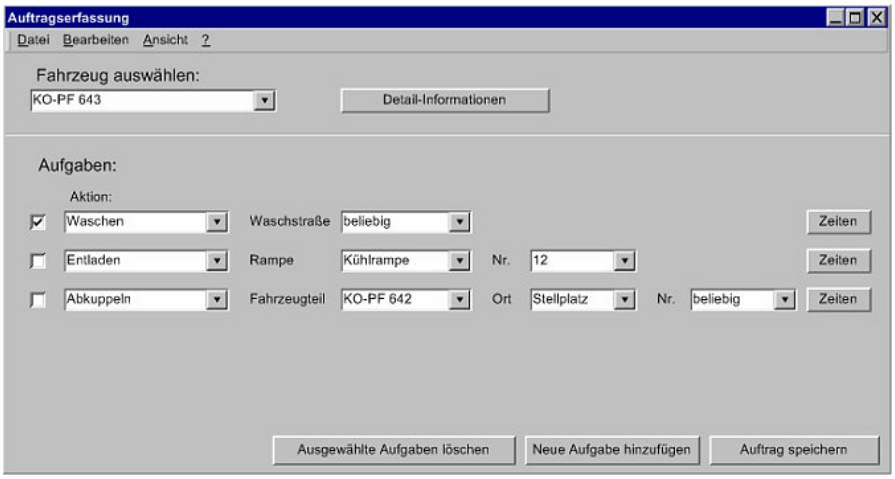


Abbildung 6.1: Beispiel einer grafischen Eingabemaske für Fahrzeugaufträge

Solch ein Modul bedarf einiger Änderungen am Leitstand. Für die Abfrage der Daten muss dieser alle möglichen Informationen zur Verfügung stellen. Viele dieser Informationen könnten durch das Ereignisverteilerkonzept der EZauto-Bibliothek ohne Änderungen aus dem laufenden System abgegriffen werden, da sie Teil von Nachrichten sind, die schon jetzt verschickt werden. Das neue Modul müsste sich nur für die jeweiligen Ereignisse anmelden. Für andere Informationen müsste jedoch der Leitstand manipuliert werden. So werden bisher beispielsweise keine Informationen über die gesamte Aufgabenstruktur der Fahrzeuge verschickt.

Die Änderungen für das Abrufen der Informationen sind gering im Vergleich zu den Änderungen, die vorgenommen werden müssen, um Daten eines Fahrzeuges zu manipulieren. Dies ist relativ unkompliziert, sofern es sich um Aufträge handelt, die dem Scheduler noch nicht zur Planung übergeben worden sind. Wenn es sich jedoch um einen aktuellen Auftrag handelt, oder um einen bereits geplanten, dann muss der Leitstand so erweitert werden, dass ein Eingriff in die Daten des Schedulers möglich wird. Bestehende Pläne müssen verworfen und Einträge im Raum-Zeit-Plan gelöscht werden können. Diese Änderungen wären auch für das Löschen von Fahrzeugen notwendig. Außerdem sollte die Bedienerschnittstelle über eine visuelle Komponente verfügen, mit der die Position der Fahrzeuge auf dem Hof angezeigt werden kann. Ebenso sollten Fahrzeuge per Maus-Klick ausgewählt und somit deren Daten angefordert werden können. Hierfür könnte die in [Ses] entwickelte 3D-Visualisierung verwendet werden.

Ein solches Werkzeug wäre für eventuelle Vorführungen sehr hilfreich, da es das Gesamtprojekt abrunden würde und Interessenten die Möglichkeit geben würde, direkt auf das Geschehen auf dem Gelände einzugreifen. Dies würde den eventuellen Eindruck nehmen, die Vorgänge wären vorausberechnet und somit statisch. Eine Einflussnahme ist bisher ausschließlich über die Beschreibungsdateien in XML-Format, Änderungen am Quelltext und in sehr eingeschränktem Maße über die Visualisierung möglich. Doch nicht nur für eventuelle Vorführungen wäre eine Bedienschnittstelle hilfreich, sondern auch bereits im Entwicklungsprozess für Testdurchläufe. Solche Durchläufe dauern oftmals mehrere Stunden und müssen manchmal aufgrund kleiner Fehler abgebrochen werden. Häufig könnte der Test weitergeführt werden, wenn Fahrzeuge entfernt oder bewegt werden könnten. Das ist beispielsweise der Fall, wenn sich zwei Fahrzeuge gegenseitig blockieren und so ein Deadlock entstanden ist. Eine beispielhafte Eingabemaske für Fahrzeugaufträge ist in Abbildung 6.1 zu sehen.

6.4.5 Tiefere Fahrzeugsimulation

Eine weitere Möglichkeit das Gesamterscheinungsbild des Leitstandes zu verbessern, wäre die Detailtiefe der Simulation zu erhöhen. Damit sind keine visuellen Aspekte gemeint, sondern Abläufe der realen Welt. Bereits in Abschnitt 4.2 wurde die Möglichkeit aufgezeigt, Spritknappheit oder Defekte in die Simulation einzuarbeiten. Die Fahrzeuge würden dem Leitstand dann mitteilen, wenn der Füllstand des Tanks einen bestimmten Schwellwert unterschritten hat oder Defekte an einem Fahrzeug auftreten.

Dazu wären Änderungen sowohl an der Fahrzeugsimulation, als auch am Leitstand notwendig. Aufseiten der Fahrzeugsimulation müssten Fahrzeuge um Attribute für Tankfüllstand und Fehlerschlüssel erweitert werden. Der Tankfüllstand müsste regelmäßig neu berechnet werden. Als Grundlage könnte hier die effektive Fahrzeit jedes Fahrzeuges herangezogen werden. Fehlercodes könnten statistisch erzeugt werden. Dafür könnte die virtuelle Umgebung mit festgelegter Wahrscheinlichkeit bestimmte Defekte erzeugen und Fahrzeuge für diese zufällig auswählen. Die Fahrzeugsimulation würde dann diese Ereignisse (Defekt oder Spritknappheit) an den Leitstand melden. Dieser muss so modifiziert werden, dass er adäquat auf solche Nachrichten reagieren kann. Er müsste dann automatisch Tank- oder Reparaturaufgaben einfügen.

Tank- und Reparaturaufgaben sind an dieser Stelle als Beispiele zu betrachten. Je nach Einsatzgebiet des Leitstandes sind verschiedene Erweiterungen denkbar. Einige gehören zum Standardaufgabengebiet wie das Tanken, andere kommen vielleicht eher selten vor. Alle erhöhen

jedoch den Bezug zur Realität bei Vorführungen und erlauben es außerdem eventuelle Probleme, möglicherweise noch vor der Portierung in die reale Umgebung, zu erkennen.

6.4.6 Fehlerbehandlung

Eine Fehler- und Ausnahmebehandlung ist ein sehr wichtiger Schritt auf dem Weg zu einem einsetzbaren Softwareprojekt. Ohne Fehler- und Ausnahmebehandlung besitzt die Leitstandssoftware nur einen sehr geringen Grad an Ausfallsicherheit. Im momentanen Zustand führt jeder Fehler mittelfristig zum Stillstand des gesamten Systems. Jede Ausnahme, im Sinne einer Abweichung von den geplanten Abläufen, wird ignoriert, was ebenfalls zum Gesamtabbruch führen kann.

Die meisten Probleme entstehen im Zusammenhang mit der Planung und im Speziellen in der Raum-Zeit-Planung. Diese Thematik wurde bereits in Abschnitt 6.4.3 behandelt. Aber es gibt auch eine Vielzahl anderer Fehler, die auftreten können. Einige lassen sich wahrscheinlich nicht behandeln und erfordern entweder eine Reaktion durch den Bediener des Leitstandes oder sie sind wirklich so gravierend, dass der weitere Ablauf die Sicherheit auf dem Hof gefährdet. Ungeachtet der Schwere der Fehler muss auf sie jedoch in irgendeiner Form reagiert werden. Die bisherige Reaktion ist meist eine Ausgabe auf die Kommandozeile und selbst das ist häufig nicht der Fall. Steht beispielsweise kein Traktor in den Traktorspuren, so endet die Suche nach einem freien Traktor ergebnislos und der Auflieger, für den ein Traktor gebraucht wird, bleibt im Übergabebereich stehen, wo er im weiteren Verlauf des Szenarios zum Hindernis wird. Vielleicht hätte es einen freien Traktor gegeben, der sich nur noch nicht in die Traktorspuren begeben hat oder es wäre in Kürze ein Traktor frei geworden. In diesem Fall hätte - mit etwas Verzögerung - das Szenario ganz normal weiterlaufen können. Steht jedoch bekannterweise in nächster Zeit kein Traktor zur Verfügung, so muss der Administrator des Systems entscheiden wie mit dieser Situation verfahren werden soll.

Es wäre möglich, jedem Fehler, der theoretisch behandelt werden kann, eine eigene ID zu zuteilen. Tritt dieser Fehler auf und kann er nicht an Ort und Stelle beseitigt werden, so verschickt das Modul, welches ihn erzeugt, ein Fehler-Ereignis mit diesem Code als Attribut. In weiteren Attributen können wichtige Eigenschaften der speziellen Fehlerinstanz übertragen werden. Dieses Ereignis kann dann von einem Modul aufgegriffen werden, das sich in der Lage sieht, solche Fehler zu behandeln. Findet sich kein Modul, das zur Lösung im Stande ist, so kann der Fehler an die Bedienerschnittstelle aus Abschnitt 6.4.4 weitergegeben werden, um dem Administrator

des Leitstandes die Möglichkeit zu geben, adäquat zu reagieren. Bei sicherheitskritischen Fehlern muss möglicherweise schon vor dieser Benachrichtigung der Verkehr auf dem Hof angehalten werden.

6.4.7 Routenplanung und Korridorberechnung

Wie bereits in Abschnitt 5.10.3 angedeutet, bedarf es auch in der Routenplanung einiger weiterer Arbeiten. Wie alle Module befindet sich diese im Status einer beispielhaften Implementierung, d. h. jegliche Fehlerbehandlung im Falle einer nicht einplanbaren Route existiert nicht. In der aktuellen Version wird stets die erste gefundene Route zur Grundlage der Trajektorienberechnung herangezogen. Weitere Routen werden zwar gespeichert, aber auf diese nicht mehr zurückgegriffen. Daher wäre es zunächst der einfachste Weg, diese Routen heranzuziehen, falls die gewählte Route nicht einplanbar sein sollte. Ebenso bietet sich an dieser Stelle der Einsatz einer Routenauswahl über eine Heuristik an. So könnte unter den gefundenen Routenmöglichkeiten zunächst stets die unter bestimmten Kriterien[11] beste Route ausgewählt und einzuplanen versucht werden. Sollte dies scheitern, so könnte die zweitbeste Route ausgewählt werden. Ebenso würde diese Heuristik dann eine Rolle spielen, wenn die gefundene Route zwar befahrbar ist, aber von der Raum-Zeit-Planung nicht eingeplant werden kann, weil sie durch andere Fahrzeuge blockiert wird. An dieser Stelle könnte die Interaktion mit der Raum-Zeit-Planung erweitert werden, sodass diese eine andere, z. B. in Fahrzeit und Weg längere Route, anfragen kann.

Auch die Trajektorienberechnung kann, wie in Abschnitt 5.10.3 gezeigt, in manchen Fällen keine optimalen Ergebnisse liefern. Diese gilt es daher in ausführlichen Tests zu optimieren, da Fehler in der Trajektorie weit reichende Auswirkungen nach sich ziehen. Unmittelbar aus der Trajektorie wird der Korridor des Fahrzeuges berechnet, in dem sich dieses während seiner gesamten Fahrzeit aufhalten soll. Mit einer fehlerhaften Trajektorie ergibt sich automatisch ein fehlerhafter Korridor und damit möglicherweise Probleme bei der Einplanung in den Raum-Zeit-Plan.
Die eigentliche Korridorberechnung ist in der aktuellen Version sehr einfach gehalten, denn es wird lediglich ein Schlauch mit einer gewissen Breite um die Trajektorie herum gelegt und die Enden jeweils verlängert. Auch dies ist faktisch falsch, da der Korridor in Kurvenfahrten breiter sein muss als auf einer Geraden. So kann es in Kurvenfahrten, in denen die Fahrzeuge den fehlerhaft berechneten Korridor verlassen, zu Kollisionen mit anderen Fahrzeugen und somit zu

[11] z. B. die Route mit der kürzesten Fahrzeit oder dem kürzesten Weg

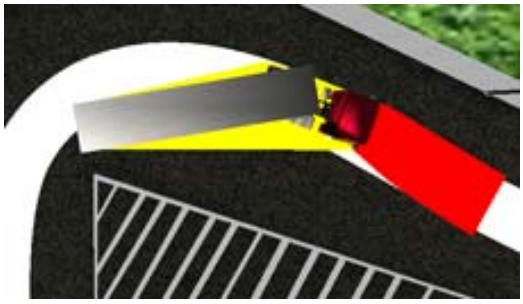


Abbildung 6.2: Verlassen des Korridors bei Kurvenfahrten

Deadlocks kommen. Abbildung 6.2 zeigt das Verlassen des Korridors (weiße Fläche) durch ein Fahrzeug in einer Kurvenfahrt. Zur Berechnung des Korridors sollte daher nicht nur die Trajektorie alleine, sondern auch die aktuellen Fahrzeugdaten, wie der Lenk- und Einknickwinkel mit einbezogen werden.

6.5 Schlussbetrachtung

Im Rahmen dieser Arbeit wurde ein allgemeines Konzept für Fahrerlose Transportsysteme, das beispielhaft auf den Anwendungsfall eines Logistikhofes angewandt wurde, auf einen neuen Anwendungsfall, dem der Rollenden Landstraße, portiert. Während dieses Vorgangs musste das System um Fähigkeiten erweitert werden, die unbedingt nötig waren, um ein Terminal der RoLa mit autonomen Fahrzeugen zu betreiben. Für diesen Fall wurde beispielhaft das Terminal in Regensburg herangezogen. Seine Topologie wurde in einer speziellen Karte nachempfunden und so modifiziert, dass unbegleiteter Verkehr abgewickelt werden kann. D. h., dass die Fahrer ihre Fahrzeuge nicht mehr wie bisher auf der ganzen Fahrt begleiten, sondern entweder das komplette Fahrzeug (sofern es für autonome Fahrt gerüstet ist) oder nur den Anhänger oder Sattelauflieger abgeben, welcher dann von einem Terminaltraktor übernommen wird. Die 3D-Visualisierung, die ebenfalls zu den gegebenen Grundlagen gehört, wurde so modifiziert, dass sie alle Vorgänge, die für den Leitstand wichtig sind, darstellen kann.

Während der Arbeit wurde der Begriff „Allgemeiner Leitstand“ definiert und es wurde herausgearbeitet, welche Funktionalitäten zu ihm gehören und welche den speziellen Anwendungsfällen zuzurechnen sind. Im weiteren Verlauf wurden die zugrunde liegende Software um wichtige Fä-

higkeiten erweitert und einige Fehler beseitigt, wodurch die Qualität des Allgemeinen Leitstands verbessert wurde. Auf Grundlage dieser Arbeiten konnten am Ende die qualitativen Eigenschaften des Leitstands, sowie zum Teil der EZauto-Bibliothek, welche vom Leitstand in umfangreichem Maße genutzt wird, unter dem Aspekt einiger softwaretechnischer Kriterien untersucht und bewertet werden. Dabei lag das Hauptaugenmerk auf der Wiederverwendbarkeit des Systems, da das Gesamtprojekt darauf abzielt, ein möglichst allgemeines Konzept für Fahrerlose Transportsysteme zu entwickeln, das in vielen Umgebungen Anwendung finden können soll.

Durch die vorliegende Arbeit konnte gezeigt werden, welches Potenzial im Konzept des Leitstands steckt. Es wurde jedoch auch aufgedeckt, an welchen Stellen dringender Nachholbedarf existiert, bzw. wo die Grenzen dieses Systems liegen. Des Weiteren wurden Möglichkeiten aufgezeigt, den Leitstand und dessen Simulation weiter abzurunden und eine Mechanik vorgestellt, wie das System auf andere Anwendungen portiert werden kann. Bis auf die Simulation der Mischbereiche können alle Vorgänge auf einem RoLa-Terminal durch die durchgeführten Erweiterungen dargestellt werden. Unter Berücksichtigung der in dieser Arbeit gegebenen Hinweise und Vorschläge, kann die Leitstandssoftware so überarbeitet werden, dass sie sich noch flexibler an eventuelle neue Anwendungsgebiete anpassen lässt.

Kapitel 7

Anhang

7.1 Hinweise zur Übersetzung und Inhalte der CD

In diesem Abschnitt soll kurz der Umgang mit dem Quelltext dieser Arbeit, der sich auf der beiliegenden CD befindet, erläutert werden. Die CD enthält die folgenden Ordner:

- *Ausarbeitung* - enthält dieses Dokument im Pdf-Format und die zugehörigen Latex-Quelldateien.
- *ControlCenter* - enthält den Quelltext des Leitstandes für autonomes Fahren in der Spezialisierung dieser Arbeit mit dem Stand vom 16. Januar 2007.
- *Doxygen-Dokumentation* - enthält die mit Doxygen generierte Kommentierung des C++ - Quelltextes. Erstellt wurde diese am 16. Januar 2007 und basiert auf den Kommentierungen der Projekte EZsped und EZrola. Diese kann über die Startseite *Doxygen-Kommentierungen.html* in einem Webbrowser angesehen werden.
- *Regensburg* - enthält die Karte des RoLa-Terminals in Regensburg, die dankenswerterweise durch die Betreiber des Terminals zur Verfügung gestellt wurde. Ebenso finden sich an dieser Stelle die Modellierungen des Terminalgeländes in Quelldateien der Open-Source-Software Blender wieder. Des Weiteren enthält dieser Ordner einen Unterordner mit Bildern, die am Tage der Besichtigung des Terminals in Regensburg gemacht worden sind.
- *Visualisierung* - enthält alle Quelldateien der Visualisierung sowie weitere zur Darstellung notwendige Dateien.

Soll die Simulation gestartet werden, so bietet sich folgendes Vorgehen an. Im Ordner *ControlCenter* befinden sich mehrere Unterordner. Der Quelltext des Leitstandes befindet sich im Ordner *ControlCenter*, während die Ordner *OrderAcceptance, SimulatedTrack* und *VirtualEnvironment* den Quelltext aller externen Module enthalten. Bevor die Simulation nun gestartet werden kann, muss der Quelltext zunächst in einen ausführbaren Programmcode übersetzt werden. Dazu wurde im Rahmen dieser Arbeit stets das Programm Visual Studio 2005[1] von Microsoft verwendet. Die dazu gehörende Projektdatei befindet sich im Ordner *project* und nennt sich *ControlCenterAll.sln*. Wird diese in Visual Studio geöffnet, so kann auf alle zugehörigen Programmkomponenten zugegriffen werden. Die externen Module sind ebenfalls in die Projektdatei eingebunden.
Die Grundlage zur Übersetzung des Leitstandes bildet die EZauto-Bibliothek. Diese muss daher ebenfalls in übersetzter Form vorliegen. Der Unterordner *EZ* vom Ordner *ControlCenter* dient daher als Platzhalter für die angesprochene Bibliothek. Sie kann in diesem Ordner gespeichert und übersetzt werden. Auch hierzu bietet sich die Verwendung des Visual Studios an, da ebenso alle Elemente in einer Projektdatei zusammengefasst sind. Liegt diese nun in übersetzter Form vor, so kann im Anschluss der Leitstand im gewünschten Modus (*Debug* oder *Release*) übersetzt werden.
Zur Ausführung des Leitstandes im *Release*-Modus bietet sich die Verwendung der Datei *startAllRelease.bat* an. Diese startet nacheinander die Ereignisverteiler und alle Komponenten des Leitstandes. Dabei muss nach dem Starten jeder Komponente der Start der Nächsten durch das Drücken einer beliebigen Taste initiiert werden. In der aktuellen Verwendung ist der Einsatz eines einzigen Rechners vorgesehen, sodass alle Module auf diesem ablaufen. Dies stellt eine sehr hohe Last für den jeweiligen Rechner dar, sodass eine Verteilung auf mehrere Rechner von Vorteil ist. In den Testszenarien dieser Arbeit wurden dazu meist drei Rechner eingesetzt. Der erste Rechner führte dabei alle Ereignisverteiler und die externen Module (mit Ausnahme der Visualisierung) aus, während auf dem Zweiten die Module des Leitstandes gestartet wurden. Der dritte Rechner diente zur Ausführung der Visualisierung. Soll dies in einem Testszenario weiterhin so gehandhabt werden, so muss die Datei *startAllRelease.bat* angepasst und die Module auf die verschiedenen Rechner verteilt werden. Beim Starten ist es wichtig, die Module in fester Reihenfolge nacheinander zu starten, so wie diese in der Startdatei vorgegeben ist. Ebenso müssen vorher die Netzwerkeinstellung angepasst werden.
Das ablaufende Szenario kann über die zu den einzelnen Modulen gehörenden XML-Dateien be-

[1] Eine Ausnahme bildet die Visualisierung. Hier empfiehlt sich dringend der Einsatz des Visual Studio 2003, vgl. Abschnitt 7.7.1

einflusst werden. So besitzen die Ordner *ControlCenter, OrderAcceptance, SimulatedTrack* und *VirtualEnvironment* jeweils einen eigenen Unterordner namens *Data*, der die zugehörigen XML-Dateien enthält. Deren Aufbau sowie die notwendigen Modifikationen werden in den folgenden Abschnitten 7.3 bis 7.5 genauer erläutert.

Der Ordner *Visualisierung* enthält ebenfalls eine Reihe von Unterordnern, die als Platzhalter für die notwendigen Bibliotheken dienen. Welche dies genau sind, wird in Abschnitt 7.7.1 erläutert. Auch hier findet sich die EZauto-Bibliothek als Grundlage wieder und sollte ebenso in den richtigen Ordner integriert und dort übersetzt werden. Der eigentliche Quelltext der Visualisierung befindet sind im Unterordner *Visualisierung*|*CSVisualization*|*src*. Auch in diesem Falle existiert im Unterordner *Visualisierung*|*CSVisualization*|*bin*|*project* eine Projektdatei zur Bearbeitung mit Visual Studio. Alle zur Darstellung notwendigen Dateien, wie z. B. Texturen oder die Geländemodellierung, müssen im Ordner *Visualisierung*|*CSVisualization*|*bin*|*data* abgelegt sein. Der übersetzte Programmcode kann auch hier durch die zugehörige Datei *StartCSVisualization.bat* gestartet werden.

7.2 Lageplan und Kapazität des Geländes

Die Zielsetzung für den Entwurf des Terminal-Geländes war es, alle gestellten Anforderungen so realistisch wie möglich in das Gelände zu integrieren. Dabei sollte garantiert werden, dass die eingeplanten Fahrwege sowie alle Stellflächen von realen Fahrzeugen befahren werden können. Dabei sollte das Terminal-Gelände aber so effizient wie möglich ausgelastet werden. Um einen autonomen Betrieb überhaupt zu gewährleisten, mussten folgende Anforderungen an das Gelände gestellt werden:

- Die Einfahrt zum Gelände hin muss breit genug sein, sodass im Bereich der Übernahme zwei Lkw nebeneinander parken können.
- Es müssen zwei Übergabebereiche geschaffen werden, sodass jeweils ein gesamtes Fahrzeug darin Platz findet.
- Unmittelbar hinter den Übergabebereichen müssen Wendekreise für die Zugmaschinen der Kunden garantiert werden, sodass diese problemlos nach dem Abkuppeln das Gelände

verlassen können.

- Die Wartespuren für Terminaltraktoren mit Sattelaufliegern sollten mindestens 3 m Breite aufweisen, jede Spur muss anfahrbar sein, auch wenn in benachbarten Spuren bereits geparkt wird.
- Die Parkspuren für die Traktoren müssen mindestens eine Breite von 3 m aufweisen.
- Die Parkplätze für die Abholer benötigen für Rangierfahrten eine Mindestlänge von 20 m, sowie eine Mindestbreite von 3 m.
- Der Wendekreis im Abholerbereich für Kundenfahrzeuge mit Sattelauflieger muss gewährleistet sein.
- Die Ein- und Ausfahrt des Abholerbereiches muss für einzelne Zugmaschinen sowie komplette Züge befahrbar sein.
- Alle Fahrwege sowie Parkplätze müssen ausreichend Platz bieten um problemlos befahren werden zu können.

Um zu garantieren, dass alle Fahrwege sowie Parkplätze befahrbar sind, wurden Straßenbauvorschriften als Orientierung herangezogen. Alle einzuhaltenden Mindestmaße wurden der Quelle [Neu], Seiten 417 bis 427, entnommen. Aus diesen Richtlinien entstand der in Abbildung 7.1 dargestellte CAD-Entwurf, der die wesentlichen Maße der verwendeten Geländenutzung zeigt. Auf dieser Basis wurde im Anschluss die dreidimensionale Umgebung als Darstellung in der Visualisierung entworfen. Die Gesamtkapazität des Geländes sieht derzeit somit

- 2 Übergabebereiche,
- 32 Stellplätze in 7 Wartespuren für Traktoren mit Sattelauflieger,
- 18 Parkplätze für Traktoren in 3 Parkspuren,
- 25 Parkplätze im 45 Grad Winkel im Übergabebereich

vor.

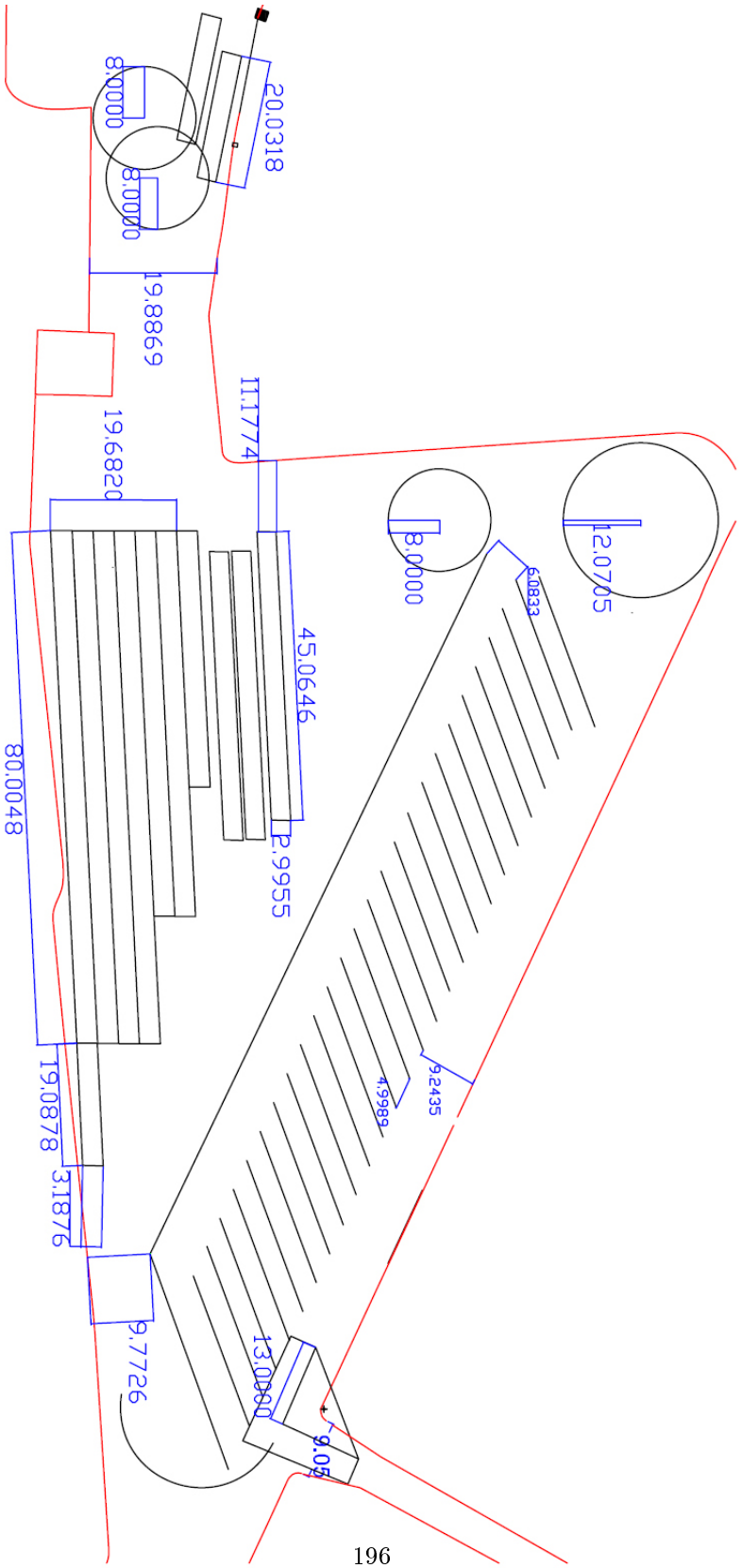


Abbildung 7.1: Lageplan des Terminals Regensburg

7.3 Statische Geländebeschreibung

Die Grundlage eines Szenarios ist das Gelände, das diesem zugrunde liegt. Die Beschreibung des Geländes (auf abstrakter, nicht auf visueller Ebene) erfolgt in einer Beschreibungsdatei, die beim Start vom Leitstand eingelesen wird. Die Topologie des Hofes wird im Fall von EZrola reduziert auf das sog. Leitliniennetz. Fahrzeuge, die sich über den Hof bewegen, richten sich immer grob nach Leitlinien. Soll ein Lkw von Punkt A nach Punkt B fahren, so werden zuerst Leitlinien gesucht, die sich in der Nähe dieser Punkte befinden. Anschließend wird ein Pfad gesucht, der ausgehend von der Startleitlinie zur Endleitlinie führt. Zu diesem Zweck sind Leitlinien über Vorgänger-Nachfolger-Beziehungen miteinander verbunden. Dazu dienen die sog. Leitlinienkomponenten. Eine Leitlinie kann bis zu vier Leitlinienkomponenten besitzen. Diese geben an, in welche Richtung die Leitlinie befahren werden kann und ob sich das Fahrzeug dabei vorwärts oder rückwärts bewegen darf. Jede dieser Möglichkeiten ist gespeichert in einer bestimmten Komponente. Die vier Möglichkeiten sind also:

1. in Linienrichtung vorwärts
2. in Linienrichtung rückwärts
3. gegen Linienrichtung vorwärts
4. gegen Linienrichtung rückwärts

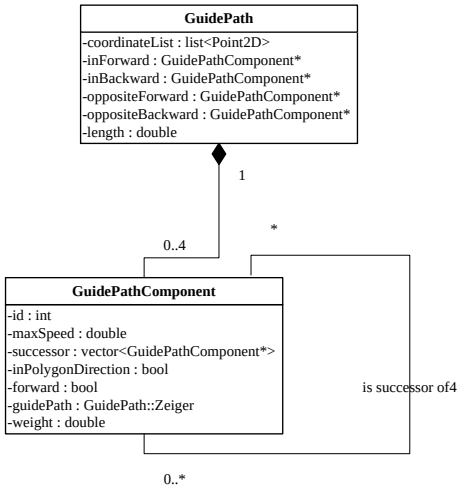


Abbildung 7.2: Statische Beziehung zwischen Leitlinien und Leitlinenkomponenten

Die Vorgänger-Nachfolger-Beziehung der Leitlinien beziehen sich auf diese Komponenten. Eine Leitlinie kann auch mit sich selbst in Beziehung stehen. Angenommen Leitlinie n habe die Komponenten (1) und (4) und (4) ist Nachfolger von (1) so kann das Fahrzeug auf dieser Leitlinie vorwärts fahren und dann auf derselben Linie wieder rückwärts. Im RoLa-Terminal Regensburg wird genau diese Beziehung benutzt, um z. B. Traktoren rückwärts in die Traktorspuren einfahren zu lassen. Sie fahren auf einer Linie so weit vorwärts an der Einfahrt der Spur vorbei, dass sie rückwärts in die Spur einlenken können.

Die Beschreibungsdatei enthält also Objekte vom Typ *GuidePath* und solche vom Typ *GuidePathComponent*. In Listing 7.1 ist ein Beispiel für eine Leitlinie, die aus zwei Punkten besteht, zu finden. Sie besteht außerdem aus zwei Komponenten, *inForward* und *oppositeBackward*. Es handelt sich hierbei um den rechten Kuppelplatz. Dieser muss rückwärts befahrbar sein, damit Traktoren rückwärts an den Auflieger heranfahren können, den sie abholen.

Listing 7.1: Beispiel einer Leitlinie

```
<!--Kuppelplatz rechts-->
  <Objekt id="guidepath152" typ="GuidePath">
    <Gruppe name="Coordinates">
      <int name="number">2</int>
      <Gruppe name="0" typ="Point2D">
        <double name="x">-47.147</double>
        <double name="y">234.405</double>
      </Gruppe>
      <Gruppe name="1" typ="Point2D">
        <double name="x">-26.018</double>
        <double name="y">229.710</double>
      </Gruppe>
    </Gruppe>
    <Objektreferenz name="inForward">guidepathcomponent1521
    </Objektreferenz>
    <Objektreferenz name="inBackward">NULL</Objektreferenz>
    <Objektreferenz name="oppositeForward">NULL</Objektreferenz>
    <Objektreferenz name="oppositeBackward">guidepathcomponent1524
    </Objektreferenz>
  </Objekt>
```

In Listing 7.2 ist eine der zur Leitlinie passenden Komponenten zu sehen, diejenige die in Linienrichtung vorwärts zu befahren ist. Als Nachfolger ist hier nur *guidepathcomponent1531* eingetragen. Andere Komponenten sind von dieser Komponente aus unerreichbar. In jeder Komponente ist noch eine Rückbeziehung zur Leitlinie gespeichert. In der XML-Datei sind die IDs für Objekte frei wählbar, müssen jedoch eindeutig sein. Schon vor dieser Arbeit wurde es zum Standard, die Leitlinienkomponenten nach der Leitlinie zu benennen, zu der sie gehören und eine Zahl (1 bis 4) anzuhängen, die den Typ der Leitlinie angibt. Aus dem Namen *guidepathcomponent1521* lässt sich folgern, dass es sich um eine Leitlinienkomponente handelt. Diese gehört zu einer Leitlinie, die den Namen *guidepath152* trägt und lässt sich in Linienrichtung vorwärts befahren.

Listing 7.2: Beispiel einer Leitlinienkomponente

```
<Objekt id="guidepathcomponent1521" typ="GuidePathComponent">
    <int name="id">1521</int>
    <double name="speed">3.0</double>
    <bool name="hasSuccessors">true</bool>
    <Gruppe name="successors">
      <int name="number">1</int>
      <Objektreferenz name="0">guidepathcomponent1531</Objektreferenz>
    </Gruppe>
    <bool name="inpoly">true</bool>
    <bool name="forward">true</bool>
    <Objektreferenz name="guidepath">guidepath152</Objektreferenz>
</Objekt>
```

Das Terminal Regensburg besteht in der Endversion dieser Arbeit aus 161 Leitlinien und 248 Leitlinienkomponenten. Alle sind in der Datei „*ControlCenter\Data\StaticAreaStructure.xml*“ gespeichert. Abbildung 7.3 auf der nächsten Seite zeigt das Leitlinien-Netz des Terminals. Jedes dieser Elemente muss momentan von Hand eingetragen werden, daher wäre die Entwicklung eines Werkzeugs, das diese Arbeit erleichtert, wünschenswert.

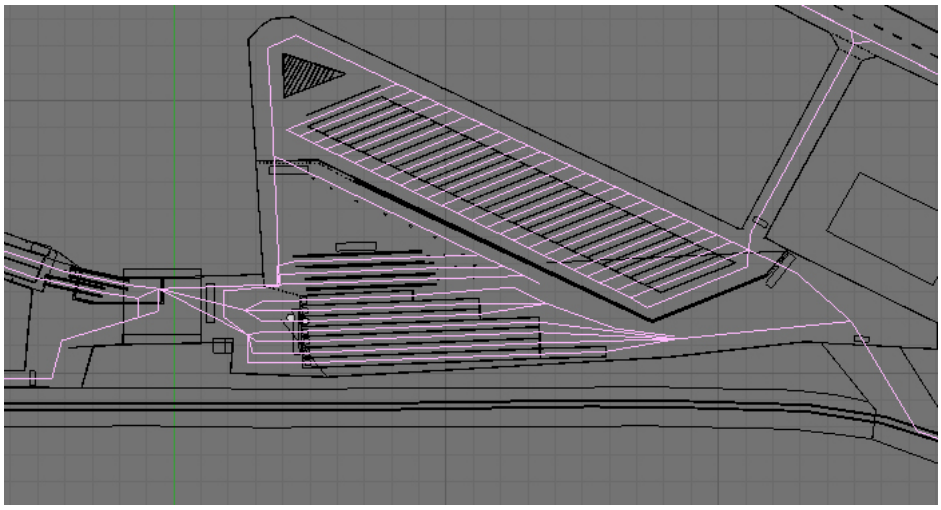


Abbildung 7.3: Leitlinien-Netz des Terminals Regensburg

7.4 Statische Aufgabenstruktur

Die statische Aufgabenstruktur ist gespeichert in der Datei „*ControlCenter\Data\StaticTaskStructure.xml*“. Diese Datei beinhaltet die Informationen über alle Aufgaben, die auf dem Gelände des RoLa-Terminals ausgeführt werden können. Die Aufgabenstruktur muss ein gerichteter azyklischer Graph sein. In ihrer momentanen Form hat sie eine Baumstruktur. Das bedeutet, dass kein Knoten mehr als einen Vorgänger hat. Die Wurzel des Graphen ist vom Typ *TaskNodeList*, enthält also eine Liste weiterer Nachfolger. Die Wurzel dient als Unterteilung in die verschiedenen Aufgabenkategorien. Sie ist in Listing 7.3 auf der nächsten Seite zu finden. Die ID ist *1*. Das Objekt beinhaltet außerdem ein Attribut zur Beschreibung des Tasks (*Description*) und eine Information darüber, ob die Planung dieser Aufgabe vor Beendigung der vorangehenden Tasks möglich ist (*waitForCompletionOfPredecessor*). Als Nächstes werden die Referenzen auf die Nachfolger gesetzt. In der vorliegenden Version werden fünf Aufgabenkategorien unterschieden:

- *tasknode11* beinhaltet Fahraufgaben zu bestimmten Punkten auf dem Hof. Z. B. zu den beiden Übergabeplätzen oder zu bestimmten Punkten außerhalb des Hofes, wo die Fahrzeuge dann vom Leitstand abgemeldet werden.
- *tasknode17* beinhaltet Transformationsaufgaben.
- *tasknode18* enthält die Aufgabe „*Verlasse den Gleisbereich*“.

- *tasknode10* ist als Platzhalter für dynamische Tasks zu sehen. Dazu sind Aufgaben zu zählen, die nicht von außen vorgegeben werden und der Abwicklung der Kundenaufträge dienen, sondern automatisch vom Leitstand erzeugt werden. Im Falle dieser Arbeit zählt hierzu das Aufrücken in Traktorspuren. Nähere Informationen, wie Zielpunkte, werden dann erst zur Laufzeit ausgefüllt.
- *tasknode14* beinhaltet alle Parkaufgaben.

Listing 7.3: Wurzel der Aufgabenstruktur

```
<Objekt id="ROOT" typ="TaskNodeList">
        <int name="ID">1</int>
        <string name="Description">root</string>
        <bool name="waitForCompletionOfPredecessor">false</bool>
        <Gruppe name="nodelist">
                <int name="number">5</int>
                <Objektreferenz name="0">tasknode11</Objektreferenz>
                <Objektreferenz name="1">tasknode17</Objektreferenz>
                <Objektreferenz name="2">tasknode18</Objektreferenz>
                <Objektreferenz name="3">tasknode10</Objektreferenz>
                <Objektreferenz name="4">tasknode14</Objektreferenz>
        </Gruppe>
</Objekt>
```

Jeder dieser Knoten ist entweder vom Typ *StaticTask*, wenn es sich um ein Blatt des Baumes handelt, oder er ist - wie die Wurzel auch - vom Typ *TaskNodeList*. Zur näheren Erläuterung soll hier ein Weg durch den Baum weiter verfolgt werden. Dazu wird hier beispielhaft das Parken in einer Wartespur ausgewählt. Zu diesem Zweck muss das Objekt mit dem Namen *tasknode14* besucht werden. Es ist in Listing 7.4 auf der nächsten Seite zu finden.

Listing 7.4: „Park-Knoten“ der Aufgabenstruktur

```
<Objekt id="tasknode14" typ="TaskNodeList">
        <int name="ID">14</int>
        <string name="Description">park</string>
        <bool name="waitForCompletionOfPredecessor">false</bool>
        <Gruppe name="nodelist">
                <int name="number">4</int>
                <Objektreferenz name="0">tasknode140</Objektreferenz>
                <Objektreferenz name="1">tasknode141</Objektreferenz>
                <Objektreferenz name="2">tasknode143</Objektreferenz>
                <Objektreferenz name="3">tasknode144</Objektreferenz>
        </Gruppe>
</Objekt>
```

Dieser Knoten enthält wieder vier weitere Knoten:

- *tasknode140* für das „gewöhnliche“ Parken auf Stellplätzen, die für Abholer reserviert werden
- *tasknode141*, von der Struktur wie *tasknode140*, jedoch für Fahrzeuge, die auf einen Folgezug warten
- *tasknode143* für das Parken in Wartespuren
- *tasknode144* für das Parken in Traktorspuren

An dieser Stelle ist bereits zu beobachten, dass die Vergabe der IDs, der Übersichtlichkeit wegen, gewissen Konventionen unterworfen wurde. Die ID des Vorgängers wird immer vorangestellt. So ist aus der ID des Knotens für das Parken in Wartespuren (*tasknode143*) herauszulesen, dass es vom Knoten mit der ID *14* abstammt, der wiederum vom Knoten mit der ID *1* abstammt.

tasknode143 hat wiederum sieben Nachfolger, von denen jeder einer bestimmten Wartespur entspricht. Diese haben die IDs *1431* bis *1437*. Davon wird jetzt die erste Wartespuren herausgegriffen und deren Eintrag (Listing 7.5 auf der nächsten Seite) genauer betrachtet:

Listing 7.5: Aufgabenknoten für das Parken in der ersten Wartespur

```
<Objekt id="tasknode1431" typ="StaticTask">
        <int name="ID">1431</int>
        <string name="Description">park on waiting lane 1</string>
        <bool name="waitForCompletionOfPredecessor">true</bool>
        <Gruppe name="StaticSubtasks">
                <int name="number">2</int>
                <Objektreferenz name="0">subtask14311</Objektreferenz>
                <Objektreferenz name="1">subtask14312</Objektreferenz>
        </Gruppe>
</Objekt>
```

Hier handelt es sich um ein Blatt der statischen Aufgabenstruktur, was am Typ des Objektes (*StaticTask*) zu erkennen ist. Diese Aufgabe darf erst verplant werden, wenn ihre Vorgängeraufgabe erfüllt wurde, da das Attribut *waitForCompletionOfPredecessor* den Wert *true* hat. Da es sich um ein Blatt des Baumes handelt, enthält dieses Objekt keine weiteren Knoten mehr.Hier befindet sich jedoch die genaue Beschreibung der Aufgabe und damit die Unterteilung in die atomaren Teilaufgaben, die sog. *StaticSubtasks*. Das Parken in einer Wartespur besteht also aus zwei solchen Teilaufgaben, deren Bezeichnung wieder den oben genannten Konventionen in der Nummerierung folgt:

- *subtask14311* führt das Fahrzeug zum Anfangspunkt der Wartespur
- *subtask14312* führt das Fahrzeug zu seinem Endpunkt in der Spur

Es ist wichtig, an dieser Stelle den Unterschied zwischen den *TaskNodeLists* und den *StaticTasks* zu erkennen. Beide enthalten zwar Nachfolger, jedoch haben diese unterschiedliche Bedeutungen. Die Nachfolger von *TaskNodeLists* sind entweder vom gleichen Typ, oder sie sind *StaticTasks* (beide sind abgeleitet von *TaskNode*). Die Erfüllung der Aufgabe eines beliebigen Nachfolgers erfüllt per Definition auch die Aufgabe des Vorgängers. Das bedeutet, dass nur einer der Nachfolger ausgeführt werden muss. *StaticTasks* besitzen zwar auch Nachfolger, diese sind jedoch vom Typ *StaticSubtask* und müssen alle der Reihe nach ausgeführt werden, um die Aufgabe des *StaticTasks* zu erfüllen.

Die Teilaufgaben gehören nicht mehr zum Aufgabenbaum, sondern sind eher als Bestandteile der statischen Aufgaben zu betrachten. Bei beiden Teilaufgaben handelt es sich um Fahraufgaben (*StaticDrivingTasks*), die von der Klasse *StaticSubtask* abgeleitet sind. *subtask14311* ist in Listing 7.6 auf der nächsten Seite zu finden. Dort sind alle Daten gespeichert, die zur Erfüllung der

Fahraufgabe notwendig sind. Dazu zählen eine Referenz zur Zielposition *vehicleposition14311*, die maximale Geschwindigkeit, eine eventuell feste Trajektorie und optionale Beginn- oder Endenachrichten. Bei diesen handelt es sich um Nachrichten, die verschickt werden, wenn die Teilaufgabe gestartet, bzw. beendet wird.

Listing 7.6: Erste Teilaufgabe beim Fahren in Wartespur 1

```
<Objekt id="subtask14311" typ="StaticDrivingTask">
        <int name="ID">14311</int>
        <Objektreferenz name="Targetposition">vehicleposition14311
        </Objektreferenz>
        <double name="MaxSpeed">10</double>
        <Gruppe name="Track">
                <int name="number">0</int>
        </Gruppe>
        <string name="StartMessage"></string>
        <string name="EndMessage"></string>
</Objekt>
```

Die Fahrzeugposition wird ebenfalls in dieser Datei gespeichert. Sie beinhaltet die Zielposition, die vorgeschriebene Ausrichtung des Fahrzeuges an dieser Stelle und den Referenzpunkt, mit dem die Zielposition angefahren werden soll. In diesem Fall (vgl. Listing 7.7) soll der Zielpunkt mit der Front des Fahrzeuges angefahren werden.

Listing 7.7: Fahrzeugposition der ersten Fahraufgabe für Wartespur 1

```
<Objekt id="vehicleposition14311" typ="VehiclePosition">
        <Gruppe name="Position" typ="Point2D">
                <double name="x">47.412</double>
                <double name="y">203.857</double>
        </Gruppe>
        <Gruppe name="Orientation" typ="Point2D">
                <double name="x">111.447</double>
                <double name="y">4.136</double>
        </Gruppe>
        <string name="ReferencePoint">Front</string>
</Objekt>
```

Die zweite Teilaufgabe gleicht der ersten strukturell. Ihr Zielpunkt liegt auf dem Ende der Wartespur. Der Zielpunkt wird durch die Parkplatzverwaltung[2] für jedes Fahrzeug überschrieben, da dieser jeweils nur für das erste Fahrzeug gilt, das in die Spur einfährt. Die Zielpunkte der nächsten Fahrzeuge stehen nicht vor der Laufzeit fest und werden daher dynamisch generiert.

Einige Aufgaben der statischen Aufgabenstruktur benötigen Informationen, die vor der Laufzeit nicht verfügbar sind. Diese Informationen werden dem System dann entweder als *AdditionalInformation* gesondert übermittelt (diese können beispielsweise in der Auftragsdatenbank (vgl. Abschnitt 7.5.6) gespeichert werden) oder sie bestehen, wie im oben beschriebenen Fall, aus Daten, die zwar im Aufgabenbaum enthalten sind, jedoch später überschrieben werden. Zu diesem Zweck gibt es auch einen sog. dynamischen Fahrauftrag (*tasknode101*). Dieser stellt keine eigentliche Aufgabe dar und dient nur dazu, dynamisch generierten Aufgaben (wie dem Aufrücken in einer Traktorspur) eine Repräsentation im statischen Aufgabenbaum zu liefern. Die dazugehörende Fahrzeugposition beinhaltet daher auch keine sinnvollen Daten, wie in Listing 7.8 zu sehen ist.

Listing 7.8: Fahrzeugposition der dynamischen Fahraufgabe

```
<Objekt id="vehicleposition1011" typ="VehiclePosition">
        <Gruppe name="Position" typ="Point2D">
                <double name="x">1.0</double>
                <double name="y">1.0</double>
        </Gruppe>
        <Gruppe name="Orientation" typ="Point2D">
                <double name="x">1.0</double>
                <double name="y">1.0</double>
        </Gruppe>
        <string name="ReferencePoint">Front</string>
</Objekt>
```

[2] vgl. Abschnitt 5.5

7.5 Erzeugung eines Szenarios

Ein Szenario ist definiert durch den Inhalt der externen Beschreibungsdateien der einzelnen Module. Zugrunde liegt jedem Szenario ein Gelände, das durch ein Leitliniennetz beschrieben ist[3]. Des Weiteren muss der statische Aufgabenbaum[4] auf das Gelände angepasst werden, da auch darin feste Punkte auf dem Hof referenziert werden.

Um ein Szenario für einen gegebenen RoLa-Bahnhof zu ändern, bzw. ein Neues zu erzeugen, müssen folgende Datengrundlagen bearbeitet werden:

- die Beschreibungen der zu verwendenden Fahrzeuge
- die Datei, die die sog. *EntryAreas* beschreibt; dabei handelt es sich um die Punkte auf dem Hof, auf denen die Fahrzeuge erzeugt werden
- die Konfigurationsdatei der virtuellen Umgebung, die angibt, welche Fahrzeuge zu welcher Zeit auf welcher EntryArea erzeugt werden sollen
- die Konfigurationsdatei der Simulation des Gleises
- der Zugfahrplan
- die Zuordnung der Aufträge zu den einzelnen Fahrzeugen

Einige Änderungen betreffen mehr die Umgebung des Szenarios, während sich andere direkt auf Vorgänge im Szenario auswirken. Liegen die Fahrzeugbeschreibungen, EntryAreas und die Konfigurationsdateien vor, so müssen für das eigentliche Szenario nur noch der Zugfahrplan[5] und die Zuordnung der Aufträge[6] angepasst werden.

Auch eine gemeinsame Uhrzeit muss festgelegt werden. Die Kontrollkomponente des Leitstandes startet momentan mit der Zeit *2005-06-27 12:00:00*. Andere Module legen ihre eigene Startzeit ebenfalls selbst fest, können sich jedoch mit der Systemzeit der Kontrollkomponente synchronisieren, indem sie die Zeit über einen sog. *TimeStampRequest* erfragen. Die Antwort enthält einen sogenannten *TimeStamp*, der die Zeit der Kontrollkomponente beinhaltet, zu dem Zeitpunkt, als die Antwort erzeugt wurde. Diese Nachricht kann im Netzwerk verzögert werden,

[3] vgl. Abschnitt 7.3
[4] vgl. Abschnitt 5.4
[5] vgl. Abschnitt 7.5.5
[6] vgl. Abschnitt 7.5.6

was den Wert verfälscht. Abweichungen, die sich im Bereich unter einer Sekunde bewegen, sind jedoch in den meisten denkbaren Fällen hinnehmbar. Daher findet kein komplizierterer Uhrenabgleich statt. Das Setzen der Uhr ist beispielhaft in Listing 7.9 zu finden. Die Synchronisation oder genauer gesagt der Umgang mit einem Timestamp in Listing 7.10.

Listing 7.9: Stellen der Uhr

```
Clock* m_clock;
m_clock =          Clock::getInstance();
m_clock->setSimulationMode(true);
m_clock->setTime("2005-06-27 12:00:00");
```

Listing 7.10: Synchronisation der Uhr

```
bool Scheduler::handleITimeStamp(double timestamp, double clockspeed)
{
        time_t t = (time_t) timestamp;
        std::cout << "TimeStamp received: " << ctime(&t);
        // set the clock according to the timestamp information.
        if(             !m_clock->setTime(timestamp)
                        ||!m_clock->setClockSpeed(clockspeed))
                        return false;
        return true;
}
```

7.5.1 Beschreibung der Fahrzeuge

Es muss dafür gesorgt werden, dass alle Fahrzeuge zur Verfügung stehen, die für das Szenario benötigt werden. Für das RoLa-Terminal musste beispielsweise der Terminaltraktor (*TraktorSIMU*) entworfen werden. Dieser ist dem *SattelschlepperSIMU* nachempfunden. In der Visualisierung ist der Unterschied zwischen diesen beiden Fahrzeugen die Fahrerkabine, die beim TraktorSIMU fehlt. Objektbeschreibungen sind in der Datei „*VirtualEnvironment\Data\ObjectDescriptions.xml*“ vorzufinden und sind Teil der virtuellen Umgebung.

Der Aufbau des Traktors ist in Listing 7.11 zu sehen. Es handelt sich hierbei um eine aktive Fahrzeugkomponente (Zeile 3) mit den drei vorhandenen Referenzpunkten *Back*, *Front* und *BackCoupling* (Zeilen 4 - 24). Außerdem wird unter anderem auch festgelegt, welche Fahrzeugteile sich mit diesem Fahrzeug verbinden lassen. In diesem Fall ist es der *AnhaengerSIMU* (Zeilen

32 - 35). Es folgen zwei referenzierte Objekte, zum einen die räumliche Ausdehnung (ab Zeile 38) und zum anderen die kinematischen Eigenschaften des Terminaltraktors (ab Zeile 60).

Listing 7.11: Objektbeschreibung des Terminaltraktors

```
<Objekt id="objectdescriptionAVC3" typ="VObjectDescriptionXML">
  <string name="typeID">TraktorSIMU</string>
  <string name="objectType">ActiveVehicleComponent</string>
  <Gruppe name="referencePoints">
    <int name="number">3</int>

    <string name="0a">Back</string>
    <Gruppe name="0b" typ="Point2D">
      <double name="x">-1.12</double>
      <double name="y">0.0</double>
    </Gruppe>

    <string name="1a">Front</string>
    <Gruppe name="1b" typ="Point2D">
      <double name="x">5.06</double>
      <double name="y">0.0</double>
    </Gruppe>

    <string name="2a">BackCoupling</string>
    <Gruppe name="2b" typ="Point2D">
      <double name="x">0.25</double>
      <double name="y">0.0</double>
    </Gruppe>
  </Gruppe>
  <Objektreferenz name="dimension">dimension_TraktorSIMU
  </Objektreferenz>
  <Objektreferenz name="staticKinematic">staticKinematic_TraktorSIMU
  </Objektreferenz>
  <Gruppe name="possibleTransportcombinations">
    <int name="number">0</int>
  </Gruppe>
  <Gruppe name="possibleTrailers">
```

```
      <int name="number">1</int>
34    <string name="0">AnhaengerSIMU</string>
    </Gruppe>
36 </Objekt>

38 <Objekt id="dimension_TraktorSIMU" typ="Polygon2D">
    <Gruppe name="vertices">
40    <int name="number">4</int>
      <Gruppe name="0" typ="Point2D">
42      <double name="x">5.06</double>
        <double name="y">1.67</double>
44    </Gruppe>
      <Gruppe name="1" typ="Point2D">
46      <double name="x">-1.12</double>
        <double name="y">1.67</double>
48    </Gruppe>
      <Gruppe name="2" typ="Point2D">
50      <double name="x">-1.12</double>
        <double name="y">-1.67</double>
52    </Gruppe>
      <Gruppe name="3" typ="Point2D">
54      <double name="x">5.06</double>
        <double name="y">-1.67</double>
56    </Gruppe>
    </Gruppe>
58 </Objekt>

60 <Objekt id="staticKinematic_TraktorSIMU" typ="StaticKinematic">
    <bool name="hasFrontAxis">true</bool>
62  <double name="offsetFrontAxis">3.65</double>
    <double name="maxSteeringAngleFront">0.87</double>
64  <bool name="hasBackAxis">true</bool>
    <double name="offsetBackAxis">0.00</double>
66  <double name="maxSteeringAngleBack">0.0</double>
    <bool name="hasFrontCoupling">false</bool>
68  <double name="offsetFrontCoupling">0</double>
```

```
    <double name="maxAngleFrontCoupling">0</double>
70  <bool name="isFrontCouplingSeparable">true</bool>
    <bool name="hasBackCoupling">true</bool>
72  <double name="offsetBackCoupling">0.25</double>
    <double name="maxAngleBackCoupling">1.57</double>
74  <bool name="isBackCouplingSeparable">true</bool>
</Objekt>
```

Für nähere Informationen zu Objektbeschreibungen sei auf [LuT] Abschnitt XV.VIII auf Seite 215 verwiesen.

7.5.2 Festlegung der „EntryAreas“

In der Datei „*VirtualEnvironment\Data\entryareas.xml*“ werden die Punkte auf dem Hof festgelegt, auf denen die Fahrzeuge erzeugt werden sollen. Für das RoLa-Terminal Regensburg enthält die Datei vier wichtige Einträge:

- zwei Punkte an der Haupteinfahrt, für Kunden die ihren Auflieger abgeben; die Punkte liegen direkt hintereinander
- ein Punkt für die Abholer der Auflieger
- ein Punkt für die Fahrzeuge, die mit dem Zug ankommen

In Listing 7.12 ist der Eintrag für die Abholer zu sehen. Neben der ID, anhand derer die virtuelle Umgebung diesen Punkt identifiziert, sind noch die Position, an der das Fahrzeug erzeugt werden soll, und die Ausrichtung, die das Fahrzeug an diesem Punkt haben soll, angegeben. Der eingetragene Punkt enthält die Position des ersten Abholers für jeden Zyklus. Diese wird von der virtuellen Umgebung für jedes weitere Fahrzeug überschrieben, da die Fahrzeuge hintereinander erzeugt werden. Für jeden weiteren Zyklus wird der Punkt wieder zurückgesetzt. In Listing 7.13 ist der Eintrag für Fahrzeuge, die auf dem Zug ankommen, zu sehen. Zu beachten ist hier, dass die Werte für Position und Ausrichtung jeweils den Wert $(0, 0)$ haben. Die angegebenen Punkte werden für jedes Fahrzeug einzeln berechnet und erst zur Laufzeit bestimmt.

Listing 7.12: Entryarea für Abholer

```
<Objekt id="mainEntranceArea3" typ="VEEntryArea">
  <int name="entryAreaID">147</int>
  <Gruppe name="position" typ="Point2D">
    <double name="x">242.572</double>
    <double name="y">329.245</double>
  </Gruppe>
  <Gruppe name="orientation" typ="Point2D">
    <double name="x">97.792</double>
    <double name="y">-49.561</double>
  </Gruppe>
</Objekt>
```

Listing 7.13: Entryarea für Fahrzeuge vom Zug

```
<Objekt id="trainEntranceArea" typ="VEEntryArea">
  <int name="entryAreaID">144</int>
  <Gruppe name="position" typ="Point2D">
    <double name="x">0.0</double>
    <double name="y">0.0</double>
  </Gruppe>
  <Gruppe name="orientation" typ="Point2D">
    <double name="x">0</double>
    <double name="y">0</double>
  </Gruppe>
</Objekt>
```

7.5.3 Konfigurationsdatei der virtuellen Umgebung

In der Konfigurationsdatei der virtuellen Umgebung kann festgelegt werden, welche Fahrzeuge zu welcher Zeit, auf welcher Entryarea erzeugt werden sollen. Hierfür muss die Datei „*VirtualEnvironment\Data\configfile.xml*“ bearbeitet werden. Dort gibt es für jedes Fahrzeug, das erzeugt werden soll, einen eigenen Eintrag. Ein Beispiel für solch einen Eintrag ist in Listing 7.14 auf der nächsten Seite zu finden.

Listing 7.14: Erzeugung eines Fahrzeuges

```
<Objekt id="vehicle2" typ="VEVehicle">
  <string name="arrivaltime">2005-06-27 12:10:00</string>
  <int name="entryAreaID">3</int>
  <Gruppe name="vehicledescriptionchain">
    <int name="number">1</int>
    <Gruppe name="0">
      <string name="typeID">TraktorSIMU</string>
      <string name="label">Traktor1</string>
      <string name="owner">Terminal</string>
      <Gruppe name="transportAllocations">
        <int name="number">0</int>
      </Gruppe>
  </Gruppe>
  </Gruppe>
</Objekt>
```

Zuerst wird festgelegt, zu welcher Zeit das Fahrzeug erstellt werden soll, hier z. B. um 12:10:00 Uhr am 27.06.2005. Die Zeit wird also absolut und nicht relativ zur Zeit des Szenarios festgelegt. Als Nächstes wird die ID der EntryArea angegeben, auf der das Fahrzeug erzeugt werden soll. Hier muss die ID eingetragen werden, die auch in der Datei *entryareas.xml* der virtuellen Umgebung angegeben wurde. Im Anschluss folgt die Fahrzeugbeschreibung des Fahrzeuges, das erzeugt werden soll. Es muss aus Fahrzeugteilen bestehen, die in der Datei *ObjectDescriptions.xml* beschrieben werden, für deren Identifikation das Attribut *typeID* zuständig ist.

Wie in Abschnitt 5.7 beschrieben, ist die Anzahl der Fahrzeuge, die im Anwendungsfall dieser Arbeit erzeugt werden müssen, so hoch, dass nicht für jedes Fahrzeug ein eigener Eintrag angelegt werden kann. Die Datei besteht daher aus einigen Standardeinträgen. Solch ein Standardeintrag ist in Listing 7.15 zu finden. Es handelt sich dabei um den Eintrag für Fahrzeuge, die auf dem Zug erzeugt werden. In diesem Eintrag spielt die Erzeugungszeit keine Rolle, daher wird sie weit in die Zukunft gesetzt. Hier wird jedoch festgelegt, welchen Typs die Fahrzeuge sind, die bei der Zugankunft verwendet werden sollen. Es handelt sich dabei also um die Kombination „*TraktorSIMU*“ - „*AnhängerSIMU*“. Für jedes dieser Fahrzeuge wird die EntryArea mit der ID *144* genutzt.

Listing 7.15: Standardeintrag in der Konfigurationsdatei der virtuellen Umgebung

```
<Objekt id="TrainEntrance" typ="VEVehicle">
  <string name="arrivaltime">2020-06-27 12:00:10</string>
  <int name="entryAreaID">144</int>
  <Gruppe name="vehicledescriptionchain">
    <int name="number">2</int>
    <Gruppe name="0">
      <string name="typeID">TraktorSIMU</string>
      <string name="label">ST0</string>
      <string name="owner">ST</string>
      <Gruppe name="transportAllocations">
        <int name="number">0</int>
      </Gruppe>
    </Gruppe>
    <Gruppe name="1">
      <string name="typeID">AnhaengerSIMU</string>
      <string name="label">ST1</string>
      <string name="owner">ST</string>
      <Gruppe name="transportAllocations">
        <int name="number">0</int>
      </Gruppe>
    </Gruppe>
  </Gruppe>
</Objekt>
```

7.5.4 Konfigurationsdatei der Gleissimulation

In der Konfigurationsdatei der Gleissimulation werden einige allgemeine Einstellungen dieses Moduls vorgenommen. Sie ist in Listing 7.16 zu finden und befindet sich in der Ordnerstruktur des Leitstandes an der Stelle „*SimulatedTrack\Data\config.xml*“. Dort werden grundlegende Dinge, wie z. B. die Länge und Breite eines Waggons, festgelegt. Außerdem wird die Position des Gleises auf dem Hof angegeben. Dazu dienen *refPoint1* und *refPoint2*. Die Gerade durch diese beiden Punkte beschreibt den Verlauf des Gleises. Die Fahrzeuge befahren den Zug in der Richtung $(refPoint2 - refPoint1)$. Der Zug kommt an einer zufallsgenerierten Stelle zwischen den beiden Referenzpunkten zum Stehen. Außerdem kann in dieser Datei die Uhrzeit für das Modul festgelegt

werden. Das Attribut *secondsTillCollection* gibt an, wie lange es nach der Ankunft eines Zuges dauert, bis die Zugmaschinen zur Abholung der Auflieger erzeugt werden. Dieser Wert darf nicht zu niedrig gewählt werden, da sonst die Auflieger nicht rechtzeitig zur Verfügung stehen. Wird der Wert zu hoch gewählt, so vergeht zwischen der Ankunft des Zuges und der Erzeugung der Abholer unter Umständen Zeit, in der der Hof stillsteht. Ebenso könnte der nächste Zug bereits einfahren, während die Auflieger des vorherigen Zuges noch die Parkplätze blockieren. In diesem Fall stehen dann nicht genügend Parkplätze zur Verfügung. Das Attribut hat momentan den Wert *600*, entspricht also zehn Minuten, was sich für einen Zug mit 18 Waggons in den Testfällen als der minimal zu empfehlende Wert herausstellte. Bei der Erstellung des Zugfahrplans ist immer darauf zu achten, dass eine ausreichende Anzahl an Stellplätzen für die Fahrzeuge bereitsteht. Des Weiteren sollten ausreichend viele freie Terminaltraktoren in den Traktorspuren stehen, um die Auflieger der Kunden abholen zu können.

Listing 7.16: Konfigurationsdatei der Gleissimulation

```
<XMLObjektarchiv version="2">
  <Objekt id="SimulatedTrackConfiguration"
  typ="SimulatedTrackConfiguration">
    <double name="wagonLength">20</double>
    <double name="wagonBreadth">4</double>
    <Gruppe name="refPoint1" typ="Point2D">
      <double name="x">331.177</double>
      <double name="y">157.874</double>
    </Gruppe>
    <Gruppe name="refPoint2" typ="Point2D">
      <double name="x">352.286</double>
      <double name="y">151.099</double>
    </Gruppe>
    <string name="startTime">2005-06-27 12:00:00</string>
    <int name="secondsTillCollection">600</int>
 </Objekt>
</XMLObjektarchiv>
```

7.5.5 Der Zugfahrplan

Im Zugfahrplan muss für jeden Zug, der das Gelände erreichen soll, ein Eintrag erzeugt werden. Der Plan wird in der Datei „*SimulatedTrack\Data\TimeTable.xml*“ gespeichert. Ein beispielhafter Eintrag aus dieser Datei ist in Listing 7.17 zu finden. Dort ist festgehalten, wann der Zug ankommt und wann er das Gelände wieder verlassen soll. Die Einträge sollten geordnet nach Ankunftszeit in der Datei gespeichert werden. Auch sollte die Ankunft eines Zuges nicht vor der Abfahrt des vorherigen Zuges liegen. Zu jedem Zug muss angegeben werden, aus wie vielen Waggons er besteht. Das Attribut *numberOfTrucks* gibt an, wie viele Lkw sich auf dem Zug befinden. Dagegen gibt das Attribut *numberOfStreetVehicles* an, wie viele Fahrzeuge für diesen Zug von außen, also von Kunden, gebracht werden sollen. Für den ersten Zug werden keine Fahrzeuge gebracht, da noch keine Traktoren zur Verfügung stehen. Der erste Zug dient daher hauptsächlich dazu, den Hof mit Traktoren zu versorgen. Zudem gibt es noch das Attribut *trainID*, das eine eindeutige Identifikation dieses Zuges enthalten muss. Diese wird benötigt, weil Fahrzeuge die Aufgabe bekommen können, auf einen bestimmten Zug zu warten, um mit diesem das Gelände wieder zu verlassen.

Listing 7.17: Eintrag aus dem Zugfahrplan

```
<Objekt id="entry2" typ="TimeTableEntry">
  <string name="arrival">2005-06-27 13:30:30</string>
  <string name="departure">2005-06-27 13:45:35</string>
  <int name="numberOfWagons">20</int>
  <int name="numberOfTrucks">16</int>
  <int name="trainID">1</int>
  <int name="numberOfStreetVehicles">10</int>
</Objekt>
```

7.5.6 Aufträge der Fahrzeuge

Die Aufträge für die Fahrzeuge werden in der Datei „*OrderAcceptance\Data\orders.xml*“ gespeichert. Diese Datei wird zu Beginn von der *OrderAcceptance* eingelesen. Ein beispielhafter Eintrag ist in Listing 7.18 auf der nächsten Seite zu finden. Zur Identifikation dient das Kennzeichen des Fahrzeuges. Dieser Eintrag gilt dem Lkw mit dem Kennzeichen *SimulatedTrack0-01-1*. Es handelt sich um ein Fahrzeug, das auf dem Zug ankommt. In der Gruppe mit dem Namen *tasks* werden die Aufträge als Referenzen auf andere Objekte in dieser Datei angegeben. Es handelt sich um die

Aufträge *leaveTrack* und *goToWaitingLane*. Diese Aufträge (ebenfalls in Listing 7.18 zu sehen) sind Standardaufträge. Den ersten Auftrag muss dabei jedes Fahrzeug ausführen, das vom Zug kommt. Das Attribut *staticTaskNodeID* in der Objektbeschreibung der Aufträge ist eine direkte Referenz in den statischen Aufgabenbaum. Dort ist also festgehalten, dass dies ein Fahrauftrag ist, der das Fahrzeug aus dem Gleisbereich führt. Der zweite Auftrag (*goToWaitingLane*) führt das Fahrzeug in eine Wartespur.

Listing 7.18: Eintrag aus der Auftragsdatenbank

```
<Objekt id="order3" typ="IOrder">
  <string name="licencePlateNumber">SimulatedTrack0-01-1</string>
  <double name="intendedArrivalTime">0</double>
  <double name="latestEndTime">0</double>
  <Gruppe name="rules">
    <int name="number">0</int>
  </Gruppe>
  <Gruppe name="tasks">
    <int name="number">2</int>
    <Objektreferenz name="0">leaveTrack</Objektreferenz>
    <Objektreferenz name="1">goToWaitingLane</Objektreferenz>
  </Gruppe>
</Objekt>

<Objekt id="leaveTrack" typ="ITask">
  <int name="staticTaskNodeID">18</int>
  <Gruppe name="additionalInformations">
    <int name="number">0</int>
  </Gruppe>
</Objekt>
<Objekt id="goToWaitingLane" typ="ITask">
  <int name="staticTaskNodeID">143</int>
  <Gruppe name="additionalInformations">
    <int name="number">0</int>
  </Gruppe>
</Objekt>
```

Bei einigen Aufträgen in dieser Datei handelt es sich um Standardaufträge, also Aufträge, die immer wieder vergeben werden. Identifiziert werden auch sie anhand des Kennzeichens. Das entsprechende Kennzeichen sollte nie tatsächlich vergeben werden. Der Standardauftrag für Fahrzeuge, die auf dem Zug ankommen, ist in Listing 7.19 zu finden. Es umfasst im Gesamten vier Einzelaufträge. Der Erste führt das Fahrzeug vom Zug, während der Zweite, *getCollected*, das Fahrzeug auf einen Abholerparkplatz schickt und zusätzlich die Information enthält, dass für dieses Fahrzeug ein Abholer erzeugt werden muss. Die weiteren Aufträge betreffen dann hauptsächlich das Zugfahrzeug. Dieses soll den Anhänger abkuppeln (*SimulatedTrackTask3*) und sich anschließend in eine Traktorspur begeben (*SimulatedTrackTask4*).

Listing 7.19: Standardauftrag für Fahrzeuge die mit dem Zug ankommen

```
<Objekt id="SimulatedTrackStandard" typ="IOrder">
  <string name="licencePlateNumber">SimulatedTrack</string>
  <double name="intendedArrivalTime">0</double>
  <double name="latestEndTime">0</double>
  <Gruppe name="rules">
    <int name="number">0</int>
  </Gruppe>
  <Gruppe name="tasks">
    <int name="number">4</int>
    <Objektreferenz name="0">SimulatedTrackTask1</Objektreferenz>
    <Objektreferenz name="1">getCollected</Objektreferenz>
    <Objektreferenz name="2">SimulatedTrackTask3</Objektreferenz>
    <Objektreferenz name="3">SimulatedTrackTask4</Objektreferenz>
  </Gruppe>
</Objekt>

<Objekt id="SimulatedTrackTask1" typ="ITask">
  <int name="staticTaskNodeID">18</int>
  <Gruppe name="additionalInformations">
    <int name="number">0</int>
  </Gruppe>
</Objekt>

<Objekt id="getCollected" typ="ITask">
```

```
  <int name="staticTaskNodeID">140</int>
  <Gruppe name="additionalInformations">
    <int name="number">1</int>
    <Objektreferenz name="0">trainInfoCollector</Objektreferenz>
  </Gruppe>
</Objekt>

<Objekt id="trainInfoCollector" typ="ITaskTrainInfo">
  <int name="trainID">1</int>
  <bool name="waitingForTrain">false</bool>
  <bool name="getsCollected">true</bool>
</Objekt>

<Objekt id="SimulatedTrackTask3" typ="ITask">
  <int name="staticTaskNodeID">172</int>
  <Gruppe name="additionalInformations">
    <int name="number">0</int>
  </Gruppe>
</Objekt>

<Objekt id="SimulatedTrackTask4" typ="ITask">
  <int name="staticTaskNodeID">144</int>
  <Gruppe name="additionalInformations">
    <int name="number">0</int>
  </Gruppe>
</Objekt>
```

Der Hof funktioniert komplett mit Standardaufträgen, demnach kann das Szenario geändert werden, ohne dass die Datei für die Aufträge modifiziert werden muss. Es müssen nur Einträge erzeugt, bzw. bearbeitet werden, wenn besondere Aufträge erwünscht sind. Wird beispielsweise ein zusätzlicher Zug im System eingefügt, dessen Fahrzeuge (genauer gesagt deren Auflieger) abgestellt und durch Kunden abgeholt werden sollen, so sind keine Änderungen an der Auftragsdatenbank nötig.

7.6 Erweiterungen

7.6.1 Abfolge der Ereignisse

In diesem Abschnitt soll die Abfolge der Ereignisse auf dem RoLa-Terminal in Hinblick auf das Zusammenspiel der verschiedenen Module näher erläutert werden. Dazu wird der Nachrichtenaustausch zwischen ihnen herangezogen. Es handelt sich um Ereignisse und Abläufe, die in den Abschnitten über die Simulation des Gleises[7], die Auftragsverwaltung[8] und dem Abschnitt über die virtuelle Umgebung[9] beschrieben werden.

Ausgangspunkt (zeitlich gesehen) soll der initiale Zustand sein. Auf dem Hof befinden sich keinerlei Fahrzeuge und die Ankunft des ersten Zuges wird erwartet. Die Abläufe sind, soweit sie zwischen den Modulen *SimulatedTrack*, *OrderAdministration* und *VirtualEnvironment* verlaufen, in Abbildung 7.4, in Form eines Sequenzdiagrammes dargestellt. Die Umsetzung der Nachrichten zwischen internen und externen Modulen, die im Teilmodul *ExternalCommunication* durchgeführt wird, wird dabei außer Acht gelassen. Außerdem werden Ereignisse wie Nachrichten behandelt, die einen festen Empfänger haben. Das entspricht nicht dem Konzept der Kommunikation über den Ereignisverteiler[10], lässt aber die Abläufe leichter und verständlicher veranschaulichen.

Alles beginnt mit der Ankunft des ersten Zuges. Die Auftragsverwaltung wird darüber von der Gleissimulation mit der Nachricht *ETrainReady* informiert. Darin enthalten sind einige Informationen, die für folgende Abläufe noch von Bedeutung sind:

- der Aufbau des Zuges (um später die Fahrzeuge an die richtige Position des Zuges leiten zu können)
- die ID des nächsten Zuges (wichtig für Fahrzeuge, die auf einen bestimmten Zug warten, da dieser über die ID identifiziert wird)
- die Anzahl der Fahrzeuge auf dem Zug (um entscheiden zu können, wann alle Fahrzeuge den Gleisbereich verlassen haben)

[7] vgl. Abschnitt 5.2
[8] vgl. Abschnitt 5.3
[9] vgl. Abschnitt 5.7
[10] zu näheren Informationen zum externen Ereignisverteiler vgl. Abschnitt 4.1.3.1

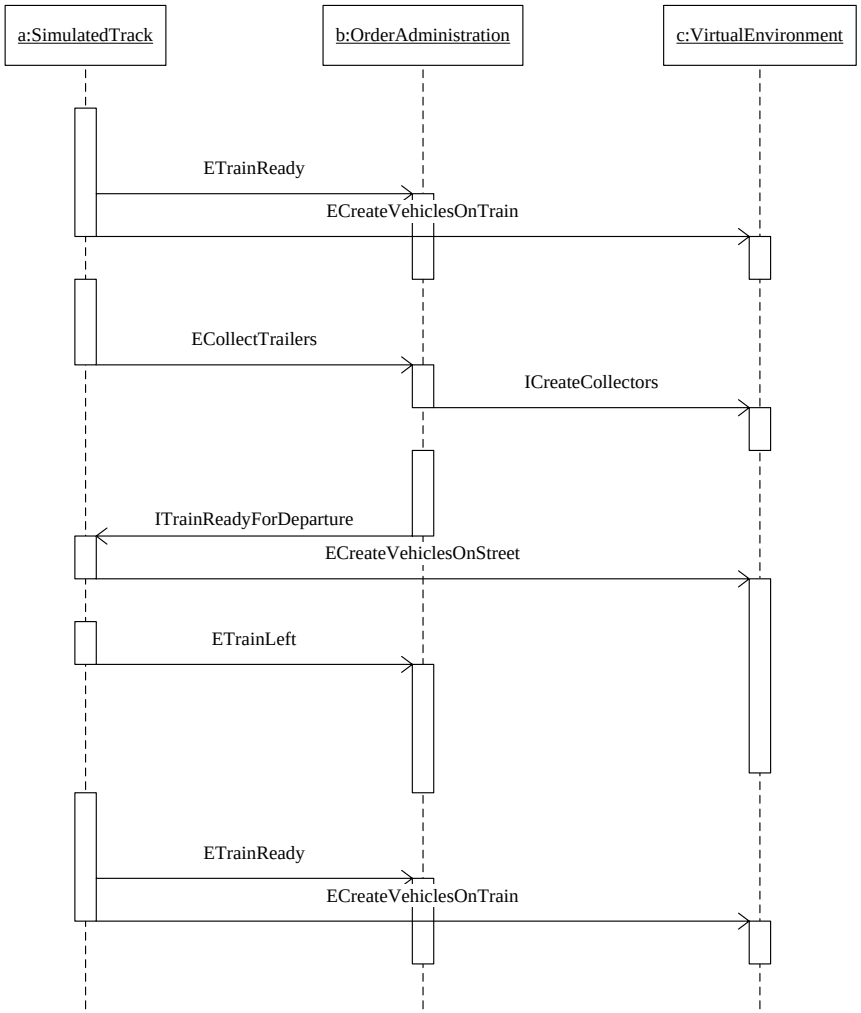


Abbildung 7.4: Abläufe auf einem RoLa-Terminal

Außerdem wird die virtuelle Umgebung dazu veranlasst, die Fahrzeuge auf dem Zug zu erzeugen, damit diese sich dann beim Leitstand anmelden. Dies geschieht mit der Nachricht *ECreateVehiclesOnTrain*. Die Fahrzeuge verlassen im Anschluss den Zug und erfüllen ihre Aufgaben auf dem Gelände.

Haben alle Fahrzeuge den Gleisbereich verlassen, so können die wartenden Fahrzeuge (aus den Wartespuren) auf den Zug auffahren. Für alle wartenden Fahrzeuge werden entsprechende Fahraufträge generiert. Da dies ein Ablauf komplett innerhalb der Auftragsverwaltung ist, wird

keine Nachricht verschickt.

Die meisten vom Zug kommenden Fahrzeuge werden für die Kunden zur Abholung bereitgestellt. Zehn Minuten nach Ankunft des Zuges verschickt die Gleissimulation daher die Nachricht *ECollectTrailers* an die Auftragsverwaltung, um die Abholung auszulösen. Die Auftragsverwaltung schickt daraufhin die Nachricht *ICreateCollectors* an die virtuelle Umgebung. Der direkte Weg von der Gleissimulation an die virtuelle Umgebung ist nicht möglich, da die Gleissimulation keine Kenntnis über die Aufträge der Fahrzeuge hat und somit nicht entscheiden kann, wie viele Abholer erzeugt werden müssen. An dieser Stelle sei nochmals darauf hingewiesen, dass die Erzeugung der Abholer nicht in den Aufgabenbereich des Leitstandes fällt, vielmehr ist sie Teil der Umgebungssimulation, wurde aber, um den Rahmen dieser Arbeit nicht zu sprengen, vorerst vom Leitstand übernommen. Diese Thematik wird in Abschnitt 6.4.2.2 näher erläutert.

Von den Fahrzeugen, die vor einiger Zeit ihre Fahraufträge auf den Zug erhalten haben, wurde die ID des Fahrauftrags des letzten Fahrzeuges gespeichert. Hat dieser sein Ziel erreicht, so sendet die Auftragsverwaltung die Nachricht *ITrainReadyForDeparture* an die Gleissimulation. Der Zug darf allerdings erst abfahren, wenn auch dessen Abfahrtszeitpunkt im Zugfahrplan erreicht ist. Im Normalfall wird er darauf noch einige Zeit warten.

Die Auftragsverwaltung beginnt direkt nach dem Ende der letzten Fahrt auf den Zug damit, die Fahrzeuge, die auf den nächsten erwarteten Zug warten, in die Wartespuren zu beordern. Dies geschieht durch den internen Aufruf der Funktion *leadParkingVehiclesOnWaitingLane()*.

Direkt nach Erhalt der Nachricht, dass der Zug abfahren kann, leitet die Gleissimulation die Erzeugung der Fahrzeuge ein, die von den Kunden für den nächsten Zug zum Terminal gebracht werden sollen. Wie viele Fahrzeuge erzeugt werden sollen, entnimmt das Modul dem Zugfahrplan. Der Wert ist für jeden Zug angegeben. Für den ersten Zug können keine Auflieger abgegeben werden, da noch keine Traktoren zur Verfügung stehen. Die Gleissimulation schickt also die Nachricht *ECreateVehiclesOnStreet* an die virtuelle Umgebung. Auch an dieser Stelle übernimmt der Leitstand momentan Aufgaben aus der Umgebungssimulation. Die Fahrzeuge der Kunden müssten vor ihm verborgen bleiben. Ausschließlich die Auflieger dürften dem Leitstand bekannt gemacht werden, sobald diese für die Abholung durch einen Traktor bereitstehen.

Ist auch die planmäßige Abfahrtszeit des Zuges erreicht, so schickt die Gleissimulation die

Nachricht *ETrainLeft* an die Auftragsverwaltung, um das Verlassen anzudeuten. Die Auftragsverwaltung kann dann die Abmeldung der Fahrzeuge veranlassen, die auf diesen Zug aufgefahren sind.

Ab jetzt wird auf den nächsten Zug gewartet, während Auflieger für diesen Zug am Terminal abgegeben werden. Die oben beschriebenen Abläufe wiederholen sich jetzt für jeden Zug, der im Zugfahrplan eingetragen ist.

Die zeitliche Ordnung der verschiedenen Abläufe auf dem Hof wird in Abbildung 7.5 noch einmal verdeutlicht. Die Dauer der jeweiligen Abläufe ist als beispielhaft zu bezeichnen, da sie immer abhängig vom jeweiligen Szenario ist. Es ist zu sehen, dass die Fahrzeuge, die von Parkplätzen in die Wartespuren fahren, zeitlich mit den Abholern kollidieren. Den Sicherheitsbestimmungen zufolge ist dies absolut zu vermeiden, da es gleichzeitige autonome sowie manuelle Fahrt in bestimmten Bereichen bedeutet. Diese Problematik kann durch die Wahl eines anderen Ereignisses als Auslöser beseitigt werden oder aber sie wird durch die Implementierung von Mischbereichen[11] umgangen. Die einfachste Lösung wäre jedoch, die Abholer nicht schon nach zehn Minuten nach Ankunft des Zuges zu erzeugen, sondern beispielsweise erst nach 20 Minuten. Dies würde auch eher der Realität entsprechen. Jedoch würde es, abhängig von anderen Faktoren, unter Umständen das Szenario verlängern, was in Bezug auf Tests nicht wünschenswert ist. Für eventuelle Vorführungen wäre dies jedoch ein adäquates Mittel.

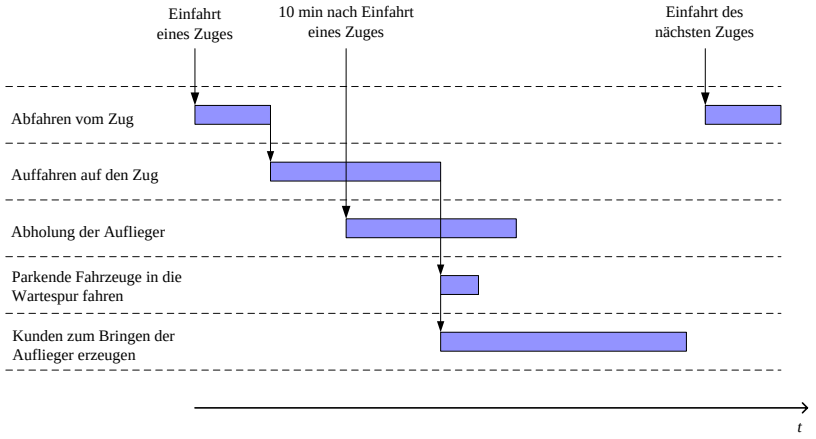


Abbildung 7.5: Zeitliche Abläufe

[11] vgl. Abschnitt 6.4.1

7.6.2 Transformationen

An dieser Stelle sollen alle für die Transformationen notwendigen Datenstrukturen, sowie Nachrichten und beteiligte Module nochmals zusammengefasst und erklärt werden. Es wird dabei mit der Erklärung der zusätzlichen Informationen begonnen. Alle beteiligten Nachrichten und deren Auswirkungen werden im Anschluss kurz zusammengefasst.

7.6.2.1 AdditionalInformation

Die *AdditionalInformation*[12] beinhalten alle zu einer Transformation notwendigen Informationen. Die vier wichtigsten Angaben sind die folgenden Variablen:

1. *Bool isAddTransformation* - gibt an, ob es sich um eine Ankuppel- (true) oder Abkuppel-Transformation (false) handelt.
2. *String targetObjectLabel* - beinhaltet das Kennzeichen oder die eindeutige Bezeichnung des ab- bzw. anzukuppelnden Sattelaufliegers oder Anhängers.
3. *String destinationObjectLabel* - wird nur im Falle der Aufnahme eines Containers benötigt und gibt das Label des Anhängers wieder, auf den der Container verladen werden soll.
4. *Point2D positionOnDestinationObject* - bezieht sich ebenso nur auf das Aufnehmen eines Containers und gibt die genaue Position auf dem im destinationObjectLabel bezeichneten Anhänger wieder[13].

Wie bereits aus der Erklärung der einzelnen Variablen ersichtlich wird, werden im Falle des EZrola-Terminals lediglich die ersten beiden Angaben benötigt, um Sattelauflieger an- und abkuppeln zu können. In diesem Falle müssen die beiden letzten Variablen nicht gesetzt werden. Sie dienen lediglich der Vorbereitung zur Aufnahme von Containern.

Da alle Fahrzeugteile leitstandsintern über ihre zugewiesene ID angesprochen werden können, reicht es für die weiteren Abläufe nicht aus, nur die oben genannten Angaben an die Module

[12] An dieser Stelle sei darauf hingewiesen, dass die zugehörige Programmstruktur *AdditionalInformations* genannt wird, was nicht der englischen Sprache entspricht. Aus diesem Grund weisen alle in dieser Arbeit enthaltenen Listings diese Schreibweise auf.

[13] im Falle, dass ein Anhänger zwei kleine Container aufnehmen kann.

weiterzuleiten. Aus diesem Grund werden die AdditionalInformation mit der Angabe der beteiligen Fahrzeug-IDs (Zugmaschine oder Traktor sowie Sattelauflieger) angereichert. Sinn dieser vollständigen Weiterleitung ist es, die Anzahl der aus einer Transformation resultierenden Nachrichten so gering wie möglich zu halten. Daher sollen weitestgehend alle Informationen an die jeweiligen Nachrichten angehängt werden. Somit soll jedes Modul nach Erhalt der Nachricht selbst in der Lage sein, die eigene Datenbank ohne weitere Nachfragen zu aktualisieren. Auch hier gilt wieder das hohe Abstraktionsprinzip und das präzise Einhalten der Schnittstellen, sodass über die Nachrichten ausschließlich elementare Datentypen ausgetauscht werden, um keine modulspezifischen Datenstrukturen weiterzuleiten.

7.6.2.2 Nachrichten und Module

An den Transformationen sind folgende Nachrichten zwischen den Modulen beteiligt:

1. *IObjectIDRequest* und *IObjectIDAnswer*
2. *ITransformationTargetPositionRequest* und *ITransformationTargetPositionAnswer*
3. *ITransformVehicle*
4. *IModifiedVehicle*
5. *VITransformation*
6. *VITransformationComplete*
7. *IDrivingJobComplete*
8. *SwitchOffSafetyForTransformation*

Die Aufgabenverteilung auf die Schichten sieht dabei wie folgt aus:

Die Aufgabenplanung sowie die Auftragsverwaltung müssen für die Vollständigkeit der *AdditionalInformation* sorgen. Dabei gibt es die Möglichkeit, diese rein statisch aus der Aufgabenplanung zu übernehmen oder aber diese dynamisch zu erzeugen. Fehlende Informationen können über die Nachrichten (1) und (2) bei der Fahrzeugverwaltung angefragt werden und bekommen die äquivalente Antwortnachricht. Aus der aktuellen Position der Anfrage (2) errechnet die

Auftragsverwaltung die Position der in der Transformation enthaltenen Fahraufgaben. Die vollständigen Angaben werden daraufhin an die Terminplanung (den *Scheduler*) weitergeleitet.

Die Terminplanung kann die Transformation nur nach Beendigung der vorherigen Aufgabe einplanen und ausführen. Mit Nachricht (3) startet sie die Transformation und informiert darüber die Fahrzeugverwaltung, die Auftragsverwaltung sowie alle externen Module. Daraufhin wartet der Scheduler nach der Aktualisierung der eigenen Fahrzeugdaten auf Nachricht (6), um alle weiteren anstehenden Aufgaben einplanen zu können.

Das Fahrzeugmanagement empfängt Nachricht (3), bereitet daraufhin die Transformation vor und informiert über Nachricht (4) die virtuelle Umgebung, sowie die Fahrzeugkontrolle. Ebenso wird Nachricht (5) an das betreffende Fahrzeug über die Fahrzeugkommunikation weitergeleitet. Wird die Nachricht (6) zum Ende der Transformation empfangen, so aktualisiert es die vorliegenden Daten und die Transformation ist damit abgeschlossen.

Die Fahrzeugkontrolle empfängt Nachricht (4) und bereitet die Transformation vor. Im Falle des Abkuppelns eines Aufliegers oder Anhängers wird das betreffende Fahrzeugteil erst nach Beendigung der direkt folgenden Fahraufgabe in die Sicherheitsüberwachung aufgenommen. Im Falle eines Ankuppelvorganges wird der Fahrzeugkontrolle vorher über Nachricht (8) mitgeteilt, nach welcher Fahraufgabe sie welchen Auflieger zum Ankuppeln aus der Sicherheitsüberwachung nehmen soll.

Das betroffene Fahrzeug empfängt mit Nachricht (5) seine neue Fahrzeugbeschreibung und sendet nach 15 s die Nachricht (6) zum Beenden der Transformation an die Terminplanung, das Fahrzeugmanagement sowie alle externen Module.

Abbildung 7.6 zeigt den Ablauf der Nachrichten zwischen den beteiligten Klassen. Die in blau dargestellte Nachricht *SwitchOffSafetyForTransformation* tritt nur im Falle einer Ankuppel-Transformation auf.

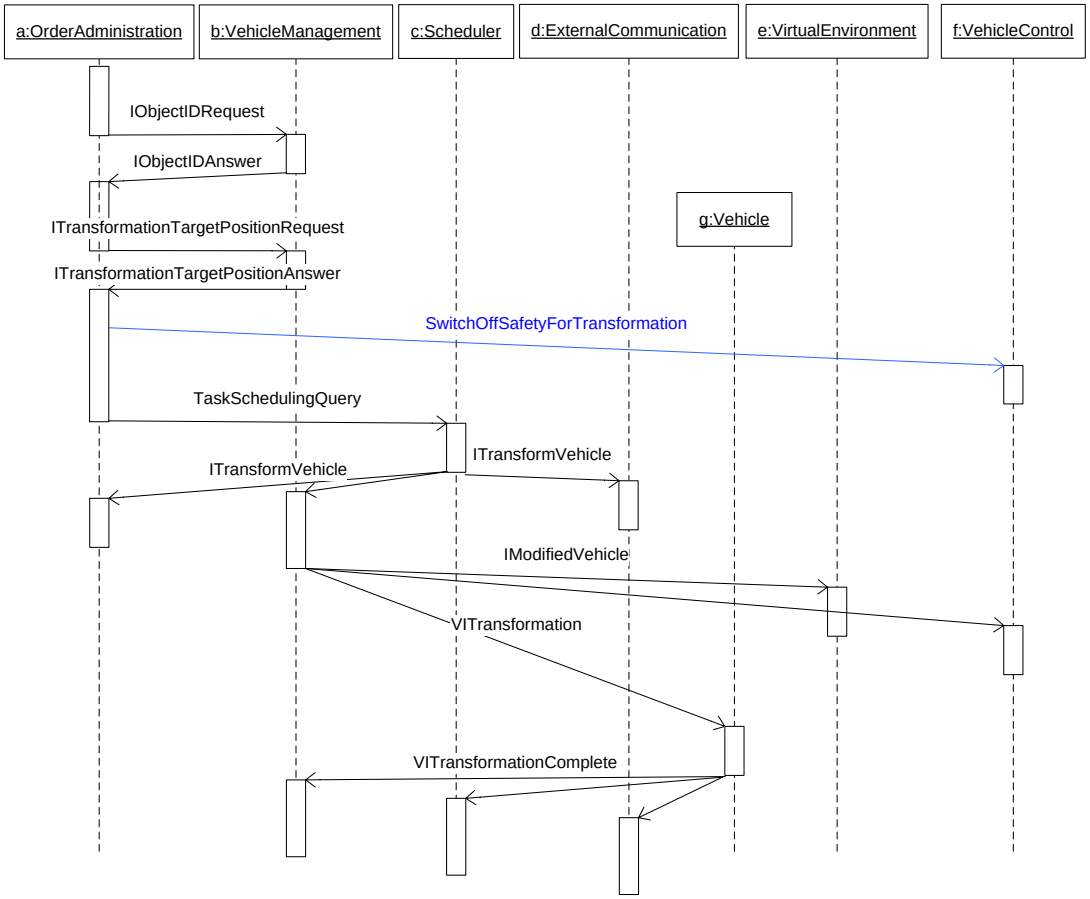


Abbildung 7.6: Sequenzdiagramm der internen Nachrichten zum Transformationsablauf

7.7 Visualisierung

In diesem Abschnitt wird nun genauer auf die Visualisierung eingegangen. Dazu wird zunächst kurz zusammengestellt, welche Einstellungen und Elemente zur Übersetzung des Quelltextes notwendig und welches die wesentlichen Datengrundlagen der Visualisierung sind. Im Anschluss werden die Quelltext-Modifikationen in Bezug auf die Transformationen genauer erläutert, ehe nochmals kurz bekannte Probleme aufgezeigt und Verbesserungsmöglichkeiten für zukünftige Arbeiten genannt werden.

7.7.1 Grundlagen und Einstellungen

Die Visualisierung basiert auf den drei folgenden Grundlagen:

- die EZauto-Bibliothek
- das wxWidgets - Framework
- das CrystalSpace - Framework, sowie den zugehörigen Bibliotheken

Um die Visualisierung übersetzen zu können, müssen alle drei Grundlagen in bereits übersetzter Form vorliegen. Während die EZauto-Bibliothek problemlos zu übersetzen ist, so sind bei der Übersetzung der beiden Frameworks vorab einige Aspekte zu beachten. Alle notwendigen Einstellungen, sowie die zu setzenden Pfade innerhalb des Programmes Visual Studio, sind in einer Hilfedatei dem Quelltext der Visualisierung beigelegt. Neben den Pfaden ist für die erfolgreiche Übersetzung von wxWidgets auch ein Patch erforderlich, der ebenfalls beim Quelltext zu finden ist und während dieser Arbeit von Philipp Wojke geschrieben wurde.
Für den Umgang mit den Grundlagen als auch der Visualisierung wird dringend die Verwendung von Visual Studio 2003 statt des neueren Visual Studio 2005 empfohlen. Der Versuch die Visualisierung auf der neueren Version zu übersetzen scheiterte und die Fehlersuche konnte im Rahmen dieser Arbeit nicht mehr fortgeführt werden.

Die Datengrundlage der Darstellung für die Visualisierung ist die mit der Open-Source-Software Blender erstellte dreidimensionale Geländekarte. Zur Erstellung dieser Karte und zur Einhaltung der Maßstäbe empfiehlt sich zunächst der Einsatz von CAD-Programmen wie AutoCAD, über das Karten des Vermessungsamtes, sofern diese vorliegen, problemlos geöffnet werden können. In dieser Umgebung bietet sich das Einzeichnen weiterer Elemente[14] an, ehe die Grundlagen der CAD-Bearbeitung in die Blender-Umgebung importiert werden. Dort kann sich nun an der zweidimensionalen Grundlage orientiert und das Geländemodell erstellt werden. Der zugrunde liegende Maßstab ist dabei so zu wählen, dass eine Blendereinheit gleich einem Meter ist. Das erstellte Geländemodell muss im Anschluss in ein der Visualisierung kompatibles Format konvertiert werden. Das dazu notwendige Programm findet sich ebenfalls beim Quelltext der Visualisierung. Im Anschluss erwartet diese die Geländekarte in der Datei *map8.csmap* im Daten-Ordner. Dort müssen ebenso alle verwendeten Texturen, sowie die XML-Datei der Leitlinien

[14] z. B. Wartespuren, Parkplätze, Sonderbereiche etc.

(*map8.xml*) des Grundstückes abgelegt werden.

7.7.2 Modifikationen des Quelltexts

Um auf die Fahrzeug-Transformationen des Leitstandes reagieren zu können, muss sich die Visualisierung zunächst, wie jedes externe Modul, für die beiden neu zu empfangenen Nachrichten *ETransformVehicle* und *ETransformationComplete* beim externen Ereignisverteiler eintragen. Bekommt sie eine solche Nachricht zugestellt, so muss ebenso die nötige Ereignisbehandlungsmethode bereitgestellt werden. Wie in Abschnitt 5.1.3 erklärt, aktualisiert die Visualisierung erst mit dem Ende einer Transformation ihre Datenbank. Daher speichert sie lediglich nach Erhalt der Nachricht *ETransformVehicle* alle notwendigen Daten in der zusätzlich angelegten Datenstruktur *infosForTransformation* im Fahrzeugmanagement. Erhält sie nun die Nachricht *ETransformationComplete*, so entnimmt sie die zuvor gespeicherten Daten und modifiziert anhand dieser ihre Fahrzeugdatenbank.
Jedes Fahrzeug wird im Fahrzeugmanagement als Objekt der Klasse *VehicleInformation* abgelegt. In dieser sind die drei wesentlichen Datenstrukturen

- die *VehicleDescriptionChain*
- die zum Fahrzeug gehörenden *typeIDs*
- die *VehicleUnit*, bestehend aus der *sFahrzeugdatenkette* und der Teilebeschreibung *sTeil*

enthalten.

Die *VehicleDescriptionChain* besteht, ähnlich zum leitstandinternen Aufbau, lediglich aus einer Abfolge von Fahrzeug-Ids, die bei der Registrierung des Fahrzeuges den Fahrzeugteilen zugewiesen wurden. Diese muss, je nach Transformation, getrennt oder zusammengefügt werden.
Die *typeIDs* geben Auskunft über die einzelnen Fahrzeugteile. Jedem mit einer ID versehenen Fahrzeugteil kann eine *typeID* zugewiesen werden. So könnte z. B. eine Zugmaschine mit Sattelauflieger die IDs 1 und 2, sowie die *typeIDs* SattelschlepperSIMU und AnhängerSIMU, besitzen. Auch diese gilt es, je nach Transformation, zu trennen oder zusammenzufügen.
Die *VehicleUnit* speichert die aktuellen Daten des Fahrzeuges[15] in der *sFahrzeugdatenkette*, sowie die genaue Zusammensetzung des Fahrzeuges in *sTeil*. Beim Trennen von Fahrzeugen gilt es,

[15] z. B. die Position, den Lenkwinkel, die Ausrichtung etc.

diese ebenso auseinander zu schneiden und den beiden neu entstandenen Fahrzeugen zuzuweisen. Analog werden beim Zusammenfügen die Daten vereint.
Um die Datenverarbeitung einer Transformation zu erleichtern, wurde größtenteils versucht, auf vorhandene Funktionen der Fahrzeugverwaltung zurückzugreifen. Beim Trennen von Fahrzeugen wurden daher die Daten des bestehenden Fahrzeuges modifiziert und ein weiteres in die Datenbank eingetragen. Beim Zusammenfügen zweier Fahrzeugteile wurden hingegen beide Fahrzeuge aus der Datenbank gelöscht und als ein neues Fahrzeug, mit den Fahrzeugdaten der beiden einzelnen Fahrzeugteile, erzeugt.

7.7.3 Problemstellungen

- Animation der Räder:
  Die Visualisierung stellte in der Ursprungsversion die Drehung der Fahrzeugräder falsch dar. Dabei wurden die Räder immer mit der gleichen Geschwindigkeit unabhängig von der Fahrtrichtung in die gleiche Richtung gedreht. Der Fehler lag dabei in der Berechnung der Interpolationspunkte der einzelnen Räder, sodass durch dessen Korrektur durch Philipp Wojke, die Raddrehungen nun der Fahrzeugbewegung entsprechen.

- Absturz nach Neustart:
  Wird die Visualisierung gestartet, so ist die voreingestellte Perspektive auf das Gelände die sogenannte *FreeView*. Wird nun ein auf dem Gelände fahrendes Fahrzeug mit dem Cursor ausgewählt, so friert die gesamte Anwendung ein. Dies geschieht jedoch ausschließlich in der initialen Kameraperspektive. Wird die Perspektive nach dem Neustart der Visualisierung sofort gewechselt, so können in allen folgenden Perspektiven die Fahrzeuge problemlos ausgewählt werden, ohne dass es zu Abstürzen führt. Vermutlich ist die Ursache eine nicht initialisierte Fenstervariable zu Anwendungsbeginn, welche erst durch das Wechseln der Perspektiven gesetzt wird.

- Passive Fahrzeugteile auf dem Gelände werden nach dem Neustart nicht dargestellt:
  Eine fehlende Funktionalität innerhalb des Leitstandes und der Visualisierung bezieht sich auf die Darstellung passiver Fahrzeugteile auf dem Gelände. Die Visualisierung meldet sich zur Darstellung der Abläufe am externen Ereignisverteiler an. Dabei bekommt sie die Daten aller sich auf dem Hof befindenden Fahrzeuge vom Leitstand mitgeteilt. Diese Daten beziehen sich allerdings nur auf aktive Fahrzeugteile und die mit diesen verbundenen

passiven Fahrzeugteile. Rein passive Fahrzeugteile werden bei der Anmeldung nicht übermittelt. Dies hat die Konsequenz, dass abgestellte Sattelauflieger oder Container nach dem Verbindungsaufbau zum Ereignisverteiler so lange nicht dargestellt werden, wie sie nicht mit einem aktiven Fahrzeugteil verbunden sind. Wird der Container aufgenommen oder der Sattelauflieger angekuppelt, so sind beide wieder mit einem aktiven Fahrzeugteil verbunden und werden von diesem Moment an dargestellt. Passive Fahrzeugteile können somit lediglich über die Verbindung zu aktiven Fahrzeugteilen in die Datenbank der Visualisierung gelangen und werden auch nur dann nach einer Trennung weiter angezeigt. Geschieht die Trennung vor der Registrierung der Visualisierung, so wird das passive Teil, wie bereits erwähnt, erst nach der Verbindung zu einem Aktiven wieder dargestellt.
Diese Problematik könnte über die Erweiterung der Kommunikation zwischen der Visualisierung und dem Leitstand behoben werden. Dazu müsste der Leitstand der Visualisierung unmittelbar nach ihrer Anmeldung auch die passiven Fahrzeugteile melden, sodass diese dargestellt werden können. Dies konnte im Rahmen dieser Arbeit nicht mehr implementiert werden.

- Performance der Darstellung:
  Während der Testphase des Leitstandes wurde das Gelände unterschiedlich stark ausgelastet. Mit steigender Anzahl der Fahrzeuge auf dem Gelände, stieg auch der Rechenaufwand zur Erstellung und Animation der Fahrzeuge an. Dabei musste festgestellt werden, dass die meiste Rechenleistung von der Visualisierung zur Erzeugung neuer Fahrzeuge aufgebracht werden musste. Dies stellte bei einer geringen Auslastung des Geländes (Anzahl der Fahrzeuge ≤ 10) zunächst keine Probleme dar, führte aber mit steigender Fahrzeuganzahl letztlich zur Asynchronität zwischen Leitstand und Visualisierung. Der Grund dafür ist Folgender.
  Die Visualisierung empfängt für jedes Fahrzeug fortwährend Nachrichten über den externen Ereignisverteiler, die ihr fahrzeugspezifische Daten übermitteln. Die Anzahl der Nachrichten nimmt somit mit der Anzahl der Fahrzeuge zu. Um die Synchronität zwischen Leitstand und Visualisierung zu gewährleisten, muss die Visualisierung in der Lage sein, alle einlaufenden Nachrichten sofort zu empfangen und auszuwerten, ohne sie eine gewisse Zeit im Postfach des Ereignisverteilers liegen zu lassen. Dies ist nur so lange gewährleistet, wie weniger oder genauso viele Nachrichten pro Zeiteinheit eintreffen, wie die Visualisierung

in der gleichen Zeiteinheit auswerten kann. Treffen jedoch mehr Nachrichten ein, so füllt sich das Postfach mit Nachrichten. Damit gerät die Visualisierung in einen zeitlichen Rückstand, da die Nachrichten eine gewisse Zeit im Ereignisverteiler gespeichert wurden, ehe sie abgeholt und ausgewertet werden konnten. Diese Tatsache wiegt gerade beim Erzeugen neuer Fahrzeuge sehr schwer, da das Einladen der Datenbanken und das Animieren des Fahrzeuges länger als die durchschnittliche Auswertung von Positionsdaten dauert. Tritt nun das Erzeugen neuer Fahrzeuge sehr schnell hintereinander auf, wie z. B. beim Eintreffen eines Zuges im Projekt EZrola, so stellt dies den Worst-Case für die Visualisierung dar, da immer mehr Nachrichten nicht zeitnah ausgewertet werden können. Dies bedeutet, dass bei konstanter Auslastung über einen längeren Simulationszeitraum gesehen, die zeitliche Lücke zwischen Leitstand und Visualisierung zunimmt. Im Rahmen dieser Arbeit betrug die Verzögerung nach einem Szenario mit zwei Stunden Laufzeit zwischen 70 und 80 s.

Ein Verbesserungsvorschlag an dieser Stelle wäre der Einsatz vereinfachter Fahrzeugmodelle, sodass das Erzeugen eines neuen Fahrzeuges in kürzerer Zeit geschehen kann. Dies konnte im Rahmen dieser Arbeit allerdings nicht realisiert werden, sodass lediglich über die Minimierung der Nachrichten eine Leistungssteigerung erzielt werden konnte. Dazu wurde die Visualisierung für den Empfang der Fahrzeugpolygone beim Ereignisverteiler ausgetragen, sodass deutlich weniger Nachrichten empfangen werden musste. Dies ist allerdings nur eine provisorische Lösung für spezielle Testfälle gewesen und so sollte nach der Modifikation der Fahrzeugmodelle der Empfang aller Nachrichten wieder ermöglicht werden.

Literaturverzeichnis

[Anh] Tuan Le-Anh und M. B. M De Koster, 2005: *„A review of design and control of automated guided vehicle systems“*, Zeitschrift: European journal of operational research, Ausgabe 171/1, Seiten 1 bis 23.

[Dui] Mark B. Duinkerken, Joseph J.M. Evers und Jaap A.Ottjes, 1999: *„TRACES: Traffic Control Engineering System“*, Summer Computer Simulation Conference, July 1999, Chicago, `http://www.ocp.tudelft.nl/tt/users/duinkerk/papers/chi9907b.pdf`, Abrufdatum: 12.01.2007.

[ECE] Economic Commission for Europe of the United Nations (UN/ECE), 2001: *„Terminologie des Kombinierten Verkehrs“*, New York und Genf, `http://www.oecd.org/dataoecd/42/32/1941816.pdf`, Abrufdatum: 15.12.2006.

[EUK] Europäische Kommission, 2004: *„Energy & Transport in Figures 2004“*, Europäische Kommission, Generaldirektion für Energie und Transport in Zusammenarbeit mit Eurostat.

[Hal] S.Hallé, F.Gilbert, J. Lauchmonier and B.Chaib-draa, 14. April 2003: *„Architectures for Collaborative Driving Vehicles: From a Review to a Proposal“*, Départment d'informatique et de génie logiciel, Faculté des Sciences Université Laval, Canada. `http://www.damas.ift.ulaval.ca/~halle/publi/LavalFCDReportV3.pdf`, Abrufdatum: 12.01.2007.

[HGB] Bundesrepublik Deutschland, November 2006: *„Handelsgesetzbuch“*.

[Kim-a] Kap Hwan Kim, Hans-Otto Günther, 2005:*„Container Terminals and Automated Transport Systems“*, Springer-Verlag Berlin Heidelberg.

[Kim-b] Kap Hwan Kim, Su Min Jeon und Kwang Ryel Ryu, 2006: *„Deadlock prevention for automated guided vehicles in automated container terminals “*, OR Spectrum Ausgabe 28,4, Seiten 659-679, Springer-Verlag Berlin Heidelberg.

[Kra] Hartmut Krasemann und Ulrich Spindel: *„Softwarearchitektur für einen Containerterminal“*, Hamburg, `http://users.informatik.haw-hamburg.de/~raasch/AKOT/CTAinJAVA-2.pdf`, Abrufdatum: 12.01.2007.

[Lim] Jae Kook Lim, Kap Hwan Kim, Kazuho Yoshimoto, Jun Ho Lee und Teruo Takahashi: *“A dispatching method for automated guided vehicles by using a bidding concept“*, [Kim-a], Seiten 325 bis 344.

[LuT] René M. Lotz und Vanessa Thewalt, 2006: *„Entwicklung eines Konzeptes für Fahrerlose Transportsysteme am Beispiel des Innovativen Logistikhofes“*, Diplomarbeit, Universität Koblenz-Landau, Daimler-Chrysler AG Stuttgart-Untertürkheim.

[Neu] Peter Neufert, 2005: *„Bauentwurfslehre“*, Vieweg.

[Oek-a] Ökombi GmbH, November 2005: *„Presseinformation: Neue ÖKOMBI startet mit ROLA durch“*, `http://www.ots.at/presseaussendung.php?ch=wirtschaft&schluessel=OTS_20051102_OTS0082`, Abrufdatum: 15.12.2006.

[Oek-b] Ökombi GmbH, *„RoLa-Verladehinweise“*, `http://www.oekombi.at/downloads/Verladehinweise%20Rola.pdf`, Abrufdatum: 15.12.2006.

[Qui] Ling Qui, Wen-Jing Hsu, Shell-Ying Huang und Han Wang, 2002: *„Scheduling and routing algorithms for AGVs: a survey“*, International Journal of Production Research, Ausgabe 40/3.

[Rus] Stuart Russell, Peter Norvig, 2003: *„Artificial Intelligence - A Modern Approach“*, Second Edition, Prentice Hall.

[SCHIG-a] SCHIG mbH, November 2005: *„Presseerklärung: Feierliche Eröffnung der neuen grenzüberschreitenden Rollenden Landstraße (ROLA) in Regensburg“*, `http://www.schig.com/pdf/Pressetext_051102_RoLa_Regensburg.pdf`, Abrufdatum: 15.12.2006.

[SCHIG-b] SCHIG mbH, August 2005: *„Imagebroschüre: Rollende Landstraße Regensburg - Ein PPP-Projekt der SCHIG mbH“*,
`http://www.schig.com/pdf/Imagebroschuere_SCHIG.pdf`,
Abrufdatum: 15.12.2006.

[Ses] Jörg Sesterhenn, 2006: *„3D-Visualisierung eines Systems zur Leitung von Speditionshöfen mit autonomen Fahrzeugen“*, Diplomarbeit, Universität Koblenz-Landau, Daimler-Chrysler AG Stuttgart-Untertürkheim.

[Tra] TRAFICO Verkehrsplanung, 2000: *„Verkehrs- und Umweltpolitische Bedeutung der Ro-La für Österreich“*, Studie im Auftrag von ÖKOMBI Ges.m.b.H & Co.KG, Wien,
`http://www.oekombi.at/downloads/News/trafico_rola_studie.pdf.`,
Abrufdatum: 15.12.2006.

[VDI] VDI-Gesellschaft Fördertechnik Materialfluss Logistik, 2005: *„VDI-Richtlinie 2510: Fahrerlose Transportsysteme“*, Beuth Verlag, Berlin.

[Ver] Alexander Verbraek, Yvo Saanen, Edwin Valentin, 2000: *„Designing Effective Terminals and their Control Systems for the Underground Logistic System Schiphol“*, 2nd International Symposium on Underground Freight Transportation by Capsule Pipelines and Other Tube / Tunnel Systems,
`http://www.betade.tudelft.nl/publications/VerbraeckSaanenValentin_ISUFT2000.pdf`,
Abrufdatum: 12.01.2007.

[Vis] Iris F. A. Vis und Ismael Harika, 2004:*“Comparison of vehicle types at an automated container terminal“*, [Kim-a], Seiten 51 bis 77.

[Woj-a] Philipp Wojke, 2006 (bisher unveröffentlicht): *„EZauto - Softwarearchitektur für Anwendungen des assistierten oder autonomen Fahrens“*, Wissenschaftlicher Beitrag, Universität Koblenz-Landau.

[Woj-b] Philipp Wojke, 2006 (bisher unveröffentlicht): *„Einsatz von autonomen Gespannen in der Rollenden Landstraße“*, Dissertation, Universität Koblenz-Landau.

[Zöb] Dieter Zöbel, 2005: *„Konzeptpapier EZrola, Stand 2005“*, Vortragsfolien, Universität Koblenz-Landau.