Carl von Ossietzky
Universität Oldenburg

Department für Informatik
Abteilung Wirtschaftsinformatik

Projektgruppe

Wintersemester 2008/2009 - Sommersemester 2009

ENDBERICHT

Gruppe:
Bernd Ahlering
Christoph Dölger
Gereon Fössing
Uwe Krisch
Fabrizio Nencini
Jörg Oelerink
Daniel Probst
Stephan Schulten

Betreuer:
Prof. Dr. Axel Hahn
apl. Prof. Dr. Jürgen Sauer

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Im Verlauf des Wintersemesters 2008/2009 und des Sommersemesters 2009 hat die Projektgruppe „Virtual Port 2“ an der Carl von Ossietzky Universität Oldenburg ein System entwickelt, mit dessen Hilfe die Planung sowie der Betrieb eines Containerterminals simuliert, gesteuert und in 3D visualisiert werden können.

Zu Beginn dieses Dokuments soll erläutert werden, wie die Projektidee entstanden ist, bevor ein Überblick über den Umfang des Gesamtsystems gegeben wird. Es folgen die Ziele und Erfolgskriterien des Projekts sowie eine Übersicht über die nachfolgenden Kapitel.

1.1 Motivation

Die Motivation zur Projektgruppe „Virtual Port II“ ergab sich aus dem Projekt JadeWeserPort, das im Jahr 2002 vom Land Niedersachsen und der Hansestadt Bremen auf den Weg gebracht wurde. Das unter der Leitung der JadeWeserPort Realisierungs GmbH & Co. KG stehende Projekt zählt zu einem der größten norddeutschen Infrastrukturvorhaben der vergangenen 50 Jahre [2]. Der zukünftige direkt am Jadebusen liegende Tiefwasserhafen in Wilhelmshaven soll neuer Dreh- und Angelpunkt für Großcontainerschiffe werden und den Anforderungen an den ständig wachsenden Containerverkehr nachkommen.

Der in den vergangenen Jahren gestiegene Containerumschlag in deutschen Seehäfen hat die Betreiber der Containerterminals vor neue Herausforderungen gestellt. Im Zuge der Wirtschaftskrise hat jedoch auch der Gütertransport auf dem Seeweg einen Rückgang hinnehmen müssen. Auf Grund der aktuellen Wirtschaftslage ist es deshalb umso wichtiger auf eine kontinuierliche Produktivität der hafenlogistischen Abläufe zu achten und die Prozesse an sich wandelnde Rahmenbedingungen anzupassen. Es ist ebenso die Effizienz in Bezug auf den Ressourcenverbrauch zu überprüfen, um dem wachsenden Kostendruck durch die momentan stagnierende Auftragslage entgegenzuwirken und bei steigenden Auftragszahlen von hoher Reaktionsfähigkeit und Effizienz zu profitieren.

Die Planung der hafenlogistischen Prozesse umfasst zahlreiche Probleme, für die es optimale Lösungen zu finden gilt. Seeseitig spielt die Stauplanung auf dem Schiff eine bedeutende Rolle, die sich mit der Anordnung der Container auf dem Schiff befasst [8]. Dieses Planungsproblem beeinflusst in erheblichem Maße die Transportkapazität, die Hafenliegezeit und den Energieverbrauch auf den Seestrecken. Ebenfalls seeseitig anzusiedeln ist die Liegeplatzplanung der ankommenden und auslaufenden Schiffe.

Auf der Landseite gibt es verschiedene Planungsprobleme, die teilweise voneinander abhängig sind. Große Bedeutung ist der Zuteilung der Containerbrücken und dem Verplanen der Transportmittel, die die zu ladenden und zu löschenden Container von den Kränen auf den Yard und in die Rückrichtung transportieren, beizumessen. Bei der Containerstellplatzplanung sind die Art des Containers sowie die Dauer der Einlagerung von Bedeutung. Das Ziel besteht wie auch bei der Stauplanung auf dem Schiff in der Vermeidung von Umstauvorgängen. Im Hinblick auf den Weitertransport eines Containers vom Containerterminal ins Landesinnere sind die LKW-, Bahn- und Schiffsabfertigung zu betrachten.

Mit Hilfe eines Containerterminal-Modells können die verschiedenen logistischen Prozesse simuliert und eine bestehende Steuerungssoftware auf eine korrekte Ausführungsweise innerhalb des Modells getestet werden. Fehlplanungen lassen sich so frühzeitig erkennen und ohne Risiko und großen Aufwand feststellen. Neben der Zeit- und Kostenersparnis durch die Erprobung der Planung in der virtuellen Realität ermöglicht eine Simulation einen Vergleich unterschiedlicher Handlungsalternativen.

Die 3D-Visualisierung tritt in vielen Bereichen wie Architektur, Bauwesen und beim Erstellen für Konstruktionsunterlagen im beispielsweise Maschinen- oder Anlagenbau immer mehr in Erscheinung. Durch die mögliche Auswahl vielfacher Perspektiven und die besondere Anschaulichkeit kann sich der Betrachter eine bessere Vorstellung der modellierten Objekte machen, wie sie zu einem späteren Zeitpunkt in der Realität aussehen und im Zusammenspiel funktionieren werden. Der Einsatz der 3D-Technologie kann somit auch die logistischen Abläufe in einem Containerterminal erfahrbar machen und diese in einer virtuellen Realität gezielt veranschaulichen.

Das System Containerterminal stellt ein komplexes Abhängigkeitsgefüge dar, dessen genaue Organistaion und Koordination in Anbetracht des globalisierten Welthandels als kritischer Wettbewerbsfaktor anzusehen ist. Um unterschiedliche Szenarien und deren Auswirkungen auf Durchsatz und Produktivität zu betrachten, werden diese Szenarien mit variabler Anzahl an einlaufenden Schiffen und unterschiedlich zugeteilten Containerbrücken erstellt, simuliert und miteinander verglichen. Auf der Basis der Simulationen können beispielsweise Erkenntnisse über die maximale Containerumschlagskapazität, die Abfertigungsdauer oder Lagerzeiten gewonnen werden.

1.2 Umfang des Systems

Auf der Basis des Konzepts der Projektgruppe „Virtual Port I“, die während des Wintersemesters 2007/2008 und des Sommersemesters 2008 stattgefunden hat, macht sich es „Virtual Port II“ zur Aufgabe, die Funktionalität der Software um eine Planungs- und Steuerungskomponente zu erweitern und die Modellierung weiterer Objekte des Hinterlands wie beispielsweise Gebäude oder Transportmittel voranzutreiben und zu visualisieren.

Die zu realisierende Software übernimmt die Planung und Steuerung der einzelnen Arbeitsschritte auf dem Containerterminal, die über eine Schnittstelle mit der Simulationssoftware, die die Prozesse ausführt, kommuniziert. Mit Hilfe der Visualisierungskomponente werden die mit der 3D-Grafik-Software Blender erstellten Objekte, zu denen der Containerterminal selbst sowie Gebäude und Transportmittel zählen, dargestellt und die auf einem Containerterminal stattfindenden logistischen Prozesse simuliert. Im Gegensatz zu bestehenden Simulationssystemen stellt „Virtual Port II“ die logistischen Abläufe in einer dynamischen Umgebung dar, die durch eine Simulationssoftware gesteuert wird.

Ein Teil des Gesamtsystems wird das operative Planungstool „virtual Terminal Operating System (vTOS)“ sein. Aufgaben dieser Planungskomponente sind:

- Erstellen von Szenarien
- Liegeplatzplanung am Kai
- Containerbrückeneinsatzplanung

- Umsetzung der Container-Stellplatzverwaltung
- Transportmitteleinsatzplanung
- Konfiguration der Visualisierungskomponente
- Belegungsübersicht des Containerterminals
- Statistikomponente

Die genannten Planungsprobleme sollen beschrieben, Lösungkonzepte vorgestellt und erläutert sowie auf unterschiedliche Szenarien angewandt werden.

Der Containerterminal soll um Flächen für Stromanschlüsse, Gefahrgut- und Leercontainer sowie für Zug- und LKW-Anbindung erweitert werden. Das Modell der Simulation muss entsprechend um Züge und LKWs ergänzt und die benötigten Objekte für die Visualisierung modelliert werden.

1.3 Ziele und Erfolgskriterien des Projekts

Ziel ist die Erweiterung der Ergebnisse aus der Projektgruppe „Virtual Port I“, auf deren Grundlage die drei Komponenten Planung (vTOS), Simulation (SIM) und Visualisierung (VIS) weiterentwickelt werden sollen. Die Planungskomponente soll auf Basis der notwendigen Angaben in der Lage sein, Auftragslisten zu erstellen und die dafür benötigten Transportmittel zur Verfügung stellen. Zu den Anforderungen der Visualisierungskomponente zählt die weitere Modellierung von Objekten des Hinterlandes. In Bezug auf die Simulation wird eine Erweiterung des Modells um die ergänzten Funktionalitäten der Planungskomponente angestrebt.

Das fertige Modell soll helfen, Optimierungspotentiale innerhalb der logistischen Abläufe aufzuzeigen. Unterschiedliche Konfigurationen eines Szenarios, das heißt exemplarisch eine variable Anzahl an ein- und auslaufenden Schiffen, können betrachtet werden. Darüber hinaus lassen sich die vorhandenenen Containerbrücken zu den anlegenden Schiffen unterschiedlich zuteilen und feststellen, welche Variante wieviel Be- und Entladezeit beansprucht. Durch die Auswahl verschiedener Lösungsverfahren für die hafentypischen Planungsprobleme kann ein Szenario auf Effizienz überpüft und mit anderen Szenarien verglichen werden. In diesem Zusammenhang sind beispielsweise Informationen über den Containerumschlag, Schiffe, Transportmittel und Positionen von Containern relevant. Die 3D-Simulation ermöglicht freien Kameraflug und Objektverfolgung.

1.4 Übersicht

Im nachfolgenden Kapitel wird auf die Vorgehensweise und den Zeitplan des Projekts eingegangen. Der sich anschließende Anforderungsteil befasst sich mit den funktionalen und

nicht-funktionalen Anforderungen, die an das Projekt gestellt wurden. Bevor die Softwarearchitektur im Entwurfsabschnitt beschrieben und erläutert wird, gibt das Kapitel Planung eine Einführung in die für einen Containerterminal typischen Planungsprobleme, die es zu betrachten und zu lösen gilt. Da die Lösung aller auftretenden Planungsprobleme innerhalb des Containerterminalbetriebs zu umfangreich und komplex für das Projekt wäre, beschränkt sich die Projektgruppe „Virtual Port II“ auf eine Auswahl der im Kapitel *Planung* dargestellten Probleme. Abschließend wird eine Zusammenfassung der Ergebnisse der Projektgruppe „Virtual Port II“ gegeben.

2. Projektmanagement

Zum Inhalt dieses Kapitels zählt eine allgemeine Beschreibung der Projektplanung. Es soll auf die Vorgehensweise und Organisation der Projektgruppe sowie auf die Aufgabenteilung und Koordination der Projektphasen eingegangen werden.

2.1 Vorgehensmodell

„Plans are nothing; planning is everything."
(Dwight D. Eisenhower)

Zu Beginn der Projektgruppe „Virtual Port II" wurde kein klassisches Vorgehensmodell wie beispielsweise das Wasserfallmodell[1] ausgewählt. Anstelle der traditionellen Vorgehensmodelle, die häufig als bürokratisch und schwergewichtig angesehen werden, sollte die gewünschte Vorgehensweise ein Höchstmaß an Flexibilität garantieren. Diese Anforderung der Flexibilität findet sich verstärkt innerhalb des agilen Projektmanagements wieder [4]. Ohne eine konkrete Form der agilen Methoden, wie beispielsweise *Scrum*, anzuwenden, orientiert sich die weitere Vorgehensweise der Projektgruppe an einigen Grundsätzen der sogenannten Agilität. Mit dem Vorwissen und den Erfahrungen der vorangegangenen Projektgruppe „Virtual Port I" sah sich die Projektgruppe mit kontinuierlichen Änderungen konfrontiert, die keinesfalls das Festhalten an starren Planvorgaben als sinnvoll erscheinen ließ. So wurde zu Anfang des Projekts kein umfangreicher, detaillierter Plan entworfen, sondern das Augenmerk darauf gelegt, in kurzen zeitlichen Abschnitten regelmäßige Planungssitzungen abzuhalten, um die vorrangigen Ziele sowie deren Fortschritt zu besprechen [6].

Rückblickend lässt sich eine Einteilung des Projekts in Phasen nennen, in denen spezielle Aufgaben bearbeitet werden sollten. Innerhalb der ersten Gruppensitzungen haben sich alle Gruppenmitglieder für einen der drei Kernbereiche Simulation, Visualisierung oder Steuerung/Planung entschieden. So konnte jede Teilgruppe zu einem späteren Zeitpunkt in ihrer Kleingruppe die Anforderungen an den jeweiligen Kernbereich formulieren, nachdem sich die Teilteams mit den Ergebnissen der vorangegangenen Projektgruppe auseinandergesetzt hatten.

2.2 Projektphasen

Als Einführung in die Projektgruppe diente die **Seminarphase**, in der jeder Teilnehmer der Projektgruppe zunächst über ein bestimmtes Thema referierte und im Anschluss daran eine schriftliche Ausarbeitung zum betreffenden Thema verfasst hat. Inhaltlich beschäftigten sich die Vorträge mit Themen aus den drei Hauptbereichen Simulation, Visualisierung und Planung/Steuerung. Der Fokus auf die drei genannten Kernbereiche ist innerhalb der Projektgruppe „Virtual Port I" entstanden und hat sich als bewährte Herangehensweise an die Thematik erwiesen. Auf die konkreten Themen, die sich im speziellen mit Planungsproblemen innerhalb der logistischen Abläufe auf einem Containerterminal, Simulationssoftwares und der Modellierung von 3D-Objekten befassen, wird im Anhang verwiesen.

[1] lineares Vorgehensmodell in der Softwareentwicklung

Auf die Seminarphase folgte die **Einarbeitungsphase** in die Ergebnisse der Vorgängergruppe und die Sichtung aller wichtigen Daten wie Quellcode und im Verlauf der Projektgruppe „Virtual Port I“ entstandene Dokumentation. Das Repository der alten Projektgruppe enthielt hierbei die meisten Informationen, auf deren Grundlage ein Verständnis für das bisher Erreichte hergestellt werden sollte. Zum besseren Verständnis sollte ebenfalls eine Präsentation und Live-Demo einiger Teilnehmer der Vorgängergruppe beitragen. Des Weiteren erklärten sich einige Teilnehmer der Projektgruppe „Virtual Port I“ dazu bereit, bei Fragen und Problemen zur Verfügung zu stehen.

Nachdem sich alle Teilnehmer für einen Kernbereich entschieden und sich die Teilgruppen mit den bisherigen Ergebnissen vertraut gemacht hatten, wurden die Anforderungen an das Projekt „Virtual Port II“ von der Gruppe spezifiziert(**Anforderungsphase**). Hieraus ließen sich Teilanforderungen für die Kleingruppen Simulation, Visualisierung und Steuerung/Planung ableiten. Bevor die einzelnen Teams ihre Arbeit aufnehmen konnten, mussten die Schnittstellen zwischen den Teilsystemen vereinbart werden, über die später ein Datenaustausch stattfinden kann. Anschließend nahmen die Teams ihre Arbeit auf und erstellten sich zunächst eine ToDo-Liste, die sequentiell abgearbeitet werden sollte.

Innerhalb der **Entwurfs- und Implementierungsphase** wurde eine neue Architektur für die Planungs- und Steuerungssoftware entworfen und anschließend mit der Implementierung begonnen.

Zum Abschluss des Projekts war die **Endphase** vom Erstellen der Dokumentation sowie von der Fehlerbehebung und vom Testen der Planungs- und Steuerungskomponente geprägt. Innerhalb der vorherigen Phasen wurden bereits einige Dinge dokumentiert, die jedoch noch präzisiert und erklärt werden mussten. Ein nicht unerheblicher Teil der Zeit wurde mit dem Testen der Planungs- und Steuerungssoftware verbracht, um die Kommunikation und Interaktion mit den anderen beiden Teilsystemen Simulation und Visualisierung zu überprüfen. Auch das Zusammenspiel der Simulations- und Visualisierungskomponente wurde auf den Prüfstand gestellt, um zu testen, ob die in der Simulation erstellten Objekte korrekt als 3D-Objekte dargestellt werden.

2.3 Organisation und Aufgabenverteilung

Zur Organisation und für die Abwicklung des Projekts sind verschiedene Aufgaben und Zuständigkeiten verteilt worden. Jedes Gruppenmitglied bekam eine Tätigkeit zugeteilt, um die es sich bis zum Projektende zu kümmern hatte. Folgende Beauftragte sind zu Projektbeginn festgelegt worden:

- **SVN-Beauftragter**
  Zur Verwaltung aller im Laufe des Projekts entstehenden Dokumente sowie des

Quellcodes wurde das Versionsverwaltungssystem *Subversion* verwendet. Die Versionierung erfolgt in einem Projektarchiv(*Repository*), auf das alle Projektteilnehmer Zugriff haben und durch ein Gruppenmitglied administriert und gewartet werden muss. Installation, Benutzerverwaltung und Wartung der Software sind die Aufgaben des SVN-Beauftragten.

- **Flyspray-Beauftragter**
  Das Bugtracking-System *Flyspray* dient der Erfassung und Dokumentation von Programmfehlern, für die sich neben Zuständigkeiten und Prioritäten weitere Eigenschaften festlegen lassen. Ebenso zählen eine Beschreibung des Fehlers sowie Status- und Fortschrittsmeldungen zum Umfang der webbasierten Software. Innerhalb der Projektgruppe wurde die Software vor allem als zusätzliches Ticketsystem verwendet, in dem zu erledigende Aufgaben und Probleme hinterlegt werden konnten. Der Flyspray-Beauftragte übernimmt die Installation, Benutzerverwaltung und sonstige Wartung des Bugtracking-Systems.

- **Wiki-Beauftragter**
  Das *Wiki* zur Projektgruppe „Virtual Port II“ stellt neben der offiziellen Webseite Informationen rund um das Projekt und die Projektgruppe dar. Die während der Seminarphase entstandenen Präsentationen und Ausarbeitungen der einzelnen Projektgruppenmitglieder geben einen Einblick in die Themen Containerlogistik, Simulationssysteme und 3D-Objektmodellierung.

- **Qualitäts-Beauftragte**
  Die Qualitäts-Beauftragten überwachen die ordnungsgemäße Anfertigung der Dokumentation innerhalb der drei Teilgruppen und übernehmen den Testbetrieb der entwickelten Planungs- und Steuerungssoftware. Alle erstellten Dokumente sind auf Vollständigkeit und Richtigkeit zu überprüfen. Hierzu zählt ebenso das während der Implementierung erstellte JavaDoc.

- **Dokumentations-Beauftragte**
  Für die Erstellung der innerhalb des Projektzeitraums entstehenden Dokumente ist es notwendig, dass auf das regelmäßige Festhalten von Teilergebnissen und das Vermerken von Notizen zu wichtigen Beschlüssen geachtet wird. Mit Hilfe einheitlicher Vorlagen wurde ein formaler Standard für die verschiedenen Dokumenttypen festgelegt, für deren Einhaltung die Dokumentations-Beauftragten Sorge zu tragen haben.

- **Web-Beauftragter**
  Die offizielle Homepage der Projektgruppe ist durch den Web-Beauftragten gestaltet worden und bietet zusammen mit dem *Wiki* zahlreiche Informationen zum Projekt und zur Projektgruppe selbst.

- **Administrator für Hard- und Software**
  Dem Administrator der Projektgruppe obliegt für die Dauer des Projekts die Administration und Wartung der vorhandenen PC-Systeme. Der für die Projektgruppe zur Verfügung gestellte Raum umfasst mehrere PC-Einzelarbeitsplätze, für deren Betreuung der Administrator verantwortlich ist. Exemplarische Aufgaben sind die Wartung und Pflege der verwendeten Hard- und Software.

2.4 Fazit

Am Schluss dieser Betrachtung des praktizierten Projektmanagements soll eine Einschätzung und Beurteilung der verwendeten Vorgehensweisen, Hilfsmittel und Verfahren vorgenommen werden. Die anfängliche Entscheidung gegen ein klassisches, statisches Vorgehensmodell erwies sich größtenteils als förderlich für das Erreichen der zu Beginn der Projektgruppe formulierten Ziele. Innerhalb der drei Teilgruppen wurde sich durchgehend ausgetauscht, so dass kleinere Planänderungen ohne wesentlichen Zeitverzug umgesetzt und berücksichtigt werden konnten. Schwierigkeitem traten lediglich dann auf, wenn eine Teilgruppe kurzzeitig weit hinter den Vorgaben lag, da dies mögliche Tests verhinderte, die für die fehlerfreie Ausführung der gesamten Simulation nötig gewesen wären.

Die Zuteilung von Aufgaben erschien als sinnvoll, da so feste Zuständigkeiten bestanden und sich jedes Gruppenmitglied auf seine Aufgaben innerhalb seiner Teilgruppe sowie innerhalb der gesamten Projektgruppe konzentrieren konnte. In Bezug auf die Dokumentation ist festzuhalten, dass es wichtig ist, sich kontinuierlich darum zu bemühen, Zwischenergebnisse und getroffene Vereinbarungen schriftlich zu erfassen, um diese zu einem späteren Zeitpunkt nachschlagen zu können.

3. Anforderungen

Diese Anforderungsdefinition ist im Rahmen der Projektgruppe „Virtual Port 2“ im Wintersemester 2008/09 an der Carl von Ossietzky Universität Oldenburg entstanden. Zu den Zielen dieser zwei Semester umfassenden Projektgruppe zählen die Entwicklung eines operativen Planungstools für einen Containerhafen sowie einer Steuerungssoftware, die die logistischen Abläufe auf einem Containerterminal simulieren. Die einzelnen auf dem Terminal ablaufenden Arbeitsprozesse werden mit Hilfe einer Simulationssoftware nachgebildet und die benötigten Objekte durch die Einbindung einer 3D-Engine visualisiert.
Die funktionalen Anforderungen stehen hier im Vordergrund und die nicht-funktionalen Anforderungen werden nur kurz angesprochen. Diese werden in späteren, während der weiteren Entwicklung entstehenden Dokumenten, ausführlich behandelt.
Die Anforderungsdefinition stellt einen Entwicklungsprozess dar, auf dem die komplette nachfolgende Entwicklung aufbaut. Das daraus hervorgehende Anforderungsdokument ist Grundlage für das Softwaredesign des entstehenden Planungstools und der Steuerungssoftware sowie des Simulationsmodells und der 3D-Visualisierungs-Komponente.

3.1 Funktionale Anforderungen

Virtual Port ist eine 3D-Simulierung des Jade Weser Port und besteht aus den Komponenten Planungs- und Steuerungssoftware, Simulation und Visualisierung. Die Planungs- und Steuerungssoftware wird durch das vTOS realisiert, die Simulation des Terminals erfolgt durch die Simulationssoftware *eM-Plant* und die Visualisierung realisiert die 3D Umsetzung der Simulation auf Basis der 3D-Engine *Xith3D*.

3.1.1 Planungs- und Steuerungssoftware

Das vTOS (virtual Terminal Operation System) ist die Steuerungssoftware von Virtual Port. Zur Steuerung des Terminals gehören die Yard-, Liegeplatz-, und Containerbrückeneinsatzplanung. Für die Yard-Planung erstellt vTOS Event- und Transportaufträge. Ein Event ist der Vorgang von einem Startort zu einem Zielort, der aus mehreren Tranportaufträgen bestehen kann. Ein Transportauftrag ist der Vorgang, der von genau einem Transportfahrzeug bearbeitet wird.
Mit einer Testumgebung, die in vTOS integriert ist, werden Simulationen erstellt. Dazu gehört das Erstellen von Schiffen mit Auftragslisten und das Zusammenfassen von Schiffen in Ship schedules. Aus den Auftragslisten werden dann die Event- und Transportaufträge erstellt.
Das Löschen der Container von einem Schiff erfolgt nach einem eigenen Algorithmus, der drei Faktoren berücksichtigt. Zuerst wird nach freien Containerplätzen abhängig vom Typ des Containers gesucht. Danach werden nur die Plätze berücksichtigt, unter denen sich kein Container mit einer früheren Auslieferugszeit befindet. Zur letzten Auswahl gehört die Berücksichtigung des kürzesten Weges vom Schiff zum Yard-Platz.
Nach dem ein Stellplatz für einen Container gefunden wurde, werden ein Event-Auftrag und die Transportaufträge erstellt. Die Transportaufträge werden dann an eM-Plant gesendet.

3.1.1.1 Benutzeroberfläche vTOS

Die Benutzeroberfläche enthält Übersichten über Yard und Aufträge, ermöglicht das Erstellen von Simulationskomponenten und steuert die Simulation eM-Plant. Die folgende Auflistung gibt einen überblick über die Funktionalität von vTOS.

- Edit: Soll keine komplette Simulation gestartet werden, können hier einzelne Transportaufträge erstellt und am eM-Plant gesendet werden.
- Create: Die einzelnen Komponenten Bay, Ship, Ship schedule und Simulation können hier erstellt werden.
- Simulation: Eine vorher erstellte Simulation, die in der Datenbank gespeichert wurde, kann hier ausgewählt werden.

- Database: Datenbankverbindungen können angelegt und geändert werden. Die Speicherung erfolgt in der XML-Datei vTosConfig.xml
- View: Yard- und Auftragsansichten, die sortierbar nach freien oder belegeten Plätzen und Event- oder Transportaufträgen sind.
- emPlant: Steuert eM-Plant, indem Start, Stop und Refresh möglich sind. Außerdem kann hier das Abarbeiten der Aufträge gestartet werden.
- 3D: Einstellungsmöglichkeiten für die Grafikengine. Auflösung, Farbe und Animationsoptionen.
- Development: Zur Weiterentwicklung ist ein Logger in vTOS implementiert, der an jeder Stelle aufgerufen werden kann.
- Statistics: Statistische Auswertung und grafische Darstellung von Menge und Typ von Containern sowie Typ und Menge von Fahrzeugen.
- Help: Benutzerhandbuch und Internetseite von Virtual Port können hier aufgerufen werden.

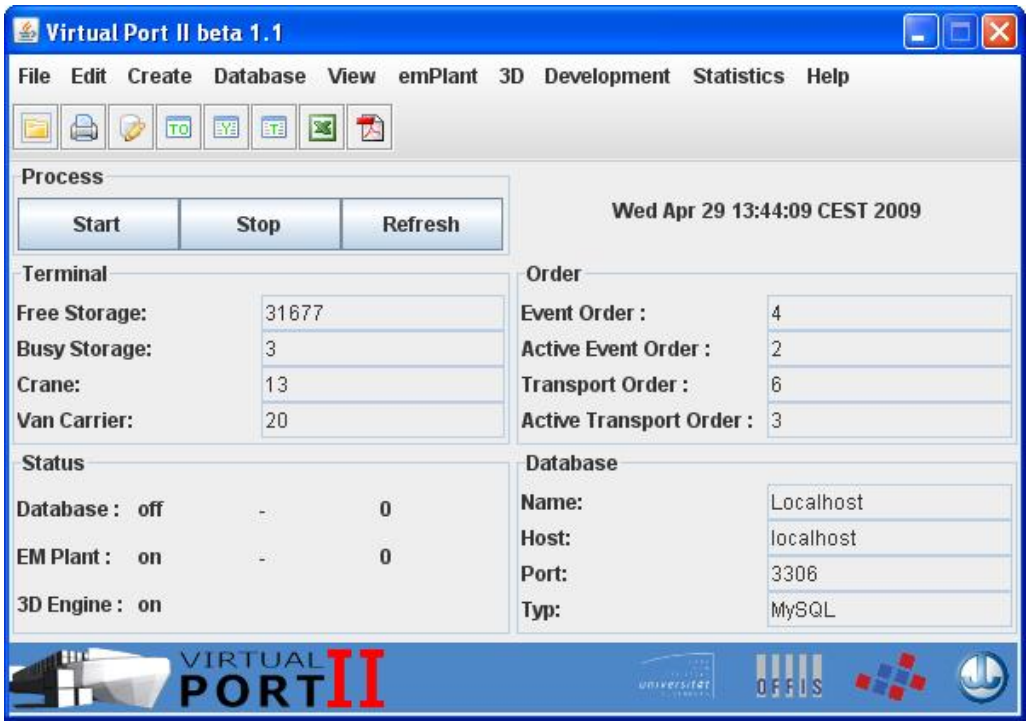


Abbildung 3.1: vTOS Hauptfenster

3.1.2 Simulation

Um die Planungsstrategien, welche von vTOS entwickelt werden, in geeigneter Weise simulieren zu können, kommt das Simulationssystem eM-Plant zum Einsatz. Mit dieser

soll ein Hafenmodell entwickelt werden, welches den Einsatz von Van Carriern, Containerbrücken und anderen logistischen Fördermitteln ermöglicht.
Zusätzlich zur Simulation, muss eM-Plant die stattgefundene Simulation an die Visualisierung weitergeben, um so die Visualisierung zu ermöglichen. Dabei ist es notwendig, die benötigten Werte in geeigneter Weise zu übergeben. Diese Werte unterscheiden sich in ihrer Art grob in statische und dynamische Werte. Statische Werte stellen die Umgebung dar und müssen daher nur einmal erfasst werden. Dynamische Werte erfassen hingegen die Bewegungen auf dem Hafengelände und müssen ständig erfasst werden.
Zur Umsetzung der beiden genannten Punkte wurde das Hafenmodell der letzten Projektgruppe „Virtual Port 1“ als Grundlage verwendet. Die Anforderungsdefinition der Simulation teilt sich in drei Oberpunkte. Die Erweiterung des vorhandenen Hafenmodells, die Kommunikation mit vTOS und die Kommunikation mit der Visualisierung.

Abbildung 3.2: Altes eM-Plant Modell

3.1.2.1 Erweiterungen des Hafenmodells

Um eine korrekte Simulation seitens vTOS durchzuführen, muss das bestehende Hafenmodell erweitert und modifiziert werden. Abbildung 3.2 zeigt das bestehende Modell der vorherigen Projektgruppe. Dabei kann die Struktur bzw. das Aussehen des Hafens übernommen werden. Geplant sind vor allem einige Felderweiterungen, neue Fahrzeuge und eine komplette Neumodellierung der gesamten Steuerung.

Felderweiterungen

Unter diesen Punkt fallen vor allem Erweiterungen, die einen direkten visuellen Effekt haben. D.h. sie verändern sichtbar das Modell. Die folgende Auflistung gibt einen Überblick über die geplanten Felderweiterungen.

- Stellplätze für Strom- und Gefahrgutcontainer:
  Einteilung der Lagerdocks entsprechend der geforderten Lagerzonen.

- Gleise und Güterbahnhof:
  Erstellen eines Schienennetzes mit einem Ein- und Ausgang. Außerdem Erstellen eines Güterbahnhofs zum Abfertigen von Containern.

- Straßennetz:
  Erneuerung des vorhanden Straßennetzes um eine reibungslose Logistik zu ermöglichen und einige Fehler zu beseitigen.

Fahrzeuge

Um den Anforderungen von „Virtual Port 2“ gerecht zu werden, müssen Vehikel wie Containerbrücken, Van Carrier und Containerschiffe erstellt werden. Optional können auch Transtainer, Züge und LKWs implementiert werden.

- Containerbrücken:
  Dienen zum Ent- und Beladen der Schiffe und haben einige Restriktionen. Dürfen sich nur am Kai bewegen und sich nicht gegenseitig überholen. Außerdem muss festgelegt werden, wie viele Stellplätze die Kräne für Container zur Verfügung haben. Also wie viele Container er nacheinander entladen kann, ohne dass ein Container abgeholt wird.

- Van Carrier:
  Dienen zum Transportieren der Container. Müssen Aufträge annehmen können und diese ausführen. Aufträge bestehen aus einem Ziel und einer zu tätigenden Aktion. Van Carrier können sich im gesamten Hafenbereich frei bewegen.

- Containerschiffe:
  Vehikel, die aus Containerstellplätzen bestehen. Sie können sich nur auf den dafür vorgesehenen Wasserstraßen bewegen. Die Ladeliste der Schiffe befindet sich in der Datenbank.

- Transtainer:
  Sind Container-Stapelkräne. Dienen lediglich dem Ent- und Beladen der Züge und können sich nur auf dem Bahnhofsgelände bewegen. Gegenseitiges Überholen ist auch nicht erlaubt.

- Züge:
  Ein Zug besteht aus mehreren Waggons, von denen jeder jeweils eine festgelegte Menge Container fassen kann. Er kann sich nur auf dem Schienennetz fortbewegen. Überholen ist generell verboten.

Steuerungen

Um eine reibungslose Logistik der verschiedenen Fahrzeugtypen zu gewährleisten, werden diese jeweils von einer einheitlichen Steuerung geleitet. Diese Steuerung soll zum einen die Auftragssteuerung und zum anderen die Wegfindung umfassen.

- Auftragssteuerung:
  Annahme der Aufträge durch die Socket Kommunikation. Unterscheidung der Aufträge nach dessen Typ, z.B. Abholauftrag eines Van Carriers oder Entladeauftrag einer Containerbrücke. Danach Verteilung an die zuständigen Steuerung. Diese soll den Auftrag dann direkt an das zuständige Vehikel weitergeben.

- Wegfindung:
  Zur Wegfindung soll ein A Stern Algorithmus verwendet werden. Außerdem bietet eM-Plant Möglichkeiten, um Kollisionen von Fahrzeugen zu verhindern und die Wegfindung effizient zu unterstützen.

Kommunikation

Da eM-Plant als Schnittstelle zwischen vTOS und der Visualisierung dient, sollen hier auch nur diese beiden Schnittstellen betrachtet werden.

- Kommunikation vTOS:
  Die gesamte Kommunikation zu vTOS soll über Sockets geregelt werden. Dabei soll es einen eingehenden und ausgehenden Kanal geben. Über den eingehenden Kanal sollen Aufträge angenommen werden können. Über den ausgehenden Kanal sollen Auftragsbestätigungen verschickt werden.

- Kommunikation Visualisierung:
  Die gesamte Kommunikation mit der Visualisierung soll indirekt über die Datenbank stattfinden. Dabei müssen bei der Initialisierung von eM-Plant alle Elemente des Containerhafens in die Datenbank mit der spezifischen Position geschrieben werden. Damit ist die Visualisierung in der Lage ein vollständiges Modell des Containerhafens ohne Bewegungen zu laden. Um die Bewegungen darstellen zu können, müssen sämtliche in eM-Plant getätigten Bewegungen auch in die Datenbank geschrieben werden.

3.1.3 Visualisierung

Im nachfolgenden Abschnitt werden die funktionalen Anforderungen an die 3D-Visualisierung beschrieben.

- Darstellung auf 3D Beamer: Die Simulation wird als stereoskopes Bild auf einem 3D Beamer wiedergegeben

- Darstellung auf Monitor: Die Simulation kann alternativ auf einem Monitor wiedergegeben werden

- Paralleler Betrieb Monitor/3D Beamer: Die Simulation wird parallel auf Monitor/3DBeamer dargestellt

- Modellierung und Darstellung von Objekten:

  - Gelände: Hafenboden, Straßen, Containerlagerplätze, Schienensysteme
  - Umgebung: Hinterland, Wasser, Himmel, Sonne, Wolken
  - Fahrzeuge: Schiffe, Kräne, Van Carrier, Transtainer, Züge, LKWs, PKWs
  - Gebäude: Bürogebäude, Tankstationen, Aufladestationen, Terminalgebäude
  - Gegenstände: Container (Kühlcontainer, Container mit Stromanschluss), Stromanschlüsse, Lampen

  Die Objekte sollen die tatsächliche Situation der Simulation widerspiegeln. Dabei soll sich das Verhalten der Objekte soweit wie möglich an der Realität bzw. an der Simulation orientieren.

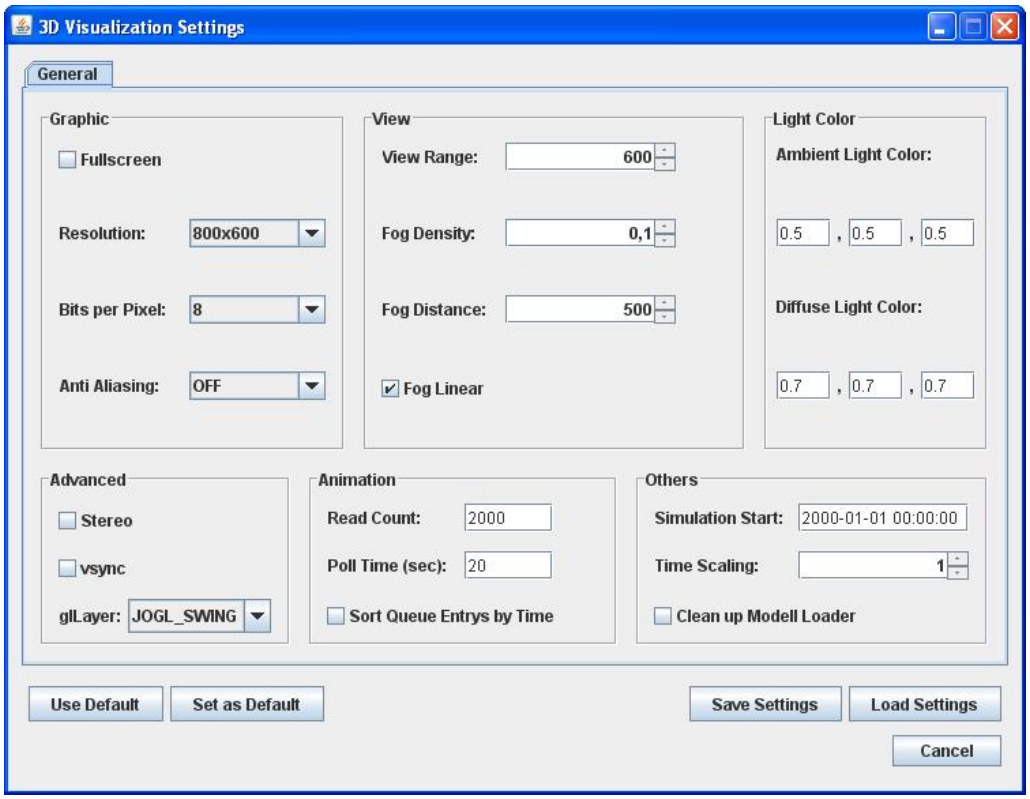


Abbildung 3.3: 3D Konfigurationsfenster

Einstellungsmöglichkeiten Graphic:

Fullscreen	Aktiviert den Vollbilmodus
Resolution	Einstellung der Auflösung des Views
Bits per Pixel	Einstellung der Farbtiefe
Anti Aliasing	Aktivierung von Anti Aliasing

Einstellungsmöglichkeiten View:

View Range	Einstellung
Fog Density	Einstellung der Nebeldichte
Fog Distance	Entfernung vom Nebel zum View
Linear Fog	Nebel nimmt mit Abstand linear zu

Unter *Light Color* können die RGB-Werte der verwendeten Lichtarten *Ambient* und *Diffused* variiert werden. *Advanced* ermöglicht weitere Optionen für die Grafikausgabe. Die optionale stereoskopische Ausgabe und der *vsync-Modus* zur Synchronisation mit Röhrenmonitoren können hier aktiviert werden. Außerdem ist die Auswahl verschiedener *glLayer* möglich.

Animation erlaubt die Konfiguration des Animationsmanagers. *Read Count* gibt die Größe des Animationsqueues an. *Poll Time* gibt die Zeit zwischen den Updates des Animationsmanagers an.

Die *Simulation Start*, das sich in der Gruppe *Others* befindet gibt den Timestamp an, ab dem die Simulationsdaten aus der Datenbank benutzt werden. Mit *TimeScaling* kann der Zeitfaktor zwischen Simulationsdaten und deren Darstellung im View variiert werden. *Clean up Modelloader* sorgt dafür, dass die temporären Modeldateien, die bei der Initialisierung anfallen, gelöscht werden.

3.2 Nichtfunktionale Anforderungen

3.2.1 Bedienbarkeit

Nachdem die Planungssoftware installiert wurde, kann der Benutzer alle Aktionen über die Maus und Tastatur ausführen, die für eine reibungslose Planung benötigt werden. Die Bedienung wird möglichst einfach und überschaubar gehalten, so dass neue Benutzer die Applikation intuitiv bedienen können. Zur besseren Erfassung der Funktionalität wird die Anwendung mit sogenannten Tooltips ausgestattet, die beispielsweise Funktionalitäten von Schaltflächen direkt beschreiben. Benutzer, die mit der Planungssoftware nicht vertraut sind, können das Handbuch zum Erlernen der Bedienung und zum Kennenlernen der Funktionalität nutzen.

3.2.2 Zuverlässigkeit

Das Programm muss gewährleisten, dass Fehleingaben des Benutzers abgefangen werden, damit das System weiterhin korrekt arbeitet. Außerdem muss das System frei von Sicherheitslücken sein, so dass unbefugte Zugriffe verhindert werden. Ausnahmefehler sollen vom System so abgefangen werden, dass dem Benutzer keine unverständlichen Fehlermeldungen angezeigt werden.

3.2.3 Implementierung

Die Implementierung wird auf dem noch zu erstellenden Entwurfsmodell basieren. Aus den verschiedenen Inhalten wird mittels Modelltransformation das Grundgerüst des Quelltextes generiert.

Folgende Werkzeuge werden bei der Implementierung und Realisierung benutzt:

- Eclipse (Entwicklungsumgebung)
- Jude (UML-Modellierung)
- emPlant (Simulationssoftware)
- Blender (3D-Grafik-Software)

Folgende Dienste und Sprachen werden benutzt:

- Java (objektorientierte Programmiersprache)
- MySQL mit phpMyAdmin (Bereitstellung und Verwaltung der relationalen Datenbank)
- SQL (Datenbank-Sprache)
- JDBC (Schnittstelle zwischen SQL und Java)
- SVN (Versionsmanagement)
- JUnit (Testumgebung)

3.2.4 Schnittstelle

Die bereits von der Projektgruppe „Virtual Port“ bestehenden Schnittstellen zwischen den einzelnen Systemen sind überarbeitet und teilweise ersetzt worden. Zwischen den drei Komponenten Planung, Simulation und Visualisierung wird es spezifizierte Schnittstellen für die Kommunikation geben, die in den nachfolgenden Dokumenten genauer beschrieben und verdeutlicht werden.

3.2.5 Betrieb

Während des Projekts stellt die Gruppe den Betrieb des Systems sicher. Das System wird so realisiert, dass es langfristig mit wenig Wartungsaufwand lauffähig und intuitiv konfigurierbar ist.

3.2.6 Rechtliches

Die Projektgruppen „Virtual Port“ und „Virtual Port II“ sind Urheber und somit Eigentümer des Quelltextes.

3.2.7 Systemmodelle

3.2.8 Anwendungsfallmodell

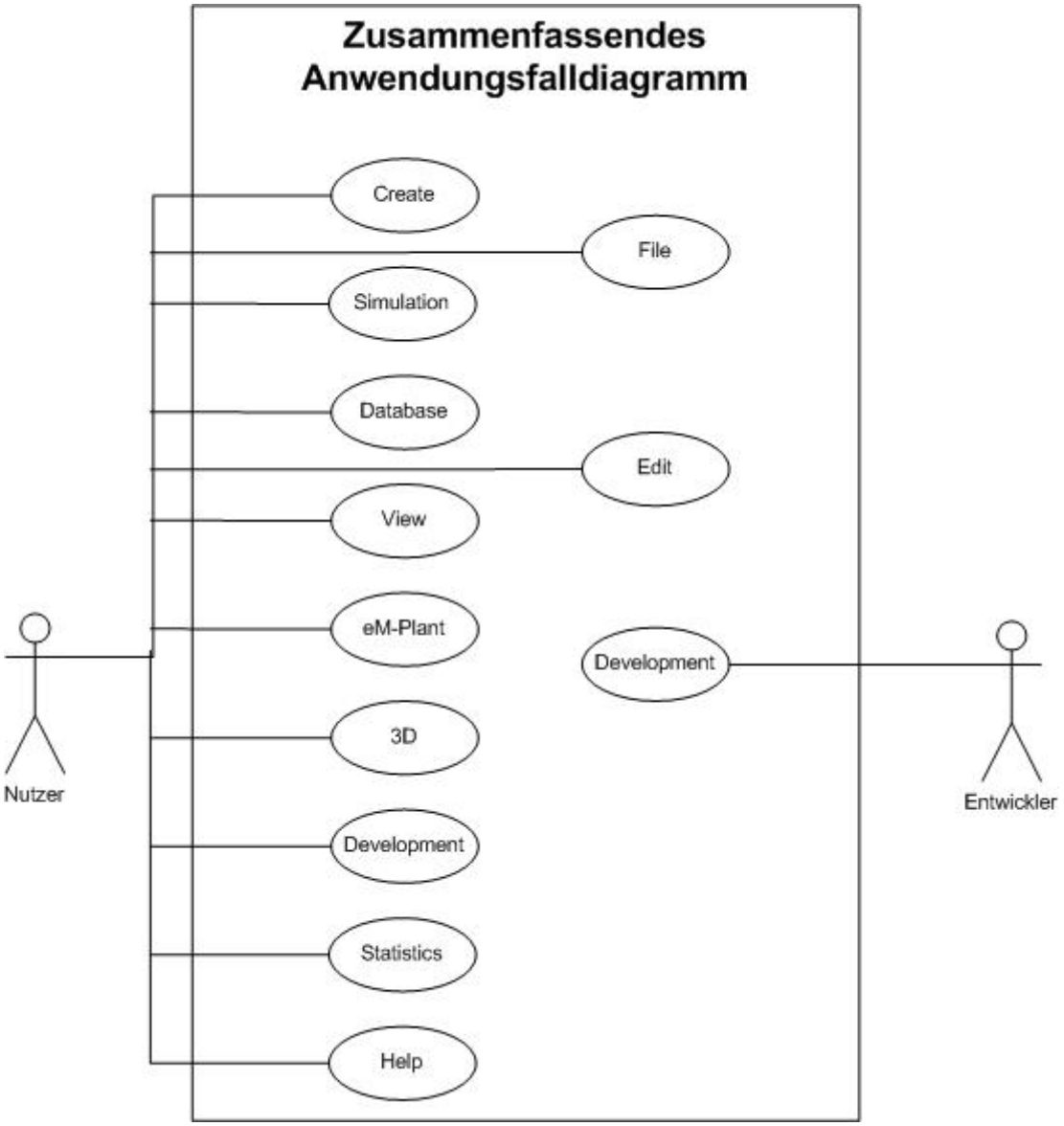


Abbildung 3.4: Zusammenfassendes Anwendungsfalldiagramm vTos

Das zusammenfassende Anwendungsfalldiagramm gibt einen Überblick über die Funktionen des vTOS. Die dargestellten Anwendungsfälle sind die Menüs, die jeweils weitere Anwendungsfälle beinhalten, die weiter unten genauer aufgeführt sind.

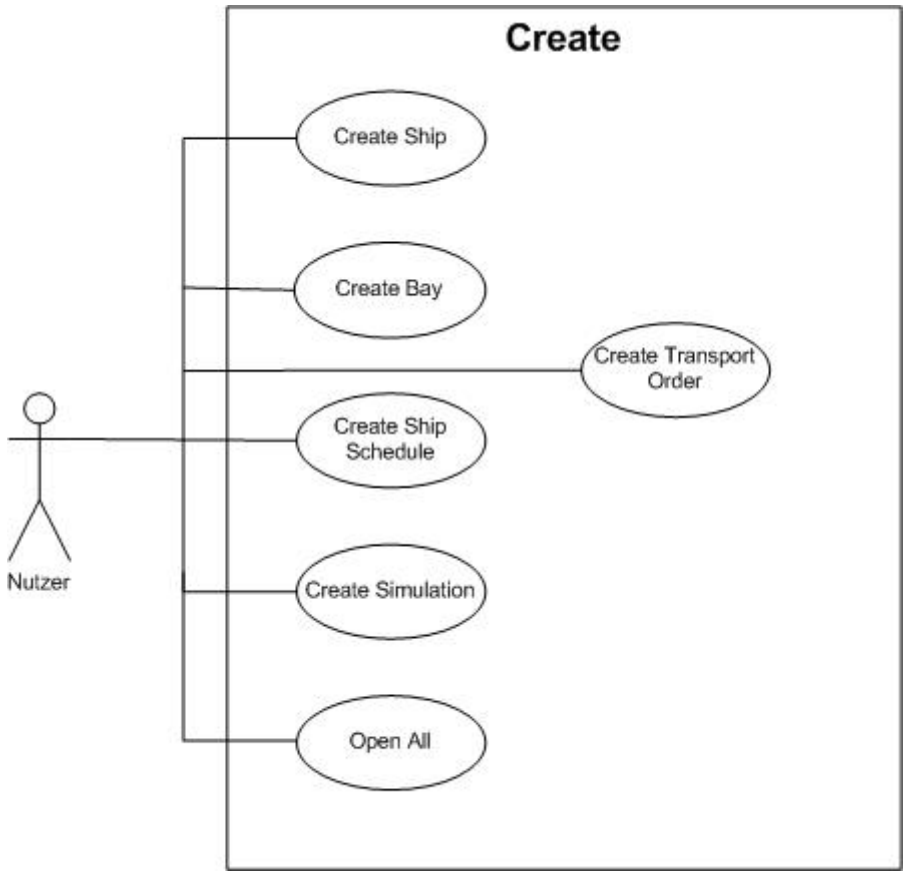


Abbildung 3.5: Anwendungsfälle „Create“

Die Anwendungsfälle „Create“ gehören zu der Testumgebung von „Virtual Port II“. Hier werden die einzelnen Komponenten, die für eine Simulation nötig sind, erzeugt und in der Datenbank gespeichert. Es existieren bestimmte Ahängigkeiten, die zu folgender Reihenfolge zur Erstellung einer Simulation führen:

1. Create Bay: Zuerst wird eine Bay erstellt, die später einem Schiff zugewiesen werden kann.
2. Create Ship: Ein Schiff wird erzeugt, dem Bay's zugewiesen werden können.
3. Create Ship Schedule: Der Schiffplan wird erstellt. Er beinhaltet mehrere Schiffe.
4. Create Simulation: Hier wird eine Schedule ausgewählt und die verschiedenen Planungsalgorithmen eingestellt.

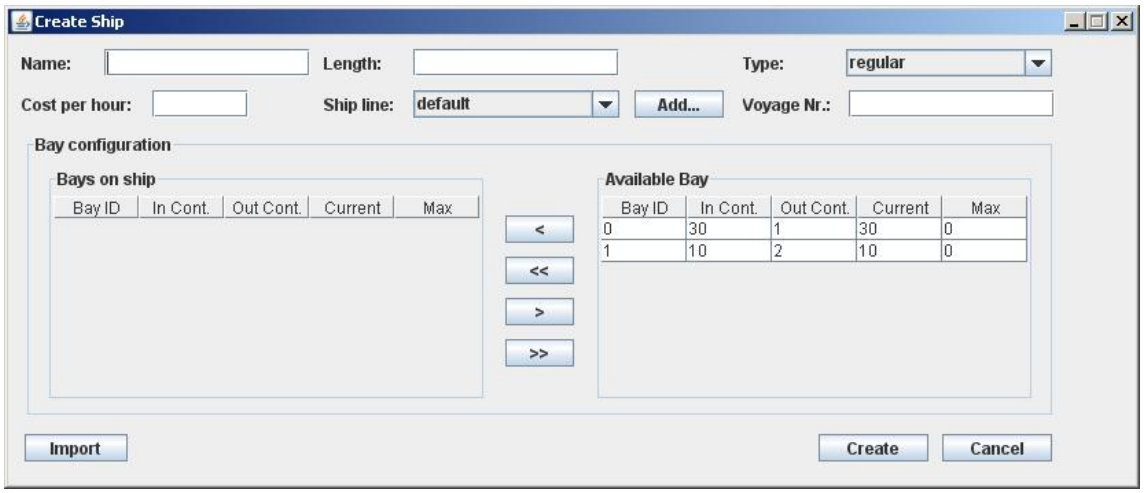


Abbildung 3.6: Create Ship

Anwendungsfallname	Create Ship
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Create Ship“-Funktion. 2. System zeigt Create Ship Formular an. 3. Nutzer füllt Formular aus. 4. System prüft Daten und schreibt sie in die Datenbank.
Anfangsbedingungen	Schiff existiert noch nicht in der Datenbank.
Abschlussbedingungen	Schiff ist in Datenbank gespeichert.
Qualitätsanforderungen	Angaben aller Felder nötig.

Tabelle 3.1: Anwendungsfall Create Ship

Der Anwendungsfall „Create Ship“ erzeugt ein Schiff und speichert dieses in der Datenbank. Dem Schiff können mehrere Bay's zugewiesen werden, die vorher erstellt und in der Datenbank gespeichert wurden. Neben Name, Länge und Routennummer, die alle Pflichtfelder sind, müssen auch die Kosten pro Stunde angegeben werden, die für spätere Berechnungen verwendet werden.

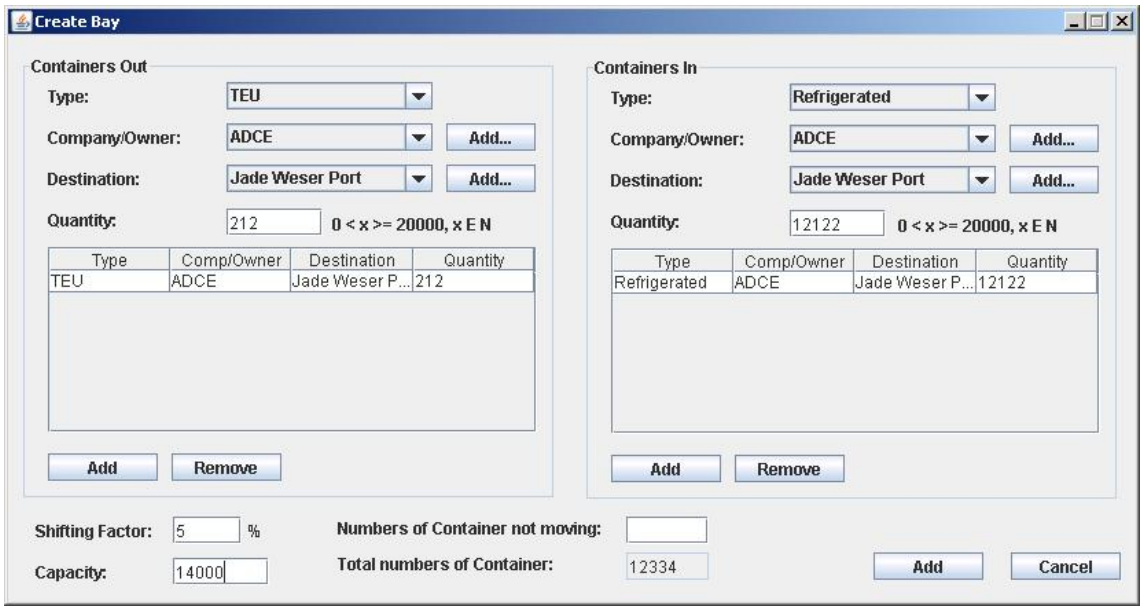


Abbildung 3.7: Create Bay

Anwendungsfallname	Create Bay
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Create Bay"-Funktion. 2. System zeigt Create Bay Formular an. 3. Nutzer füllt Formular aus. 4. System prüft Daten und schreibt sie in die Datenbank.
Anfangsbedingungen	-
Abschlussbedingungen	Bay ist in Datenbank gespeichert.
Qualitätsanforderungen	Angaben aller Felder nötig.

Tabelle 3.2: Anwendungsfall Create Bay

Um ein Schiff mit Bay's auszustatten, müssen diese zuerst unter „Create Bay" angelegt und in der Datenbank gespeichet werden. Aus der Beladung eines Schiffs erzeugt vTOS dann die Auftragsliste der Container. Eine Bay kann in mehreren Schiffen verwendet werden, so dass nicht jedes mal zur Erzeugung eines Schiffs neue Bay's angelegt werden müssen. Die Bay enthält Informationen über Typ der Container, Menge der zu löschenden sowie zu ladenden Container und das Ziel(*Destination*).

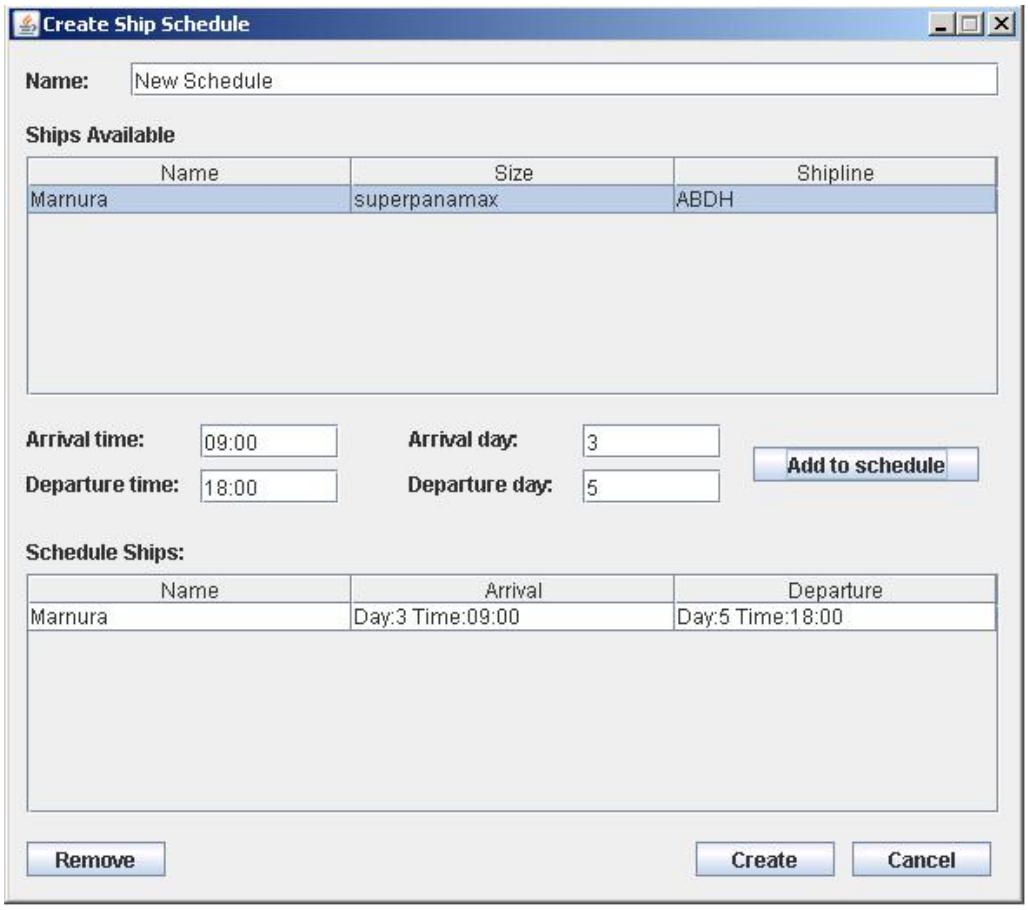


Abbildung 3.8: Create Ship Schedule

Nachdem Schiffe in der Datenbank gespeichert wurden, können Schiffspläne erstellt werden. Ein Schiffsplan enthält mehrere Schiffe mit Ankunftstag und -zeit und Abfahrtstag und -zeit. Jede Simulation enthält genau einen Schiffsplan.

Anwendungsfallname	Create Ship Schedule
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Create Ship Schedule“-Funktion. 2. System zeigt Create Ship Schedule Formular an. 3. Nutzer füllt Formular aus. 4. System prüft Daten und schreibt sie in die Datenbank.
Anfangsbedingungen	Schiff existiert noch nicht in der Datenbank.
Abschlussbedingungen	Schiff ist in Datenbank gespeichert.
Qualitätsanforderungen	Angaben aller Felder nötig.

Tabelle 3.3: Anwendungsfall Create Ship Schedule

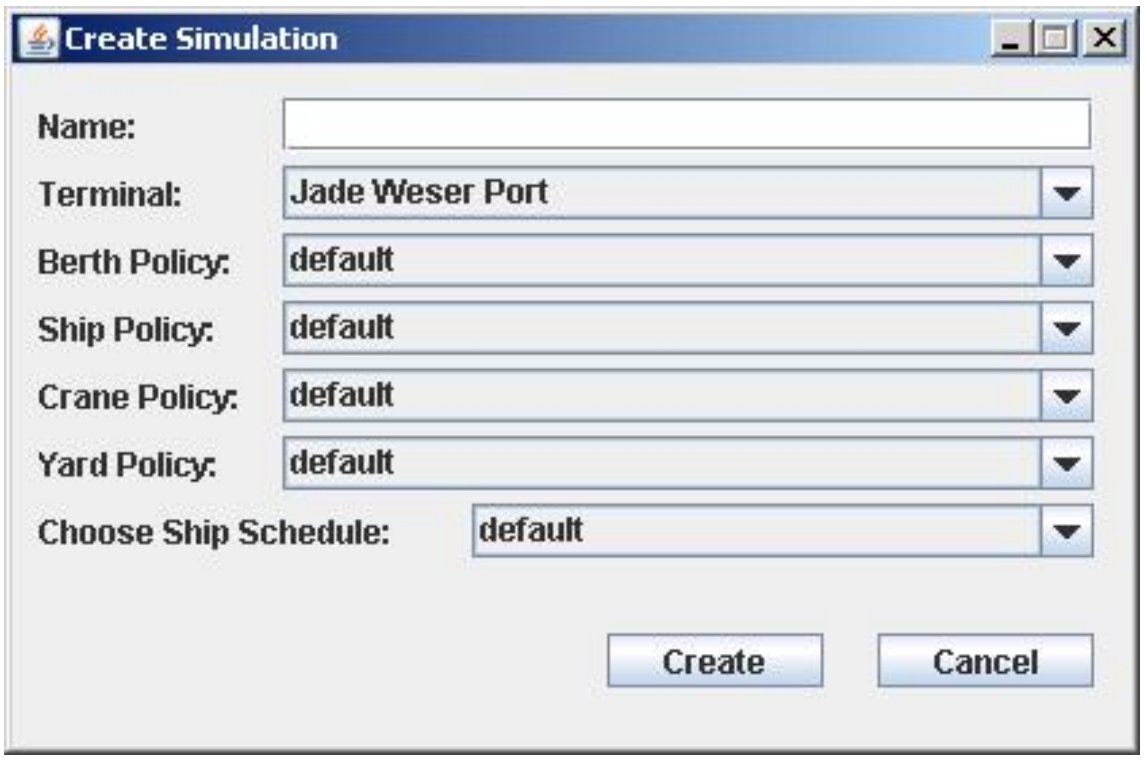


Abbildung 3.9: Create Simulation

Die Simulation zu erzeugen ist der letzte Schritt im Create-Prozess. Neben Namen und Terminal werden die Algorithmen der Planungskomponenten angegeben. Es stehen mehrere Planungsverfahren für Liegeplatzplanung, Containerbrückeneinsatzplanung, Schiffsreihenfolge, Yardplanung zur Auswahl. Außerdem wird der Simulation ein Schiffsplan zugeordnet.

Anwendungsfallname	Create Simulation
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Create Simulation“-Funktion. 2. System zeigt Create Simulation Formular an. 3. Nutzer füllt Formular aus. 4. System prüft Daten und schreibt sie in die Datenbank.
Anfangsbedingungen	Mindestens ein Ship Schedule existiert in der Datenbank.
Abschlussbedingungen	Simulation ist in Datenbank gespeichert.
Qualitätsanforderungen	-

Tabelle 3.4: Anwendungsfall Create Simulation

Abbildung 3.10: Create Transport Order

Um einzelne Containertransporte zu simulieren, können manuell Transportaufträge erstellt werden. Dazu werden Start- und Endkoordinaten, Fahrzeug und Auftragsnummer eingegeben. Nachdem eM-Plant gestartet wurde, kann der Transportauftrag gesendet werden.

Zum Generieren einer Simulation mit allen Komponenten kann mit „Open All“ alle dafür notwendigen Create-Fenster geöffnet werden.

Anwendungsfallname	Create Transport Order
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Create Transport Order“-Funktion. 2. System zeigt Create Transport Order Formular an. 3. Nutzer füllt Formular aus. 4. System prüft Daten und schreibt sie in die Datenbank und sendet Daten an eM-Plant.
Anfangsbedingungen	Container muss sich an Startposition befinden.
Abschlussbedingungen	Transport Order ist in Datenbank gespeichert und an eM-Plant gesendet.
Qualitätsanforderungen	Start- und Zielposition vorhanden.

Tabelle 3.5: Anwendungsfall Create Transport Order

Anwendungsfallname	Open All
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Open All“-Funktion. 2. System öffnet alle Create-Formulare.
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	-

Tabelle 3.6: Anwendungsfall Open All

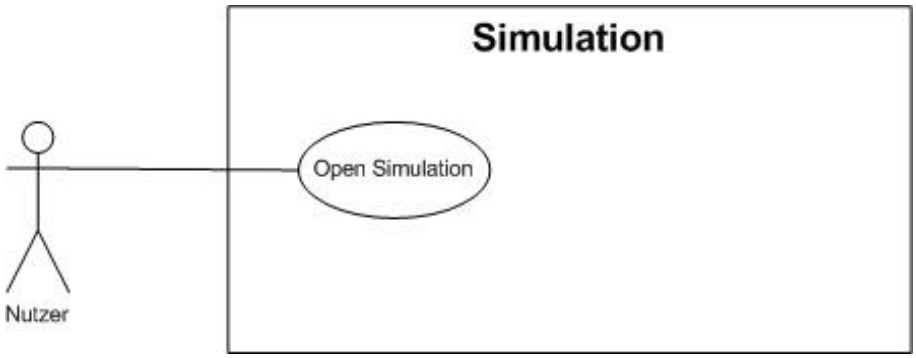


Abbildung 3.11: Open Simulation

Anwendungsfallname	Open Simulation
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Open Simulation“-Funktion. 2. System öffnet Open-Simulation-Formular.
Anfangsbedingungen	-
Abschlussbedingungen	Ausgewählte Simulation wird in Datenbank gespeichert.
Qualitätsanforderungen	Es existiert min. eine Simulation in der Datenbank.

Tabelle 3.7: Anwendungsfall Open Simulation

Um eine Simulation zu starten, wird diese zunächst ausgewählt. Im Simulationsfenster werden alle Informationen zu den in der Datenbank gespeicherten Simulationen angezeigt; Planungsverfahren, Dauer, Anzahl Schiffe, Anzahl umzuschlagender Container etc. Die ausgewählte Simulation wird im Info-Bereich des vTOS-Hauptfensters angezeigt.

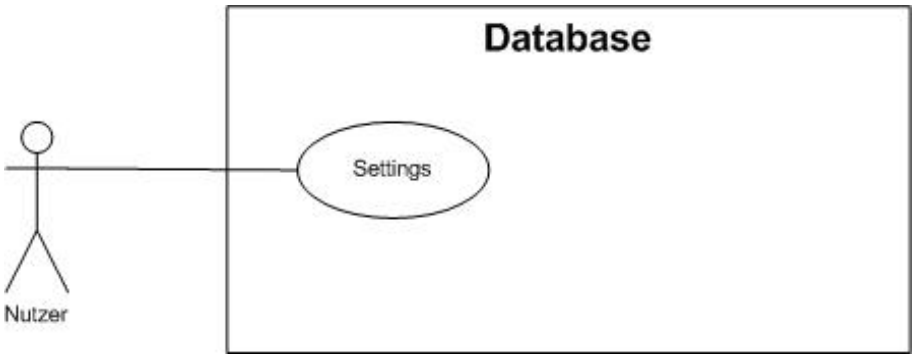


Abbildung 3.12: Database Settings

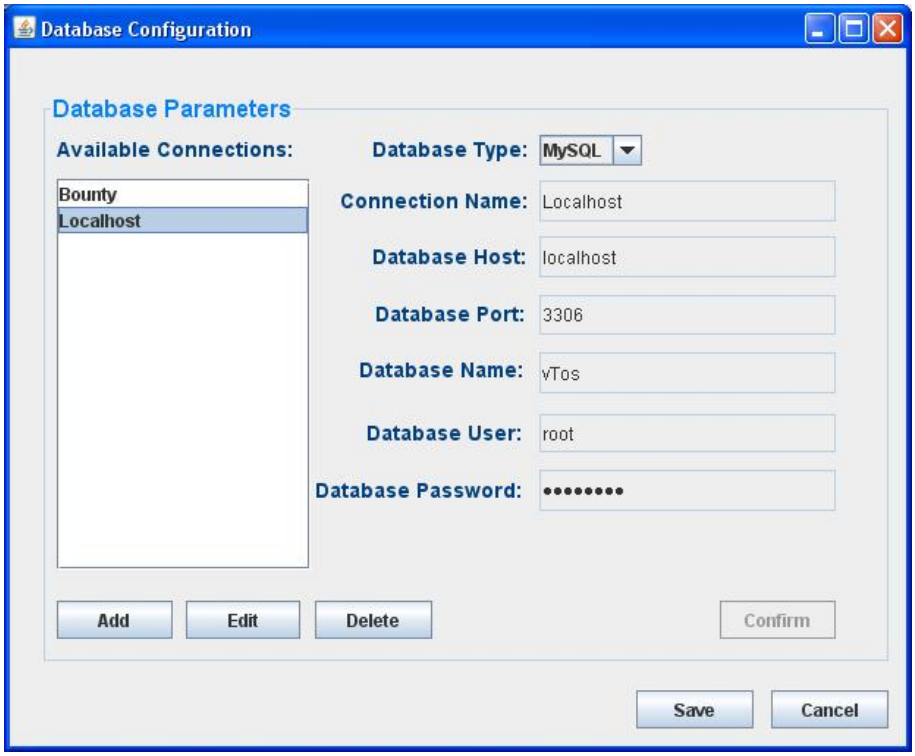


Abbildung 3.13: Database Settings Fenster

Anwendungsfallname	Database Settings
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Settings“-Funktion. 2. System öffnet Settings-Formular.
Anfangsbedingungen	-
Abschlussbedingungen	Datenbankdaten werden in einer XML-Datei gespeichert.
Qualitätsanforderungen	-

Tabelle 3.8: Anwendungsfall Database Settings

Der Konfigurationsdialog „Database Settings“ stellt mehrere Auswahlmöglichkeiten zur Verfügung. Neben der Datenbankverbindung lassen sich die gewünschte Datenbank, in der die Simulationsdaten erfasst werden, auswählen sowie ein Datenbankuser und das entsprechende Passwort für den Zugriff auf die Datenbank angeben oder ändern. Ein Nutzer besitzt die Möglichkeiten Datenbankverbindungen hinzuzufügen, zu löschen oder zu editieren. Bei den Datenbanktypen werden MySQL, Oracle und MS SQL unterstützt. Jede Datenbankverbindung hat einen eindeutigen Namen und wird mit ihren Verbindungsdaten in einem XML-Konfigurationsdokument gespeichert.

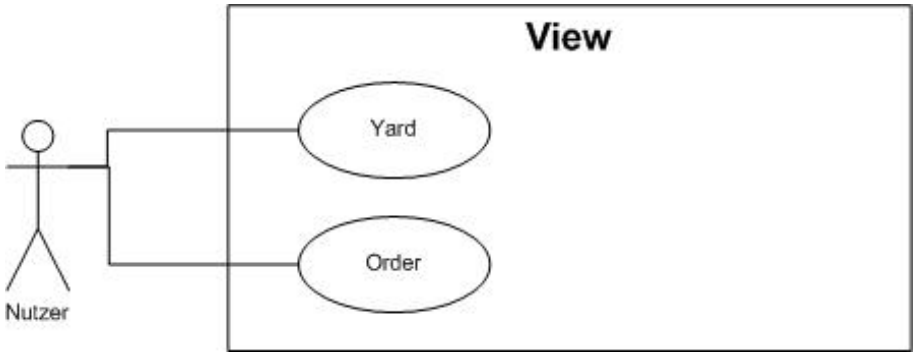


Abbildung 3.14: View Yard und View Order

Anwendungsfallname	Yard-View
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Yard“-Funktion. 2. System öffnet Yard-View.
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	-

Tabelle 3.9: Anwendungsfall Yard-View

Die Yard-View gibt einen Überblick über die Belegung der gesamten Containerstellfläche auf dem Containerterminal. Jedem Containerstellplatz ist eine eindeutige Bezeichnung zugeordnet und dieser lagert eine bestimmte Art von Container. Zu den

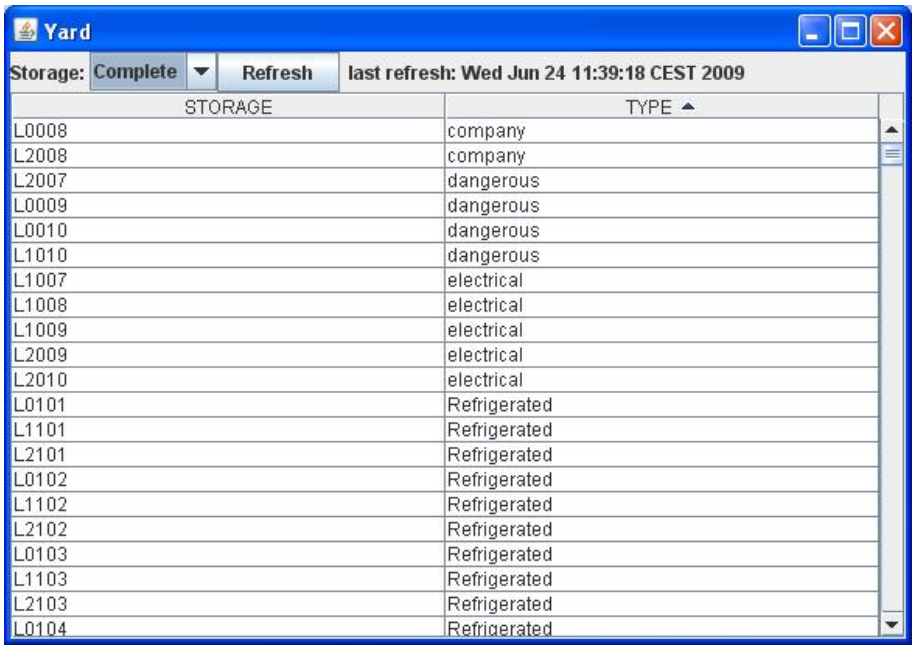


Abbildung 3.15: Open Yard View

unterschiedlichen Arten von Containern gehören Kühlcontainer(*refrigerated*), Gefahrgutcontainer(*dangerous*) sowie die ISO-Container TEU(*Twenty-foot Equivalent Unit*) und FEU(*Fourty-foot Equivalent Unit*. In der Übersicht kann sich der Nutzer anzeigen lassen, welcher auf dem Yard welchen Containertyp lagert.

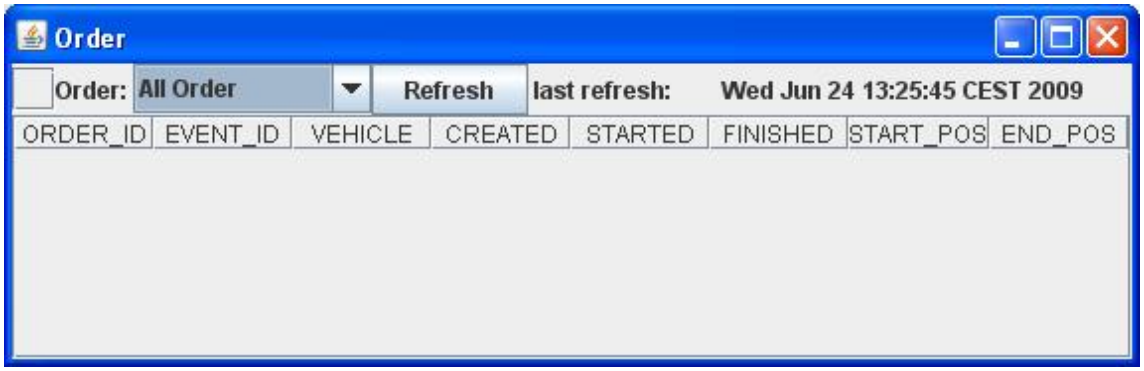


Abbildung 3.16: Open Order View

In der Tabelle werden die Transportaufträge mit den für die Bearbeitung notwendigen Informationen angezeigt. Zu den verfügbaren Informationen gehören eine Auftragsnummer, die involvierten Transportmittel, das Erfassungs-, Bearbeitungsstart- und Fertigstellungsdatum des jeweiligen Auftrags sowie die Start- und Endposition.

Das vTOS Reporting Modul bietet dem Nutzer die Möglichkeit, sich auswählbare Statistiken in Form eines Tortendiagramms ausgeben zu lassen. Durch das Betätigen der Schaltfläche *Create Selected Reports* werden die gewünschten Reports erstellt und in einem separaten Fenster angezeigt. Mit *Abbrechen* wird das vTOS Reporting Modul geschlossen.

Anwendungsfallname	Order-View
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Order“-Funktion. 2. System öffnet Order-View.
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	-

Tabelle 3.10: Anwendungsfall Order-View

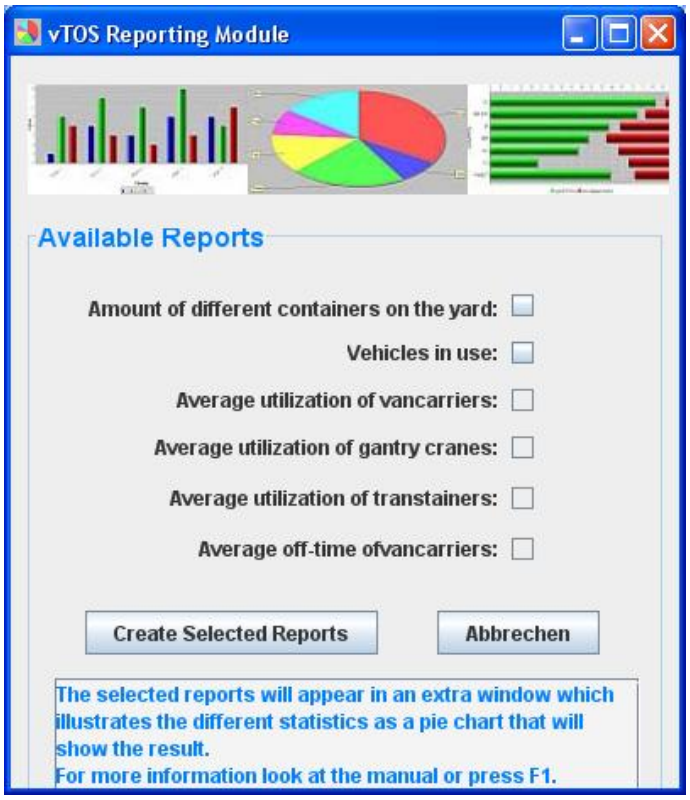


Abbildung 3.17: vTOS Reporting-Modul

Anwendungsfallname	Statistics
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Statistics-Reporting“-Funktion. 2. System öffnet Statistik-Dialog. 3. Nutzer selektiert die gewünschten Reports.
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	Fehlereingaben des Nutzers müssen abgefangen werden.

Tabelle 3.11: Anwendungsfall Statistics

Abbildung 3.18: Offizielle Webseite der Projektgruppe Virtual Port 2

Anwendungsfallname	Help-About
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Help-About“-Funktion. 2. System öffnet Webseite des Projekts im Standard-Browser.
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	-

Tabelle 3.12: Anwendungsfall Help-About

Über den Menüeintrag *About* gelangt der Nutzer auf die Webseite des Projekts *Virtual Port II*, auf der er Informationen zur Projektgruppe und dem entwickelten System erhält.

Anwendungsfallname	Help-Manual
Akteure	Initiierender Akteur: Nutzer
Ereignisfluss	1. Nutzer betätigt „Help-Manual“-Funktion. 2. System öffnet das Programm-Handbuch(PDF).
Anfangsbedingungen	-
Abschlussbedingungen	-
Qualitätsanforderungen	-

Tabelle 3.13: Anwendungsfall Help-Manual

Das Handbuch zur Software ist für den Nutzer über das Menü *Help* und den Eintrag *Manual* zu erreichen. Es wird mit dem Standard-PDF-Betrachter des jeweiligen Nutzers geöffnet und gibt einen Überblick über die Funktionalitäten und den Aufbau der Programmoberfläche.

3.2.9 Dokumentationsanforderungen

Das Produkt „Virtual Port II“ soll am Projektende durch eine entsprechende Dokumentation komplettiert werden. Hierzu zählen die verschiedenen Dokumente und Modelle, die während des Entwicklungsprozesses entstehen und nachfolgend aufgeführt sind:

- Anforderungsdefinition
- Entwurfsdokument mit Architektur- und Schnittstellenbeschreibung
- Installations- und Wartungsunterlagen
- Benutzerhandbuch
- Sourcecode mit JavaDoc
- Objektmodelle der Visualisierung
- Simulationsmodell

4. Planung

In diesem Kapitel werden die Planungsprobleme Liegeplatzplanung, Containerbrckeneinsatzplanung sowie die Yard-Planung beschrieben. Dazu werden zunächst einige Ansätze aus der Literatur aufgeführt und anschließend wird jeweils ein Lösungsansatz genau erläutert.

4.1 Liegeplatzplanung (Berth Allocation Problem)

Bei der Liegeplatzplanung wird jedem Schiff genau ein Liegeplatz am Kai zugeordnet. Dabei wird festgelegt, wann und wo ein Containerschiff am Kai anlegen soll. Die Länge des Schiffs ist hierbei eine bekannte Größe, während die Operationszeit, die möglichst gering ausfallen soll, geschätzt werden muss. Der Liegeplatz für ein Containerschiff soll nach Möglichkeiten schon feststehen, wenn das Schiff am Hafen ankommt. Dabei haben die jeweiligen Kaipositionen jeweils unterschiedlichen Nutzen für jedes Schiff.

4.1.1 Ansätze in der Literatur

Für Reeder sind geringe Wartezeiten ein entscheidungsrelevantes Produktivitätskriterium. Hiernach werden Häfen und Terminals entlang der Route ausgewählt. Genau deshalb ist das Berth Allocation Problem (BAP) in der Literatur so häufig zu finden und es existieren viele wissenschaftliche Modelle, die Lösungsansätze bieten. Einige dieser Ansätze werden im Folgenden vorgestellt.

Die Liegeplatzplanung nach KIM und MOON:

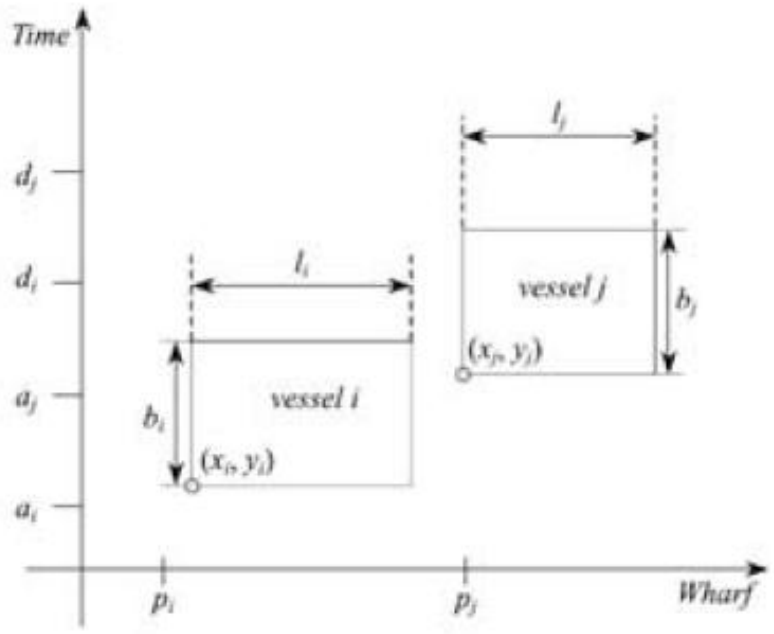


Abbildung 4.1: Idealtypische Liegeplatzplanung mit time-space-Diagramm

Das Liegeplatzproblem wird am häufigsten als time-space-Diagramm dargestellt. Die Ordinante zeigt den Zeitpunkt an, an dem das Containerschiff einem Kaiplatz zugeordnet wird und die Abzisse gibt die Position am Kai an.
Eine konkrete Darstellung eines Planes für die Zuordnung von Liegeplätzen für Containerschiffe zeigt die nächste Abbildung.

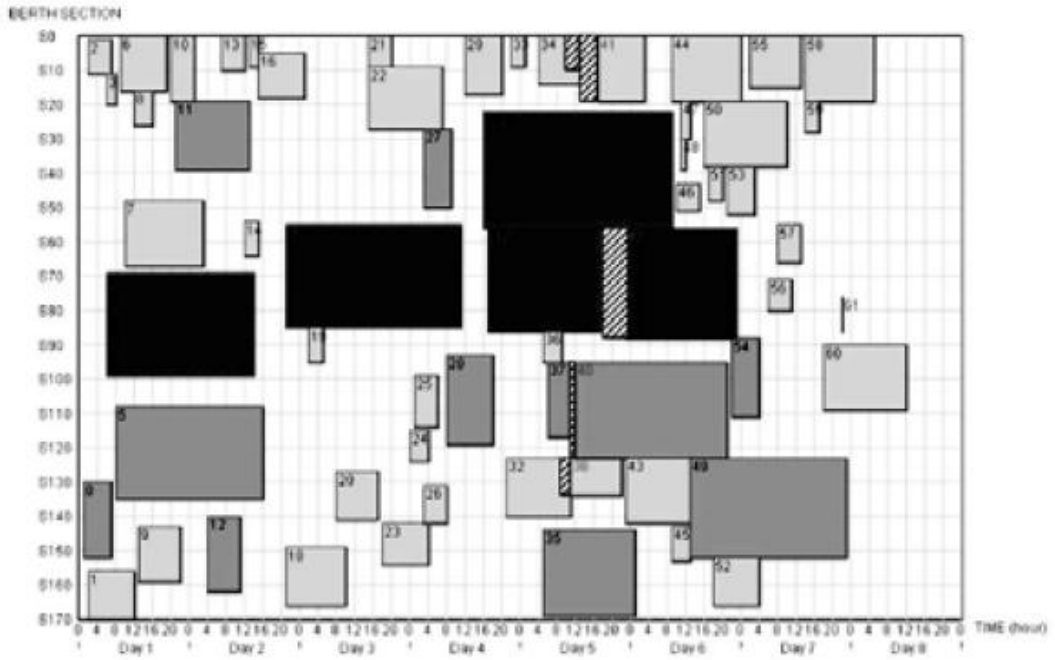


Abbildung 4.2: Planung von Liegeplätzen für Containerschiffe

In Abbildung 4.2 ist die Abzisse die Zeitachse und die Ordinate bildet die Kaifläche ab. Die Containerschiffe haben eine gegebene Größe und auch eine gegebene Operationsdauer. Somit wird sichtbar, wann ein Schiff an welcher Position am Kai anliegt und für welche Dauer dieses Schiff den Liegeplatz belegt. Über die Zeitachse ist zu sehen, wann welches Schiff als nächstes an einen Liegeplatz anlegt. Eine Lösung dieser Planung ist dann gültig, wenn sich die Schiffe nicht überschneiden, d.h. es dürfen nicht mehrere Schiffe zur selben Zeit an einem Liegeplatz liegen.
In der Abbildung haben einige Schiffe schraffierte Felder. Bei diesen Containerschiffen weicht die erwartete Ankunftszeit von dem zugeordneten Anlegezeitpunkt ab. Hier entsteht also eine Wartezeit für das Schiff. Der schraffierte Bereich ist dabei die Größe des Schiffs und die Differenz des Zuordnungszeitpunktes. Diese Wartezeit gilt es nach Priorität der einzelnen Schiffe zu minimieren. Es wäre z.B. möglich Containerschiffe von Reedereien mit höherem Aufkommen zu bevorzugen. Natürlich müssen ggf. rechtliche Rahmenbedingungen bei der Bevorzugung berücksichtigt werden.
Wird der Kai mit den Liegeplätzen als eine endliche Anzahl von Ressourcen (z.B. Liegeplatz 1 bis 10) betrachtet, so kann er diskret modelliert werden.

Ein weiterer Ansatz wird von IMAI, NAGAIWAN und CHAN als in polynomialer Zeit lösbare statische Formulierung des Berth Allocation Problems gezeigt.

4.1.2 Berth Allocation Problem nach GUAN und CHEUNG

Die Lösung des Problems ist als eine zusammengesetzte Heuristik modelliert. Dabei wird eine Tree-Search-Procedure(TSP) und eine Heuristik mit paarweisem Austausch(PWE)

von Containerschiffen verwendet. Die Tree-Search-Procedure liefert für kleine Instanzen, die eine maximale Größe von 15 Schiffen pro Gruppe nicht überschreiten, eine optimale Lösung. Allerdings wird dazu angenommen, dass die Schiffe dieselbe Ankunftszeit haben und zudem keine Präferenzen bezüglich einer Kaiposition besitzen. Für die Ankunftszeit werden die Schiffe in kleinere *Batches* zusammengefasst. Diese können sich etwa auf denselben Wochentag beziehen. Für diese zusammengefassten Schiffe wird dann eine gemeinsame Ankunftszeit gesetzt. Durch dieses Verfahren gehen zwar Informationen über die genauen Ankunftszeiten der Schiffe verloren, allerdings lässt sich so mit der Tree-Search-Procedure für den jeweiligen *Batch* eine optimale Lösung finden.

Entscheidungsvariablen (Positionen relativ):

- u_i legt den Startzeitpunkt der Operationen am Containerschiff fest
- v_i gibt die Start-Liegeplatzsektion des Containerschiffes an
- c_i stellt die Fertigstellung der Bearbeitung des Containerschiffs fest. Diese Variable ergibt sich aus der Summe von u_i und der angenommenen Operationszeit des Schiffes p_i.

Eigenschaften der Schiffe:

- p_i die Operationszeit des Schiffes i
- s_i die Größe des Schiffes i
- a_i die Ankunftszeit des Schiffes i
- w_i die relative Bedeutung des Schiffes i

Allgemeine Variablen:

- N Anzahl Schiffe, die den Hafen erreichen
- T Planungszeitraum
- S Kailänge

Damit sich die Schiffe an den zugewiesenen Kaiplätzen nicht überlappen können, werden noch zwei weitere Variablen benötigt:

- σ_{ij} hat den Wert 1 wenn sich das Containerschiff i vollständig links von Containerschiff j befindet (keine Überschneidung), sonst 0

- δ_{ij} hat den Wert 1 wenn sich das Containerschiff i vollständig unterhalb von Containerschiff j befindet, d.h. wenn Schiff i an einem Kaiplatz anliegt, liegt Schiff j nicht an. Die Schiffe überscheiden sich nicht.

Ein Containerschiff liegt genau dann vollständig von Containerschiff j, wenn der Anlegezeitpunkt von j nach dem Fertigstellungszeitpunkt von i liegt. Somit liegt i nicht mehr an der Kaiposition, wenn j anlegen will, d.h.

$c_i < c_j - p_j$.

Ein Containerschiff i liegt vollständig oberhalb(Abbildung 4.1)[unterhalb(Ankunftzeit später)] eines anderen Containerschiffes j, wenn

$v_i > v_j + s_j[v_i < v_j - s_j]$.

Die Formulierung des Modells nach GUAN und CHEUNG sieht wie folgt aus:

Zielfunktion: Minimiere $\sum_{i=1}^{N} w_i(c_i - a_i)$

Folgende Nebenbedingungen sind zu beachten:

1. $u_j - u_i - p_i - (\sigma_{ij} - 1)T \geq$,
2. $v_j - v_i - si - (\delta_{ij} - 1)S \geq 0$,
3. $\sigma_{ij} + \sigma_{ji} + \delta_{ij} + \delta_{ji} \geq 1$
4. $\sigma_{ij} + \sigma_{ji} \leq 1$
5. $\delta_{ij} + \delta_{ji} \leq 1$
6. $p_i + u_i = c$
7. $u_i \in [a_i, T - p_i + 1], v_i \in [1, S - s_{i+1}]$
8. $\sigma_{ij} \in \{0, 1\}, \delta_{ij} \in \{0, 1\}$

Für (1)-(5) und (8) gilt $\forall i, j$ und für (6) und (7) gilt $\forall i$.

Der folgende Pseudocode zeigt die Heuristik von GUAN und CHEUNG, die aus der Tree-Search-Procedure und der Pairwise-Exchange-Heuristic besteht.

Listing 4.1: Die Lösungsprozedur von GUAN und CHEUNG als Pseudocode

```
procedure CompositeHeuristic

0. Obtain initial Assignment for each batch

1. Sequence batches according to their batch arrival time

2. for all batches B_i do

        2.1 for each pair of vessels with one in B_i and one in
        B_{i+1}, if the pair-wise exchange heuristic can reduce
        the objective value, then declare this pair as a
        candidate. If there is no candidate go to step 2.3

        2.2 Select the candidate with the maximum reduction
        and implement the exchange.

        2.3 Apply the tree-search procedure in B_i and move
        vessel rectangles to the left as much as possible.
        Set i = i + 1 and go to step 2.

3. If the objective value can be reduced in step 2,
   then set i = 1 and go to step 2. Otherwise terminate.

end procedure
```

Das Zusammwirken der Tree-Search-Procedure und der Pairwise-Exchange-Heuristic wird in der nächsten Abbildung deutlich. Die Composite Heuristic basiert auf dem Zusammenwirken der Tree-Search-Procedure und der Pairwise-Exchange-Heuristic. Innerhalb der *Batches* werden mit TSP den Liegeplätzen zugeordnet. Zuvor werden die Containerschiffe gruppiert. Die PWE bestimmt wie in dem Pseudocode gezeigt, ob, und wenn ja, welche Elemente zwischen zwei aufeinander folgenden *Batches* getauscht werden.[5]

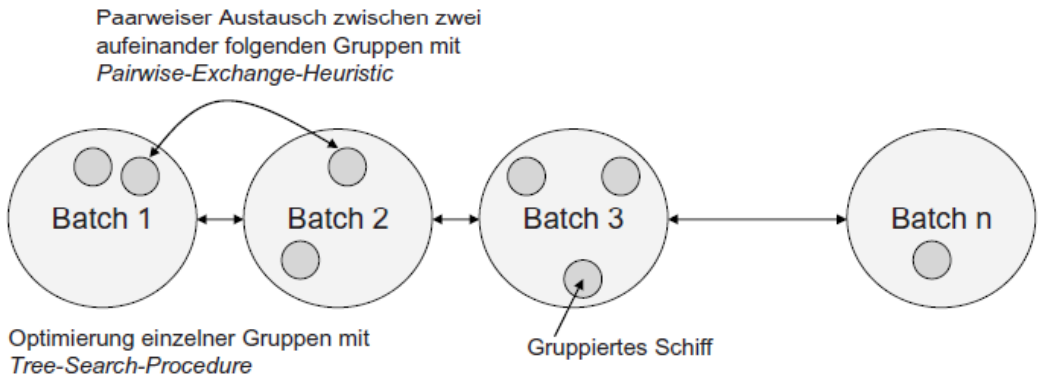


Abbildung 4.3: Composite Heuristik von GUAN und CHEUNG

4.2 Containerbrückeneinsatzplanung (Crane Scheduling Problem)

Bei der Containerbrückeneinsatzplanung werden die einzelnen Containerbrücken einem am Kai liegenden Containerschiff zugewiesen. Auf einem Containerschiff befinden sich mehrere Bays. Jede Bay bekommt eine Containerbrücke zugewiesen. Die Zuordnung von Containerbrücken zu den Schiffen beeinflusst direkt die Operationszeit.

4.2.1 Ansätze in der Literatur

Die Bezeichnung Crane Scheduling Problem (CSP) bezieht sich auf die erste wissenschaftliche Darstellung dieses Problems (DAGANZO, C.F. 1989). Zum CSP existieren bereits einige Untersuchungen, die sich bezüglich Zielsetzung, Methoden der Optimierung und verschiedenen Schwerpunkten in der Modellierung unterscheiden. Im Folgenden werden einige dieser Ansätze vorgestellt.

Das Modell von BIERWIRTH und MEISEL klassifiziert CSP in vier Stufen nach ihrem Detailierungsgrad:

1. In der einfachsten Stufe wird das Containerschiff in Areale aufgeteilt, die beladen und gelöscht werden sollen. Innerhalb dieser Areale sind mehrere Bays zusammengefasst. Jedes Areal wird von einer Containerbrücke bearbeitet und optimiert. Dadurch dass jeweils eine Containerbrücke genau einem Areal zugewiesen ist, sind Behinderungen zwischen den Containerbrücken ausgeschlossen.

2. Eine gegenüber der ersten Stufe detailliertere Stufe teilt das Containerschiff in Reihen bzw. Bays ein, die von den Containerbrücke bearbeitet werden. In dieser Stufe muss aber berücksichtigt werden, dass sich die Containerbrücken gegenseitig beeinflussen können.

3. Eine Erweiterung der vorigen Stufe führen BIERWIRTH und MEISEL so an, dass sich die Arbeitslast einer Reihe des Containerschiffs auf mehrere Containerbrücken verteilen lässt. Als Gründe für diese Aufteilung werden die verschiedenen Containereigenschaften wie Bestimmungsort, Gewicht und Größe genannt, nach denen die Container gruppiert werden. Jede Gruppe könnte dann von einer Containerbrücke über mehrere Reihen auf dem Schiff bearbeitet werden.

4. Die letzte Erweiterung sieht eine individuelle Betrachtung der Container mit ihren speziellen Eigenschaften vor. Aufgrund der hohen Anzahl von Containern erscheint diese Betrachtung jedoch nicht zweckmäßig. Hierbei würden auch die Informationen zur Gruppierung vernachlässigt. Der Erfolgsfaktor „Unit-Load“ spricht ebenfalls dafür, dass eine individuelle Betrachtung der Container nicht sinnvoll erscheint.

Die erste wissenschaftliche Darstellung zum Crane Scheduling Problem wird von DAGANZO beschrieben. Hierbei wird zwischen einem statischen und einem dynamischen Ansatz differenziert. Zuvor wurden Entscheidungen bezüglich dem Einsatz der Containerbrücken nach Erfahrungen der Entscheidungsträger begründet. In dem Ansatz werden folgende Bedingungen vorausgesetzt.

- Interferenzen werden nicht berücksichtigt, d.h. Leistungsfähigkeit, Bewegunngsfreiheit, Mindestdistanz, Austausch von Positionen und der zeitliche Mindestabstand der Containerbrücken werden nicht berücksichtigt.
- Kräne können sich ohne Zeitverlust bewegen.
- Vorgänger-Beziehungen werden nicht berücksichtigt, d.h. wenn ein bestimmter Job erst begonnen, werden kann wenn ein anderer Job bereits abgeschlossen worden ist.

Nach dem Modell werden alle Operationen eines Areals des Containerschiffes zusammengefasst. Außerdem lassen sich Jobs unterbrechen, die dann von anderen Kränen übernommen werden können. Das statische Problem werden nach dem Modell für kleine Instanzen optimale Lösungen gefunden, bei der dynamischen Variante und großen Instanzen wird eine Heuristik vorgeschlagen, die auf *Crane scheduling principles* basiert und gute Lösungen liefern soll.

Eine Erweiterung des Modells von DAGANZO zeigen PETERKOFSKY und DAGANZO. Hierbei wird das *Branch und Bound-Verfahren* für einen verbesserten Ansatz für das CSP mit größeren Instanzen verwendet.

In der folgenden Abbildung ist ein Beispiel für eine Vorgänger-Beziehung zu sehen. Die Containerbrücken-Operationen an den Bays(iii) und (iv) an Containerschiff (2) können

erst dann beginnen, wenn die Bays(i) und (ii) an Schiff(1) vollständig abgearbeitet worden sind. Schiff(2) kann erst anlegen, wenn Schiff(1) die Ressourcen am Kai frei gibt. Zu erkennen ist dies an der Ordinate.

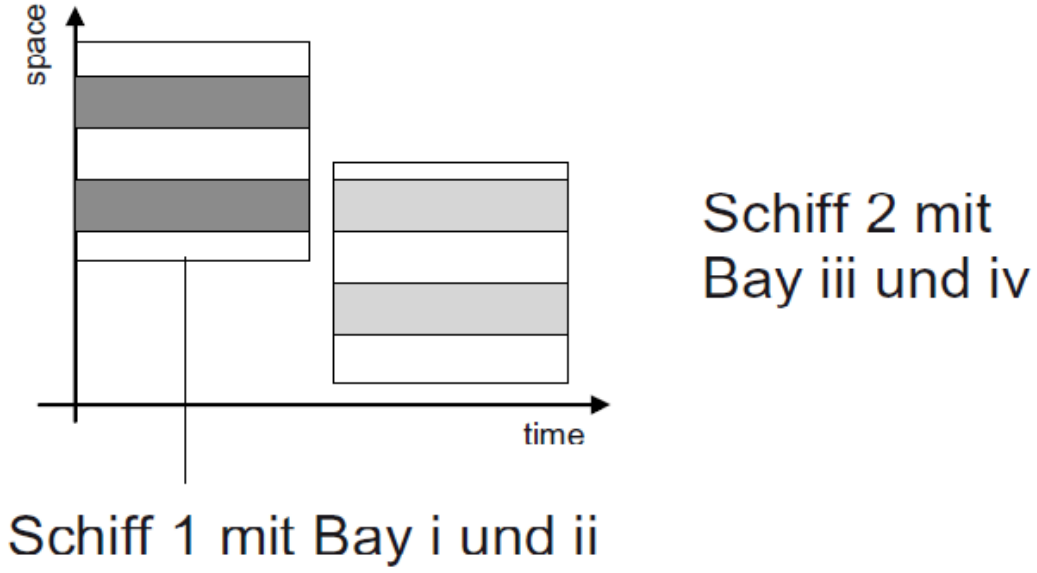


Abbildung 4.4: Vorgänger-Beziehung

4.2.2 Crane Scheduling nach ZHU und LIM

Das Modell von ZHU und LIM berücksichtigt, dass sich Containerbrücken nicht überkreuzen können, da sie auf den selben Schienen fahren. Als Zielfunktion wird die Minimierung der Fertigstellung des letzten Auftrags verwendet.

Eingangsdaten:

- m, die Anzahl eingesetzter Containerbrücken
- n, Anzahl der Bays aller zum Simulationszeitpunkt angemeldeten Containerschiffe. Die Bays werden auch als Jobs bezeichnet.
- p_i, die (geschätze) Bearbeitungszeit einer Bay i.

Diese Daten werden in dem isolierten CSP-Modell von ZHU und LIM als nicht veränderbar vorausgesetzt. Die Operationszeit ist jedoch stark von Entscheidungen anderer Problemstellugnen abhängig. Offen bleibt deshalb, ob die Prognose durch dieses Modell ausreicht und es nicht eventuell zu einer Verwerfung der Ergebnisse kommt. Dies kann passieren, wenn im Laufe der übergreifenden Planung die Annahmen nicht mehr zutreffend sind. Vgl. [5].

Folgende Varibalen sind zu bestimmen:

- c_i, der Fertigstellungszeitpunkt von Bay i (Job i erledigt)
- C_{max}, der letzte Fertigstellungszeitpunkt aller n Bays (alle Jobs von allen angemeldeten Containerschiffen erledigt)
- x_ik, gibt an, ob Bay i durch Containerbrücke k bearbeitet wird (binäre Variable). Es gilt $1 \leq i \leq n, 1 \leq k \leq m$
- y_{ij}, gibt an, ob Bay i vor dem Start der Operation an Bay j beendet ist, d.h. $c_i \leq c_j - q_j$, mit $1 \leq i, j \leq n$
- z_{ijkl}, gibt an, ob Bay i durch Containerbrücke k und Bay j durch Containerbrücke l bearbeitet wird, d.h. die nebenstehenden Containerbrücken bearbeiten auch die nebenstehenden Bays.

Nun kommmen noch einige Variablen dazu, die sich aus der vorangegangenen Liegeplatzplanung ergeben:

- a_i, Zeitpunkt, an dem die Bearbeitung der Bay beginnen kann. Dieser Zeitpunkt ergibt sich aus der Liegeplatzplanung. In dem Modell nach ZHU umd LIM wird angenommen, dass alle Containerschiffe zum Beginn des Planungszeitraums zur Verfügung stehen. Da dies jedoch in der Realität nicht der Fall ist, wird diese Variable benötigt.
- pos_i, gibt die Position des Schiffes am Kai an, die durch die Liegeplatzplanung zugeordnet wurde.

Diese Hinzunahme von diesen Parametern führt zu einer integrierten Modellvariante, die sich aus den Parametern der Liegeplatzplanung ergibt. Vgl. [5].

Nach der Festlegung aller Parameter kann nun das Modell von ZHU und LIM formuliert werden:
Zielfunktion: Minimiere C_{max}
Die Nebenbedingungen dazu sind:

1. Definition von C_{max}: $C_{max} \geq c_i$, mit $1 \leq i \leq n$
2. Definition von c_i: $c_i - p_i \geq 0$, mit $1 \leq i \leq n$

3. $\sum_{k=1}^{m} x_{ik} = 1$, mit $1 \leq i \leq n$

   In Bedingung (3) wird durch die Variable x sichergestellt, dass jede Bay genau einer Containerbrücke zugeordnet wird. Dadurch ist ein Abbruch während der Bearbeitung einer Bay nicht möglich. Die Bay kann nicht an eine andere Containerbrücke übergeben werden. Dies ist in der Realität aber auch nicht üblich.

4. $z_{ijkl} \leq x_{ik}$, mit $1 \leq i, j \leq n, 1 \leq k, l \leq m$

5. $z_{ijkl} \leq x_{jl}$, mit $1 \leq i, j \leq n, 1 \leq k, l \leq m$

   Mit 4 und 5 wird die binäre Varibale z definiert. Werden Bay i von Containerbrücke k und Bay j von Containerbrücke l übernommen, nimmt z den Wert 1 an. Für alle anderen Fälle ist z=0;

6. $x_{ik} + x_{jl} - 1 \leq z_{ijkl}$, mit $1 \leq i, j \leq n, 1 \leq k, l \leq m$

   Wenn die Variable z den Wert 1 besitzt, dann haben die Variablen x_{ik} und x_{jl} auch diesen Wert.

7. $c_i - (c_j - p_j) + y_{ij}M > 0$, mit $1 \leq i, j \leq n$

8. $c_i - (c_j - p_j) - (1 - y_{ij})M \leq 0$, mit $1 \leq i, j \leq n$

   Die Bedingungen (7) und (8) definieren die Variable y, M steht für eine Integerzahl. Endet Job i vor j (Bay i ist dann vor Job j fertig bearbeitet), nimmt die Variable y (Binärvariable) den Wert 1 an, sonst 0.

9. $y_{ij} + y_{ji} \geq z_{ijkk}$, mit $1 \leq i, j \leq n, i \neq j, 1 \leq k \leq m$

   y_{ij} und y_{ji} können nicht gleichzeitig den Wert 1 annehmen, denn entweder wird Bay i vor Bay j oder Bay j vor Bay i fertig. Somit hat immer mindestens eine Variable den Wert 0. z_{ijkk} bekommt den Wert 1, wenn sowohl i als auch j von der Containerbrücke k bearbeitet werden. Die Bedingung (9) stellt sicher, dass dies nicht gleichzeitig erfolgt.

10. $y_{ij} + y_{ji} \geq z_{ijkl}$, mit $1 \leq i, j \leq n, 1 \leq l < k \leq m$

    Die Variable z_{ijkl} nimmt genau dann den Wert 1 an, wenn Bay i von Containerbrücke k und Bay j von Containerbrücke l bearbeitet werden. Mit $1 \leq l < k \leq m$ wird sichergestellt, dass l links von k operiert. Die Containerbrücken sind dabei durchnummeriert auf einer Linie. Das Modell fordert, dass sich die Containerbrücken nicht schneiden. Wenn $l < k$ dürfen Bay i mit $(i \leftarrow k)$und Bay j mit $(i \leftarrow k)$nicht gleichzeitig bearbeitet werden, denn dies würde bedeuten, dass sich

die Containerbrücken schneiden würden. $(a \leftarrow b)$bedeutet, dass Bay a von Containerbrücke b übernommen wird.[5]

4.3 Transportmittelplanung (Vehicle Dispatching Problem)

Die Transportmittelplanung befasst sich mit der Transportplanung jedes einzelnen Containers. Ziel ist es, alle Fahrzeuge und Kräne auszulasten, um die Liegezeiten der Containerschiffe am Kai zu minimieren.

4.3.1 Ansätze in der Literatur

Bei den Lösungsansätzen in der Literatur gibt es sehr viele Abstaktionsunterschiede. Diese lassen sich in zwei Gruppen unterteilen: nach Transportmitteln und nach dem Layout des Containerterminals. Die Transportmittel sind Daten, die erforderlich sind. Die Berücksichtigung des Layouts des Containerterminals dagegen ist nicht zwingend erforderlich, kann bei großen Containerterminals jedoch zusätzlich Zeit einsparen.

4.3.2 Vehicle Dispatching Problem nach BÖSE ET AL.

Das Beispiel anhand BÖSE ET AL. berücksichtigen die Transportmittel Vancarrier und Containerbrücken. Jede einzelne Containerbrücke kann in einer bestimmten Zeit eine bestimmte Menge an Containerjobs erledigen. Die Vancarrier fahren die Container mit einer bestimmten Geschwindigkeit zum Zielort. Aus diesen Informationen kann man die Bearbeitungszeit eines jeden Containers bestimmen.
Jede Containerbrücke hat zudem mehrere Containerstellplätze, auf dem Container abgestellt werden können. Diese dienen als Puffer, damit die Containerbrücke nicht warten muss, bis der Container von einem Vancarrier abgeholt wird. Sollte der gesamte Puffer mit Container gefüllt sein, muss die Containerbrücke warten, bis ein Container abgeholt wird. BÖSE ET AL. beschäftigen sich hauptsächlich mit der Minimierung der Pufferzeiten. Vancarrier und Containerbrücken sollen gleichmäßig ausgelastet werden.
Die Vancarrier können wiederum nach zwei Strategien verteilt werden. Bei der statischen Strategie werden Vancarrier nur bestimmten Containerbrücken zugeordnet und bei der semi-dynamischen Strategie werden die Vancarrier auf alle Containerbrücken verteilt. BÖSE ET AL. befassen sich mit der dynamischen Strategie. Weiter zu beachten ist, dass zuerst die Löschcontainer und anschließend das Laden der Container bearbeitet wird.
Somit lässt sich anhand folgender Formel der Entstehungszeitpunkt eines Containerjobs berechnen: (1) kennzeichnet den Löschvorgang und (2) den Ladevorgang. Mit *i* ist der *i*-te Job der Containerbrücke *g* dargestellt. $t_g^{initial}$ gibt den Zeitpunkt für den beginn der Operationen für die Containerbrücke *g* an. $t_{g,i}^{trans}$ beschreibt die Zeit, die für die Durchführung dieses Transports durch den Vancarrier benötigt wird. [5]

$$t_{i,g}^{birth} = \begin{Bmatrix} t_g^{initial} + i * \bar{t}_g & (1) \\ t_g^{initial} + (i-1) * \bar{t}_g - t_{g,i}^{trans} & (2) \end{Bmatrix}$$

4.4 Yardplanung (Storage Location Problem)

In diesem Abschnitt soll die Yardorganisation betrachtet werden, bei der es um die Verteilung ankommender Containerströme auf Yard-Lagerplätze geht. Zunächst werden einige Ansätze aus der Literatur vorgestellt und anschließend wird ein Ansatz genauer erläutert.

4.4.1 Ansätze in der Literatur

Bei der Lagerplanung werden zwei verschiedene Lagerstrategien betrachtet. Ein bestimmter Bereich auf dem Yard kann dabei genau einem Schiff bzw. einem bestimmten Container von dem Schiff zugewiesen werden. Die andere Möglichkeit, *scattered stacking* genannt, ist die Zuordnung eines Bereichs zu einem Liegeplatz anstatt eines Schiffes. Die erste Strategie benötigt bestimmte Umschlagsinformationen während *scattered stacking* die den Nachteil der fixierten Reservierung ausgleichen soll.
Im Weiteren wird genauer auf die Verfahren eingegangen, die sich mit Strategien beschäftigen, die eine Blockplanung verwenden. Dabei wird der Yard in mehrere Blöcke eingeteilt. Eine genaue Zuordnung innerhalb dieser Blöcke ist dabei nicht vorgesehen. Abbildung 4.5 zeigt einen Block und die einzelnen Lagerplätze. Mit steigenden Um-

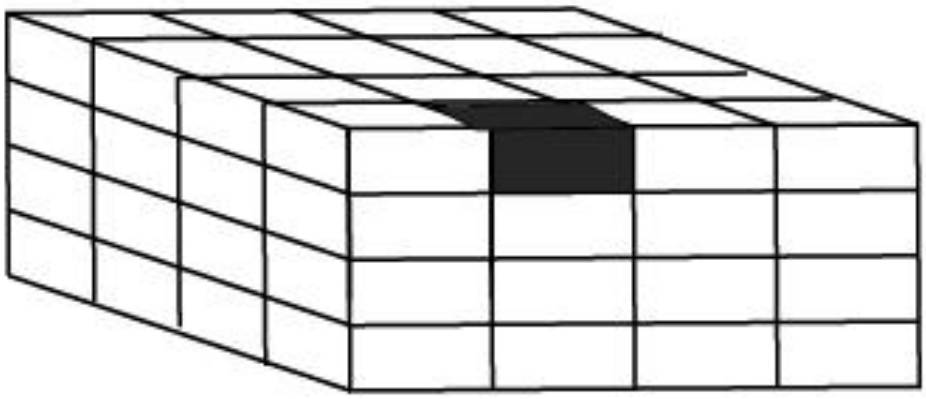


Abbildung 4.5: Yard Block mit Containerlager

schlagszahlen steigen auch die Anfragen zur Abfertigung von Containerschiffen. Daraus ergibt sich die Frage, wie hoch Container auf dem Yard gestapelt werden sollen. Mit steigender Höhe wird der Zugang zu darunterliegenden Containern aufwändiger. Somit steigt in Folge dessen die Operationszeit. Es gilt also die optimale Höhe und Anzahl zu stapelnder Container zu finden. Die Frage geht auf JIN, LIU und GAO (Vgl. JIN, C.; LIU, X.; GAO, P. (2004).) zurück.

Die eingesetzten Fahrzeuge sind ein weiterer Faktor, der die maximale Anzahl an zu lagernden Containern auf einer gegebenen Fläche beeinflusst. Abbildung 4.6 zeigt, wie verschiedene Yardkräne und -transporter sich auf die Containeranzahl auswirken. Die Angaben sind in TEU/Hektar. Aus der Abbildung ist ersichtlich, dass passive Fahrzeuge,

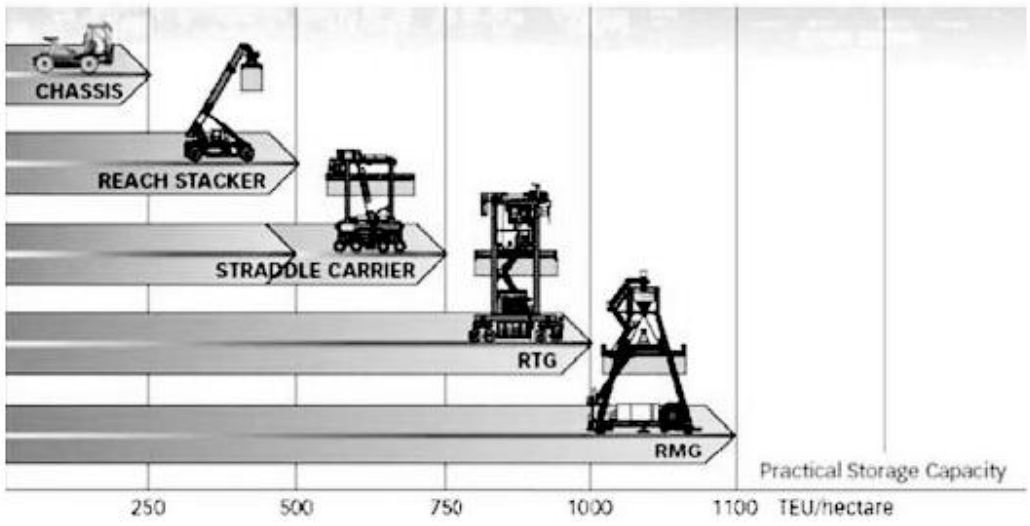


Abbildung 4.6: Einfluss der Transportmittel auf die Lagerkapazitat

die also alleine nur den Container transportieren, ihn aber nicht verladen, im Einsatz die geringste Platzgröße für Container ermöglichen. Es ist ein weiteres Fahrzeug zum Stapeln der Container nötig, wodurch viel Platz benötigt wird, der zum Lagern der Container nicht mehr zu Verfügung steht. Die *Reach Stacker* sind aktive Fahrzeuge und können den Container transportieren und stapeln. Sie müssen aber seitlich an den Block heranfahren und benötigen somit auch viel Platz. Dazu besser geeignet sind die Van Carrier, die jedoch aufgrund ihrer Größe, die für Zwischenräume noch mehr Platz benötigen als RTG und RMG.

4.4.2 Storage Location Problem nach ZHANG ET AL.

Das Modell nach ZHANG behandelt die Zuordnung der Yard-Blöcke in zwei Stufen. Dabei wird die Liegeplatzplanung als gelöst vorausgesetzt.
In der ersten Stufe werden die Container klassifiziert. Dabei findet die Einteilung entsprechend ihrer Transportrichtung statt. Es gibt vier verschiedene Typen:

- VSDS - diese Container erreichen den Terminal von der Wasserseite.
- VSLD - diese Container verlassen den Terminal wassersetig.
- CYGD - diese Container erreichen den Terminal von der Landseite.

- CYPI - diese Container verlassen den Terminal landseitig.

Im Folgenden werden die Parameter aufgeführt, die von dem Modell verwendet werden:

- B, die Anzahl der Blöcke im Yard
- T, die Anzahl der Planungsperioden
- C_i die Lagerkapazität innerhalb von Block i, $1 \leq i \leq B$
- V_{i0} die Startbelegung innerhalb von Block i, d.h., diejenigen Container, die von Beginn an in diesem Block gelagert sind, $1 \leq i \leq B$
- P^0_{it}, die erwartete Anzahl an CYPI-Containern, die in Block i lagern und in Periode t für den landseitigen Transport aufgenommen werden sollen, $1 \leq t \leq T, 1 \leq i \leq B$
- L^0_{it}, die erwartete Anzahl von VSLD-Containern, die in Block i lagern und in Periode t auf ein Containerschiff geladen werden sollen, $1 \leq t \leq T, 1 \leq i \leq B$
- $\tilde{G}_{tk}$, die erwartete Anzahl an CYGD-Containern, die in t den Terminal erreichen und in t+k auf ein Containerschiff geladen werden sollen, $1 \leq t \leq T, 0 \leq k \leq T-t$
- $\tilde{D}_{tk}$, die erwartete Anzahl an VSDS-Container, die in Periode t wasserseitig den Terminal erreiche nund diesen in t+k landseitig wieder verlassen, $1 \leq t \leq T, 0 \leq k \leq T-t$
- $\tilde{R}_{tk}$, die erwartete Anzahl an Transshipment-Container, die den Terminal in Periode t wassersetig erreichen und diesen in t+k ebenfalls wassersetig verlassen, $1 \leq t \leq T, 0 \leq k \leq T-t$
- α_{it}, die erwartete Anzahl an VSDS-Container, die in Periode t den Terminal erreichen, Block i zugeordnet sind und außerhalb des Planungszeitraums (>T) auf ein Containerschiff geladen werden, $1 \leq t \leq T, 1 \leq i \leq B$
- β_{it}, die erwartete Anzahl an VSDS-Container, die in Periode t den Terminal erreichen, Block i zugeordnet sind und deren Auslieferungszeitpunkt unbekannt bzw. außerhalb des Planungszeitraums liegt, $1 \leq t \leq T, 1 \leq i \leq B$
- γ_{it}, die erwartete Anzahl an Transshipment-Container, die den Terminal in Periode t wasserseitig erreichen und Block i zugeordnet sind, mit einem unbekannten bzw. außerhalb des Planungszeitraums liegenden (wasserseitigen) Aufnahmezeitpunkt, $1 \leq t \leq T, 1 \leq i \leq B$

Mit diesen Parametern wird eine Verbindung zur Liegeplatzplanung hergestellt. Entsprechend muss die Liegeplatzplanung vorliegen und Ergebnisse liefern. Zu den bereits aufgeführten Parametern kommen noch weitere Entscheidungsvalriablen hinzu:

- G_{itk}, die Anzahl von CYGD-Containern mit vollen Informationen, die in Block i gelagert werden und den Terminal in Periode t erreichen und in t+k wasserseitig wieder verlassen, $1 \leq i \leq B, 1 \leq t \leq T, 0 \leq k \leq T - t$
- G_{it}, die Anzahl von CYGD-Containern, die in Block i gelagert werden und den Terminal in Periode t erreichen, $1 \leq t \leq T, 1 \leq i \leq B$
- D_{itk}, die Anzahl der VSDS-Container mit vollen Informationen, die in Block i gelagert werden, in Periode t wasserseitig den Terminal erreichen und diesen in t+k landseitig wieder verlassen, $1 \leq i \leq B, 1 \leq t \leq T, 0 \leq k \leq T - t$
- D_{it}, die Anzahl aller VSDS-Container, einschliesslich Transshipment-Container, die in Block i gelagert werden und in Periode t wasserseitig den Terminal erreichen, $1\ leqt \leq T, 1 \leq i \leq B$
- R_{itk}, die Anzahl an Transshipment-Containern mit vollem Informationen, die in Block i gelagert werden, den Terminal in Periode t erreichen und diesen nach k Zeiteinheiten ebenfalls wasserseitig verlassen, $1 \leq i \leq B, 1 \leq t \leq T, 0 \leq k \leq T-t$
- L_{it}, die Anzahl von VSLD-Containern, Transshipment-Container eingeschlossen, die in Block i gelagert un in Periode t auf ein Schiff verladen werden, $1 \leq t \leq T, 1 \leq i \leq B$
- P_{it}, die Anzahl von CYPI-Containern, die in Block i gelagert werden und in Periode t landseitig den Terminal verlassen, $1 \leq t \leq T, 1 \leq i \leq B$
- V_{it}, die Anzahl der Container in Block i am Ende der Periode t, $1 \leq t \leq T, 1 \leq i \leq B$

$$\text{Zielfunktion: } \textit{Minimiere} \rightarrow \sum_{t=1}^{T} \left\{ \begin{array}{l} w_1 \left[\max_{\{i\}}(D_{it} + L_{it}) - \min_{\{i\}}(D_{it} + L_{it}) \right] + \\ w_2 \left[\max_{\{i\}}(D_{it} + L_{it} + G_{it} + P_{it}) - \min_{\{i\}}(D_{it} + L_{it} + G_{it} + P_{it}) \right] \end{array} \right\}$$

Abbildung 4.7: Zielfunktion nach ZHANG ET AL.

5. Entwurf

In diesem Kapitel erfolgt die Beschreibung des Entwurfs, zu dessen Inhalten beispielsweise die Software-Architektur, die Schnittstellen zwischen den einzelnen Teilsystemen sowie das zugrunde liegende Datenbankmodell gehören.

5.1 Überblick

Nachdem in den vorherigen Kapiteln die funktionalen und nichtfunktionalen Anforderungen formuliert wurden, widmet sich dieser Abschnitt den Entwurfszielen und den Konzepten der geplanten Implementierung und denen der Simulations- sowie Visualisierungsmodelle. Abbildung 5.1 verdeutlicht die Gesamtarchitektur mit den drei einzelnen Komponenten Planung und Steuerung(vTOS), Simulation(SIM) sowie Visualisierung(VIS). Innerhalb dieses Entwurfskapitels erfolgt eine detaillierte Beschreibung der Schnittstellen, die zwischen den drei Komponenten bestehen. Ebenso wird auf jeden der drei Teilbereiche eingegangen und dieser näher beschrieben. Neben den Aspekten der Implementierung umfasst die Software-Architektur den zugrundeliegenden Datenbankentwurf sowie dessen explizite Beschreibung.

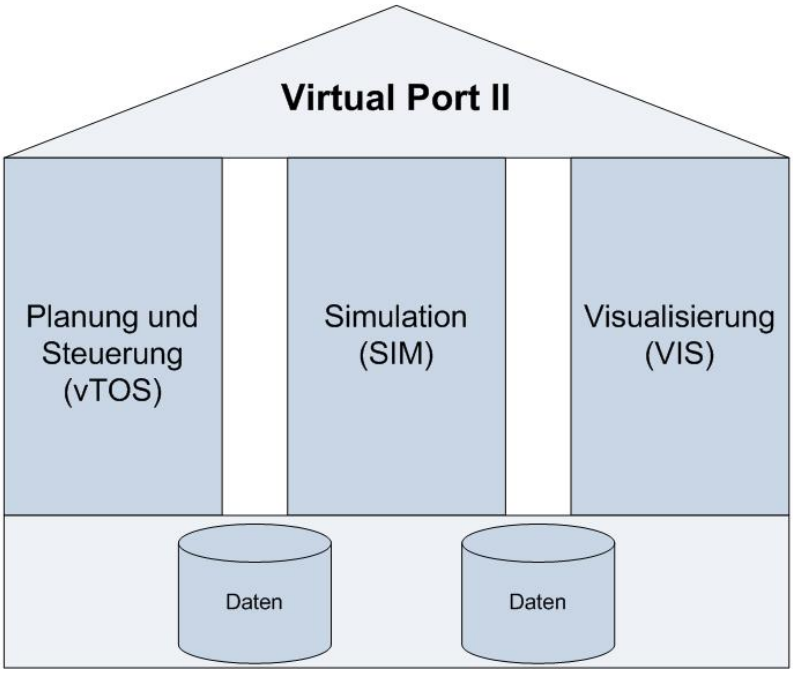


Abbildung 5.1: Allgemeine Architektur des Gesamtsystems „Virtual Port II“

5.2 Software-Architektur

Im Folgenden wird die Softwarearchitektur von Virtual Port II erläutert. Hier werden zunächst die verwendeten Pakete vorgestellt und beschrieben. Anschließend werden die Geschäftsklassen der einzelnen Pakete aufgeführt und wichtige Methoden beschrieben.

5.2.1 Packages

Das Projekt vTOS beinhaltet mehrere Packages, die nachfolgend aufgezählt und deren Aufgaben kurz beschrieben werden. Eine detaillierte Klassenbeschreibung wird die Funktionalitäten der Klassen näher erläutern. Abbildung 5.2 zeigt eine Übersicht aller in Virtual Port II verwendeten Pakete, die im Weiteren genauer beschrieben werden.

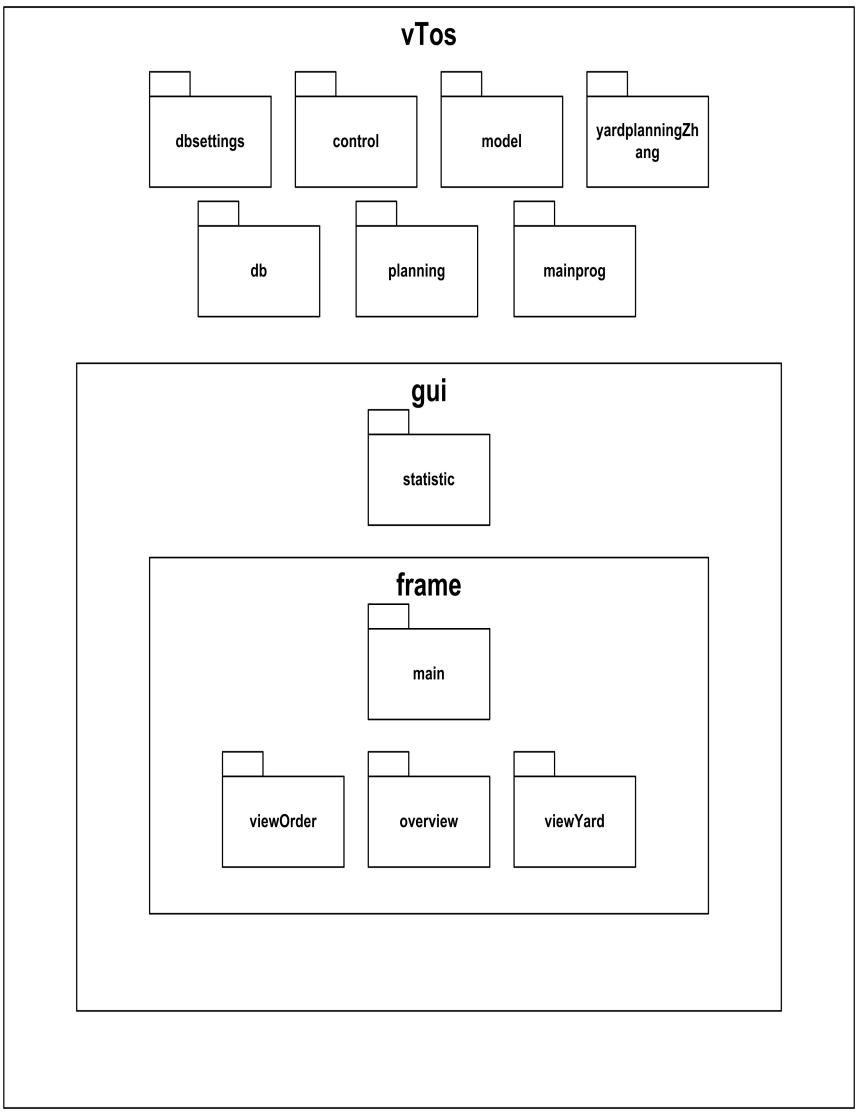


Abbildung 5.2: Paketübersicht

5.2.1.1 EmPlant

- **emPlant**
  Klassen, die für die Kommunikation mit der Software eM-Plant zuständig sind. Hier wird auch entschieden, welche Aufträge wann an eM-Plant gesendet werden.

5.2.1.2 vTos

- **vTos.db**
  Stellt eine JDBC-Datenbankschnittstelle zur Verfügung und wickelt die Kommunikation zwischen der Datenbank und den Anwendungen ab.

- **vTos.dbsettings**
  Die enthaltenen Klassen bieten die Möglichkeit Datenbankverbindungen zu editieren oder zu löschen und neue Verbindungen anzulegen.

- **vTos.mainprog**
  Klassen übernehmen die Darstellung des Programmstartfensters und bieten verschiedene Auswahlmöglichkeiten, auf die innerhalb der entsprechenden Klassenbeschreibungen näher eingegangen wird.

- **vTos.yardplanningZhang**
  Enthält diejenigen Klassen, die die Verteilung und Organsiation der zu ladenden und zu löschenden Container auf dem Yard übernimmt und regelt.

- **vTos.control**
  Enthält Controller-Klassen. Hier sind alle Actionlistener implementiert.

- **vTos.model**
  Die Klassen dieses Packages repräsentieren einige Model-Klassen, die entsprechende Daten bereitstellen.

- **vTos.planning**
  In diesem Paket geschieht die eigentliche Planung. Es werden Aufträge erstellt und in der Datenbank abgelegt.

- **vTos.gui.statistic**
  Das Package Statistik erlaubt es einem Benutzer verschiedene Auswertungen in Bezug auf terminaltypische Kennzahlen zu selektieren und sich grafisch anzeigen zu lassen.

- **vTos.gui.frame.main**
  Umfasst alle Klassen, die die GUI des Menüs Szenarien repräsentieren.

- **vTos.gui.frame.overview**
  Die enthaltenen Klassen ermöglichen eine dynamische Darstellung der Belegung des Terminals nach Blöcken und Etagen.

- **vTos.gui.frame.viewOrder**
  Enthält Klassen, die für die grafische Oberfläche zum Anzeigen aller Orders zuständig sind.

- **vTos.gui.frame.viewYard**
  Enthält Klassen, die für die grafische Oberfläche zum Anzeigen der Containerplätze auf dem Yard zuständig sind.

5.2.1.3 BerthAllocationCraneSchedulingSimulation (bacs)

Diese Komponente wurde innerhalb einer Bachelorarbeit parallel zu Virtual Port II entwickelt und integriert. Sie enthält die Liegeplatzplanung, Containerbrückeneinsatzplanung sowie die Szenarienerstellung.

- **bacs.BerthAllocation**
  Die Liegeplatzplanung besteht aus drei Planungsverfahren, die während der Erstellung eines Szenarios ausgewählt werden können. Die Liegeplatzplanung kann entweder nach dem First in First out Prinzip, nach Schiffslänge oder nach Containerumschlagsmenge des Schiffes stattfinden.

- **bacs.CraneScheduling**
  Dieses Paket enthält Klassen zur Containerbrückeneinsatzplanung. Es stehen die Verfahren nach Schiffslänge und nach Containerumschlagsmenge zur Verfügung.

- **bacs.Database**
  In diesem Paket sind Klassen zur Datenbankkommunikation enthalten.

- **bacs.Simulation**
  Das Paket Simulation enthält alle Klassen, die zur Erstellung eines Szenarios erforderlich sind. Dazu zählen auch alle GUI-Elemente.

5.2.2 Klassenbeschreibung

emPlant:

- *EmPlantControl*: Controller, der aus vTOS eine Fernsteuerung der Simulationssoftware eM-Plant eröglicht und die Kommunikation sowie das Senden ausgewählter Befehlssequenzen erlaubt.
- *EmPlantMessage*: Modell, welches das Grundgerüst einer EmPlantMessage festlegt.
- *EmPlantOrderManager*: Der EmPlantOrderManager generiert Auftragslisten, auf denen Transportmittel, Positionen sowie zu ladende und zu löschende Container inklusive Start- und Zielposition festgehalten sind.
- *EmPlantSocketClient*: Diese Klassen repräsentiert das Modell der Socket-Verbindung.
- *EmPlantSocketListener*: Ermöglicht es auf Anfragen innerhalb der Socket-Verbindung zu reagieren und daraufhin weitere Aktionen einzuleiten.

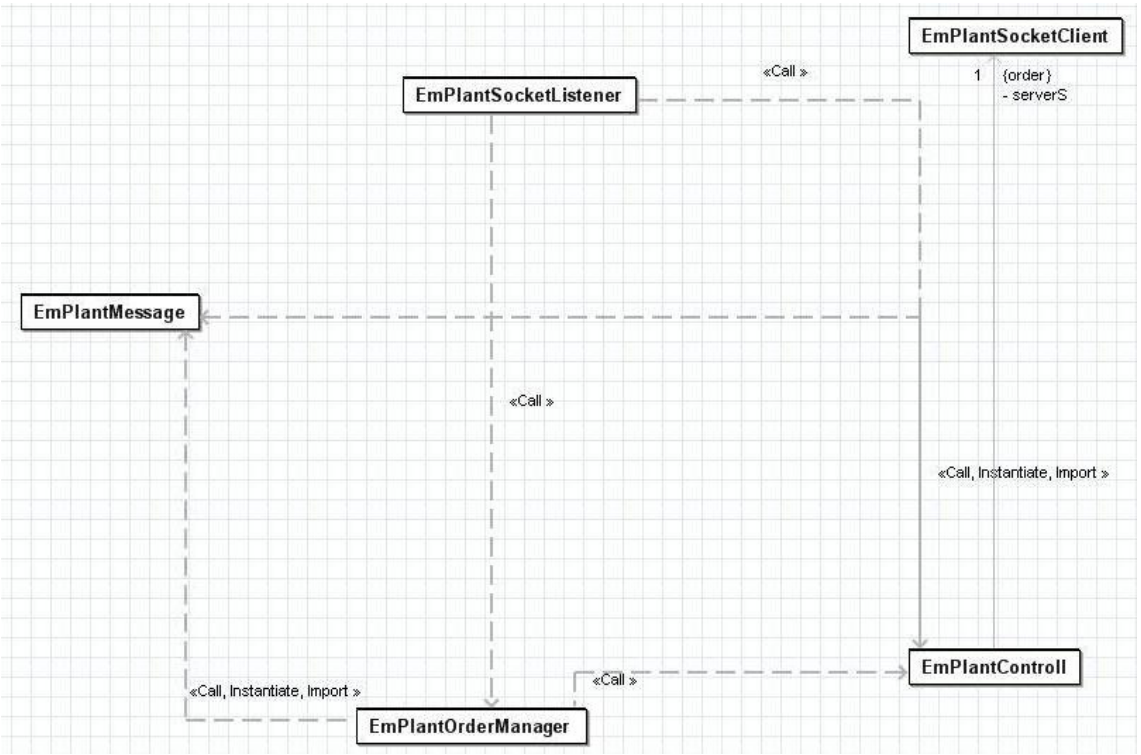


Abbildung 5.3: Klassendiagramm zum Package emPlant

vTos.db:

- *DataBase*: Beinhaltet die JDBC-Schnittstelle, regelt den Auf- und Abbau der Verbindung und stellt einige unterschiedliche Methoden zur Ausführung azsgewählter Queries zur Verfügung.

- *DataBaseEmPlant*: Diese Klasse regelt den datenbankspezifischen Zugriff auf die Datenbank der Simulationssoftware eM-Plant und stellt einige zusätzliche Hilfsfunktionen bereit.

- *DBMainPanel*: Die Klasse ermittelt einige wichtige Werte, die im Hauptfenster der Szenarienerstellung in Textfeldern ausgewiesen sind und einige Eigenschaften des Terminals darstellen.

- *DBOrder*: Beinhaltet einige Funktionen zum Abfragen, Aktualisieren und Speichern bestimmter Werte, die in Verbindung mit den Auftragslisten stehen.

- *DBPlanning*: Diese Klasse repräsentiert wichtige Funktionen, die für die Planungskomponenten benötigt werden. Hierbei handelt es sich um spezifische Informationen, die abegefragt werden müssen, um eine effiziente Planung durchführen zu können und eine unmittelbare Grundlage hierfür darstellen.

- *DBSzenario*: Klasse, die ein Szenario mit seinen Parametern in der Datenbank ablegt.

- *DBYard*: Lädt eine Übersicht des Yards und stellt Funktionen bereit, die beispielsweise die Anzahl aller freien Stellplätze oder nach Containertyp ausgeben.

vTos.dbsettings:

- *MainDBSettings*: Die Klasse repräsentiert die Datenbank-Konfiguration und bietet verschiedene Möglichkeiten zur Auswahl. Das Löschen oder Editieren bestehender sowie das Anlegen neuer Datenbankverbindungen sind möglich. Zu den Eigenschaften der Verbindungen gehören ein Verbindungsname, ein Datenbanktyp, Host, Port, ein Datenbankname sowie ein Datenbankuser und ein zugehöriges -passwort. Alle Verbindungen werden mit ihren Parametern in einer XML-Datei gespeichert.

- *CtrlDBSettings*: Beinhaltet die Funktionen zum Lesen und Schreiben der XML-Datei mit den einzelnen Datenbankverbindungen sowie das eigentliche Modell einer Datenbankverbindung. Desweiteren umfasst die Klasse geeignete Datenstrukturen zur Repräsentation der Datenbankverbindungen.

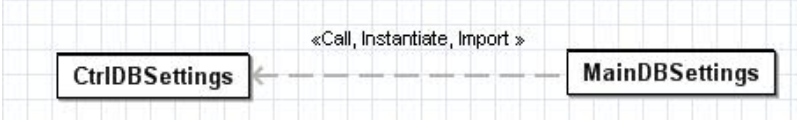


Abbildung 5.4: Klassendiagramm zum Package vTos.dbsettings

vTos.mainprog:

- *MainProgram*: MainProgram stellt das Start- und Begrüßungsfenster dar, das über verschiedene Funktionalitäten verfügt, deren weiterer Funktionsumfang in den entsprechenden Packages und Klassen erläutert wird.
- *AboutFrame*: Eine Klasse, die ein Info-Fenster erzeugt, in dem Copyright-Hinweise sowie ein Hyperlink auf die projekteigene Webseite angezeigt werden.

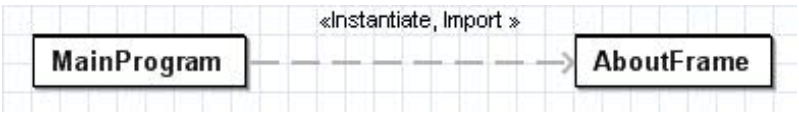


Abbildung 5.5: Klassendiagramm zum Package vTos.mainprog

vTos.yardplanningZhang:

- *MinAlgorithmus*: Diese Klasse greift die in Kapitel4.4 vorgestellte Yardplanung auf und behandelt das sogenannte „Storage Allocation Problem“. Inhalt und Aufgabe ist die Minimierung der in Abbildung4.7 dargestellten Zielfunktion zur Lösung des Problems. Der Aufbau des Algorithmus sowie welche Parameter und Vorbedingungen eine Rolle spielen, werden in Kapitel4.4 behandelt und erläutert.
- *YardConf* : Repräsentiert einen modellhaften Yard mit allen Stellplätzen, auf den der oben angesprochene Verteilungsalgorithmus angewendet werden kann.

vTos.control:

- Hier befinden sich alle Controller-Klassen von vTos. In diesen Klassen werden die Objekte erzeugt. Des Weiteren enthalten diese Klassen Methoden, die auf Benutzereingaben reagieren.

vTos.model:

- In diesem Paket sind alle Models, wie z.B. Container, Order, Yard, Ship, Scenario, etc. implementiert.

vTos.planning:

- *Order*: Hier werden Aufträge erstellt, die für das Be- und Entladen der Schiffe zuständig sind.
- *SimLoader*: Diese Klasse erstellt eine ausgewählte Simulation. Zuerst werden alle Aufträge erstellt (Order), anschließend werden die Container in das Szenario geladen. Alle Informationen werden anschließend in die Datenbank gespeichert.

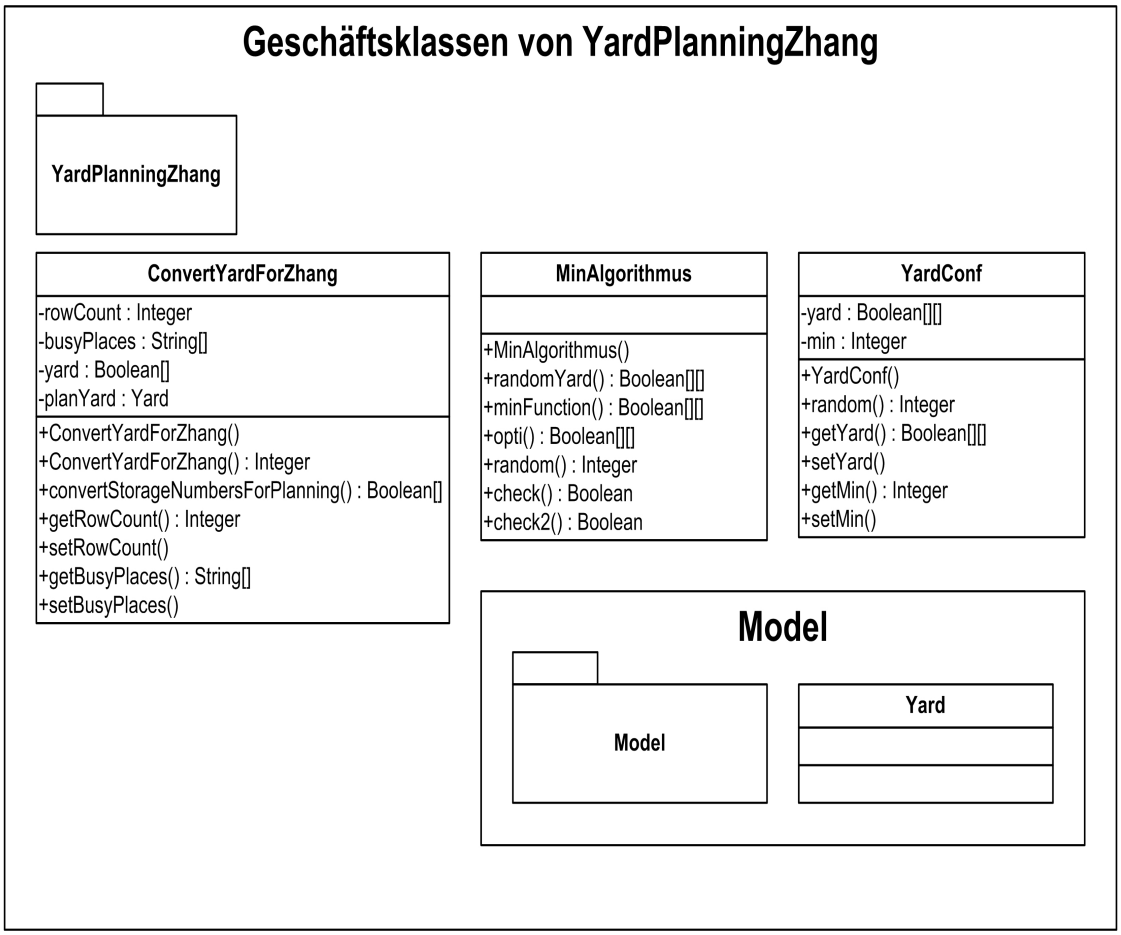


Abbildung 5.6: Klassendiagramm zum Package YardPlanningZhang

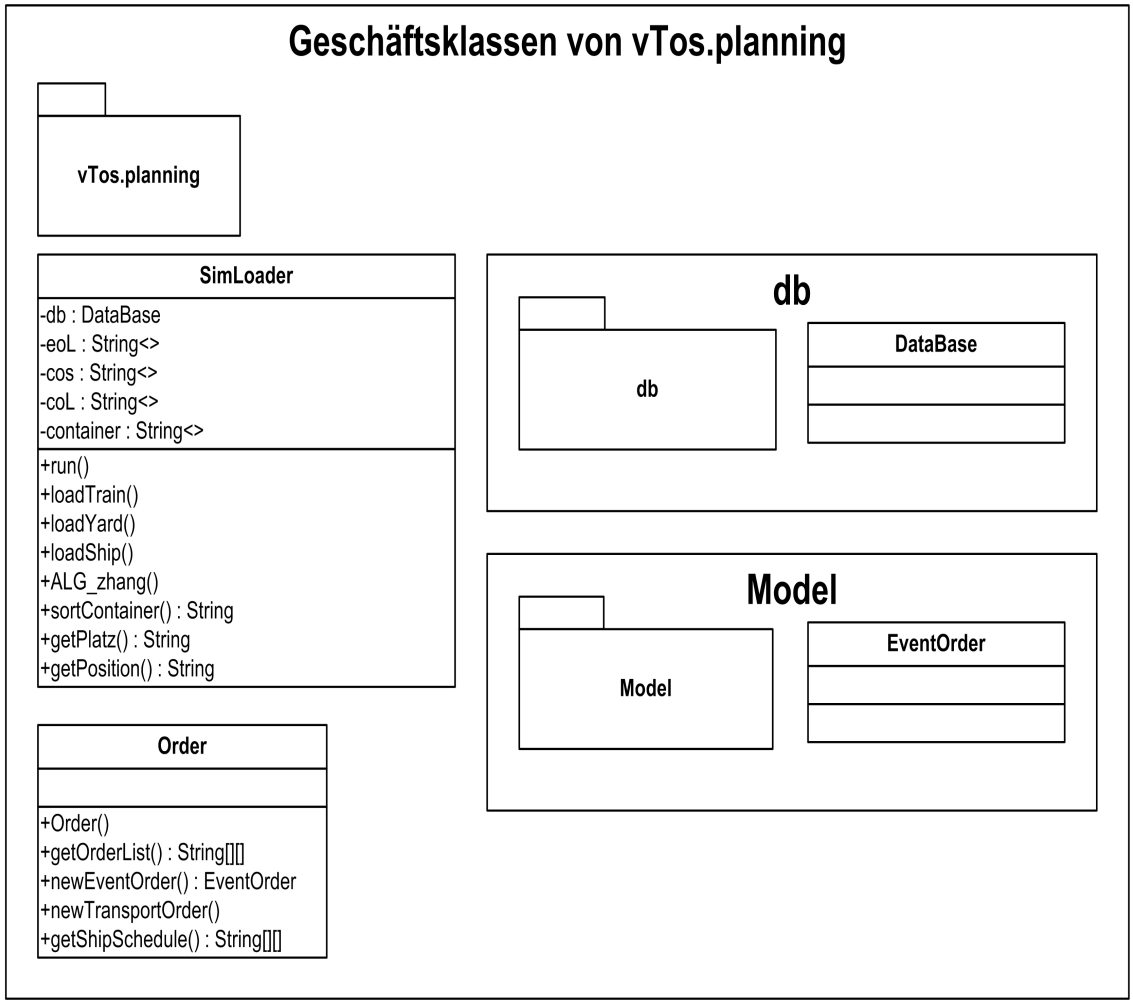


Abbildung 5.7: Klassendiagramm vTos.planning

vTos.gui.statistic:

- *MainWindowStatistic*: Eine Auswahl verschiedener statistischer Auswertungen erlaubt die Klasse MainWindowStatistic, die typische Kennzahlen eines Containerterminals in grafischer Form veranschaulicht. Zahlreiche Statistiken lassen sich selektieren und ausgeben.
- *JDBCPieChart*: Mit Hilfe der freien Java Chart Library JFreeChart[1] ermöglicht die Klasse das Erstellen der ausgewälten Statistiken in grafischer Form. Um die Statistiken dynamisch zu generieren, beinhaltet die Klasse eine JDBC-Datenbankschnittstelle, so dass für jede Statistik auf aktuelle Informationen aus der Datenbank zugegriffen werden kann.

Abbildung 5.8: Klassendiagramm zum Package vTos.gui.statistic

vTos.gui.frame.main:

- *FrameMain*: Diese Klasse dient der Darstellung der Benutzeroberfläche des Menüs „Szenarien“und aller hierin enthaltenen Komponenten.
- *CtrlFrameMain*: Controller-Klasse, die die Actions und Kontrollmechanismen innerhalb des Menüs „Szenarien“ beinhaltet.

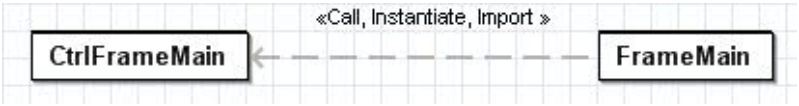


Abbildung 5.9: Klassendiagramm zum Package vTos.gui.frame.main

vTos.gui.frame.overview:

- *FrameOverview*: Die Klasse FrameOverview visualisiert die Datenbanktabelle (vgl. cargohold_land5.2.3), in der die Belegung der Containerstellplätze auf dem Terminal nach Block, Containertyp und Etage hinterlegt ist. Neben der Auswahl der Blöcke A-V bietet die Klasse die Möglichkeit, die Anzahl der gesamt sowie der je Block freien und belegten Stellplätze zu erfahren. Durch das dynamische Auslesen der genannten Tabelle wird eine aktuelle Belegungsansicht sichergestellt.

vTos.gui.frame.viewOrder:

[1] http://www.jfree.org

- *Button*: Alle Buttons, die zum Frame ViewOrder gehören.
- *Combobox*: Alle Comboboxes, die zum Frame ViewOrder gehören.
- *Label*: Alle Labels, die zum Frame ViewOrder gehören.
- *Table*: Alle Tables, die zum Frame ViewOrder gehören.
- *FrameTransportOrder*: Erzeugt das Frame, mit Parametern und enthält Attribute der oben aufgeführen Klassen.

vTos.gui.frame.viewYard:

- *Button*: Alle Buttons, die zum Frame ViewYard gehören.
- *Combobox*: Alle Comboboxes, die zum Frame ViewYard gehören.
- *Label*: Alle Labels, die zum Frame ViewYard gehören.
- *Table*: Alle Tables, die zum Frame ViewYard gehören.
- *FrameYard*: Erzeugt das Frame, mit Parametern und enthält Attribute der oben aufgeführen Klassen.

bacs.BerthAllocation:

- *Berth*: Das Model, das in der Liegeplatzplanung verwendet wird. Es enthält als Attribute Informationen zum Kai, Schiff und Container. Neben einem Default-Constructor sind hier nur alle Getter uns Setter für die Attribute implementiert. Abbildung 5.10 zeigt die Klasse Berth.
- *BerthAlgorithm*: In dieser Klasse werden die verschiedenen Planungverfahren für die Liegeplatzplanung berechnet. Diese sind: „First in first out“, „Schiffslänge“ und „Containerumschlangsmenge pro Schiff‘. Weiter sind Hilfsmethoden zur Initialisierung des Yards implementiert.

bacs.CraneScheduling:

- *Crane*: Das Model, das in der Containerbrückeneinsatzplanung verwendet wird. Es enthält als Attribute Informationen zum Schiff, Schiffsposition und Container. Neben einem Default-Constructor sind hier nur alle Getter und Setter für die Attribute implementiert. Abbildung 5.11 zeigt die Klasse Crane.
- *CraneSchedulingAlgorithm*: In dieser Klasse werden die verschiedenen Planungverfahren für die Containerbrückeneinsatzplanung berechnet. Diese sind: „Schiffslänge“ und „Containerumschlangsmenge pro Schiff‘.

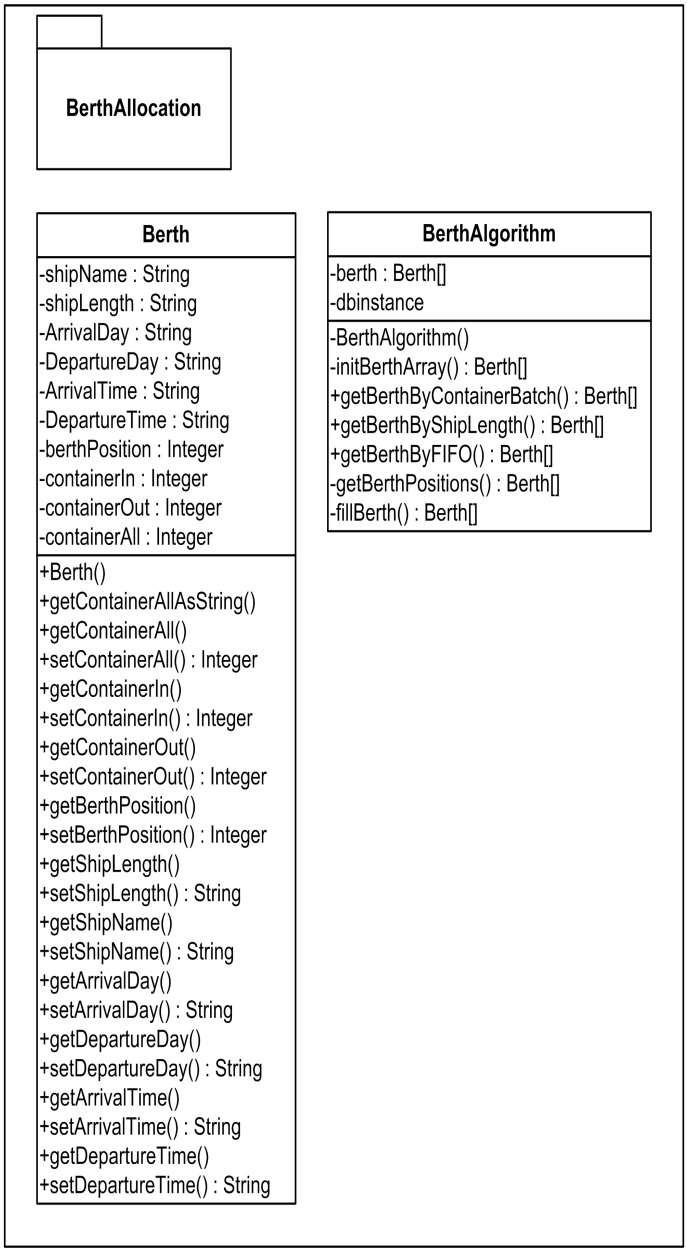


Abbildung 5.10: Klassendiagramm der Liegeplatzplanung

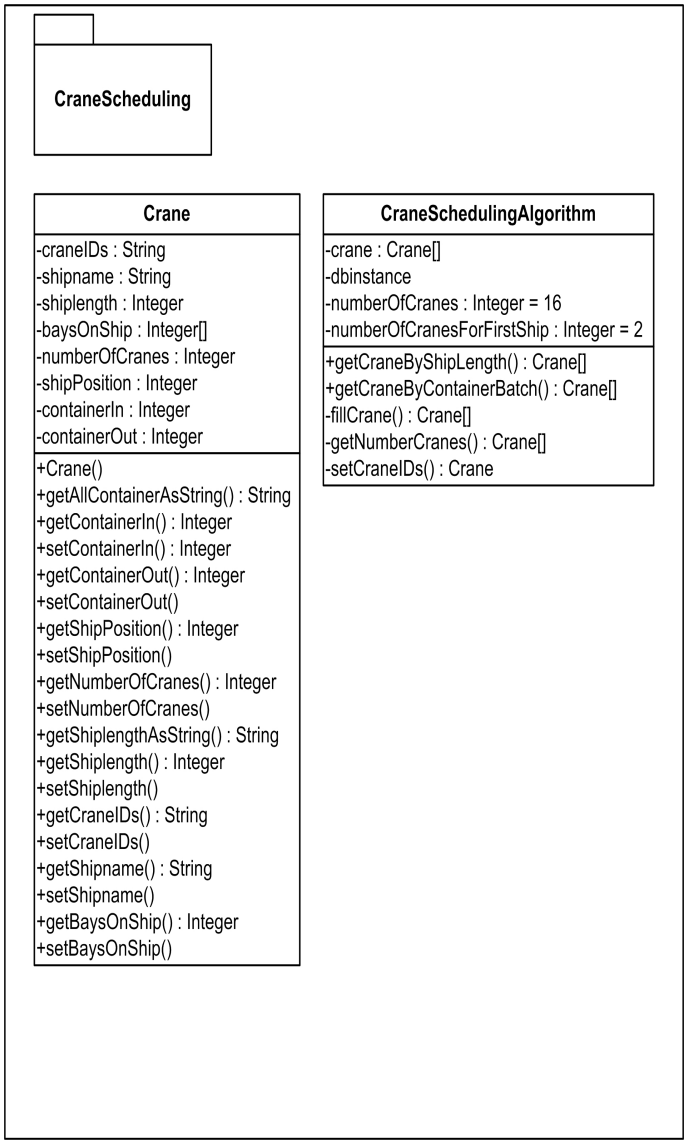


Abbildung 5.11: Klassendiagramm der Containerbrückeneinsatzplanung

bacs.Database:

- *Database*: Diese Klasse enthält als Attribute Verbindungsdaten für die Datenbank, ein Default-Contructor und Methoden, die unterschiedliche Queries ausführen.

bacs.Simulation:

- *Bay*: Das Model, das Informationen über die Anzahl Container und Kapazität der Bay enthält.
- *CreateBay*: Die grafische Oberfläche zur Erstellung der Bay und anschließende Speicherung in der Datenbank.
- *CreateShip*: Die grafische Oberfläche zur Erstellung eines Schiffes und anschließende Speicherung in der Datenbank.
- *CreateShipSchedule*: Die grafische Oberfläche zur Erstellung eines Schiffsplans und anschließende Speicherung in der Datenbank.
- *CreateSimulation*: Die grafische Oberfläche zur Erstellung einer Simulation und anschließende Speicherung in der Datenbank.
- *MyInputVerifier*: Eine Klasse, die falsche oder unzulässige Eingaben abfängt und den Benutzer entsprechende Hinweise in Form von Dialogfenstern liefert.
- *Ship*: Das Model, das Informationen, wie Name, Länge, Kosten pro Stunde, etc. über ein Schiff enthält.
- *ShipSchedule*: Ein Model, welches Informationen zum Schiffsplan enthält.
- *Simulation*: Das Model, das Informationen, wie Name, Schiffsplan, Terminal, und alle Planungsverfahren über eine Simulation enthält. Es enthält außer Getter und Setter keine weiteren Methoden.
- *StartSimulation*: Grafische Benutzeroberfläche, um eine zuvor in der Datenbank gespeicherte Simulation auszuwählen.

5.2.3 MySQL Datenbank

In diesem Projekt werden zwei Datenbanken benutzt. In der Logik werden die Umgebung und die Aufträge gespeichert und in der Simulation werden die Objekte für die 3D-Simulation erzeugt. Nachfolgend werden die Tabellen beschrieben, die für die Logik benutzt werden:

cargohold_land							
id	pos_x	pos_y	pos_z	container	type	storage_nr	reserved

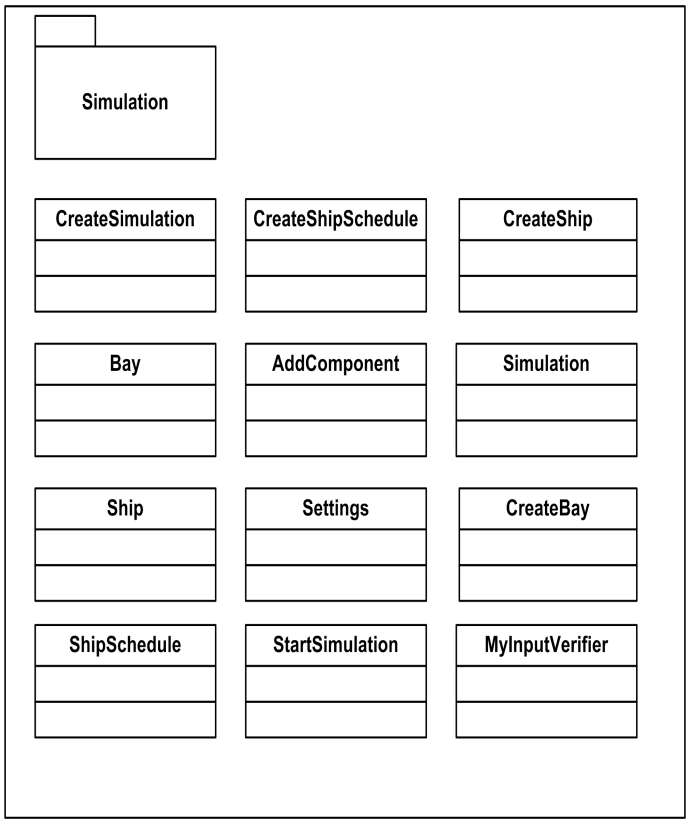


Abbildung 5.12: Klassendiagramm der Simulation

1. **id** Die ID des Datensatzes
2. **pos_x, pos_y, pos_z** Damit wird die Position auf dem Containeryard angegeben
3. **container** Die ID des Containers, der auf dieser Position steht. Wenn dort kein Container steht, wird eine 0 eingetragen.
4. **type** Typ des Containers, z.B.: TEU, FEU, Refrigerated, Dangerous
5. **storage_nr** Um den Liegepatz eines Containers schneller finden zu können, hat jeder platz eine storage_nr, die so generiert ist, dass man anhand dieser Nummer schnell den Platz auf der Übersichtskarte finden kann.
6. **reserved** Wenn ein Container auf dem Weg zu diesem Platz ist, wird reserved=1 gesetzt. Damit wird verhindert, dass dieser Platz doppelt vergeben wird.

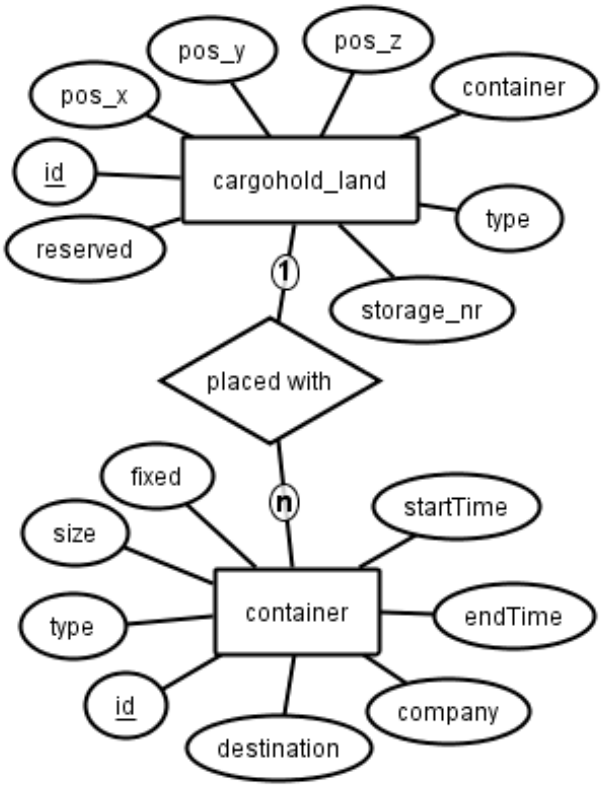


Abbildung 5.13: Tabellen cargohold_land und container

container							
id	type	fixed	size	startTime	endTime	destination	company

1. **id** Die ID des Containers
2. **type** Typ des Containers, siehe type bei cargohold_land
3. **pos_x, pos_y, pos_z**
4. **fixed** Wenn auf diesem Container ein anderer Container steht, wird dieser mit fixed=1 markiert.
5. **startTime** Zeitstempel an dem der Container auf dem
6. **endTime** Zeitstempel, wann der Container den Hafen wieder verlässt
7. **destination** Zielort des Containers

orders						
id	container_id	charge_nr	event_id	bay_id	vehicle	createTime
executeTime	endTime	start_pos	target_pos			

1. **id** Die ID der Order
2. **container_id** Die ID des Containers
3. **charge_nr** Auftragsnummer
4. **event_id** ID des dazugehörigen Events
5. **bay_id** ID der Bay
6. **vehicle** Fahrzeug, welches den Container befördert
7. **createTime** Zeitpunkt, an dem dieser Order erstellt wurde
8. **executeTime** Startzeit, an dem dieser Order ausgeführt wurde
9. **endTime** Zeitpunkt, an dem dieser Order abgeschlossen wurde
10. **start_pos** Startposition des Containers
11. **target_pos** Zielposition des Containers

events							
id	bay_id	end_pos	start_pos	createTime	executeTime	end_time	container_id

1. **id** Die ID des Transport-Orders

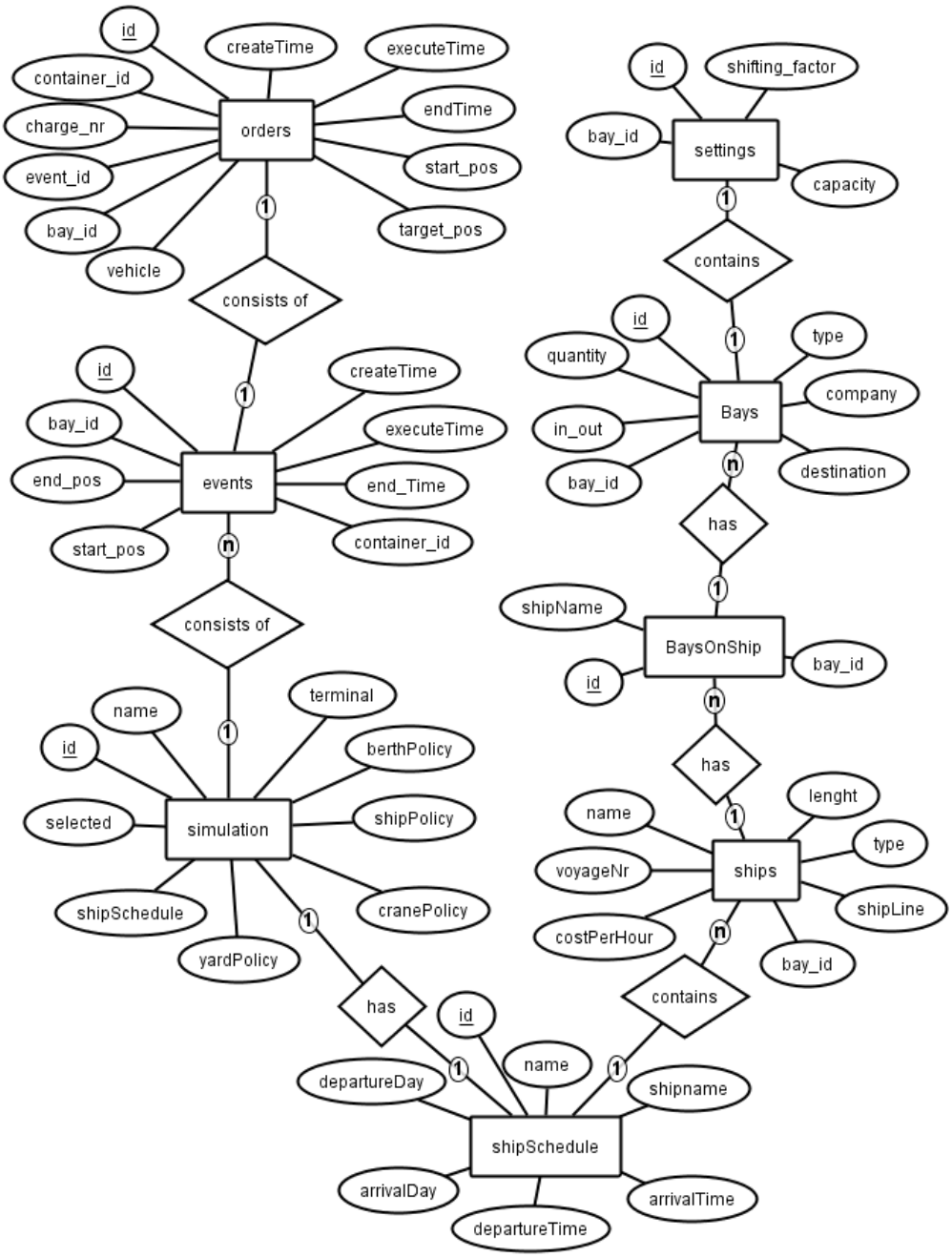


Abbildung 5.14: Tabellen, die für die Simulation zuständig sind

2. **bay_id** ID der Bay
3. **end_pos** Zielposition des Containers
4. **start_pos** Startposition des Containers
5. **createTime** Zeitpunkt, an dem dieser Order erstellt wurde
6. **executeTime** Zeitpunkt, an dem dieser Order abgeschlossen wurde
7. **end_time** Zeitpunkt, an dem dieser Order abgeschlossen wurde
8. **container_id** ID des Containers

simulation								
id	name	selected	shipSchedule	yardPolicy	cranePolicy	shipPolicy	berthPolicy	terminal

1. **id** Die ID der Simulation.
2. **name** Name der Simulation.
3. **selected** 1, wenn diese Simulation von vTos ausgewählt wurde, sonst 0.
4. **shipSchedule** Der verwendete Schiffsplan innerhalb der Simulation.
5. **yardPolicy** Die verwendete Yardplanung innerhalb der Simulation.
6. **cranePolicy** Die verwendete Containerbrückeneinsatzplanung innerhalb der Simulation.
7. **shipPolicy** Die verwendete Stauplanung innerhalb der Simulation.
8. **berthPolicy** Die verwendete Liegeplatzplanung innerhalb der Simulation.
9. **terminal** Der verwendete Terminal innerhalb der Simulation.

shipSchedule						
id	name	shipname	arrivalTime	departureTime	arrivalDay	departureDay

1. **id** Die ID des Schiffsplans.
2. **name** Name des Schiffsplans.
3. **shipname** Name des Schiffes.
4. **arrivalTime** Geplante Ankunftszeit des Schiffes.
5. **departureTime** Geplante Abfahrtszeit des Schiffes.
6. **arrivalDay** Ankunftstag des Schiffes.

7. **departureDay** Abfahrtstag des Schiffes.

ships						
name	voyageNr	costPerHour	bay_id	ship_line	type	length

1. **name** name des Schiffes.
2. **voyageNr** Seelininen-Nummer
3. **costPerHour** Kosten pro Stunde, in der das Schiff am Dock liegt.
4. **bay_id** Id der Bay.
5. **ship_line** Schiffslinie.
6. **type** Schiffstyp (Panamax, Super Panamax, etc.).
7. **length** Schiffslänge.

BaysOnShip		
id	shipName	bay_id

1. **id** Id.
2. **shipName** Schiffsname.
3. **bay_id** Id der Bay.

Bays						
id	quantity	in_out	bay_id	destination	company	type

1. **id** Id.
2. **quantity** Menge der Container.
3. **in_out** 0, wenn Container vom Schiff gelöscht werden. 1, wenn Container auf das Schiff geladen werden.
4. **bay_id** Id der Bay.
5. **destination** Ziel der Bay.
6. **company** Firmenname der Container, falls vorhanden.
7. **type** Typ der Container.

settings			
id	bay_id	capacity	shifting_factor

1. **id** Id.
2. **bay_id** Id der Bay.
3. **capacity** Kapazität der Bay.
4. **shifting_factor** Umladefaktor inneralb der Bay.

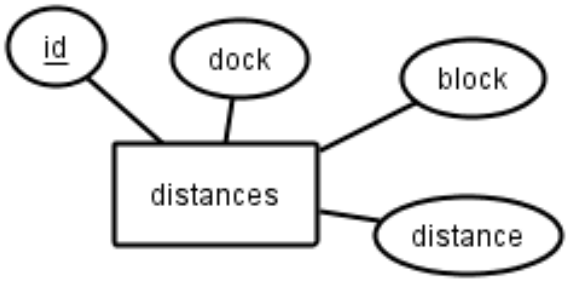


Abbildung 5.15: Tabelle, in der die Abstände vom Dock zu den Blöcken gespeichert sind

distances			
id	dock	block	distance

1. **id** Die ID
2. **dock** von 1 bis 98.
3. **block** von A bis V
4. **distance** entfernung vom Dock zum Block

5.2.4 Abbildung auf Hardware-/Software-Komponenten

Folgende Hardwarekomponenten werden für Planungs- und Simulationssoftware verwendet:

- DELL Optiplex GX620
- Intel Pentium D und CPU 2,80 GHz
- 2,00 GB Hauptspeicher

Folgende Hardwarekomponenten werden für die Visualisierung verwendet:

- Intel Core i7 CPU 950
- NVIDIA GeForce GTX295
- 6,00 GB Hauptspeicher

Folgende Softwarekomponenten werden verwendet:

- MySQL Datenbank in der Version 5.0.38
- Sun Java SE in der Version 6 Update 13 incl. Java Runtime Environment in der Version 6 Update 13
- Xith3D in der Version 0.9.5 alpha1

Die Entwickler verwenden zur Abstimmung der Arbeiten folgende Mittel:

- Flyspray in der Version 0.9.9.6 als Fehler- und Aufgabenmanagementsystem
- Subversion in der Version 1.4.5
- einen E-Mailverteiler sowie ein Wiki (Media Wiki in der Version 1.14.0

5.3 Hafenaufteilung

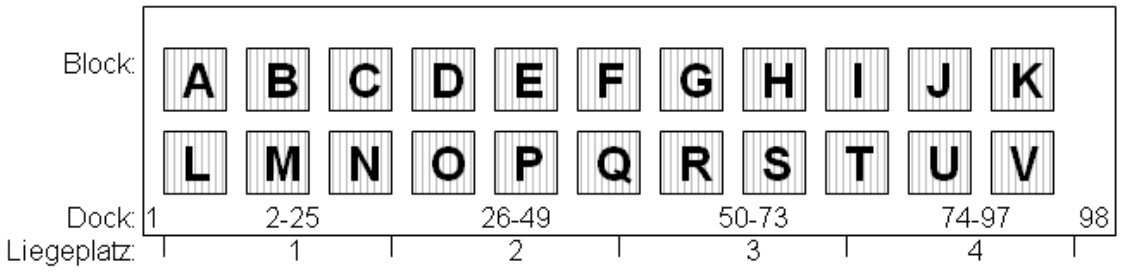


Abbildung 5.16: Aufteilung des Jade-Weser-Ports in Blöcke, Liegeplätze und Docks

5.3.1 Blöcke

Die Containerplätze auf dem Yard sind in der Datenbank mit x-,y- und z-Koordinaten eingetragen. Damit der Benutzer eine Vorstellung bekommen kann, wo ein bestimmter Container steht, gibt es für jeden Platz eine sog. “storage-nr“, aus der man die genaue Position eines Containerplatzes feststellen kann. Die “storage-nr“ besteht aus fünf Ziffern, die folgendermaßen zusammengestellt sind:

1. der Block (A bis V)

2. die Etage (0 bis 2)
3. die Platznummer (1 bis 480)

Abbildung 5.17 zeigt die Nummerierung der Plätze in einem Block. In einer Reihe liegen 24 Container nebeneinander und 20 Container hintereinander. Der Containerplatz A0023 ist als Beispiel in dem Bild in rot markiert. An der zweiten Ziffer der "storage-nr" erkennt man, dass der Platz in der ersten Etage liegt.

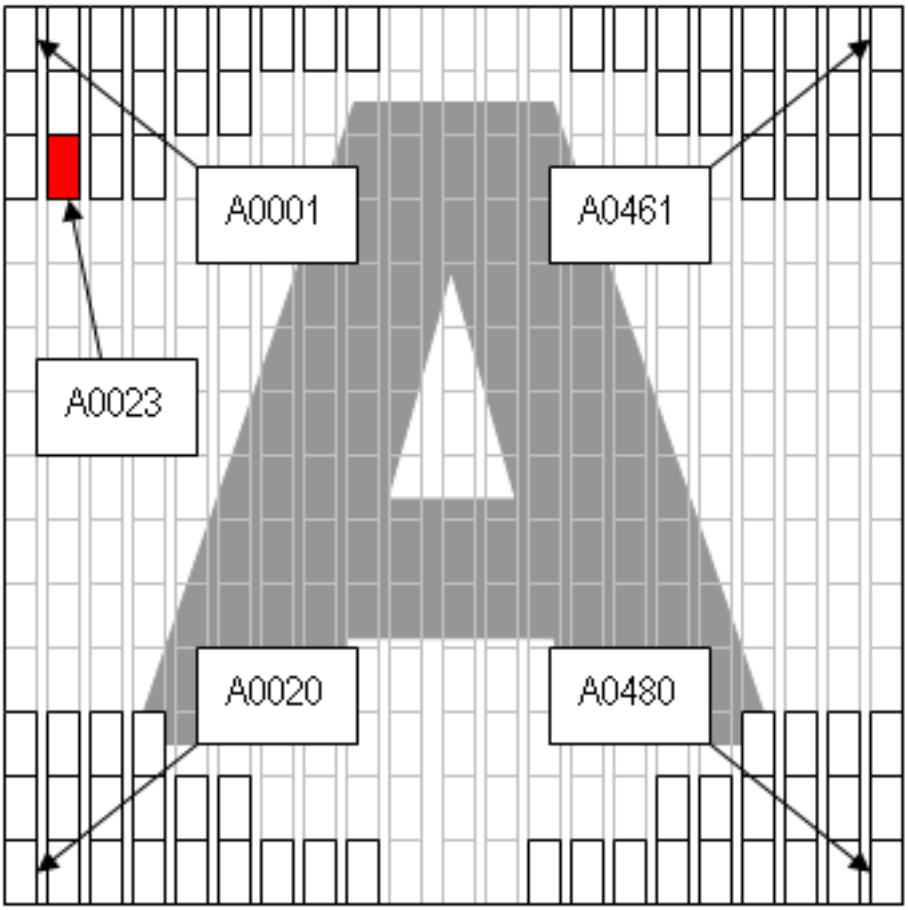


Abbildung 5.17: Aufteilung eines Blockes auf dem Containerterminal

5.3.2 Liegeplätze

Der Jade-Weser-Port hat eine Kajenlänge von 1725m. Es sind vier Liegeplätze vorgesehen, an denen die Containerschiffe andocken können. Wenn kleinere Schiffe (z.B. Feederschiffe) anlegen, können auch mehrere Schiffe auf einen Liegeplatz verteilt werden.

5.3.3 Docks

Um den Containerbrücken eine genaue Position am Dock geben zu können, wird der Dock in 98 Abschnitte eingeteilt. Diese Abschnitte sind in der Simulations-Software eM-Plant implementiert und haben eine feste größe von 17,60m. D.h. die 16 Containerbrücken können auf 98 verschiedenen Positionen verteilt werden.

5.4 Simulation

In diesem Kapitel wird der Entwurf der Hafensimulation durch eM-Plant vorgestellt. eM-Plant ist ein Simulationstool, welches bereits eine große Menge an Bausteinen besitzt, um eine spezifische Fertigung zu simulieren. Bausteine sind Simulationsobjekte, wie z.B. Wege, Fahrzeuge, Lager oder einfache Methoden.
Für die Entwicklung des Hafenmodells, werden jedoch einige weitere Bausteine benötigt, die noch nicht in eM-Plant vorhanden sind. Jedoch werden genügend Möglichkeiten zur Erstellung weiterer Bausteine angeboten. Zur Programmierung der Bausteine wird die Programmiersprache SimTalk verwendet.
Die Abbildung 5.18 zeigt einen ersten Grobentwurf der Hafensimulation. Dieser Grobentwurf soll in diesem Kapitel verfeinert werden.

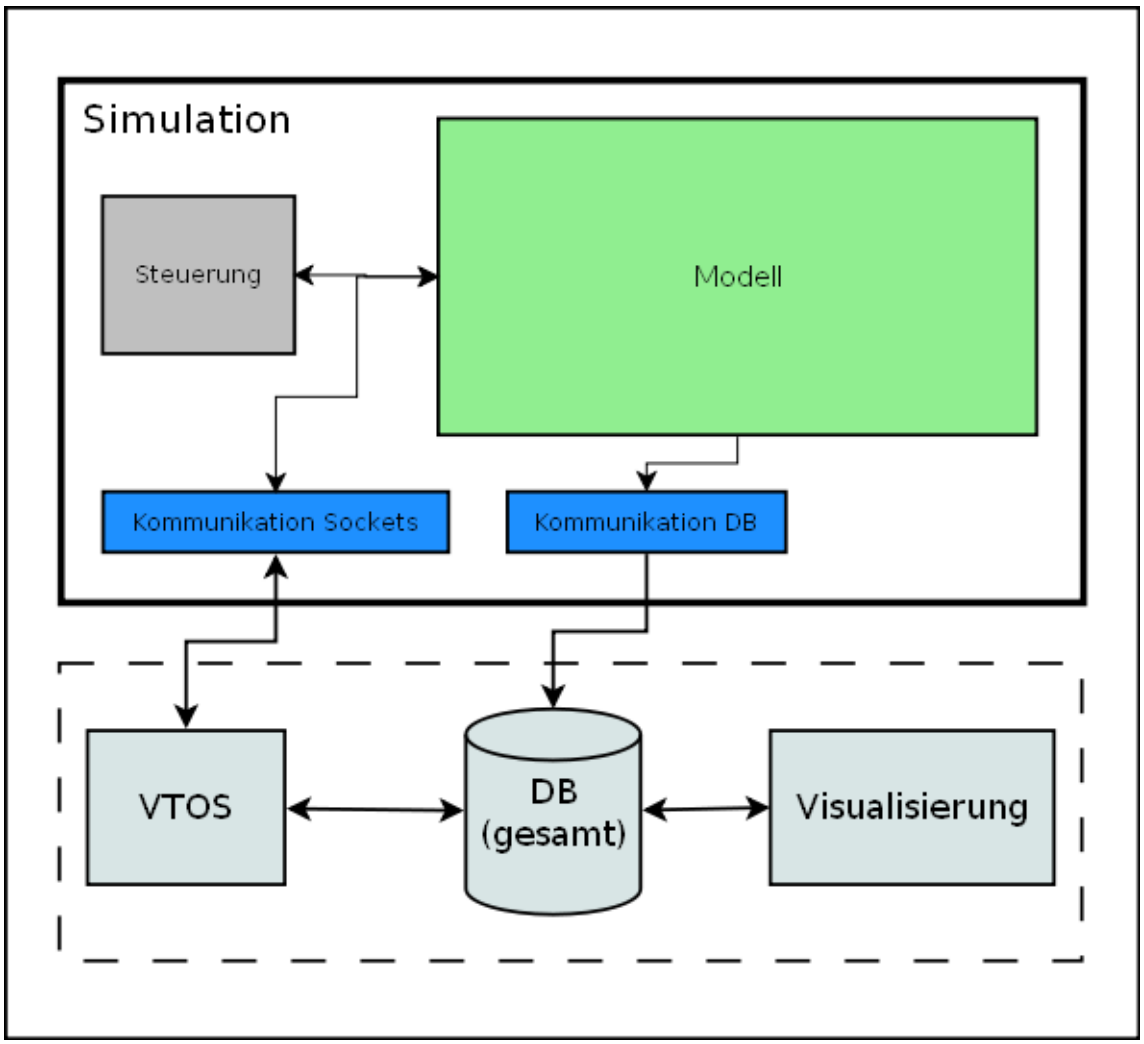


Abbildung 5.18: Grobe Architektur der Simulation

5.4.1 Modell

Dieser Abschnitt beschäftigt sich mit dem Entwurf des Simulationsmodells, also der Testumgebung. Dabei wird das alte Modell der Projektgruppe „Virtual Port 1“ als Grundlage

verwendet. Da jedoch die meisten Methoden unseren Anforderungen nicht genügen, wird nur die Oberfläche weiter verwendet. Das Simulationsmodell bestimmt auch das Aussehen des Containerhafens in der Visualisierung. Die Anordnung der Objekte in eM-Plant ist dafür ausschlaggebend. Der Grund dafür ist, dass deren Positionen in die Datenbank geschrieben werden und die Visualisierung diese zur Positionierung der 3D Objekte nutzt.

5.4.1.1 Grundriss des Modells

Abbildung 5.19 zeigt den Grundriss des alten Modells. Die roten Rechtecke mit den Zahlen sollen hier zur Erklärungshilfe der geplanten Änderungen dienen.

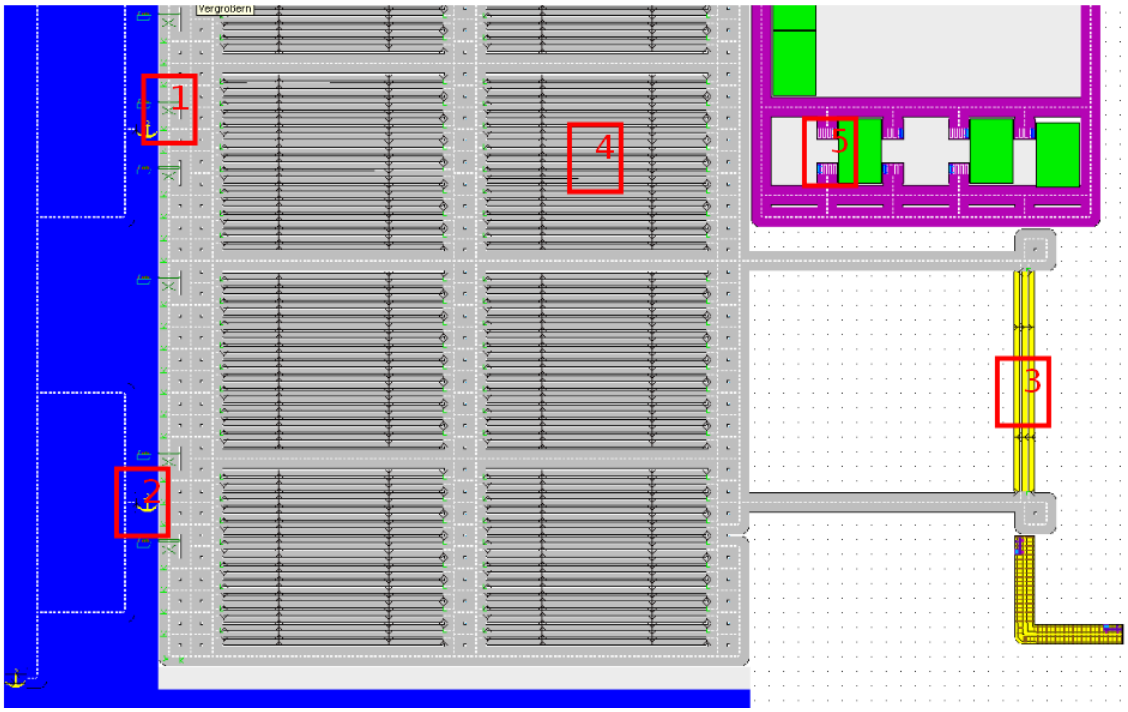


Abbildung 5.19: Altes eM-Plant Modell

1. Im alten Modell sind die Containerbrücken feste Bestandteile der Netzwerke *katze_dock_?*. D.h. sie sind nur gezeichnet und können sich aus diesem Grund auch nicht bewegen oder sonst irgendwie visuell agieren. Damit sich die Containerbrücken bewegen können, ist es notwendig, diese als eigenständige Fahrzeuge zu implementieren. Im Abschnitt Fahrzeuge wird darauf weiter eingegangen. An dieser Stelle soll lediglich festgelegt werden, dass die Netzwerke *katze_dock_?* nun durch die einfachen Netzwerke *vc_kreuzung_katze_?* ersetzt werden. Letzteres muss jedoch erweitert werden. Es muss ein Gleis für die Containerbrücken eingerichtet werden.

2. Dieser Punkt ist ähnlich dem Vorherigen. Schiffe können nur auf einigen wenigen dafür vorgesehenen Punkten am Kai anlegen. Dies ermöglicht jedoch keine korrekte *ShipToShore* und muss daher geändert werden. Es müssen neue Wasserwege

eingeführt werden, die den gesamten Kai abdecken. Ein Schiff muss in der Lage sein, an jedem Punkt des Kais anlegen zu können. D.h. sie müssen auch an jedem Punkt des Kais entladen werden können. Dafür werden Methoden benötigt, die das korrekte Entladen über die gesamte Länge des Schiffs ermöglichen.

3. Der alte Bahnhof entspricht nicht unseren Anforderungen und muss daher entfernt und neu implementiert werden. Er soll aus mehreren Gleisen bestehen und eine Zufahrtsstraße besitzen. Zum Be- und Entladen müssen noch Transtainer implementiert werden. Außerdem müssen noch Gleise erstellt werden, auf denen sich die Züge bewegen können. Es ist geplant, dass es zwei Gleise geben soll, auf denen sich die Züge vor und zurück bewegen können. Dabei müssen Methoden geschrieben werden, die eine Kollision durch Sperren des jeweiligen Gleises vermeiden. Eine spezifischere Beschreibung erfolgt im späteren Verlauf dieser Ausarbeitung.

4. Die Lager müssen um einen Mechanismus erweitert werden, der es erlaubt ein Lager vor einem Simulationslauf als Lager mit Stromanschlüssen oder als ein Lager nur für Gefahr-Transportgut zu bestimmen. Des Weiteren müssen die Methoden für das An- und Ablagern von Containern in dem Lager überarbeitet werden. Bisher wird keine Angabe für einen speziellen Ablagerungsort im Lager ermöglicht. Dies ist insbesondere für eine vernünftige Planung zwingend erforderlich.

5. Die Gebäude müssen angepasst werden. Diese haben einen falschen Standort in der Gesamtstruktur des Hafens. Außerdem werden noch weitere Gebäude eingefügt.

5.4.1.2 Fahrzeuge

Um in dem Simulationslauf eine funktionsfähige Logistik einzurichten, ist es notwendig, verschiedene Fahrzeuge zu implementieren. In eM-Plant sind Fahrzeuge bewegliche Objekte. Sie können sich auf dafür vorgesehenen Baustein, wie z.B. einem *Track*, bewegen. Die verschiedenen Fahrzeuge werden alle abgeleitet von dem eM-Plant Objekt *Fahrzeug*, welches wiederum aus einer fest vorgegebenen Menge von Attributen und Methoden besteht. Um einen neuen Fahrzeugtyp zu definieren, müssen diese Methoden und Attribute dementsprechend erweitert bzw. verändert werden. Im Nachfolgenden soll der Entwurf der benötigten Fahrzeugtypen kurz skizziert werden, dabei soll dessen Steuerung bzw. Wegfindung jedoch vorerst beiseite gelassen werden und erst im Abschnitt Steuerung näher erläutert werden.

- **Containerbrücken**
  Abbildung 5.20 zeigt das geplante aussehen der Containerbrücken. Diese dürfen sich wie schon in den Anforderungen festgelegt nur entlang des Kais bewegen. Um dies zu realisieren, muss ein Schienennetz aus *Tracks* angelegt werden, auf denen sich nur die Brücken bewegen dürfen. Kollisionen müssen hierbei nicht betrachtet

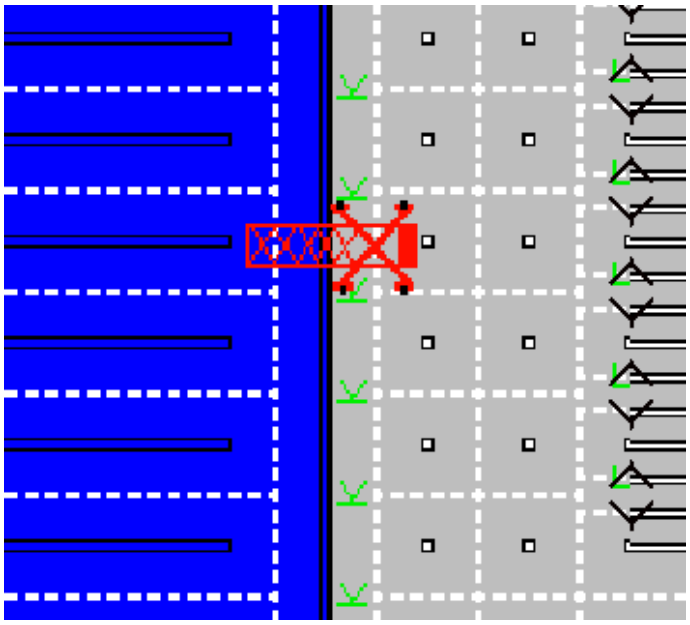


Abbildung 5.20: Containerbrücke

werden, da eM-Plant diese automatisch verhindert. Das Ent- und Beladen soll zum größten Teil von vTOS übernommen werden. Die Simulation soll lediglich einen Auftrag bekommen, in dem eine feste Container ID und die jeweilige Handlung steht. Diese Container ID muss identisch mit einem entsprechenden Eintrag in der Datenbank sein. Ansonsten würde es bei der Visualisierung zu einem Fehler kommen. Die Handlung soll angeben, ob es sich um einen Ent- oder Beladevorgang handelt. Beim Beladen wird der entsprechende Container auf das jeweilige Schiff in die Datenbank eingetragen und in eM-Plant gelöscht. Wobei der Container durch Van Carrier angeliefert werden muss. Der Entladevorgang soll gegensätzlich arbeiten, d.h. der Container wird in eM-Plant neu erstellt und wird in ein Lager auf dem Kai übertragen.
Noch zu erwähnen ist, dass sämtliche Bewegungen und Vorgänge in die Datenbank geschrieben werden müssen.

- **Van Carrier**
  Die Van Carrier haben die Aufgabe, Container zu deren Bestimmungsorten zu transportieren. Um dies zu realisieren, benötigt das Modell ein großes Straßennetz. Dieses kann jedoch aus dem altem Modell komplett übernommen werden. Es sind lediglich einige kleine Änderungen geplant. Abbildung 5.21 zeigt beispielhaft wie zwei Van Carrier aneinander vorbeifahren.
  Die Aufnahme von Containern bzw. Fördergütern stellt in eM-Plant kein Problem dar. Beim Ent- und Beladen muss dies jedoch in die Datenbank geschrieben werden um die korrekte Position der Container zu speichern. Außerdem muss jede Bewegung der Van Carrier in die Datenbank geschrieben werden.

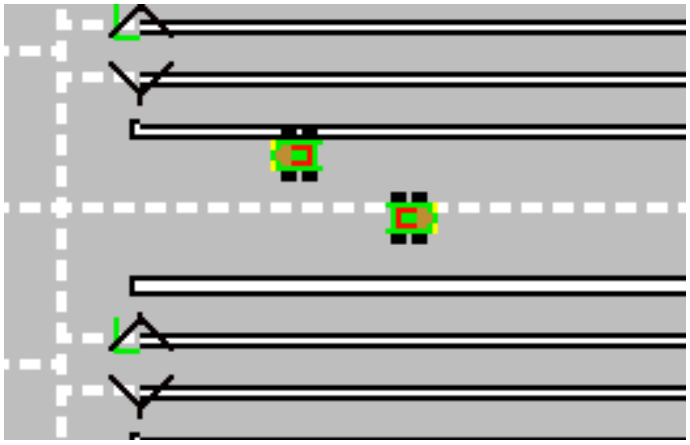


Abbildung 5.21: Van Carrier

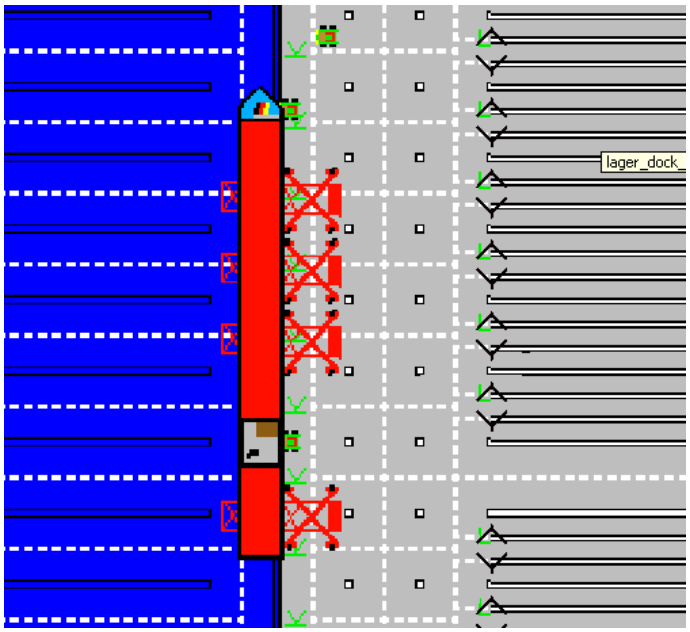


Abbildung 5.22: Containerschiff

- **Containerschiffe**
  Diese Schiffe haben in der Simulation keine große Bedeutung. Ihre einzige Aufgabe besteht darin, die Bewegung für die Visualisierung in die Datenbank zu schreiben. Ihre Beladung spielt in eM-Plant keine Rolle. Die Ladeliste wird vollständig von vTOS verwaltet. Container, die in der Simulation auf- oder abgeladen werden, werden gelöscht oder neu erstellt. Abbildung 5.22 zeigt ein solches Containerschiff in eM-Plant.
  Das Meer besteht aus einer Senke und einer Quelle. Wird ein Schiffsauftrag empfangen, soll an der Quelle ein Schiff erstellt bzw. an der Senke ein Schiff gelöscht werden.

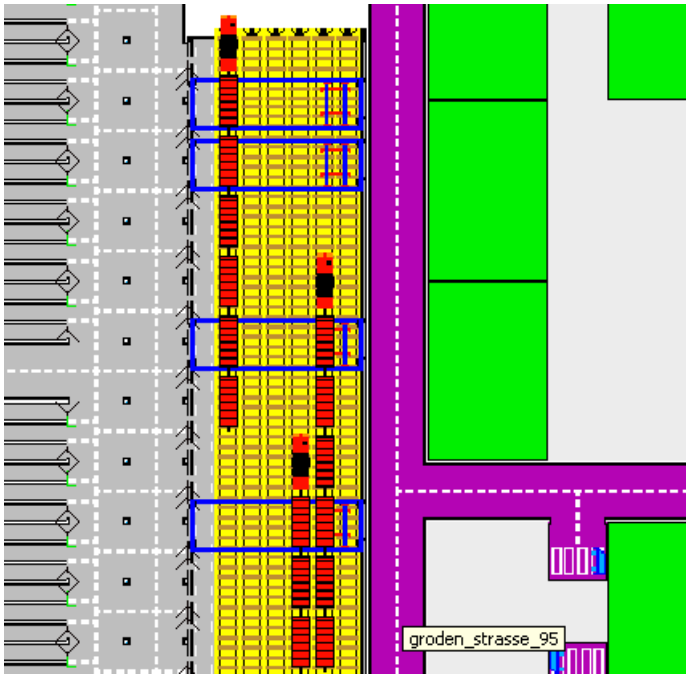


Abbildung 5.23: Bahnhof mit Zügen und Transtainern

- **Transtainer**
  Abbildung 5.23 zeigt den geplanten Güterbahnhof mit einigen Transtainern. Diese Vehikel ähneln stark den Containerbrücken, daher soll hier die Beschreibung der Containerbrücken auch gelten.

- **Züge**
  Abbildung 5.23 zeigt den geplanten Güterbahnhof mit einigen Zügen. Genau wie bei den Containerschiffen ist die Aufgabe der Züge, die Bewegungen in die Datenbank zu schreiben. Genau wie bei den Schiffen sollen Züge eine Ladeliste besitzen, die in der Simulation unberücksichtigt bleibt. Problematisch ist die Verwendung des Schienennetzes, da es zwei Zufahrtswege für die Züge gibt, die aber auch in beide Richtungen verwendet werden können. Kollisionen müssen hier aktiv vermieden werden. Dies soll jedoch im Abschnitt Steuerung näher erläutert werden.

5.4.2 Steuerung der Komponenten

Um den verschiedenen Fahrzeugtypen in eM-Plant Aufgaben zu erteilen, wird eine Steuerung benötigt. Diese Steuerung muss Aufträge annehmen, weitergeben, steuern und überwachen.
Wenn z.B. ein Van Carrier bei Punkt A einen Container aufladen und diesen bei Punkt B abladen soll, findet folgender Ablauf statt. Es wird von vTOS ein Auftrag gesendet. Dieser Auftrag enthält u. a. einen eindeutige Van Carrier ID, einen Abholpunkt A, einen Zielpunkt B und eine eindeutige Container ID. Dieser Auftrag wird dann an das Netzwerk bzw. Klasse *Vancarrierverwaltung* weitergeleitet. Hier wird der Auftrag im richtigen Format an den entsprechenden Van Carrier weitergegeben. Dieser wird den Abholpunkt nun als primäres Ziel definieren und wird einfach losfahren. Bevor er einen Baustein verlässt, der mehr als einen Nachfolgeweg besitzt, soll die Methode Wegberechnung ausgeführt werden. Diese berechnet auf Basis des A Stern Algorithmus den kürzesten Weg und schickt den Van Carrier diesen entlang. Erreicht er sein primäres Ziel, prüft er ob er einen Container beladen soll. Ist dies der Fall, wird der Container beladen und sein primäres Ziel durch den Zielpunkt ersetzt. Der Rest erfolgt analog.
Im Nachfolgenden werden die Steuerungen bzw. Verwaltungen, die für eine korrekte Simulation mit den gegebenen Anforderungen benötigt werden, näher erläutert.

5.4.2.1 Wegberechnung

Die Wegberechnung soll für die verschiedenen Fahrzeuge den kürzesten Weg zwischen einem Start und einem Endpunkt finden. Hierbei ist es von besonderer Bedeutung diesen Algorithmus möglichst einfach zu halten, damit die Geschwindigkeit nicht unnötig leidet, bei einer Simulation mit vielen Fahrzeugen. Da der Hafen quadratisch ist und bis auf die Lager auch nur quadratische Flächen(Straße, Dock, Wasser) enthält, bietet sich eine Wegberechnung nach der Luftlinie an. Dies ist sehr simpel zu realisieren und erfüllt die oben genannten Bedingungen. Eine Wegberechnung, nach A* würde hier zu unnötigem Verbrauch von Rechenkapazitäten führen. Der Algorithmus wird nun anhand des folgenden Pseudocodes erklärt.

- Bestimme x und y vom Ziel
- Bestimme alle Nachbarbausteine, der derzeitigen Position
- Bestimmt jedes x und y von den Nachbarbausteinen
- Anhand des x und y berechne den Abstand zum Ziel
- Wähle den Baustein mit dem geringstem Abstand zum Ziel

Dieser Algorithmus wird bei jeder Kreuzung aufgerufen. Durch die Einfachheit kann ein tieferes Gucken, z.B. nach dem nachfolgendem Nachbarn ersparrt werden. Ein kleines

Beispiel ist in der Abbildung 5.24 dargestellt. In dem erstem Abschnitt ist die aktuelle Position in grün dargestellt, hier ist der Start. Das Ziel ist ein Lager. Nun werden alle benachbarten Bausteine ermittelt, diese sind blau unterlegt (Abschnitt 2). Anschließend werden die Abstände der Bausteine zum Ziel ermittelt. Der Baustein mit dem kürzestem Abstand zum Ziel wird angefahren, dargestellt als blauer Pfeil. Die Abstände sind durch die durchgezogenen Linien dargestellt. Der grüne Pfeil stellt den kürzesten Weg dar, die restlichen sind rot gekennzeichnet, die optimale Route ist somit grün hervorgehoben. Dieser Vorgang wiederholt sich bei dem nachfolgendem Baustein (Abschnitt 3). Die Bausteine, die rausfallen, sind rot durchkreuzt. Anschließend folgt Abschnitt 4, der Zielbaustein ist nun direkt erreichbar, da er ein Nachbarbaustein der aktuellen Position ist, und wird somit direkt angesteuert.

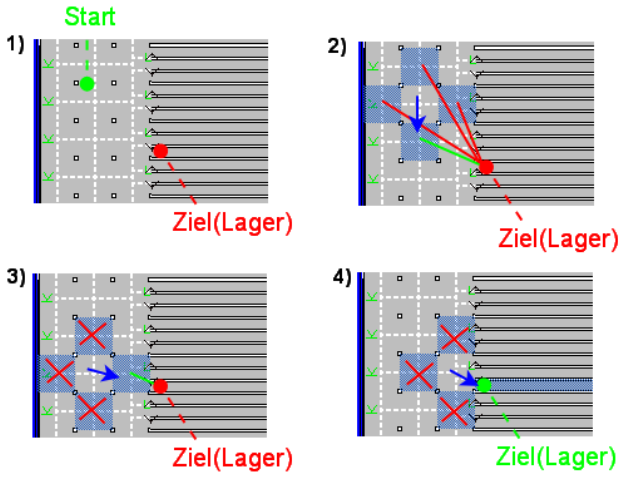


Abbildung 5.24: Beispiel zur Wegberechnung

5.4.2.2 Van Carrier Verwaltung

Die beiden Methoden in den Listings 5.1 und 5.2 zeigen die geplante Funktionsweise der Fahrzeugverwaltung in Pseudocode. Die Methode Fahrzeugerstellung muss von Beginn an ausgeführt werden. Sie erstellt 20 Fahrzeuge und trägt dessen Namen in eine Tabelle ein. Über diese Tabelle können Fahrzeuge direkt angesprochen werden. Außerdem werden die Fahrzeuge mit ihrem Namen, ihrer Source, ihrer Rotation und ihrer Position in die Datenbank geschrieben. Source ist dabei der Metaobjektname mit dem die Visualisierung den Objekten die Modelle zuweist.

```
1 FOR a:=0 TO 20 LOOP;
2     erstelle Fahrzeug;
3     trage Fahrzeugnamen in Tabelle;
4     trage Fahrzeug in die Datenbank;
5 END;
```

Listing 5.1: Methode Fahrzeugerstellung in Pseudocode

Die Methode FahrzeugJobVerteiler soll ausgeführt werden, wenn EmPlant einen Van Carrier Auftrag von vTOS erhält. Dabei wird dann das entsprechende Fahrzeug über die oben genannte Tabelle ausgewählt. Diesem Fahrzeug wird dann ein primäres und sekundäres Ziel zugewiesen. Zusätzlich muss jedem Ziel eine Aktion zugeordnet werden. Aktionen sind in diesem Fall Aufladen, Abladen und Leerfahrt. Auf- und Abladen muss jeweils auch eine genaue Container ID zugeordnet werden.

```
1 IF Auftrag erhalten THEN
2     waehle Fahrzeug;
3     schicke Bestaetigung an vTOS zurueck;
4     trage primaeres Ziel ein;
5     trage primaere Aktion ein;
6     trage sekundaeres Ziel ein;
7     trage sekundaere Aktion ein;
8 END;
```

Listing 5.2: Methode FahrzeugJobVerteiler in Pseudocode

5.4.2.3 Schiffsverwaltung

Für einen korrekten Simulationslauf ist es notwendig, dass vTOS Schiffe in eM-Plant erstellen kann. Das Listing 5.3 zeigt die Methode Schiffserstellung, diese soll über die Sockets angesprochen werden können. Mit dieser Methode können Schiffe in eM-Plant zeitnah erstellt werden. Die Funktionsweise ähnelt der Fahrzeugverwaltung. Als erstes wird ein Schiff erstellt. Diesem wird eine Länge und nur ein Ziel zugeteilt. Ziel ist eine Stelle am Kai an dem das Schiff mittig platziert werden soll. Hiernach folgt ein Datenbankeintrag und der Befehl zum Starten des Schiffs.

```
1 IF Auftrag erhalten THEN
2     erstelle Schiff;
3     trage Laenge ein;
4     trage Ziel ein;
5     trage Schiff in die Datenbank;
6     starte Schiff;
7 END;
```

Listing 5.3: Methode Schiffserstellung in Pseudocode

5.4.2.4 Kranverwaltung

Die Kranverwaltung muss abgesehen von der Kranerstellung, zwischen zwei Arten von Aufträgen unterscheiden, zum einen dem einfachen fahrenden Auftrag, mit dem eine

Containerbrücke bewegt werden kann. Zum anderen den Ladeaufträgen, mit denen die Container auf- bzw. abgeladen werden (vom Containerschiff). Die Listings 5.4, 5.5 und 5.6 zeigen die wichtigsten Methoden in Pseudocode.
Die Kranerstellung basiert auf der Idee, die Kräne gleichmäßig am Kai zu verteilen. Der Kai besteht in eM-Plant aus 98 Bausteinen und es gibt 16 Containerbrücken. Um daher eine möglichst gleichmäßige Verteilung zu erreichen, ist es notwendig alle sechs Bausteine mit einem Kran zu positionieren. Nachdem die Kräne erstellt und auf die richtige Position gesetzt wurden, wird der Name wieder in eine Tabelle eingetragen und der Kran mit den wichtigsten Eckdaten in die Datenbank geschrieben. Diese Eckdaten sind analog zu denen der Van Carrier.

```
1 variable := 0;
2 FOR a:=0 TO 16 LOOP;
3     variable := variable + 6;
4     erstelle Kran;
5     setze Kran an Kaiposition(variable);
6     trage Krannamen in Tabelle;
7     trage Kran in die Datenbank;
8 END;
```

Listing 5.4: Methode Kranerstellung in Pseudocode

Die Kranbewegung ist lediglich eine simple Methode zum Bewegen der Kräne.

```
1 IF Auftrag erhalten THEN
2     waehle Kran;
3     trage Ziel ein;
4     starte Kran;
5 END;
```

Listing 5.5: Methode Kranbewegung in Pseudocode

Die Methode Krancontainer unterscheidet als erstes zwischen Auf- und Abladen. Nach der Unterscheidung wird die jeweilige Aktion ausgeführt und dann entsprechend in die Datenbank, genauer in die Tabelle Queue, geschrieben. Es werden die Spalten id, typ, building_block, vc, action1 und timestamp gefüllt. Typ gibt dabei die Art der Animation an, z.B. 3 für Entladen und 2 für Beladen. VC steht für das jeweilige Fahrzeug. In Action muss die genaue Container ID eingetragen werden und timestamp gibt die genaue Zeit an, wann diese Handlung in der Visualisierung ausgeführt werden soll.

```
1 IF Auftrag erhalten THEN
```

```
2     IF Aufladen THEN
3         Lade Container X auf;
4         Trage Vorgang in die Datenbank;
5     END;
6     IF Abladen THEN
7         Entlade Container X;
8         Trage Vorgang in die Datenbank;
9     END;
10 END;
```

Listing 5.6: Methode Krancontainer in Pseudocode

5.4.2.5 Zugverwaltung

Die Zugverwaltung läuft analog zu der Schiffsverwaltung. Lediglich beim Verteilen der Züge auf ein bestimmtes Schienennetz gibt es einige Unterschiede. Da es zwei Zufahrtsgleise gibt und Züge sich nicht kreuzen können, ist es notwendig die Gleise zu sperren, wenn diese befahren werden.

5.4.2.6 Transtainerverwaltung

Die Transtainerverwaltung muss, abgesehen von der Initialisierung der Transtainer, zwischen zwei Arten von Aufträgen unterscheiden, zum einen dem einfachen fahrenden Auftrag, mit dem ein Transtainer bewegt werden kann. Zum anderen den Ladeaufträgen, mit denen die Container auf- bzw. abgeladen werden. Die Listings 5.7, 5.8 und 5.9 zeigen die wichtigsten Methoden in Pseudocode.
Die Transtainererstellung basiert auf der Idee die Transtainer gleichmäßig am Verladebahnhof zu verteilen. Der Bahnhof besteht in eM-Plant aus 58 Bausteinen und es soll erst einmal mit fünf Transtainern gearbeitet werden. Um daher eine möglichst gleichmäßige Verteilung zu erreichen, ist es notwendig alle elf Bausteine einen Transtainer zu positionieren. Nachdem die Transtainer erstellt und auf die richtige Position gesetzt wurden, wird der Name wieder in eine Tabelle eingetragen und mit den wichtigsten Eckdaten in die Datenbank geschrieben. Diese Eckdaten sind analog zu denen der Van Carrier.

```
1 variable := 0;
2 FOR a:=0 TO 5 LOOP;
3     variable := variable + 11;
4     erstelle Transtainer;
5     setze Transtainer auf Bahnhof(variable);
6     trage Transtainernamen in Tabelle;
7     trage Transtainer in die Datenbank;
8 END;
```

Listing 5.7: Methode Transtainererstellung in Pseudocode

Transtainerbewegung ist lediglich eine simple Methode zum Bewegen der Kräne.

```
1 IF Auftrag erhalten THEN
2     waehle Transtainer;
3     trage Ziel ein;
4     starte Transtainer;
5 END;
```

Listing 5.8: Methode Transtainerbewegung in Pseudocode

Die Methode Transtainercontainer unterscheidet als erstes zwischen Auf- und Abladen. Nach der Unterscheidung wird die jeweilige Aktion ausgeführt und dann entsprechend in die Datenbank, genauer in die Tabelle Queue, geschrieben. Es werden die Spalten id, typ, building_block, vc, action1 und timestamp gefüllt. Die Werte sind ähnlich wie bei der Kranverwaltung. Jedoch sind noch keine Animationen für die Transtainer vorhanden. Es müssen also zwei neue Typen von Animationen erstellt werden, um diese in der Visualisierung korrekt darstellen zu können.

```
1 IF Auftrag erhalten THEN
2     IF Aufladen THEN
3         Lade Container X auf;
4         Trage Vorgang in die Datenbank;
5     END;
6     IF Abladen THEN
7         Entlade Container X;
8         Trage Vorgang in die Datenbank;
9     END;
10 END;
```

Listing 5.9: Methode Transtainercontainer in Pseudocode

5.4.3 Kommunikation

Die Kommunikation teilt sich in eM-Plant in zwei Bereiche. Zum einen die Kommunikation mit der Visualisierung über die Datenbank und zum anderen die mit vTOS über die Sockets. Im Folgenden wird ein grober Entwurf beider Arten vorgestellt.

5.4.3.1 Kommunikation Sockets

Nachdem eine Verbindung mit vTOS erstellt wurde, soll diese gehalten werden und die Befehle in einfache Strings kodiert werden. Dabei ist es wichtig, dass die Nachrichten schnell verschickt und bearbeitet werden. Um dies zu gewährleisten, sind zwei separate Kanäle geplant. Unterteilt in eingehende und ausgehende Nachrichten. Jede Nachricht soll genau eine Anweisung für eM-Plant bzw. eine Rückmeldung für vTOS enthalten. Dadurch wird es zu einem erhöhten Nachrichtenverkehr kommen. In diesem Abschnitt sollen kurz die wichtigsten Methoden, auf Seiten eM-Plants, skizziert werden.

Nachrichteneingang Zum Abarbeiten der Aufträge soll die Methode Nachrichteneingang dienen, welche in Listing 5.10 durch einfachen Pseudocode beschrieben wird. Wenn eine Nachricht ankommt, soll diese als erstes auf dessen Richtigkeit geprüft werden. Welche Mechanismen dazu zum Einsatz kommen sollen, ist an dieser Stelle noch nicht klar. Wurde ein Fehler erkannt, wird eine Fehlermeldung zurückgeschickt und damit ein erneutes Senden der Nachricht erzwungen. Ist die Nachricht richtig, wird ein OK mit einer entsprechenden Nachrichten ID zurückgeschickt, damit der Sender weiß, dass die Nachricht angekommen ist. eM-Plant kann nun den String entsprechend dekodieren und an die entsprechende Verwaltung weiterleiten. Die Kodierung der Nachrichten wird am Ende dieses Abschnitts erläutert.

```
1  IF Nachricht erhalten THEN
2      pruefe Nachricht;
3      IF Fehler THEN
4          schicke Fehlermeldung;
5          BREAK;
6      ELSE
7          schicke OK mit Nachrichten ID;
8      END;
9      dekodiere Nachricht;
10     gebe Nachricht weiter;
11 END;
```

Listing 5.10: Methode Nachrichteneingang in Pseudocode

Nachrichtenausgang

Damit vTOS den Stand der Simulation jederzeit kennt, ist es notwendig Statusmeldungen bei der Bearbeitung der Aufträge zurückzuschicken. Ein gewöhnlicher Auftrag für ein VC, zum Abholen und Abliefern eines Containers, soll folgende Rückmeldungen verschicken:

- Auftrag richtig angekommen.

- VC mit entsprechender Nummer fährt los.
- VC hat entsprechenden Container aufgeladen und fährt weiter.
- VC hat Ziel erreicht; Container abgeladen.

Nach dem oberen Muster werden sämtliche Aufträge bearbeitet. Listing 5.11 zeigt die Methode Nachrichtenausgang in Pseudocode. Diese Methode dient dazu, Nachrichten zu verschicken und gleichzeitig für die korrekte Übertragung zu sorgen. Korrekte Übertragung heißt, dass es zu keiner Doppelbelegung des Ausgangssockets kommt und dass die Strings fehlerfrei verschickt werden. Auch wenn UDP und TCP dies z. T. unterstützen, ist dies notwendig, da nicht bekannt ist, wie eM-Plant damit umgeht.
Damit es zu keiner Doppelbelegung kommt, soll jeder Auftrag durch die Variable „belegt“ prüfen, ob die Leitung frei ist. Ist dies der Fall, kann dies mit entsprechendem Setzen der Variable „belegt“ gesendet werden. Das fehlerfreie Verschicken der Aufträge wird dadurch erreicht, dass der Empfänger die Nachricht auf Korrektheit prüft und dann ein „OK“ versendet. Kommt es hier zu einem Fehler, wird die Nachricht einfach wieder verschickt.

```
1  Globale Variable belegt := FALSE;
2  Variable  nachrichtverschickt := FALSE;
3  IF Nachricht zu verschicken THEN
4      WHILE belegt THEN
5          warte kurz;
6      END;
7      belegt := TRUE;
8      WHILE !nachrichtverschickt THEN
9          verschicke Nachricht;
10         IF Antwort = OK THEN
11             nachrichtverschickt := TRUE;
12         END;
13     END;
14     belegt := FALSE;
15 END;
```

Listing 5.11: Methode Nachrichtenausgang in Pseudocode

5.4.3.2 Kommunikation Datenbank

Die Kommunikation mit der Datenbank soll zum größten Teil von der Vorgruppe übernommen werden. Da in der Visualisierung die Tabellen der Vorgruppe verwendet werden, reichen die alten Datenbankverbindungen von eM-Plant vollkommen aus. Lediglich neue Einträge für Züge und Schiffe müssen nach dem alten Muster erstellt werden. In den folgenden EER-Diagrammen werden die hierfür genutzten Tabellen dargestellt.

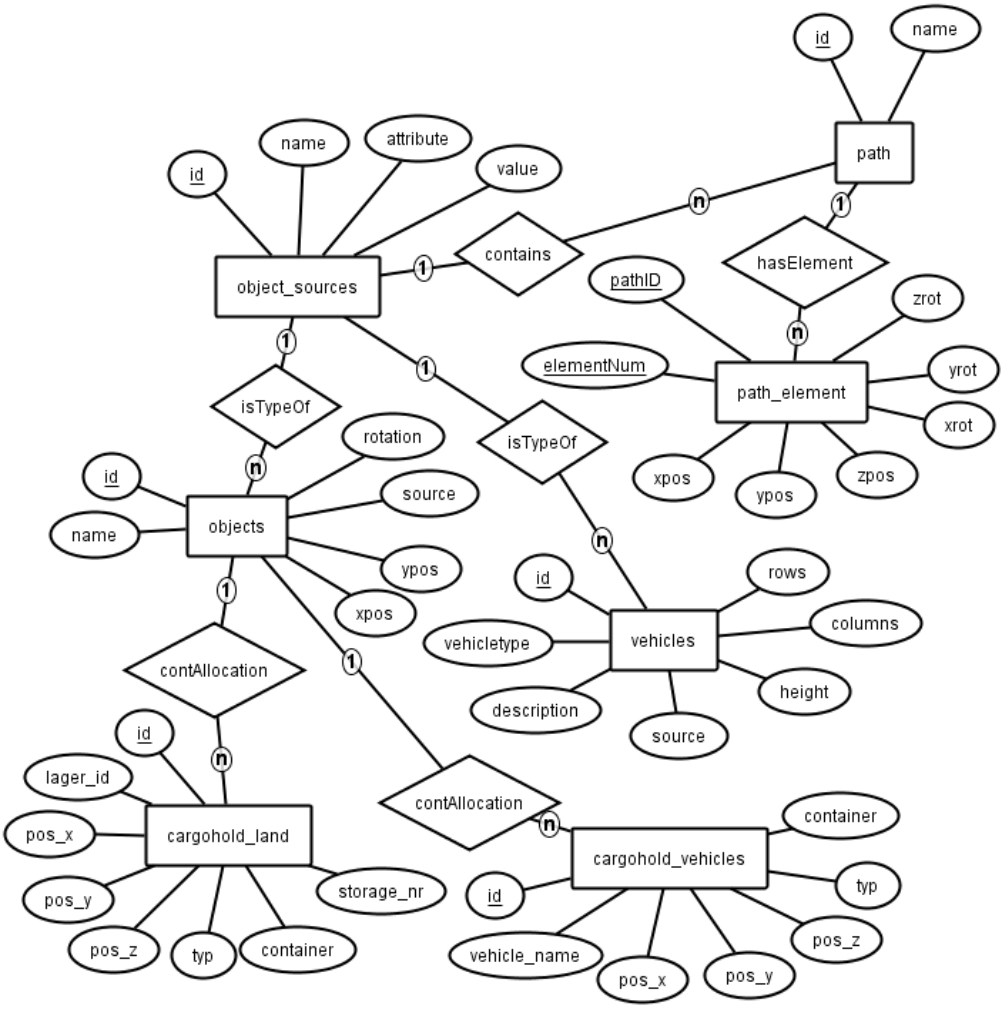


Abbildung 5.25: Die Objectdefinition für die 3D Visualisierung.

object_sources			
id	name	attribute	value

1. **id** Die ID des Datensatzes
2. **name** Name des beschriebenen Objektes
3. **attribute** Attributname, welches beschrieben werden soll.
4. **value** Wert des angebebenen Attributs

path	
id	name

1. **id** Die ID des Datensatzes

2. **name** Name des Pfades

path_elememt							
pathID	elementNum	xpos	ypos	zpos	xrot	yrot	zrot

1. **pathID** Die ID aus der path Tabelle
2. **elementNum** Fortlaufende Nummer für eine pathID
3. **xpos, ypos, zpos** Die Position auf dem Pfad.
4. **xrot, yrot, zrot** Die Rotation auf dem Pfadelement.

objects					
id	name	rotation	source	xpos	ypos

1. **id** Die ID des Datensatzes
2. **name** Name des Objekts
3. **source** Der Name des source Objekts aus der Tabelle object_sources.
4. **rotation** Die initiale Rotation des Objekts.
5. **ypos, xpos** Position des Objekts in der Simulation.

cargohold_land							
id	lager_id	pos_x	pos_y	pos_z	typ	container	storage_nr

1. **id** Die ID des Datensatzes
2. **lager_id** Name des Lagers aus der Tabelle objects.
3. **pos_x. pos_y, pos_z** Position im Lager.
4. **typ** Beschreibt den Typ des Containers.
5. **container** Die ID des Containers.
6. **storage_nr** Die Lagernummer aus der Planungsdatenbank.

cargohold_verhicles						
id	vehicle_name	pos_x	pos_y	pos_z	typ	container

1. **id** Die ID des Datensatzes
2. **vehicle_name** Name des Vehikel aus der Tabelle objects.

3. **pos_x. pos_y, pos_z** Position auf dem Vehikel.
4. **typ** Beschreibt den Typ des Containers.
5. **container** Die ID des Containers.

vehicles						
id	vehicletype	description	source	height	rows	columns

1. **id** Die ID des Datensatzes
2. **vehicletype** Der Typname des Vehikels. Z.B. LKW.
3. **description** Beschreibung des Vehikels.
4. **source** Die source Beschreibung aus der Tabelle object_sources.
5. **height, rows, columns** Beschreibt die Lagergröße des Vehikels.

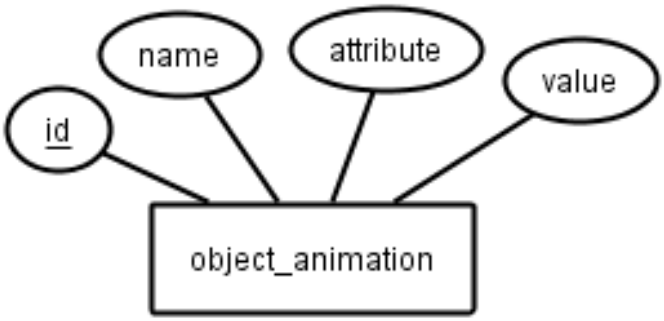


Abbildung 5.26: Die Animationen der Objekte

object_animation			
id	name	attribute	value

1. **id** Die ID der Animation
2. **name** Name der Animation.
3. **attribute** Attribut des Datensatzes
4. **value** Klassenname der Animation.

5.5 Visualisierung

Zusammenfassung

Dieser Teil des Entwurfs befasst sich mit der Visualisierungskomponente. Dabei wird zunächst die Umsetzung des 3D-Effektes über die Software und Hardware (vgl. 5.27) erläutert. Anschließend wird kurz auf das benutzte Modellierungstool, sowie auf die beiden für „Virtual Port II“ geschriebenen Programme zur Modellierung eingegangen. Die genaue Erklärung, angefangen von der Objekterstellung bis hin zum Einpflegen in die 3D-Umgebung, wird im Handbuch beschrieben. Ferner werden die einzelnen Packages der 3D Engine beschrieben und spezifiziert.

5.5.1 Stereoscopic View

Für die stereoskopische Ausgabe der 3D-Simulation des Hafens ist ein entsprechend eingerichtetes Doppelbeamersystem der Firma 3Dims vorgesehen. Dieses System basiert auf zwei Beamern, die so ausgerichtet sind, dass ihre Projektionen kohärent übereinanderliegen. Eine optische Separation der Bilder erfolgt durch entsprechend vorgeschaltete Polarisationsfilter und mit Hilfe entsprechender Betrachtungsbrillen.

Abbildung 5.27: Doppelbeamersystem

Um nun der Visualisierung durch Steroskopie Tiefeninformationen und somit Dreidimensionalität zu verleihen, ist es notwendig, die Szenerie aus zwei leicht versetzten Perspektiven zu betrachten.

Hierfür wird die gewählte 3D-Engine "Xith3D" angepasst, da sie originär nicht für diese Aufgabe konzipiert ist. Nach dieser Anpassung ist die Engine in der Lage, zwei Viewports ("Kameras") parallel anzuzeigen. Beide Viewports werden nebeneinander in einem Canvas dargestellt. Dieser Canvas kann auf dem Rechner, der an das 3D-Beamersystem angeschlossen ist, so angeordnet werden, dass jeder Beamer genau einen Viewport darstellt und somit eine optische Separierung durch die Polarfilter erfolgt.

Diese Viewports sind horizontal leicht versetzt angeordnet und leicht eingedreht, so dass sich beide Sichtachsen in einem entfernten Punkt in der weiteren Umgebung treffen bzw.

schneiden. Sie stellen also eine Referenz der Augen des Betrachters bzw. der Betrachter dar. Die Art der Viewport-Anordnung wird als "Kreuzblick" bezeichnet und erzeugt beim Betrachter ohne große Mühe den Eindruck von Räumlichkeit.

5.5.2 Blender

Das Modellierungstool der Wahl für VPort ist das Open Source Produkt Blender. Dies wurde aufgrund der kostenlosen Verfügbarkeit für mehrere Plattformen und aufgrund seiner Mächtigkeit gewählt.
Blender ist unter der Website http://www.blender.org frei herunterladbar und auf dieser Seite finden sich auch ausführliche Einführungen in dieses Modellierungstool. Mit Blender wurden alle Objekte für die Visualisierung erstellt.

5.5.3 Level of Detail

Beim Erstellen der Modelle für VPort wurden die meisten Modelle in drei Detailstufen (LOD: Level of Detail) erstellt. Die Tabelle 5.1 gibt einen Überblick über die verschiedenen LOD.

Einstellungsmöglichkeiten Grafik:

LOD	Sichtweite	Anzahl Faces am Beispiel Eisenbahnwaggon
1	200m - 390m	280
2	100m - 200m	1.118
3	0m - 100m	5.038

Tabelle 5.1: Level of Detail

LOD 3 steht also für den höchsten Detail-Grad und ist bis zu einer Entfernung von 100m sichtbar. Von 100m bis 200m ist LOD 2 zu sehen und danach LOD 3, welches die wenigsten Details zeigt. Die Anzahl der Faces bezeichnet hier die den Grad der Detaillierung. Ein Face wird von 3-4 Punkten begrenzt und stellt eine Fläche eines Modells dar. Als Beispiel für die unterschiedlichen LOD ist in Abbildung 5.28 ein Eisenbahnwaggon aus VPort in allen drei Detailstufen zu sehen.

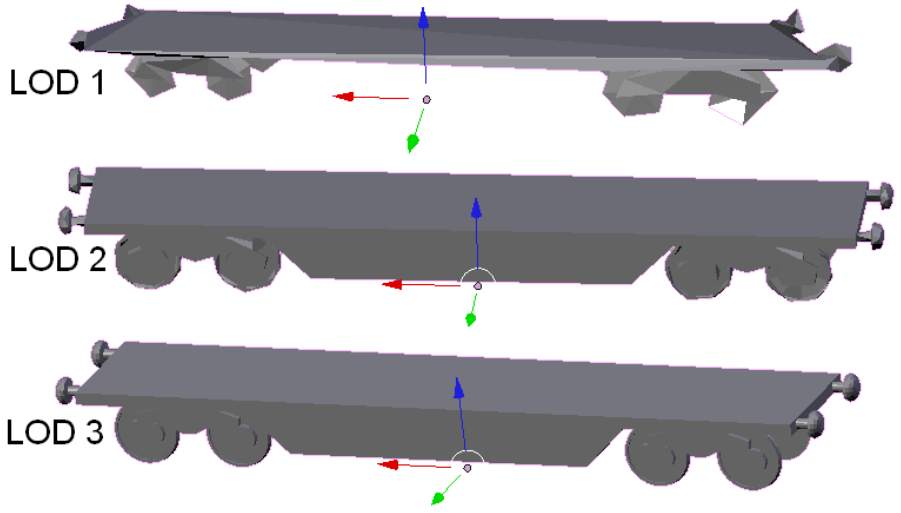


Abbildung 5.28: LOD am Beispiel des Eisenbahnwaggon.

In LOD 1 besitzt der Waggon die wenigsten Details und in LOD 3 die meisten.

5.5.4 Skalierung

Um die Modelle maßstabsgetreu zu erstellen, wurde sich auf einen Maßstab geeinigt, der die Blendereinheiten in Meter umrechnet. Der Maßstab 1m = 2 BE bedeutet, dass man

einem Würfel die Kantenlänge von zwei Blendereinheiten geben muss, damit er in VPort ein Volumen von einem Kubikmeter hat. Dieser Maßstab von 1m = 2 BE wurde für die Modellierung festgelegt.

5.5.5 Vertiefung

Ein ausführliche Beschreibung der Modellierung der 3D-Objekte behandelt das "VPort - Handbuch". In diesem wird genauer auf die beiden für VPort geschriebenen Programme zur Modellierung eingegangen. Außerdem liefert es eine genaue Anleitung für die 3D-Objekte; angefangen von der Objekterstellung bis hin zum Einpflegen in die 3D Umgebung.

5.5.6 Klassenbeschreibung

5.5.7 Package Übersicht

Die folgenden Packages befinden sich in unserer Engine und werden in den späteren Abschnitten genauer betrachtet, sofern dies nötig ist.

- **vintage**
  Die Klassen in diesem Package sind für die Initialisierung und Speicherung der aktuell verwendeten Daten zuständig.

- **vintage.db**
  Dieses Package beinhaltet eine einfache JDBC Datenbank-Schnittstelle, die während der Initialisierung genutzt wird.

- **vintage.engine**
  Klassen, die als Schnittstelle zur Xith3D-Engine verwendet werden.

- **vintage.engine.animation**
  Übernimmt die Steuerung und Erstellung von Animationen, die aus der Datenbank gelesen werden.

- **vintage.engine.animation.path**
  Spezielle Animationen, die den IPO Kurven aus Blender nachempfunden sind.

- **vintage.engine.animation.task**
  AnimationTasks sind eine Beschreibung, wie die Einträge aus der Datenbank zu interpretieren sind.

- **vintage.engine.camera**
  Steuerung der Bewegungen des Benutzers.

- **vintage.engine.envirement**
  In diesem Package werden nichtfunktionale Objekte wie Licht, Nebel, Skybox, usw. verwaltet.

- **vintage.engine.gui (vintage.engine.gui.provider)**
  Ist für das Erstellen eines Informationsfensters innerhalb der GUI zuständig.

- **vintage.engine.hud (vintage.engine.hud.commands)**
  Das HUD ist für das Darstellen von Fenstern innerhalb der 3D Welt zuständig. Zusätzlich wird eine Konsole erstellt, über die Kommandos an die Simulation übergeben werden können.

- **vintage.engine.model**
  Enthält die verwendeten Modellloader für die 3D-Modelle, sowie die Repräsentation des eigentlichen Modells.

- **vintage.engine.partion (vintage.engine.partion.bst)**
  Erstellung eines optimierenden Scenegraphen.

- **vintage.engine.scenegraph**
  Die Klassen dieses Packages bauen während der Initialisierung einen Szenengraphen auf und halten diesen während der Laufzeit auf dem aktuellen Stand.

- **vintage.events**
  Ein Event ist eine interne Aktion, die beispielsweise für das Infofenster selektierter Objekte zuständig ist.

- **vintage.input**
  Fängt die Eingabe des Benutzers ab und kann diese in Events umwandeln.

- **vintage.log**
  Initialisiert den Logger.

- **vintage.logic**
  Enthält hauptsächlich abstrakte Klassen und Interfaces, die das Verhalten von Objekten steuern.

- **vintage.objects(und Subpackages)**
  Enthalten die Implementierungen der einzelnen Objekte, wie Vancarrier, Straße, Container, Züge, usw.

- **vintage.utils**
  Hilfefunktionen, die häufiger verwendet werden.

- **vintage.utils.modeleditor**
  Eine Modelleditor GUI zum Erstellen von binären .vpm Dateien.

- **vintage.utils.patheditor**
  Eine GUI zum Erstellen von Animationspfaden, wie sie vom Package "vintage.engine.animation.path" benötigt werden.

- **vintage.utils.slotCreation**
  Tool zum Positionieren von Containern auf den Lagern und Schiffen.

- **vintage.utils.texturepack**
  Grafisches Tool zum Erstellen von alternativen Materialien für .obj Dateien.

- **vintage.utils.timer**
  Timer, dem ein Objekt übergeben wird, und dieses einmalig zu einem gegebenen Zeitpunkt an einen TimeListener weitergibt.

- **vintage.utils.vpeviewer**
  Tool zum Exportieren der Simulation in ein binäres Dateiformat, für das es zum einen ein Blender-Import-Skript gibt, zum anderen eine abgespeckte Version der Engine, die dieses Format interpretieren und darstellen kann.

5.5.8 Package Beschreibungen

5.5.8.1 vintage.engine

Die Engine ist die Schnittstelle zur verwendeten Xith3D-Engine und enthält mit der Klasse „Scene" die Klasse, die zur Initialisierung und Steuerung der Xith-Engine genutzt wird. Sie erbt von der Klasse „InputAdapterRenderLoop", welche zusätzlich einen eingebauten InputManager zur Verfügung stellt. Die Hauptaufgabe der Scene ist es, den Scenegraphen der „Xith3DEnvirement" bekannt zu machen. „VPCanvasConstructionInfo" ist eine Klasse zum Speichern der Einstellungen, mit denen das 3D Fenster erstellt wird. Hier werden unter anderem die Auflösung, die Bittiefe und ähnliche Einstellungen gespeichert. Aus diesen wird der Xith -„View" erstellt, welcher eine abstrakte Kamera darstellt. Eine

von uns hinzugefügte Erweiterung der Xith Engine ist die Klasse „StereoView“, die die Darstellung von echtem 3D mittels zweier Stereo-Beamer ermöglicht. Für eine genauere Beschreibung der Theorie bitte den entsprechenden Absatz über Stereoskopie (siehe 5.5.1) betrachten. Hier sei nur erwähnt, dass beim Stereomodus das Bild zweimal aus verschiedenen Perspektiven gerendert wird. Um dies zu bewerkstelligen, wird die Position bei jedem Durchlauf jeweils auf der lokalen X-Achse nach links und rechts verschoben. Der „PickingManager“ ermöglicht das Selektieren von Objekten in der Simulation, die das Interface Pickable implementieren.

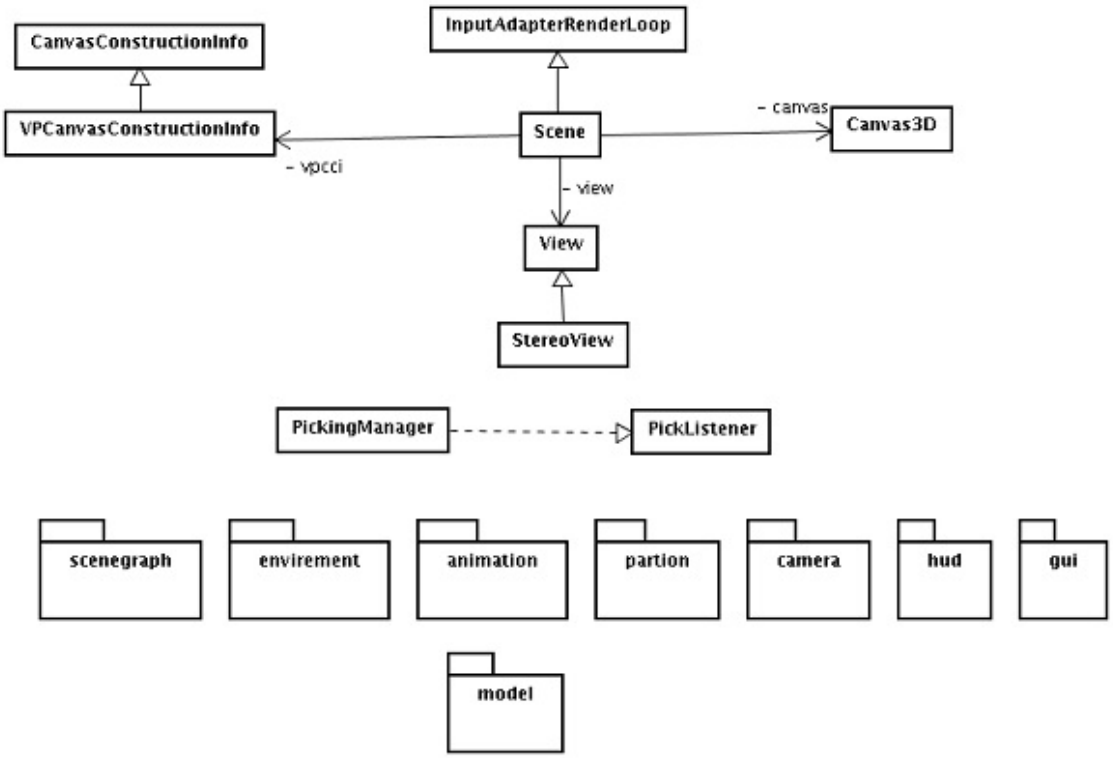


Abbildung 5.29: Klassendiagramm vintage.engine

5.5.8.2 vintage.engine.animation

Das Package vintage.engine.animation übernimmt die Initialisierung und Steuerung aller Animationen der Simulation. Die dafür zuständige Klasse ist der “AnimationManager“, der im Abschnitt Manager auf Seite 123 genauer beschrieben ist. Die zweite wichtige Klasse in diesem Package ist “AbstractAnimation“, von der alle Animationen der Engine erben müssen. Eine Animation wird, solange sie “lebendig“ ist, in jedem Frame einmal von der Xith-Klasse “Animator“ bzw. “OperationScheduler“ aufgerufen und hat die Möglichkeit, ein oder mehrere VportObjects zu manipulieren. Häufig ist es günstig, eine komplexe Animation in kleinere Teilanimationen zu zerlegen, zu diesem Zweck wurden die Animationen “ParallelAnimation“ und “SequenzAnimation“ eingeführt.

ParallelAnimation

Eine ParallelAnimation kann beliebig viele AbstractAnimations gleichzeitig ausführen. Die Animation ist solange "am Leben" wie die Hauptanimation, also die Animation die als erstes eingefügt wurde.

SequenzAnimation

Eine Sequenzanimation führt, wie der Name schon sagt, eine Folge von Animationen sequenziell aus, das heißt, die nächste Animation in der Liste wird dann ausgeführt, wenn die vorhergehende beendet wurde. Die Sequenzanimation ist solange "am Leben" bis alle Animationen abgearbeitet worden sind.

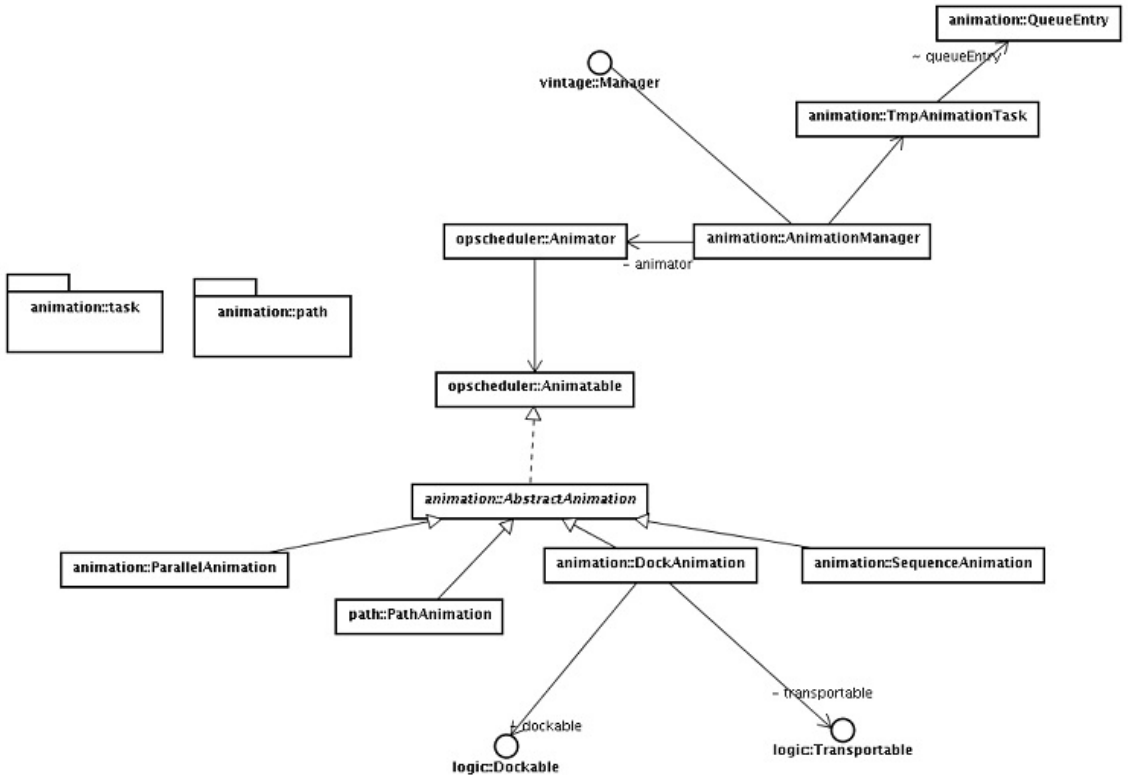


Abbildung 5.30: Klassendiagramm vintage.engine.animation

5.5.8.3 vintage.engine.animation.path

Im Package vintage.engine.animation.path befindet sich eine spezielle, aber häufig verwendete, Animationsart, die Pfadanimation. Eine Pfadanimation entspricht dabei der einer "PosRot"-IPO-Kurve aus Blender. Die Klassen DropDownAnimation und PickupAnimation wiederum sind Erweiterungen einer Pfadanimation, bei der am Ende der Animation ein "Transportable" Objekt an ein "Dockable" Objekt angedockt, bzw. die Verbindung zwischen ihnen gelöst wird. (zu "Transportable" und "Dockable" bitte im Abschnitt vintage.logic ab Seite **??** nachlesen) Die Grundlage zu dieser Animation bildet die Klasse "Path", die verschiedene Stützpunkte beinhaltet. In diesem Fall besteht ein Stützpunkt aus einer Positionsangabe und einer Rotation an der entsprechenden Position. Mit

Hilfe des "Interpolators" wird zwischen den Stützpunkten interpoliert um eine möglichst fein abgestufte Animation zu erhalten. Zur Verfügung stehen folgende Interpolatoren:

1. **LinearInterpolator:**
   Der LinearInterpolator interpoliert, wie der Name sagt, mittels einer linearen Funktion zwischen den übergebenen Stützpunkten.

2. **BezierCurve:**
   Eine Bezier Kurve ist eine mathematische Vorschrift, um zwischen zwei oder mehr Kontrollpunkten zu interpolieren. Eine Bezierkurve bestehend aus zwei Punkten, entspricht dabei einer linearen Interpolation. Bei mehr als zwei Punkten wird dann eine Kurve errechnet. Der Vorteil von Bezierkurven ist die glatte Interpolation von Kurven, die auch bei großer Skalierung "glatt" bleiben. Durch die komplexe mathematische Funktion wird allerdings im Vergleich zur linearen Interpolation sehr viel mehr Rechenleistung benötigt.

Da es teilweise leichter ist, einen Pfad erst kurz vor Start der Animation zu berechnen, wurde das Interface "PathCalculator" eingeführt. Dies ist der Fall, wenn verschiedene Animationen hintereinander ausgeführt werden sollen und sich die Position des Objektes verändert. Um dieses Interface zu nutzen, werden die Klassen DynamicPathAnimation, DynamicDropdownAnimation und DynamicPickupAnimation verwendet. Diese rufen die Methode "getPath()" des Interfaces auf, sobald die Animation gestartet wird.

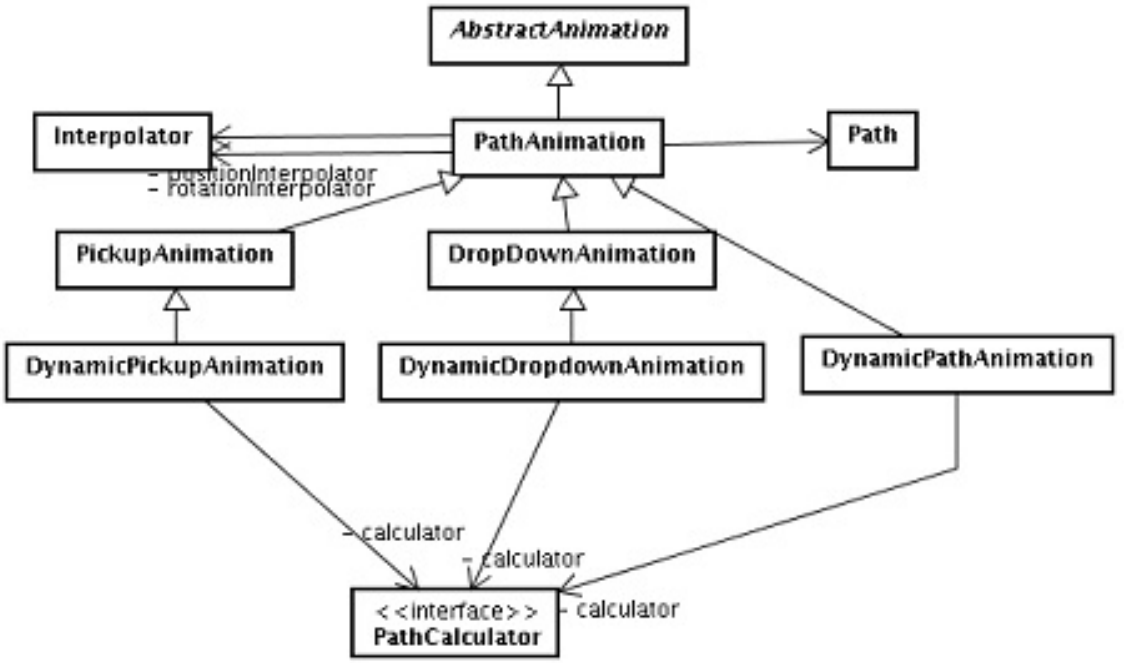


Abbildung 5.31: Klassendiagramm vintage.engine.animation.path

5.5.8.4 vintage.engine.animation.task

Ein Animationstask bildet die Vorschrift, wie die Daten aus der Tabelle "queue" zu interpretieren sind und wird genutzt um eine Animation zu erstellen, für genauere Informationen bitte im Abschnitt Schnittstellen nachlesen.

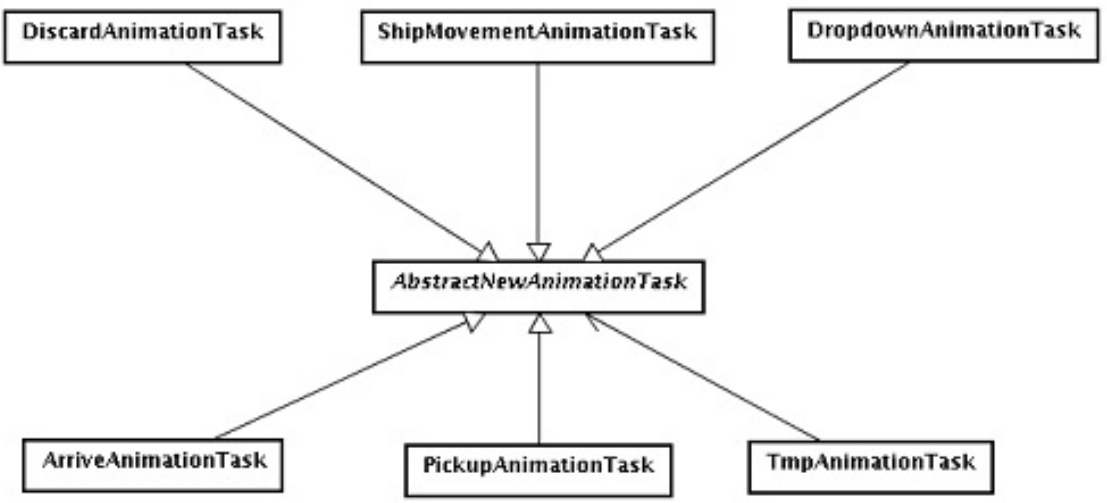


Abbildung 5.32: Klassendiagramm vintage.engine.animation.task

5.5.8.5 vintage.engine.camera

Das Package camera beinhaltet die Steuerung der Kamera und steuert damit die Bewegung des Benutzers durch die 3D Welt. Durch die Verwendung der in Xith integrierten Klasse "FirstPersonInputHandler" muss keine eigene Kamera implementiert werden, sondern nur die entsprechenden Modi umgesetzt werden.

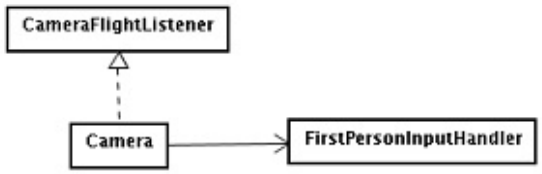


Abbildung 5.33: Klassendiagramm vintage.engine.camera

5.5.8.6 vintage.engine.envirement

In diesem Package werden die nichtfunktionalen Aspekte der 3D Welt behandelt; dazu gehören momentan das globale Licht und der Nebel, der ein sanftes Ausklingen von Objekte am Ende des Sichtbereiches ermöglicht. Da Xith, in der verwendeten Version, keine unterschiedlichen Farbwerte für ambientes, diffuses und speculares Licht ermöglichte, mussten wir die Engine an dieser Stelle um die gewünschten Eigenschaften erweitern. Außerdem werden hier die Skybox und die Hintergrundgeräusche erstellt und verwaltet.

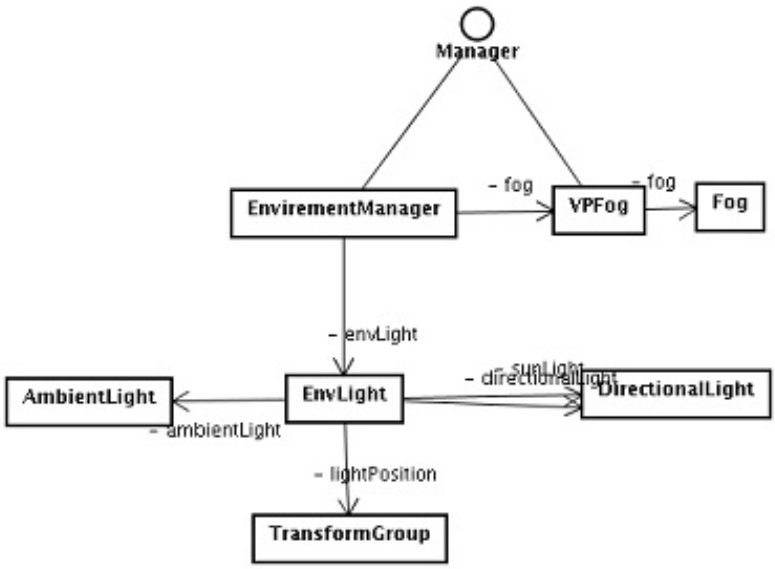


Abbildung 5.34: Klassendiagramm vintage.engine.envirenment

5.5.8.7 vintage.engine.gui

Um während der Simulation Informationen über die sichtbaren Objekte zu erhalten, wurde eine GUI entwickelt, die unter anderem Informationen aus der Datenbank (Tabelle "object_sources") in einem HUD Fenster darstellen und verändern kann.

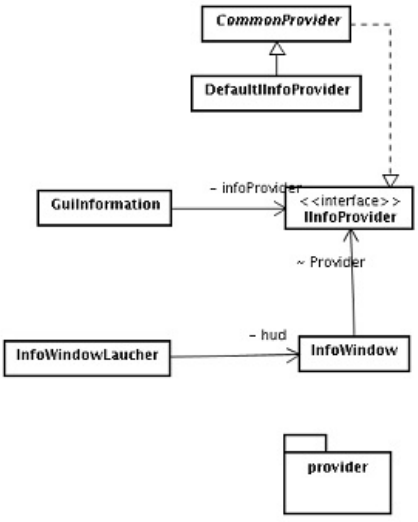


Abbildung 5.35: Klassendiagramm vintage.engine.gui

5.5.8.8 vintage.engine.hud

Xith bietet zur Darstellung von Informationen innerhalb der 3D Welt zwei verschiedene Möglichkeiten:

1. Java - Swing Fenster

2. HUDs

Die Klasse "VPHud" ist die Schnittstelle zum HUD der Xith Engine und bietet vor allem die Möglichkeit den InputManger an- bzw. auszuschalten, wenn das HUD aktiviert wird. Nur so kann eine bequeme Nutzung der GUI gewährleistet werden. Die "VPConsole" ist eine Konsole, in die während der Laufzeit verschiedene Kommandos eingegeben werden können, wie sie aus verschiedenen Computerspielen bekannt ist. Obwohl Xith die Möglichkeit bietet, Swing Fenster innerhalb der Simulation anzuzeigen, haben wir uns für das HUD System entschieden, da nach ausführlichen Versuchen mit der Swing GUI ein deutlicher Performancegewinn beobachtet werden konnte. Weiterhin ist die Struktur des HUD Systems der von Swing sehr ähnlich, was sich in einer kurzen Einarbeitungsphase wiedergespiegelt hat.

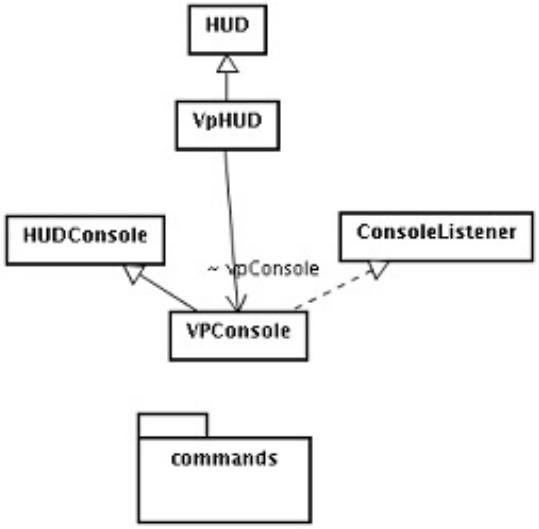


Abbildung 5.36: Klassendiagramm vintage.engine.hud

5.5.8.9 vintage.engine.hud.commands

Die "Commands" stellen die Eingaben in der VPConsole dar, wobei zwischen Kommandos ohne übergebene Parameter, Kommandos mit einem Parameter und Kommandos mit beliebig vielen Parametern unterschieden wird.

5.5.8.10 vintage.engine.model

Das Package model ist von zentraler Bedeutung in der Engine, hier werden alle verwendeten Modelle, durch die entsprechenden Modelloader geladen. Weiterhin wird durch das Interface "VPModel" das verwendete interne Modelformat spezifiziert.

- **VPModel:**
  Ein VPModel beschreibt ein darzustellendes Model. Ein Model hat immer die Attribute Position, Rotation und Scale. Für jedes der Attribute gibt es eine "TransformGroup", die die entsprechenden Transformationen beinhaltet. Dies ist zumindest für

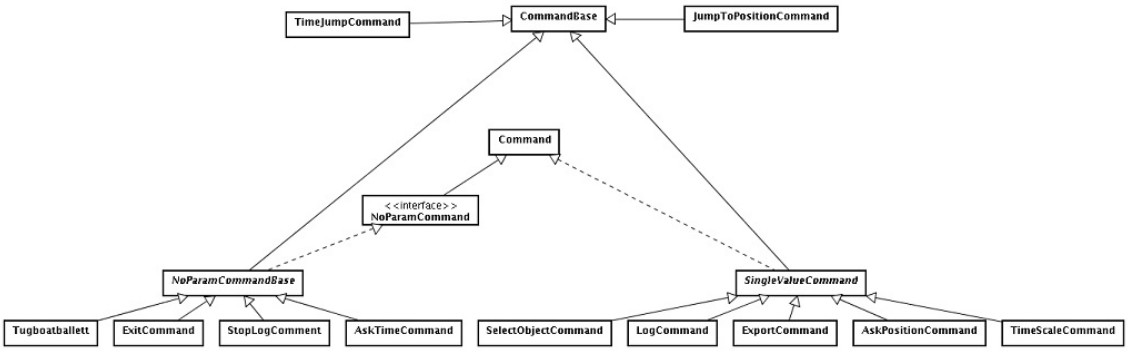


Abbildung 5.37: Klassendiagramm vintage.engine.hud.commands

die Attribute Position und Rotation unabdingbar, denn das Model soll immer um den Koordinatenursprung und nicht um die aktuelle Position rotiert werden. Ein VPModel ist somit, bei genauerer Betrachtung ein Teilbaum des Scenegraphen, bestehend aus dem Rootelement, der TransformGroup, die als Kindknoten wiederum die RotationsGroup enthält. Kindknoten der RotationsGroup sind die ScaleGroup und ggf. die SelectionGroup. Das eigentliche Modell ist als Kindknoten an die ScaleGroup gebunden. Ein weiteres Attribut des VPModel ist das VportObjekt, welches durch das Model dargestellt wird. Dies ist vor allem bei der Selektion (siehe vintage.engine.SelectionManager) nötig, um vom selektierten Model auf das Objekt schließen zu können. Gespeichert wird das VportObjekt als "UserData" der TransformGroup.

- **AbstractVPModel:**
  Da die meisten Eigenschaften des VPModel sowohl bei einem einfachen Model (SingleModel) als auch bei einem "Level Of Detail" Model (LODModel) gleich behandelt werden können, werden in dieser Klasse viele der geforderten Methoden implementiert, dazu gehört unter anderem die oben erwähnte Hierarchie von Transform, Rotations, Scalierungs und Model Gruppen.

- **SelectedGroup:**
  Die Selected Group beinhaltet ein einfaches Modell, welches angezeigt wird, wenn das Objekt vom Benutzer selektiert wurde. Somit ist es nur in solchen Modellen vorhanden, deren VPortObject das Interface "Pickable" (siehe vintage.logic.Pickable) implementiert.

- **SingleModel:**
  Ein SingleModel ist ein VPModel, welches durch genau ein Model und damit durch genau eine .obj, .3ds oder .vpm Datei beschrieben ist.

- **LODModel:**
  Ein LODModel besteht aus zwei oder mehr SingleModel Instanzen, von denen zu jedem Zeitpunkt nur eine angezeigt wird - je nach Abstand des Benutzers

vom Model. Dies ermöglicht die Verwendung von komplexen Modellen ohne starke Performance Einbrüche, da bei großer Entfernung ein vereinfachtes Object benutzt werden kann.

- **Modelloader:**
  Der Modelloader ist eine Klasse mit ausschließlich statischen Methoden zum Laden von 3D Modellen aus verschiedenen Dateiformaten. Neben dem Laden der Modelle werden bereits geladene Dateien gespeichert, um den Ladevorgang zu beschleunigen. Ist ein Modell bereits im Cache vorhanden, wird von diesem eine Kopie erstellt.

- **DefLoader, ModelDef, ObjDef:**
  Diese Klassen werden benutzt um ein XML vpm Modell zu beschreiben und anschließend mit dem Modelloader zu laden.

- **Texturepack:**
  Bei häufig geladenen Modellen wie Containern, ist es wünschenswert eine möglichst große Abwechslung in den Materialien zu haben. Um nicht für jede mögliche Kombination aus Geometrie und Material eine eigene Modeldatei zu speichern, wurde das Texturpack eingeführt, in dem alternative Materialien angegeben werden, die während des Ladevorgangs automatisch geladen werden. Momentan wird das Texturepack nur bei .obj Dateien angewendet.

- **ExtObjLoader:**
  Der ExtObjLoader ist eine Erweiterung des ObjLoaders von Xith, mit dem es möglich ist, Texturepacks anzuwenden.

- **BinaryVPMLoader:**
  Das BinaryVPM Modelformat ist ein für den VirtualPort entwickeltes Dateiformat zum Speichern von 3D Modellen. Unterstützt werden momentan LOD-Modelle, Texturepacks, Teilmodelle und die relative Positionierung von Teilmodellen. Zum Generieren der Dateien wurde ein Tool "Modeleditor" entwickelt, der das bequeme Erstellen und Manipulieren der Dateien ermöglicht.

5.5.8.11 Vintage.engine.partion/vintage.engine.partion.bst

Ein großes Problem ist das Erzeugen eines geeigneten Scenegraphen, der sowohl performant als auch übersichtlich gestaltet sein sollte. Da das gegebene Modell, der JadeWeserPort nicht von sich aus über eine geeignete hierarchische Struktur verfügt, muss diese im Nachhinein erstellt werden. Zu diesem Zweck wurde das Interface "PartionNode" eingeführt, welches eine abstrakte Vorschrift zum Erstellen eines baumartigen Graphen darstellt. Die Klasse "BSTNode" ist eine Realisierung dieses Interfaces. "BST" steht in diesem Fall für Binary Space Tree, bei dem der verfügbare Raum in eine baumartige

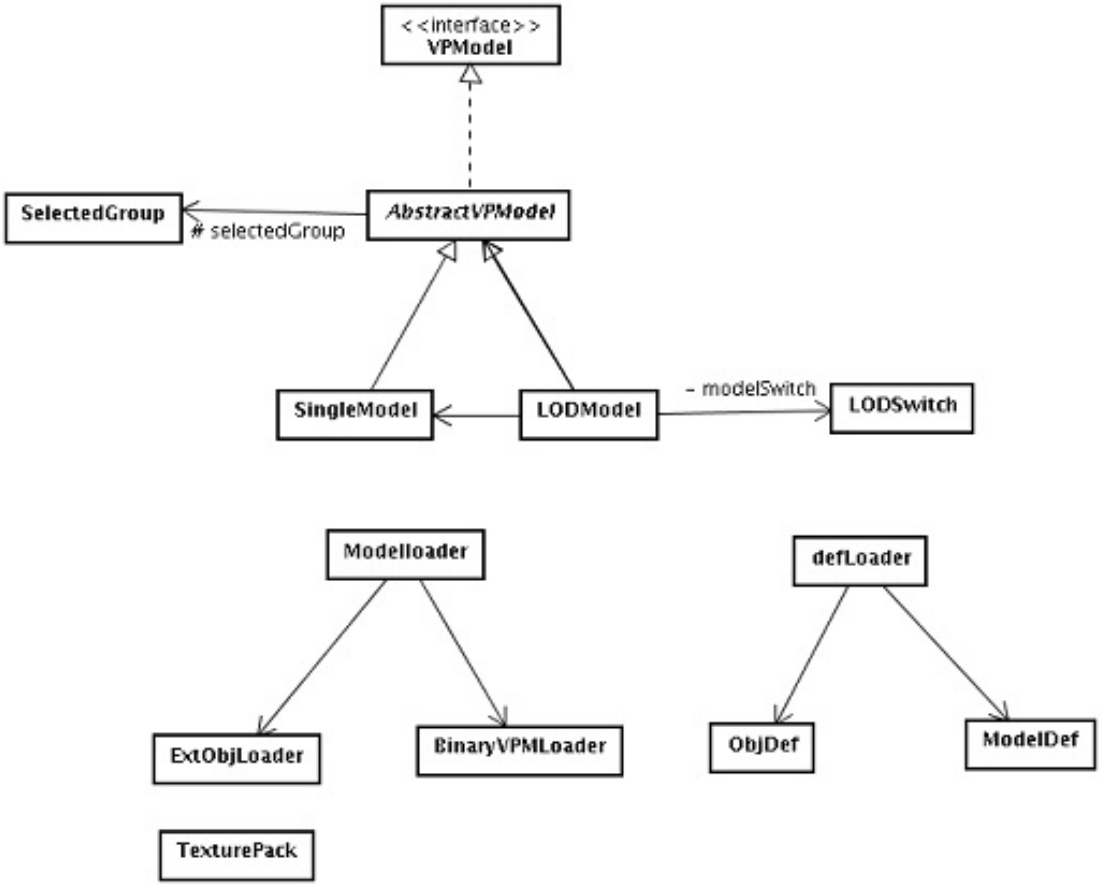


Abbildung 5.38: Klassendiagramm vintage.engine.model

Struktur gebracht wird. Dazu wird der verfügbare Raum solange an der aktuell längsten Achse unterteilt, bis eine Mindestgröße oder eine maximale Baumtiefe erreicht wurde. Eine schematische Darstellung der Unterteilung ist im folgenden Bild zu sehen. Der Vorteil

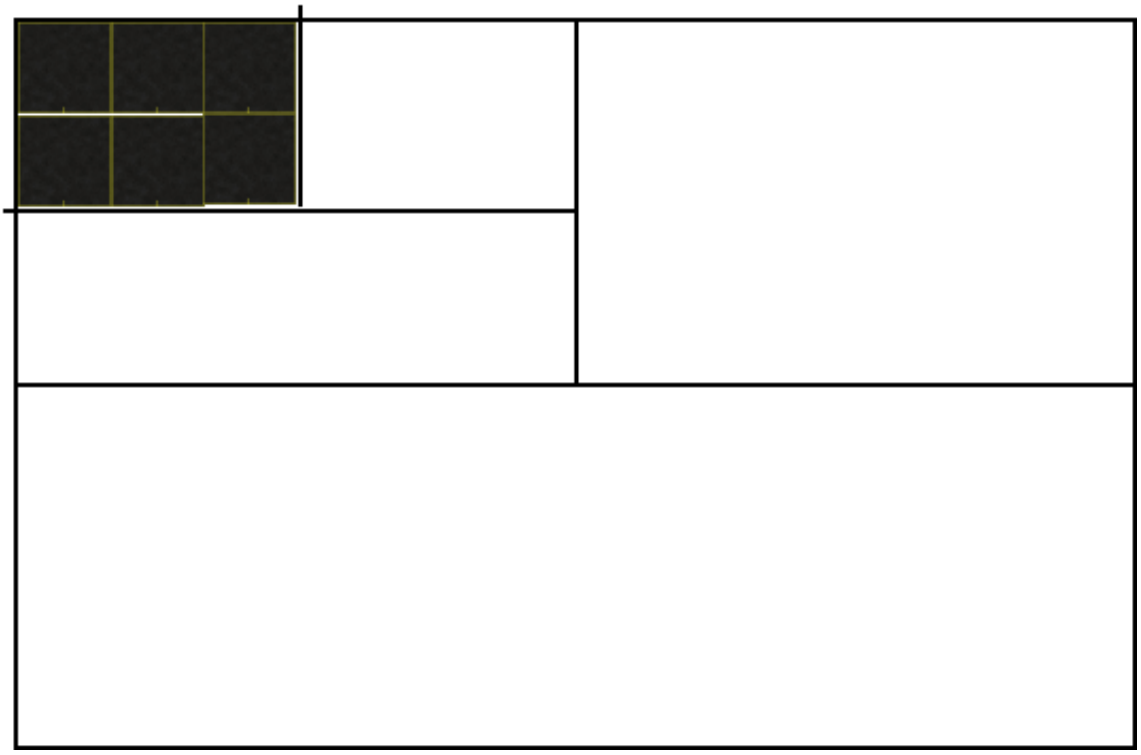


Abbildung 5.39: BSP-Tree Beispiel

einer solchen Unterteilung liegt auf der Hand; schon bei diesem kleinen Beispiel können von den 96 möglichen Straßenstücken, innerhalb von drei Schritten 90 vom Rendern und weiteren Berechnungen abgeschnitten werden, da sie für den Benutzer nicht sichtbar sind.

5.5.8.12 Vintage.engine.scenegraph

Im Package scenegraph werden hauptsächlich Wrapper zur verwendeten Xith Engine definiert. Dazu gehören die Klassen "VPNode", "VPGroup" und VPScenegraph". Dies wurde eingeführt um das Portieren auf eine andere, unterliegende Engine wie z.B. jME zu ermöglichen. Die Klasse, "ScenegraphManager" wird verwendet um den Scenegraphen zu initialisieren und während der Laufzeit zu verwalten. Der Rasterizer hat zunächst die Aufgabe den verwendeten Raum mit Hilfe eines "PartionNode" optimal zu unterteilen, um anschließend die in der Simulation verwendeten Objekte an den, für sie passenden, Stellen im Scenegraphen anzuordnen.

5.5.8.13 Vintage.event

Events sind Objekte, die zum Ausführen einer Aufgabe innerhalb der Engine genutzt werden. Sie werden an den EventManager übergeben, der die nötigen Schritte einleitet

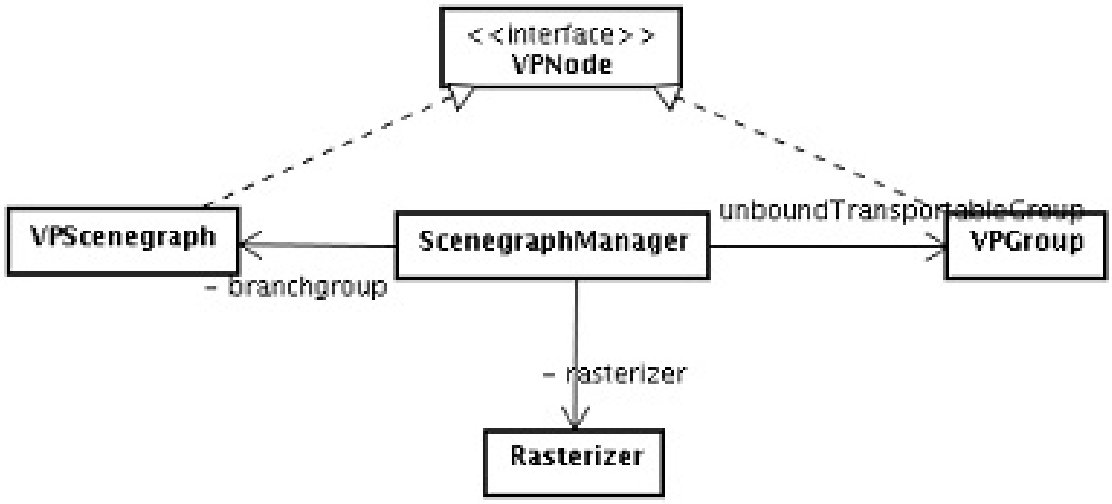


Abbildung 5.40: Klassendiagramm vintage.engine.scenegraph

um das Event ausführen zu können. Momentan werden folgende Events genutzt:
InfoEvent: Gibt an, dass von einem selektierten Objekt die Informations GUI angezeigt werden soll.
FollowEvent: Wird dieses Event ausgeführt, wechselt die Kamera in den "Thirdperson" Modus und folgt dem selektierten Objekt.
GotoEvent: Dieses Event weist die Kamera an, zum aktuell selektierten Objekt zu springen.
GUIEvent: Das GUIEvent wird verwendet um ein HUD Frame zu erzeugen, über das der Benutzer eines der oben stehenden Events auswählen kann. Zusätzlich ist es so gestaltet, dass weitere Events hinzugefügt werden können.

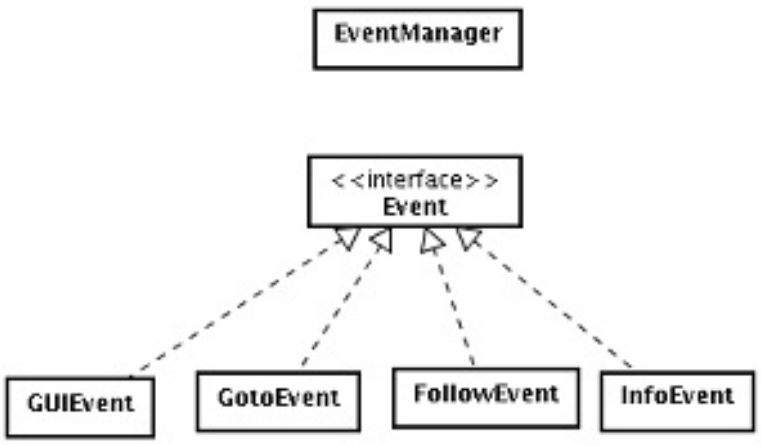


Abbildung 5.41: Klassendiagramm vintage.event

5.5.8.14 Vintage.input

In diesem Package befindet sich der InputManager, an den alle Tastatur und Mauseingaben des Benutzers weitergeleitet werden, damit dieser die zugeordneten Schritte einleiten kann.

5.5.8.15 Vintage.logic

Im package logic werden Interfaces und Abstrakte Klassen definiert, die das Verhalten von Objekten spezifizieren oder ihre Implementierung vereinfachen.

- **VPortObject:**
  Ein VportObject ist die allgemeinste Form eines Objektes. Jedes Objekt das in der Simulation genutzt werden soll, muss mindestens dieses Interface implementieren. Ein VPortObject besteht dabei aus einer eindeutigen id und einem eindeutigen Namen, über die es jederzeit identifiziert und während der gesammten Laufzeit beim ObjectManager angefragt werden kann. Weiterhin hat jedes Objekt in der Simulation, ob sichtbar oder nicht, eine Position, Rotation und Scalierung.

- **Drawable:**
  Soll ein Objekt in der 3D Welt dargestellt werden, muss es das Interface "Drawable" implementieren, welches verlangt, dass das Objekt über ein VPModel verfügt.

- **Pickable:**
  Dieses Interface wird implementiert, wenn das Objekt innerhalb der Simulation vom Benutzer selektiert werden kann. Sowohl beim Selektieren des Objektes, als auch beim Verlieren des selektierten Status, wird dem Objekt eine Nachricht, in Form eines Methodenaufrufes zugestellt.

- **Drawable:**
  Gui: Verfügt ein Objekt über Informationen, die während der Simulation angezeigt werden sollen, muss dieses Interface implementiert werden, um es der Engine bekannt zu machen.

- **Dockable:**
  Implementiert ein Objekt das Interface "Dockable", kann es ein oder mehrere transportable Objekte in den eigenen Scenegraphen mit aufnehmen. Ein Beispiel für ein dockable Objekt ist der Vancarrier, der einen Container aufnehmen kann.

- **Transportable:**
  Ein Transportable ist ein Objekt, welches an ein "Dockable" Objekt angedockt werden kann. Das Andocken an ein Dockable entspricht dabei der Parent Funktion von Blender. Ist ein Transportable "angedockt" wird seine Position und Rotation relativ zu der des Dockable Parent Objektes angegeben, dies führt dazu das Animationen auf das Dockable eine entsprechende Animation des Transportable herbeiführen.

- **Drivable:**
  Ein Drivable Objekt hat einen oder mehrere Pfade, auf dem sich Objekte bewegen können, wenn sie das Objekt betreten.

- **AnimationFactory:**
  Wird das Interface AnimationFactory implementiert, wird von dem Objekt gefordert, dass es aus einem AnimationsTask (siehe vintage.engine.animation.task) eine AbstractAnimation erstellen kann.

- **Static:**
  Dieses leere Interface wird vom ScenegraphManager und Rasterizer für die Optimierung des Scenegraphen genutzt. Es wird davon ausgegangen, dass Objekte, die dieses Interfaces realisieren, während der gesammten Laufzeit ihre Position, Rotation und Größe nicht verändern.

- **Dynamic:**
  Genau wie Static wird dieses Interface zur Optimierung des Scengraphen genutzt, wobei davon ausgeangen wird, dass sich Objekte dieses Types beliebig in der Simulation bewegen können.

- **SemiStatic:**
  Ein SemiStatic Objekt ist eine Mischform aus Static und Dynamic, bei dem davon ausgegangen wird, dass es nur relativ wenige Bewegungen während der Laufzeit gibt.

- **Animatable:**
  Soll ein Objekt während der Simulation animiert werden, muss es dieses Interface implementieren.

- **WorldEntity:**
  Die WorldEntity ist eine Abstrakte Klasse, die die vom VportObjekt geforderten Methoden implementiert. Weiterhin ist die WorldEntity hierarchisch aufgebaut, was bedeutet, dass sie weitere WorldEntitys initialisieren und verwalten kann.

- **DrawableWorldEntity:**
  Genau wie bei der WorldEntity handelt es sich bei der DrawableWorldEntity um eine Abstrakte Klasse, die automatisch Modelle laden kann, die in der Tabelle object_sources angegeben.

- **AbstractDrivableWorldEntity:**
  Bei dieser Klasse handelt es sich um eine erweiterte Version für das automatische Laden von Pfaden der DrawableWorldEntity. Es werden die Pfade geladen, die in der Tabelle object_sources mit dem Attribut “animation.path.“ angegeben sind.

Die Abstrakten Klassen WorldEntity und DrawableWorldEntity implementieren zusätzlich zu den geforderten Methoden weitere Methoden der übrigen Interfaces, sofern sich die Verwendung dieser verallgemeinern lässt.

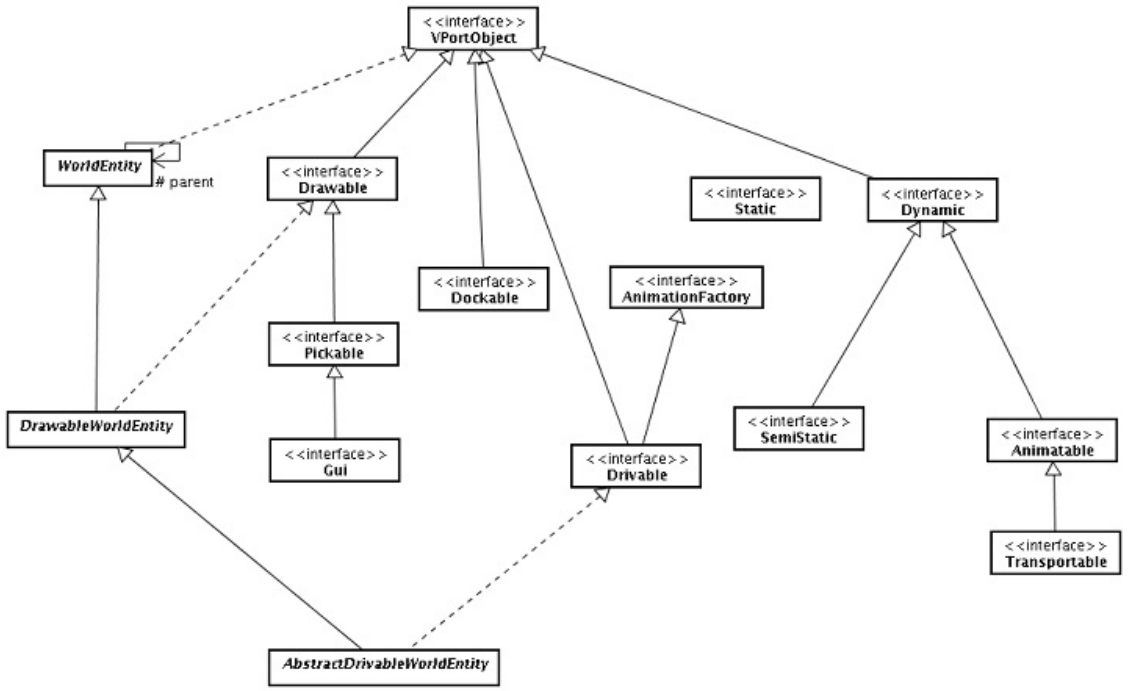


Abbildung 5.42: Klassendiagramm vintage.logic

5.5.8.16 Vintage.objects (inkl. Subpackages)

In diesem Package sowie in den Subpackages befinden sich die eigentlichen Objekte der Simulation, die die Logik der Engine darstellen. Da es sich bei nahezu jeder Klasse um speziell auf den Hafen und die Aufgabenstellung zugeschnittene Klassen handelt, werden hier nur die wichtigsten kurz beschrieben.

- **Unassigned, Floor:**
  Diese Klassen haben keine interne Logik und werden für Objekte benutzt, die zwar in der Simulation dargestellt werden sollen, aber ansonsten keine weitere Logik implementieren. Durch das Erben von der Klasse DrawableWorldEntity werden die zugeordneten Modelle automatisch geladen.

- **Street, AgvStation, Dock:**
  Diese Klassen werden für alle Objekte genutzt, auf denen VanCarrier oder Autos fahren können, wobei sich der Hauptteil der implementierten Logik innerhalb der abstrakten Klasse "AbstractDrivableWorldEntity" befindet.

- **Spreader, Ship, Dock (Lager):**
  Instanzen dieser Klassen sind in Objekte, denen es möglich ist "Transportable" Objekte in den eigenen Scenegraphen aufzunehmen, was vom Dockable Interface

gefordert wird. Eine Besonderheit hier ist der VanCarrier und die Klasse Crane (im Package vintage.objects.crane), die die zunächst für das Andocken von Containern zuständig zu sein scheinen. Sie besitzen jedoch Teilobjekte, in diesem Fall den Spreader, der die Aufgabe des Aufnehmens übernimmt.

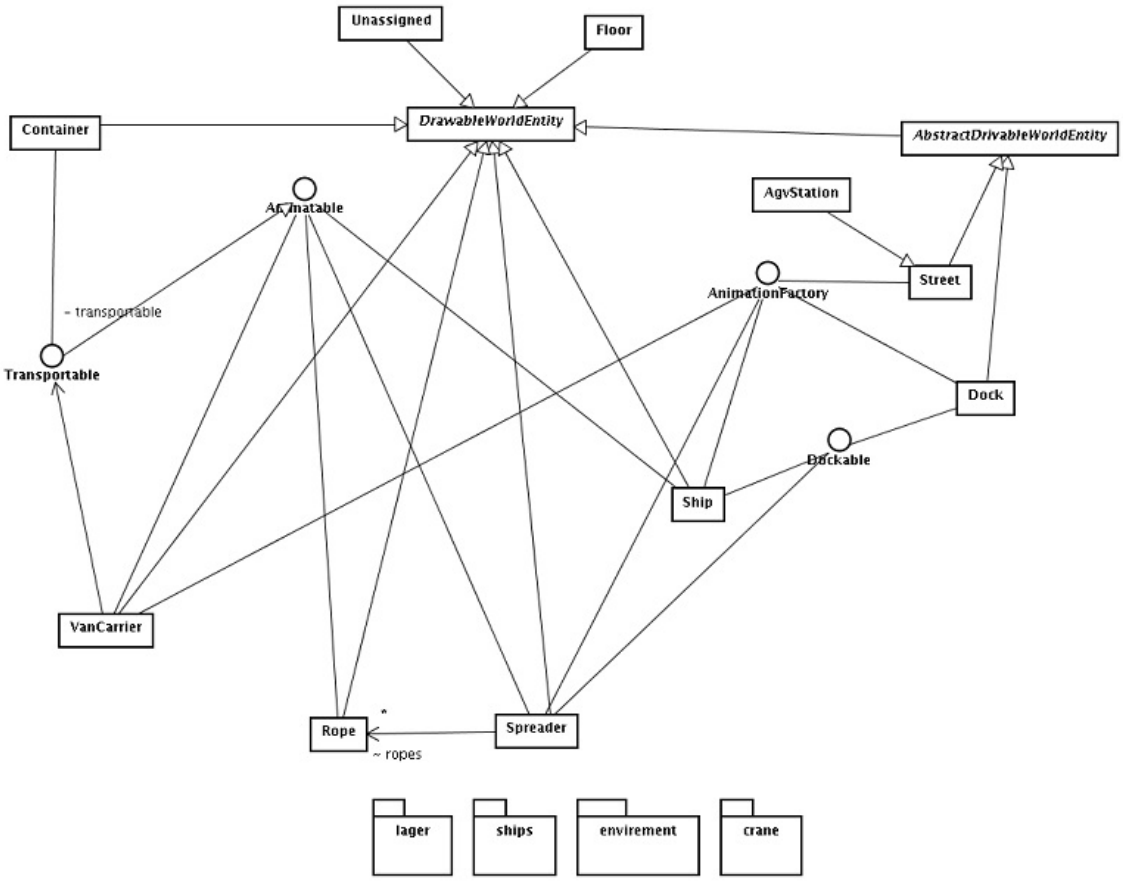


Abbildung 5.43: Klassendiagramm vintage.objects

5.5.8.17 Vintage.utils

Dieses Package beinhaltet Tools, die von mehreren Klassen genutzt werden, aber keinem anderen Package zugeordnet werden können, bzw. eigenständige Programme in den Subpackages sind.

- **SplashScreen:**
  Der Splashscreen realisiert einen Ladebildschirm, der während des Ladevorgangs angezeigt wird und den aktuellen Fortschritt anzeigt. Da das Laden aus vielen Teilaufgaben besteht, wurde es in zwei Fortschrittsbalken unterteilt, wobei der obere den Gesamtfortschritt des Ladevorgangs anzeigt.

- **Utils:**
  Diese Klasse enthält eine Sammlung von nützlichen statischen Methoden, die zwar relativ simpel sind, aber von verschiedenen Klassen benötigt werden.

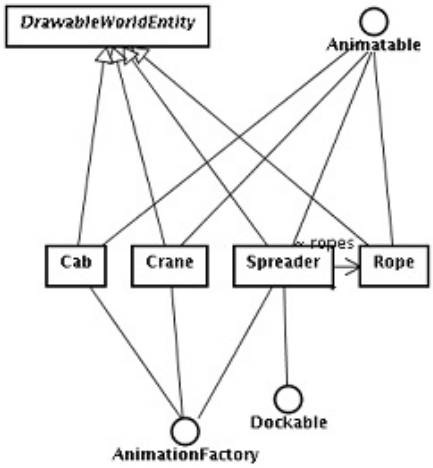


Abbildung 5.44: Zusatz Klassendiagramm vintage.objects

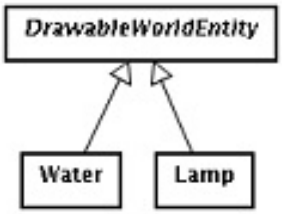


Abbildung 5.45: Zusatz Klassendiagramm vintage.objects

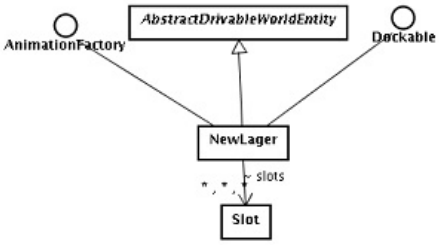


Abbildung 5.46: Zusatz Klassendiagramm vintage.objects

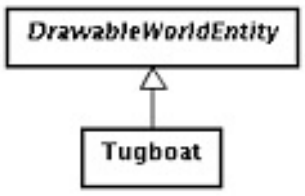


Abbildung 5.47: Zusatz Klassendiagramm vintage.objects

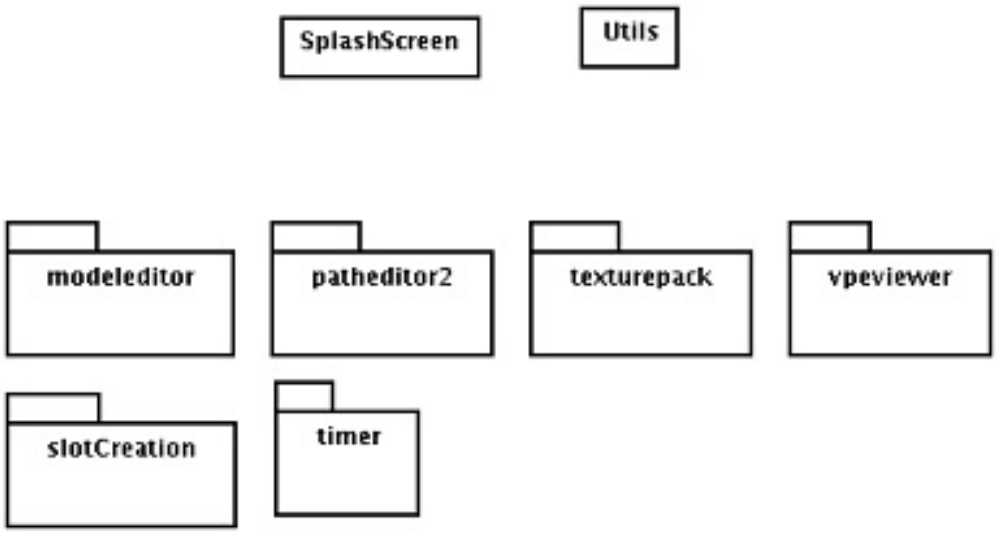


Abbildung 5.48: Klassendiagramm vintage.utils

5.5.8.18 Vintage.timer

Der für die Animation verwendete Timer wurde in diesem Package implementiert. Es handelt sich dabei um einen "OneShotTimer", das heißt ein Objekt kann sich bei dem Timer registrieren. Dies kann mit einer absoluten, oder zum Zeitpunkt der Registrierung relativen Zeit, einem TimeListener und dem zu übertragenden Objekt geschehen. Wurde ein Objekt beim Timer registriert, wird zum angegebenen Zeitpunkt eine einmalige Aktivierung des TimeListeners aufgerufen und das Objekt anschließend "vergessen". Der Timer wird aktiv, sobald das erste Objekt bei ihm registriert wird und legt sich bis zum nächsten Update-Event schlafen.

- **TimeListener:**
  Ein Timelistener ist das Objekt, welches durch den Timer zu einem bestimmten Zeitpunkt benachrichtigt wird.

- **TimerObject:**
  Das TimerObjekt ist eine private Klasse des OneShotTimers und speichert neben dem TimerObjekt, die Daten, die bei einem Update übertragen werden, und die Zeit an der das Update erfolgen soll.

5.5.8.19 Vintage.utils.modeleditor

Der Modeleditor ist ein eigenständiges Java - Swing Programm, mit dem es möglich ist, binäre .vpm Dateien zu erstellen und zu manipulieren. Nähere Informationen befinden sich im Handbuch.

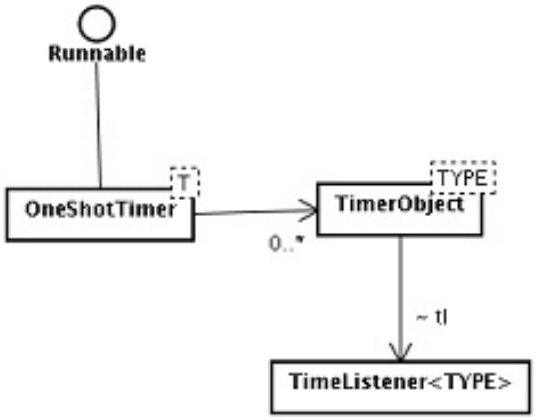


Abbildung 5.49: Klassendiagramm vintage.utils.timer

5.5.8.20 Vintage.utils.texturepack

Ein Texturepack ist eine Sammlung von .obj Materialien in einem eigenen Dateiformat. Die Klassen im Package texturepack stellen ein Java-Swing Programm dar, mit dem es möglich ist, entsprechende Dateien bequem zu editieren.

5.5.8.21 Vintage.utils.patheditor

Beim Patheditor handelt es sich um ein mit der Xith-Engine geschriebenes Programm zum Erstellen und Manipulieren von Pfaden, wie sie im package vintage.engine.animation.path benötigt werden. Erstellte Pfade werden mit dem Programm in die aktuelle Datenbank übertragen.

5.5.8.22 Vintage.utils.slotCreator

Ein Slot ist ein Platzhalter für einen Container, dem relative Koordinaten zugeordnet werden können. Weiterhin besteht die Möglichkeit, einen Slot für einen konkreten Container zu reservieren. Die Verwendung von Slots hat den Vorteil, dass für einen Container nicht während der Laufzeit die Position auf dem Dockable berechnet werden muss, was unter Umständen recht rechenintensiv ist. Der Slotcreator ist ein Tool mit dem es möglich ist Slots für die Objekte Lager, Ship, Train und Truck zu erstellen. Dem Tool werden dazu die bereits erstellten Slots, sowie eine oder mehrere Boundingboxen übergeben in denen die Slots erstellt werden. Aufgabe des Slotcreators ist es, jedem Slot eine relative Position innerhalb der übergebenen Boundingboxen zuzuordnen. Weitere Einstellungsmöglichkeiten sind die Achse auf der das Slot-Array geteilt werden kann, auf welchen Achsen es möglich ist den Mindestabstand zwischen den Slots zu erweitern, um einen visuell ansprechendes Ergebnis zu erhalten und ein automatisches Mapping zwischen Achsenunstimmigkeiten, welche durch unterschiedliche Koordinatensysteme innerhalb der Engine und des vTOS entstehen.

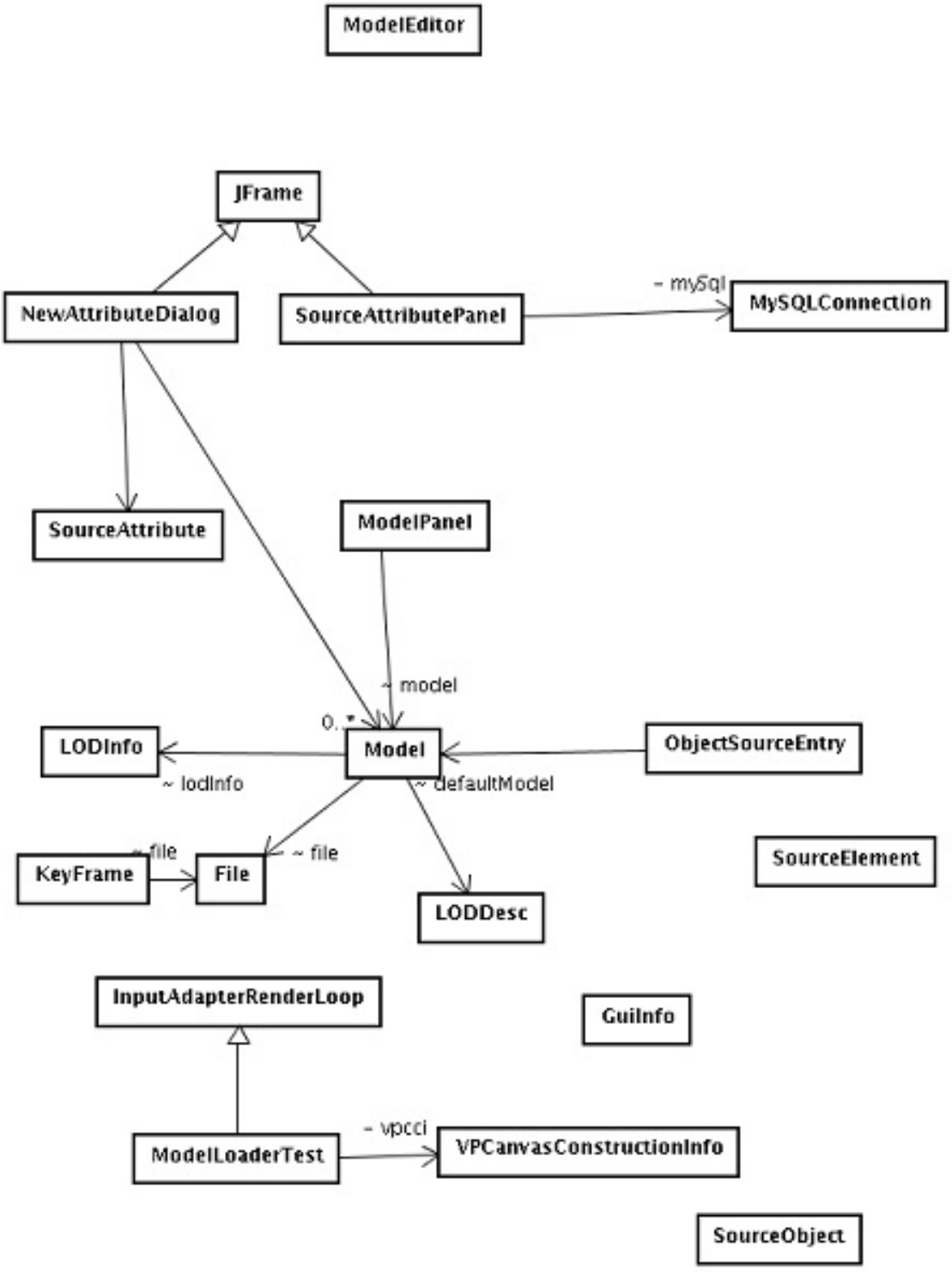


Abbildung 5.50: Klassendiagramm vintage.utils.modeleditor

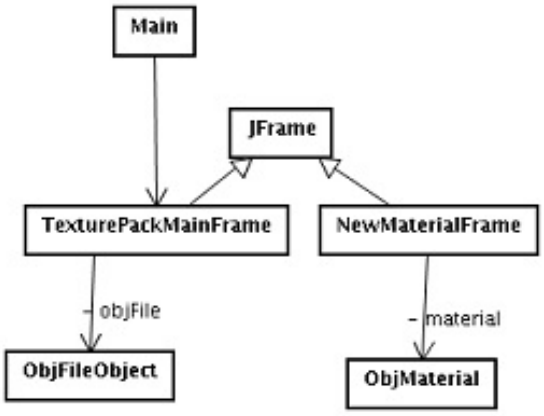


Abbildung 5.51: Klassendiagramm vintage.utils.texturepack

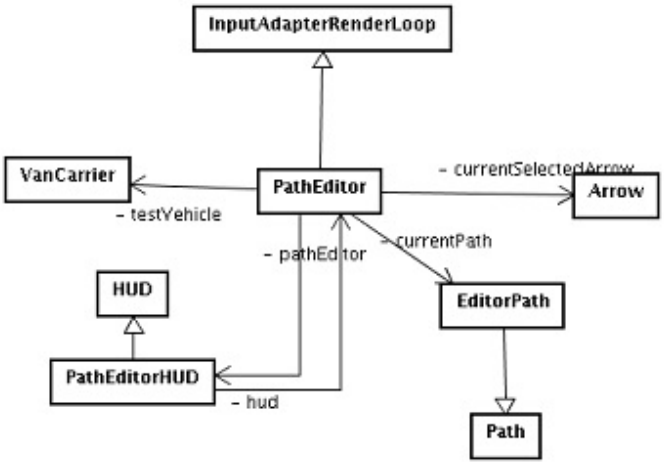


Abbildung 5.52: Klassendiagramm vintage.utils.patheditor

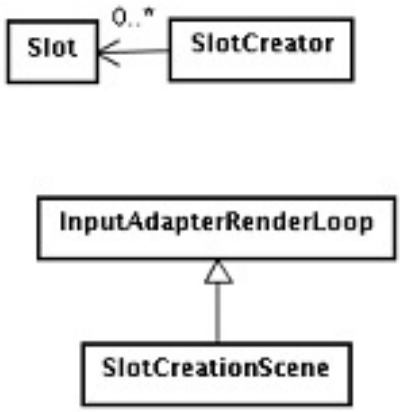


Abbildung 5.53: Klassendiagramm vintage.utils.slotCreator

5.5.8.23 Vintage.utils.vpeviewer

Der VPEViewer ist eine "light" Version des gesamten Programms, welches eine vorher aufgenommene .vpe Datei wieder darstellen kann. Innerhalb des mit dem VPEViewer dargestellten Programms kann man sich genau wie im eigentlichen Programm bewegen. Jedoch werden keine Animationen neu erstellt, sondern auf IPO - Keyframes zurückgegriffen, die bei einem Durchlauf des Originalprogramms aufgezeichnet worden sind. Da für den VPEViewer keine Datenbankanbindung nötig ist, eignet er sich besonders für die Verwendung bei Präsentationen.

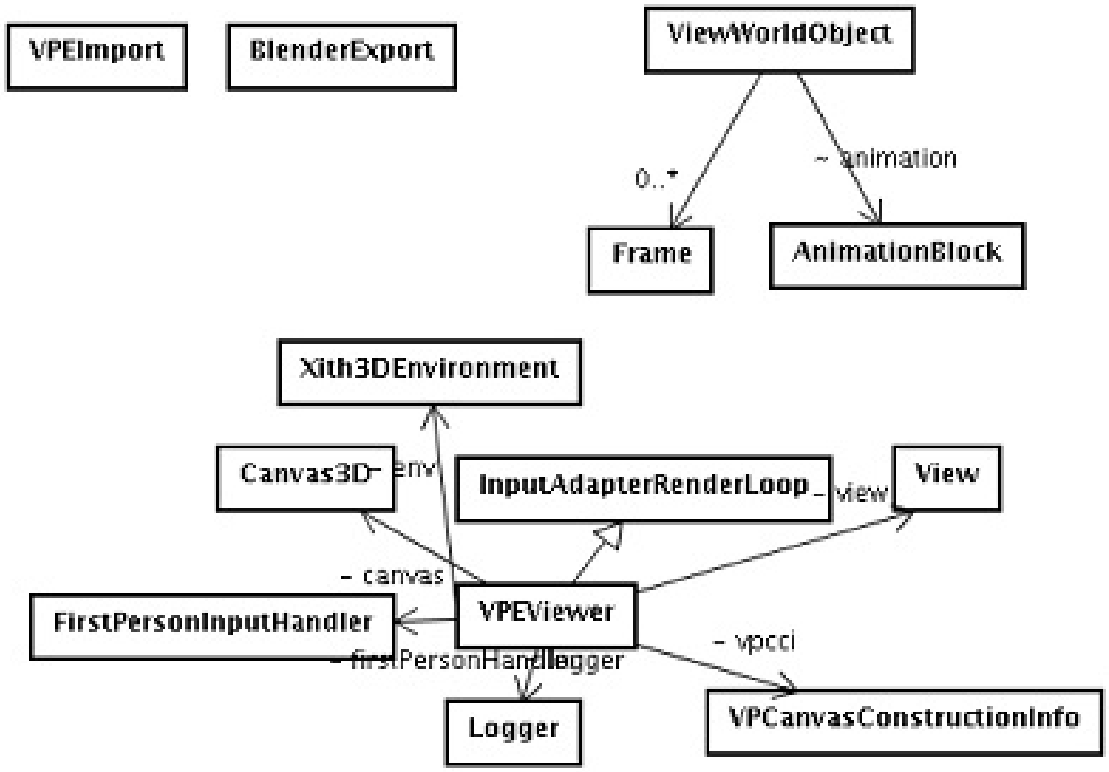


Abbildung 5.54: Zusatz Klassendiagramm vintage.engine.vpeviewer

5.5.8.24 Blender Import/Export

Der VPEViewer ist aus einem Dateiformat entstanden, welches zunächst für den Import in Blender entwickelt worden ist. Es existiert ein Pythonskript, mit dem .vpe Dateien in Blender importiert werden können, sofern die in der Simulation verwendeten Modelle als Blenderdatei vorliegen.

5.5.8.25 Manager

Im folgenden Abschnitt werden die Manager, die sich im Package "vintage" befinden genauer betrachtet, da sie einen Großteil der Logik innerhalb der Engine darstellen.

5.5.8.26 Vintage.Manager

Dieses Interface ist lediglich dafür zuständig, für eine einheitliche Initialisierung und Terminierung der Simulation zu sorgen.

5.5.8.27 Vintage.SourceManager

Die Simulation ist so aufgebaut, dass sie in der Lage ist Objekttypen zu laden, die durch die Einträge in der Tabelle "object_sources" der Datenbank definiert werden. Das Format dieser Tabelle ist möglichst allgemein gehalten, auch wenn dies bei strenger Betrachtung in einigen Fällen die erste Datenbanknormalform verletzt.
Ein Eintrag in dieser Tabelle besteht aus einer eindeutigen ID, die jedoch vom SourceManager nicht weiter beachtet wird.
Der nächste Eintrag ist der Name des Objekttypes, dieser Name kann beliebig gewählt werden und ist durch einen String repräsentiert. Die nächsten beiden Spalten definieren ein Attribut - Wert Tupel, welche zunächst beliebige Werte enthalten können (hier auch die mögliche Verletzung der Normalform).
Wie die übergebenen Attribute interpretiert werden, hängt zum großen Teil von der angegebenen Klasse ab. Alle Attribute, die zu ihrem Objekttyp gehören, werden als Parameter bei der Initialisierung übergeben.
Es gibt jedoch einen Parameter, der bei jedem Typ enthalten sein muss, "init.classname" gefolgt von dem Pfad der verwendeten Klasse im Classpath der Engine. z.B. "vintage.objects.VanCarrier".
Die einzige Einschränkung der Klasse ist, dass sie von der abstrakten Klasse "WorldEntity" des Packages "vintage.logic" erbt.
Mit Hilfe dieses Eintrags wird eine neue Instanz der entsprechenden Klasse während der Laufzeit dynamisch erstellt und mit den Werten aus der Tabelle "objects", sowie den in der Tabelle "object_sources" angegebenen Attributen, initialisiert.
Weitere Attribute, die je nach verwendeter Klasse bzw. Oberklasse, wie z. B. "DrawableWorldEntity" oder "AbstractDrivableWorldEntity" gefordert oder unterstützt werden, sind:

DrawableWorldEntity:

1. **model.defaultmodelpath (zwingend notwendig)**
   Der Wert dieses Attributes gibt den Pfad zu der Modeldatei relativ zum Rootverzeichnis an. (z.B.: "data/modelle/vancarrier/vancarrier.vpm")

2. **init.rotOffset (optional)**
   Der als Wert angegebene Vektor (Komponenten durch ; getrennt) gibt eine Rotation an, die auf jedes Objekt dieses Types addiert wird. Dieser Wert hat nur Einfluss auf das verwendete Modell und wird bei einem Aufruf von getRotation() nicht berücksichtigt.

3. **Init.posOffset (optional)**
   Wie init.rotOffset, nur für eine Positionsverschiebung.

4. **Init.scaFactor (optional)**
   Ähnlich wie posOffset und rotOffset, jedoch wird der übergebene Wert nicht zu der aktuellen Skalierung addiert, sondern mit ihr multipliziert. Ein Wert von 0;0;0 würde entsprechend das Modell "unsichtbar" machen.

5. **init.otherObjects# (optional)**
   Mit diesem Attribut können einem Objekttypen weitere Objekte zugeordnet werden. Wenn dies der Fall ist, werden diese von der WorldEntity automatisch geladen und beim ObjectManger registriert, als wäre dieses Objekt in der Tabelle "objects" aufgeführt. Jedoch beziehen sich die Koordinaten, die mit "init.rotOffset", "init.posOffset" und "init.scaFactor" angegeben werden, nun relativ zu dem aufrufenden Objekt.
   Der Wert des Attributs muss ein Name aus der Tabelle "object_sources" sein.

AbstractDrivableWorldEntity:

1. **animation.pathid.# (optional)**
   Diesem Attribut wird die ID eines Pfades aus der Tabelle "path" übergeben, der Pfad wird vom AbstractDrivableWorldEntity initialisiert und der aktuellen Position, Rotation und Größe des erstellten Objektes angepasst.

(# steht für eine beliebige Nummer, um die Attribute besser zu unterscheiden)
Da es sich beim AbstractDrivableWorldEntity um eine Kind–Klasse von DrawableWorldEntity handelt, können bzw. müssen die dort angegebenen Attribute ebenfalls gesetzt werden.

5.5.8.28 Vintage.RotationManager

Der RotationManager liest die Tabelle "rotation" der Datenbank aus und erstellt aus den Einträgen eine LookUpTabelle.
Dies ist notwendig, da das von uns verwendete Simulationstool em–Plant zwar Grundobjekte rotiert, diese jedoch nicht in Form eines Winkels, sondern einer Rotations bzw. Bildnummer angibt. Die LookUpTabelle ist entsprechend ein Mapping von Rotationsnummer zu Rotationswinkel (angegeben in Grad).

5.5.8.29 Vintage.ObjectManager

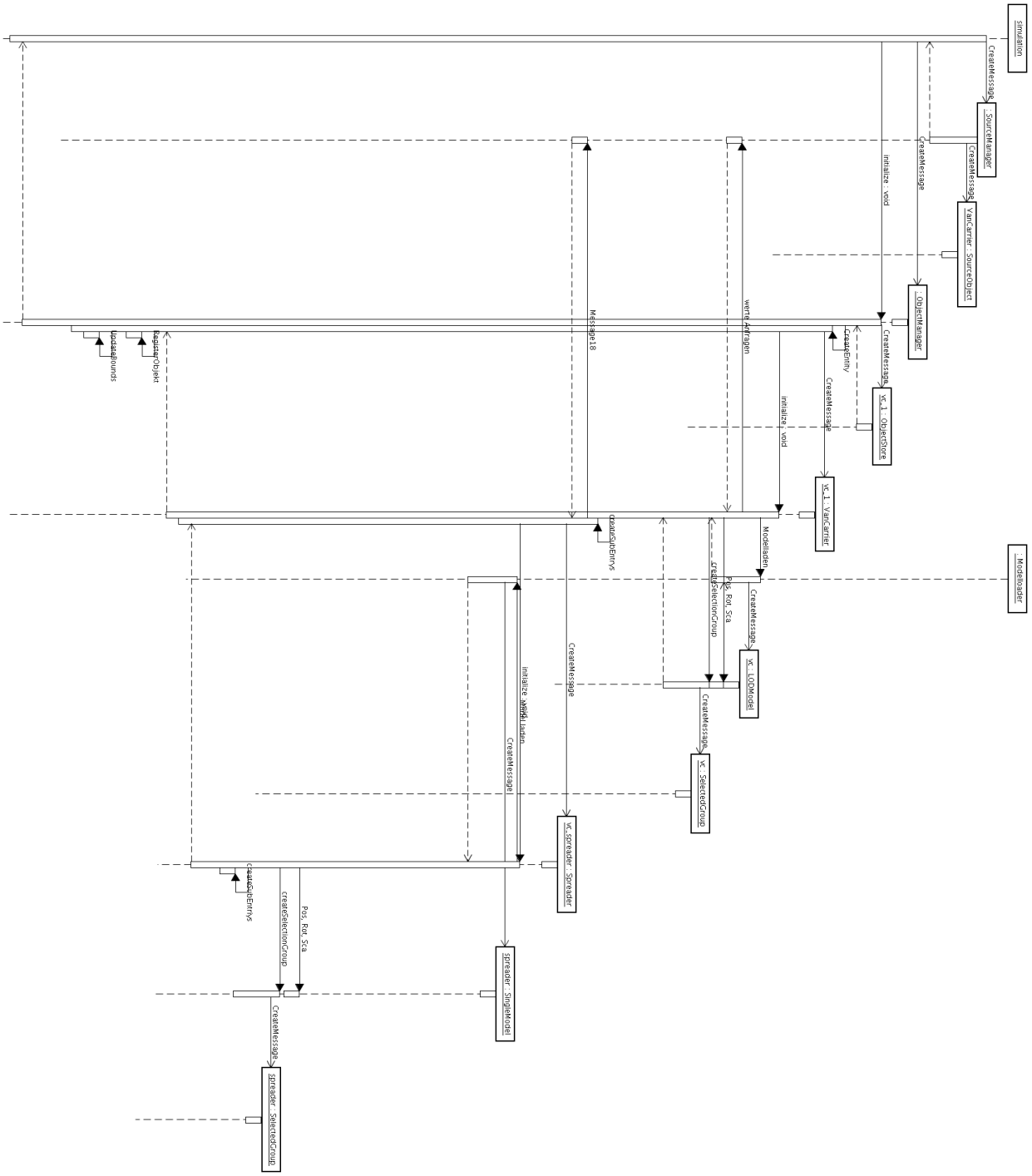
Der ObjectManager lädt und initialisiert die in der Tabelle "objects" angegebenen Objekte. Um dies bewerkstelligen zu können, ist er in der Lage neue Instanzen von Klassen aus dem Classpath zu erzeugen und diese, sofern sie Instanzen der abstrakten Klasse

“WorldEntity“ sind, zu initialisieren.
Neben der Initialisierung der Objekte erstellt er ein BoundingVolume der gesamten Simulation, welches vom ScenegraphManager dazu genutzt wird, den Scenegraphen räumlich zu unterteilen.
Während der Laufzeit ist es die Aufgabe des ObjectManagers Zugriff auf jedes Objekt der Simulation zu gewährleisten. Um einen effektiven Zugriff auf die in der Regel mehreren tausend Objekte zu ermöglichen, benutzt er effiziente Such–Strukturen, die eine Zugriffszeit von O(1) ermöglichen (HashMaps).
Durch diese Art der Initialisierung haben wir uns ein größtes Maß an Erweiterbarkeit erhalten, denn es ist möglich neue Objekte hinzuzufügen, ohne den Quellcode der Engine zu verändern, wie es z.B. bei einem Update, oder sogar einem komplett anderen Anwendungsfall nötig wäre.
Eine mögliche Erweiterung an dieser Stelle ist die Einführung eines “Hotplug“–Systems, mit dem es zusätzlich möglich ist, “Classpath–fremde“ Objekte zu laden. Das folgende Sequenzdiagramm zeigt die wichtigsten Aspekte beim Initialisieren eines Objektes, in diesem Fall eines Van Carriers. Aus Gründen der Übersichtlichkeit konnten nicht alle Methodenaufrufe dargestellt werden, weiterhin lädt der Spreader zwei weitere Objekte, was im Diagramm 5.5.8.29 jedoch nicht dargestellt ist.

5.5.8.30 Vintage.engine.AnimationManager

Der Animationsmanager ist der wichtigste Manager während der Laufzeit des Systems, denn ohne ihn würden keine Animationen gestartet werden.
Daher muss er folgende Anforderungen erfüllen können:

1. Beliebige Animationen erstellen und starten
2. Datenbank auslesen, wenn keine weiteren Animationen vorhanden sind
3. Bei leeren Tabellen stabil laufen und auf Einträge warten
4. Nahezu synchron mit der Simulationslogik (em–Plant) laufen
5. Zeitdifferenzen ausgleichen (Einträge der Datenbank, sind meist auf das Jahr 2001 datiert)
6. Zu einem beliebigen Zeitpunkt innerhalb der Simulation starten (Datenbankzustand in der Simulation herstellen)
7. Zu einem beliebigen Zeitpunkt innerhalb der Simulation springen (Datenbankzustand in der Simulation herstellen)
8. Beschleunigung und Abbremsen der Simulationsgeschwindigkeit ermöglichen


simulation
CreateMessage
: SourceManager
CreateMessage
VanCarrier_SourceObject
CreateMessage
: ObjectManager
initialize : void
CreateMessage
vc_1 : ObjectStore
CreateEntry
CreateMessage
vc_1 : VanCarrier
initialize : void
werte Anfragen
Message18
UpdateBounds
RegisterObject
createObjectEntrys
ModelLoader
ModelLoader
CreateMessage
vc : LODModel
Pos, Rot, Sca
createSelectionGroup
CreateMessage
vc : SelectionGroup
initialize : void
CreateMessage
vc_spreader : Spreader
createObjectEntrys
Pos, Rot, Sca
createSelectionGroup
spreader : SingleModel
CreateMessage
spreader : SelectionGroup

Diese Anforderungen, genau wie die generelle Problematik, Aufgaben zu einem vorgegebenen Zeitpunkt zu bearbeiten, machen den AnimationManager zu dem komplexesten Manager der Simulation.
Die Möglichkeit beliebige Animationen, die zum aktuellen Entwicklungsstand noch nicht bekannt sein müssen, zu verwenden, wurde über ein ähnliches Konzept realisiert, wie es schon im ObjektManager zum Laden beliebiger Klassen gemacht wurde.
Es wurde eine neue Tabelle mit Namen "object_animations" in der Datenbank angelegt, die ähnlich der Tabelle object_sources aufgebaut ist. In diesen Klassen wird ein Animationstyp (die ID des Eintrags) angegeben, sowie ein lesbarer Name um die Übersichtlichkeit zu gewährleisten.
Weiterhin wird der Pfad zu einem "NewAnimationTask" innerhalb des Classpath der Engine angegeben, der als Referenz dienen soll.
Das Interface NewAnimationManager fordert die Möglichkeit, eine neue Instanz der gleichen Klasse erzeugen zu können, denen die Spalten der Tabelle "queue" übergeben werden. Dies ermöglicht es die Einträge in der Tabelle "queue" unterschiedlich zu interpretieren. Beispielsweise wird der Eintrag action1 bei einer "arrive"–Animation als Eingangsname interpretiert, wobei bei einer "pickUp" Animation der Eintrag "action1" als Containername interpretiert werden kann.
Um aus dem zur Laufzeit erstellten Animationstask eine konkrete Animation erzeugen zu können, muss der Task eine "AnimationFactory" liefern, an die der Task übergeben wird. Diese erstellt aus den entsprechenden Informationen die Animation, welche an den von Xith bereitgestellten "OperationScheduler" übergeben wird. Die Funktion wird nun bei jedem Frame aufgerufen, solange sie aktiv ist.
Ziel der Engine ist es möglichst parallel mit der Simulationssoftware, in unserem Fall eM–Plant (hier kann auch eine andere eingesetzt werden), zu laufen. Dabei ist zu berücksichtigen, dass eine echte Parallelität nicht zu realisieren ist, da die Ereignisse von em–Plant erst dann übertragen werden, wenn die Aktion bereits ausgeführt worden ist. Zusätzlich kommt hinzu, dass der AnimationManager, um eine flüssige Animation zu gewährleisten, mindestens eine Animation pro animiertem Objekt hinterher hängt, um aus der Differenz der beiden Startzeiten die Animationsdauer zu berechnen. (Hier können Ausnahmen innerhalb des AnimationTask definiert werden).
Wird die Engine gestartet, ohne das Animationen in der Tabelle "queue" oder "save_queue" vorhanden sind, versucht die Engine innerhalb vorgegebener Zeitabstände erneut die Tabellen auszulesen um mit der Animation starten zu können. Dies ist auch der Fall wenn alle Animationen in den beiden Tabellen abgearbeitet worden sind.
Um den Rechenaufwand pro Auslesevorgang möglichst gering zu halten, werden pro Lesevorgang nur eine begrenzte Anzahl von Zeilen ausgelesen. Der AnimationManager berechnet automatisch die maximale Zeit, die bis zum nächsten Update vergehen kann, ohne das es zu stockenden Animationen kommt.
Um dem Benutzer die Möglichkeit zu geben, Abläufe im von uns dargestellten Hafen im Zeitraffer zu betrachten, wurde eine Skalierung der Zeit eingeführt. Hier gilt: t $==$ 1

entspricht Echtzeitdarstellung, wie es von der Simulationssoftware vorgegeben ist, t < 1 würde einer Beschleunigung der Zeit entsprechen und t > 1 einer extremen Verlangsamung.
Die Einführung der Zeitskalierung erforderte eine ständige Aktualisierung der Zeitverschiebung, die zwischen der Simulationszeit und der Realzeit besteht.
Eine weitere Aufgabe des AnimationManagers ist es, dem Benutzer die Möglichkeit zu geben, zu verschiedenen Zeitpunkten innerhalb der Simulation zu springen. So kann beispielsweise ein ganzer Tag übersprungen werden. Der AnimationManager versucht dann den Zustand der Datenbank auch in der Simulation herzustellen, indem er von jedem Objekt die jeweils letzte Aktion ausführt, die noch außerhalb der neuen Startzeit liegt.
Dies setzt voraus, dass alle Animationen (auch neue) dafür sorgen, dass ihre Endbedingungen erfüllt sind, wenn die "stop" Methode der Animation aufgerufen wird. Das folgende Sequenzdiagramm **??** gibt einen kleinen Überblick über das Zusammenspiel der einzelnen Klassen, wenn eine Animation gestartet werden soll.

5.5.9 Schnittstellen Engine

Zusammenfassung
Im folgenden Abschnitt werden die Schnittstellen zum Erweitern der Engine kurz zusammengefasst. Der Abschnitt ist als Ergänzung zu der obigen Klassenbeschreibung zu verstehen und bietet lediglich einen kleinen Einblick in die vielfältigen Erweiterungsmöglichkeiten.

5.5.9.1 Objektklassen

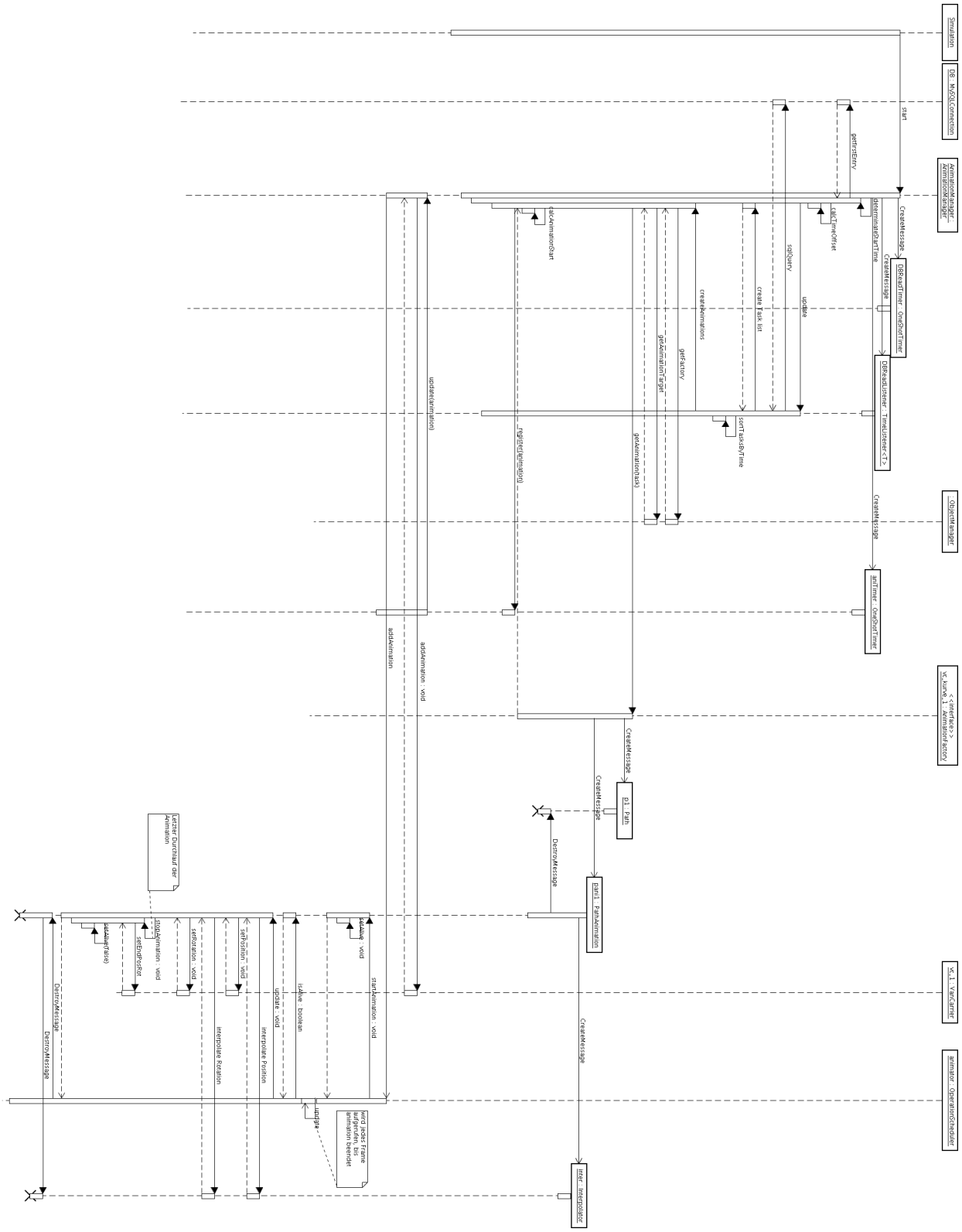
Die Tabelle object_sources definiert Objektklassen und wie diese in der Simulation dargestellt werden. Eine Klasse wird dabei durch einen eindeutigen Namen beschrieben und kann beliebig viele Attribute enthalten. Lediglich das Attribut "init.classname" wird vom ObjectManager gefordert, um einem Objekt eine neue Instanze der Klasse WorldEntity zuweisen zu können.

5.5.9.2 Angezeigte Objekte

Die in der Engine angezeigten Objekte werden in der Tabelle objects (und save_objects) definiert und müssen während des Startzeitpunktes des Programms vollständig hinterlegt sein. Ein nachträgliches Laden von Objekten ist nicht vorgesehen. Mit welchen Eigenschaften ein Objekt geladen wird, wird über die Tabelle object_sources definiert.

5.5.9.3 Logik

Die Engine arbeitet in unserem Fall mit dem Simulationswerkzeug eM-Plant zusammen. An dieser Stelle ist jedoch jede andere Simulationssoftware denkbar, solange sie sich an die entsprechenden Datenbank-Vereinbarungen hält.


Simulation
DB : MySQLConnection
AnimationManager : AnimationManager
DBReadTimer : OneShotTimer
DBReadListener : TimeListener<T>
_ObjectManager
antTimer : OneShotTimer
vc_kurve_1 : <<interface>> AnimationFactory
p1 : Path
path1 : PathAnimation
vc_1 : VanGarmer
animator : OperationsTweaker
inter : Interpolator
start
getFromEntry
CreateMessage
determineStartTime
calcTimeOffset
calcAnimationStart
sqlQuery
update
createAnimations
createTask : int
getFactory
getAnimationTarget
getAnimation(task)
registerAnimation()
startTasksByTime
updateAnimation()
addAnimation
addAnimation : void
DestroyMessage
Letzter Durchlauf der Animation
setAlive(false)
setEndPosRot
stopAnimation : void
setRotation : void
setPosition : void
update : void
interpolate Position
interpolate Rotation
isAlive : boolean
setAlive : void
stopAnimation : void
Wartet jedes Frame aufgerufen, bis Animation beendet
interpolate

Prinzipiell ist die Kommunikation zwischen Engine und Logik auf eine einseitige, von eM-Plant initialisierte, Kommunikation beschränkt, die über das vorgegebene Datenbankschema geschieht. Um dem Benutzer die Möglichkeit zu geben, direkt in die Simulation einzugreifen, wird zusätzlich eine asynchrone Socket Verbindung zu verschiedenen Logik Elementen ermöglicht.
Um eine neue, auf der Logik arbeitende Funktionalität einzubinden, muss sich das Element mit seiner Host–Adresse und Port zunächst in die Tabelle command_target eintragen.
Anschließend werden die verfügbaren Dienste in der Tabelle emplant_commands eingetragen. Die Engine überprüft bei jedem Selektieren eines Objektes, ob entsprechende Dienste für diese Objektklasse existieren. Ist dies der Fall, wird im Info-Dialog ein Knopf mit der entsprechenden Beschreibung (Spalte "desc" der Tabelle) erzeugt. Beim Betätigen dieses Knopfes wird ein Signal mit dem angegebenen Code und dem Objektnamen an das Element versendet, sofern es erreichbar ist. Dabei handelt es sich um eine "Fire And Forget" Nachricht, was bedeutet, dass ein nicht erreichbares Element zu keinem Fehler, aber auch zu keiner Meldung führt.

5.5.9.4 Animationen

Animationen werden von der Simulationslogik in der Tabelle queue bzw. save_queue als ein Tupel bestehend aus:

1. einer eindeutigen ID
2. einem Animationstyp
3. einer Animations Factory
4. einem Animations Ziel
5. bis zu drei Parametern
6. einem Startzeitpunkt

abgelegt.
Um eine neue Animation hinzuzufügen, muss zunächt in der Tabelle object_animation ein neuer Animationstyp erstellt werden. Die dort angegebene Klasse definiert eine Vorschrift zum Interpretieren der Werte aus der Tabelle "queue". Zusätzlich muss sichergestellt sein, das alle Animations Factorys, aus diesem Animationstyp eine entsprechende Animation erstellen können. Ist dies bisher nicht der Fall, muss die entsprechende Java - Klasse entsprechend abgeändert bzw. erweitert werden.

5.6 Schnittstellen

5.6.1 vTos - eM-Plant

Der Simulation stehen drei Kanäle zur Kommunikation mit den anderen Komponenten zur Verfügung. Die Simulation kommuniziert mit der vTOS Komponente über Sockets. Es existiert ein Ein- und ein Ausgabekanal. Einzelne Aufträge werden von vTOS über den genannten Eingangskanal an die Simulation geschickt. Ein Auftragseingang wird anschließend über den Ausgabekanal bestätigt. Als Beispiel nehmen wir hier einen Transportauftrag. Dieser wird von vTOS an die Simulation mit den Parametern Abholort, Zielort, Container und Fahrzeugkennung geschickt. Die Simulation bestätigt den Eingang und gibt den Auftrag an das entsprechende Fahrzeug weiter. Hat dieses Fahrzeug den Abholort erreicht und den Container geladen, wird dieses als Zwischenbericht der vTOS Komponente mitgeteilt. Ist der Auftrag abgeschlossen, wird dieses vTOS mitgeteilt und das Fahrzeug kann weiter verplant werden. Ein Auftrag besteht aus einer AuftragsID, einer Kennung und einem individuellen Teil.

AuftragsID
Jedem Auftrag/Befehl wird eine ID zugewiesen. Die eM-Plant-Software quittiert nach Abschluss eines Auftrages mit deren ID. Damit weiß vTos, um welchen Auftrag es sich handelt.

Kennung
Anhand der Kennung wird festgelegt, um welches Fahrzeug es sich handelt. Nachfolgend eine Liste der Kennungen:

1. **VC Containerbewegung**
2. **Schiffsbewegung**
3. **Zugbewegung**
4. **EmPlant Steuerung**
5. **Containerbrücke-Kranbewegung**
6. **Containerbrücke-Containerbewegung**
7. **Transtainer-Bewegung**
8. **Transtainer-Containerbewegung**
9. **Einfache VC Bewegung**

Individuelle Werte Je nach Kennung weichen die Werte nach dieser ab. Bei einer Schiffsbewegung reicht es, das Ziel anzugeben. Bei einer Vancarrierbewegung muss sowohlt der Ort, an dem der abzuholende Container steht, als auch der Ort, an dem der Container abgestellt werden soll, angegeben werden. Nachfolgend werden die individuellen Werte, sortiert nach Kennungen, erklärt:

1. **VC Containerbewegung**
   [*AuftragsID*, *Kennung*, VC_ID, Abholpunkt, Zielort, ContainerID]
   Der Abholpunkt ist der derzeitige Standort des abzuholenden Containers. Der Zielort ist der Ort, an dem der Container abgestellt werden soll. Beide Orte können auf dem Schiff, Zug, LKW, Dock oder Yard sein.

2. **Schiffsbewegung**
   [*AuftragsID*, *Kennung*, SchiffID, Länge, Ziel]
   Anhand der SchiffID und Schiffslänge weiß eM-Plant, um welches Schiff es sich handelt. Anhand der Schiffslänge wird die Größe des Schiffes berechnet. Das Ziel ist der Punkt, an dem der Bug des Schiffes stehen soll.

3. **Zugbewegung**
   [*AuftragsID*, *Kennung*, ZugID, Länge, Ziel]
   Wie bei der Schiffsbewegung kann anhand der ZugID und der Länge (Anhand der Länge wird die Anzahl der Waggons berechnet) der Zug erstellt und identifiziert werden.

4. **EmPlant Steuerung**
   [*AuftragsID*, *Kennung*, Ereignis]
   Dies ist der einzige Befehl, der kein Transportauftrag ist. Mit der Kennung 4 kann man die eM-Plant-Simulation steuern. Dabei gibt es folgende Ereignisse: Start, Stop und Reset.

5. **Containerbrücke-Kranbewegung**
   [*AuftragsID*, *Kennung*, KranID, Ziel]
   Mit dieser Kennung lassen sich Containerbrücken an einem bestimmten Dock (Ziel) rangieren.

6. **Containerbrücke-Containerbewegung**
   [*AuftragsID*, *Kennung*, KranID, Art, Ziel, ContainerID]
   Hiermit kann man eine Containerbrücke mit dem Attribut "Art" beauftragen, einen Container vom Schiff ab- oder aufzuladen.

7. **Transtainer-Bewegung**
   [*AuftragsID*, *Kennung*, TranstainerID, Ziel]
   Wie bei den Containerbrücken, lassen sich auch die Transtainer zu einem bestimmten Ziel an den Gleisen fahren.

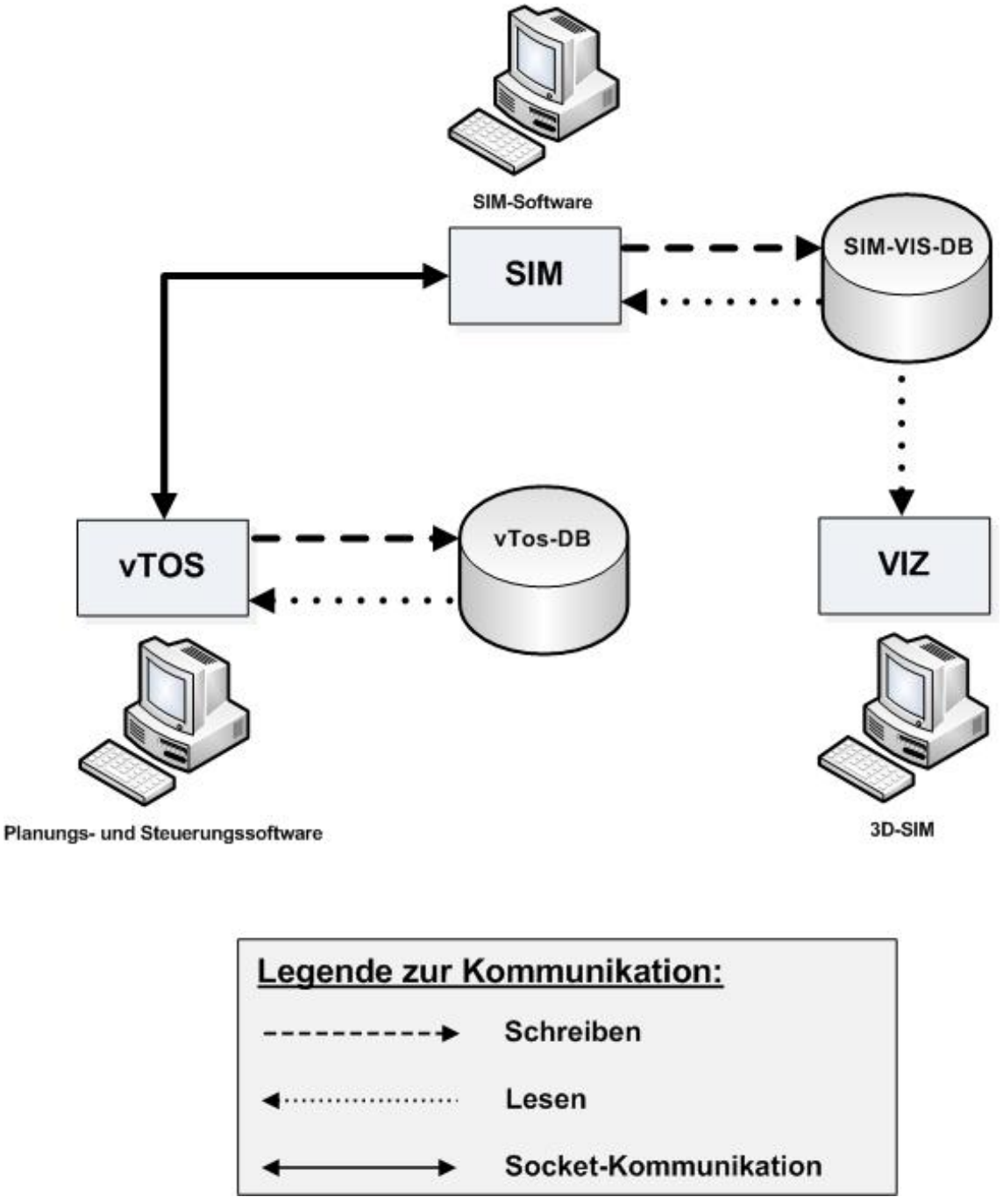


Abbildung 5.55: Schnittstellen der Teilsysteme Planung (vTOS), Simulation (SIM) und Visualisierung(VIS)

8. **Transtainer-Containerbewegung**
   [*AuftragsID*, *Kennung*, KranID, Art, Ziel, containerID]
   Bei der Containerbewegung der Transtainer, muss eine eindeutige KranID angegeben werden. Danach folgt das Attribut Art. Dieses gibt mit 1 an das ein Container aufgeladen werden muss, bzw. mit 2, dass ein Container abgeladen werden soll. Ziel und ContainerID sind identisch zu den oberen Attributen.

9. **Einfache VC Bewegung**
   [*AuftragsID*, *Kennung*, VC_ID, Zielort]
   Mit dieser Kennung kann man Vancarrier beauftragen, zu einem bestimmten Ort zu fahren, ohne dabei einen Container zu transportieren.

5.6.2 eM-Plant - Visualisierung

Die Kommunikation zwischen der Simulationssoftware "eM-Plant" und der Visualisierungs-Komponente findet über eine MySQL-Datenbank statt. Aus dieser werden zum Beispiel die Containerbelegung des Hafengeländes und der Schiffe, alle Objekte, die auf dem Hafengelände dargestellt werden, die Animationen der sich zu bewegenden Fahrzeuge sowie alle auszuführenden Aufträge gelesen.

6. Zusammenfassung und Ausblick

In diesem Kapitel sollen die Ergebnisse der Projektgruppe „Virtual Port II“ vorgestellt werden. Dabei sollen sowohl positive, wie auch negative Aspekte der Ergebnisse aufgezeigt werden. Im Anschluss folgt ein kurzer Ausblick.

6.1 Zusammenfassung

Ziel dieser Projektgruppe war die Erweiterung der Ergebnisse aus der vorherigen Projektgruppe „Virtual Port I“. Diese hatte die Grundlage gelegt, für die Visualisierung des Hafens, sowie deren Simulation über eM-Plant. Hier galt es nun eine Planungskomponente einzufügen und die Visualisierung zu erweitern. Dadurch entstanden drei Bereiche für die Projektgruppe, die Planungskomponente vTos, die Simulationskomponente mit Hilfe von eM-Plant und die Visualisierungskomponente 3D.

Die Planungskomponente musste komplett neu entwickelt werden. Aufgrund der geänderten Anforderungen wurde die Oberfläche und Steuerung komplett neu gestaltet.
Mit dieser neu umgesetzten Planungskomponente ist es nun möglich die verschiedenen Planungskomponenten, wie Liegeplatzplanung, Yardplanung und einer Containerbrückeneinsatzplanung, zu vergleichen. Zu jeder Planungskomponente existieren bereits mehrere alternative Algorithmen, diese sind durch externe Schnittstellen erweiterbar. Über diese Schnittstelle könnte man eine Stauplanung, welche die Verteilung der Container auf einem Schiff berechnet, umsetzen.
Für den Vergleich verschiedener Planungsalgorithmen steht eine statistische Analyse zur Verfügung.

Als Schnittstelle zwischen vTOS und der 3D Visualisierung dient die Simulationskomponente, realisiert mit Hilfe von eM-Plant. Das neue Modell der Simulationskomponente baut auf den Ergebnissen der Vorgruppe auf.
Die Kommunikation von vTOS zu eM-Plant geschieht über Sockets. Diese funktionieren zuverlässig über zwei Kanäle, jeweils Ein- und Ausgang. Als Schnittstelle zwischen der Visualisierung und eM-Plant dient eine Datenbank, die weitestgehend aus der Vorgruppe übernommen wurde.
Alle simulierten Fahrzeuge (Züge, Schiffe usw.) werden über eine neu entwickelte Logik gesteuert. Insbesondere die Wegberechnung der simulierten Fahrzeuge ist stark optimiert worden, sodass keine größeren Ladezeiten zur Wegberechnung entstehen. Des Weiteren ist die Wartbarkeit der neuen Logik durch einen übersichtlicheren Aufbau verbessert worden.
Mit der von uns genutzten eM-Plant Version ist es leider nicht möglich, eine wirkliche Echtzeitsimulation zu simulieren. Da der Zeitablauf in dem simuliertem Modell von der Anzahl simulierter Objekte abhängt und daher nicht konstant ist. Für weitere Simulationsprojekte könnte man aufgrund der ungenügenden Zeitsteuerung eine eigene Implementierung dem Einsatz von eM-Plant vorziehen.

Bei der Visualisierungskomponente konnte der Großteil aus dem vorherigen Projekt übernommen werden. Insbesondere die Datenbankdefinition wurde nur um Kleinigkeiten erweitert. Die Modelle der dargestellten Visualisierungsobjekte wurden fast alle erneuert

und zum Großteil auch erweitert. So gibt es nun eine Skybox, welche die Darstellung eines Himmels ermöglicht und so die Gesamtvisualisierung deutlich verbessert. Die neu hinzugekommen Fahrzeuge, wie Züge und Transtainer sind entsprechend animiert worden. Da die Visualisierung durch den 3D Effekt sehr rechenintensiv ist, ist eine gesamte Simulation mit allen Kränen, VC und maximaler Auslastung der Lager nicht möglich gewesen. Ein neuer Rechner mit leistungsfähigerer Hardware sollte dieses Problem beheben.

Zusammenfassend haben wir das Ziel der Simulierung des Jade Weser Ports größtenteils erreicht. Man kann nun statistische Aussagen über die Anordnung des Hafens durch den Vergleich verschiedener Simulationsläufe erhalten. Dies ist auch komplett in einer 3D Visualisierung darstellbar, hierdurch sind Schwächen und Stärken einer Simulation besser nachvollziehbar.

6.2 Ausblick

Das fertige vTOS unterstützt derzeit weder Stauplanung noch Planung für Logistik im Hinterland. D.h. Züge, Transtainer und LKWs werden bei der Planung nicht berücksichtigt. Züge und Transtainer sind jedoch in der Simulation und der Visualisierung schon vorhanden und können über vorhandene Schnittstellen in vTOS eingefügt werden.

Aufgrund der schon im vorherigen Abschnitt genannten Probleme von eM-Plant, bietet es sich an dieser Stelle an, weitere Arbeiten anzustreben. Diese Probleme sind zum einen, dass die Zeit nicht konstant abläuft und damit den Ablauf in Echtzeit verhindert. Zum anderen gibt es die Möglichkeit in eM-Plant, die Simulationsgeschwindigkeit rasant zu erhöhen. Hier kommt es jedoch häufig zu Fehlern, da die Kommunikation nicht schnell genug abgearbeitet werden kann. Kommt es z.B. zu einer Verzögerung bei der Socketkommunikation von mehreren Sekunden, können dies in der Simulation schon mehrere Minuten ausmachen. Dabei bleibt hier die offene Frage zurück, ob diese Probleme durch eine neuere Version von eM-Plant gelöst werden können oder ob eine eigene Implementierung sinnvoller wäre.

Eine sinnvolle Möglichkeit die Visualisierung zu erweitern, wäre die Planungsdaten und deren Analyse mit in die 3D-Darstellung aufzunehmen. Außerdem sind noch keine Pfade für die LKWs vorhanden. Zur Erweiterung würde sich auch anbieten, Wettereffekte oder dynamisches Wasser einzufügen.

A. Glossar

- **Automate Guided Vehicle** - Fahrerloses Transportfahrzeug mit eigenem Fahrantrieb, das automatisch gesteuert und berührungslos geführt wird.
- **Bay** - Bezeichnung der einzelnen Stellplätze der horizontalen Ebene auf einem Containerschiff, beginnend von vorne nach hinten.
- **Bayplan** - Eine List mit zu ladenden bzw. zu löschenden Containern einer Bay, die zusätzlich Informationen über die Container, das Schiff und den Yard enthält.
- **Containerbrücke** - Eine Krananlage mit einer Laufkatze und einer schienengebundenen Transportanlage für die verladung von ISO-Containern in Containerterminals.
- **Containerterminal** - Spezialhafen nur für Container
- **CT** - Abkürzung für Container Terminal
- **Dock** - Anlegebereich der Schiffe, in dem Frachtgut transportiert wird.
- **eM-Plant** - Simulationssoftware von Tecnomatix.
- **Face** - Fläche in einem 3D Modell
- **Feederschiff** - Frachtschiff, das als Verteiler und Zubringer für die großen Containerterminals in Seehäfen und Tiefwasserhäfen und Frachten zu kleinen Küsten-, Kanal- und Binnenhäfen transportiert.
- **FEU** - Fourty-foot equivalent unit. Hierbei handelt es sich um einen Standardcontainer, der auch als 40-Fuß-ISO-Container bezeichnet wird.
- **IPO - Kurve** - Die IPO-Kurven sind Bezier-Kurven, deren Eckpunkte von den IPO-Keys gebildet werden.
- **JadeWeserPort** - Containertiefwasserhafen in Wilhelmshaven.
- **Kai** - Befestigtes Ufer zum Laden und Löschen von Schiffen.
- **Katze** - Bewegliches Kranbauteil, das entlang eines Trägers fahren kann.
- **Ladeliste** - Eine Liste mit Containern, die geladen werden müssen.
- **LOD** - Level of Detail. Drei verschiedene Detail-Grade für die Darstellung in der Visualisierung
- **Panamax** - Schiffsgröße
- **RMG** - Rubber Mounted Gantry
- **RTG** - Rubber Tired Gantry

- **Shifting Factor** - Messgröße, die angibt, wie viele Umstauvorgänge nicht zu löschender Container auf einem Schiff stattgefunden haben.
- **Ship schedule** - Liste der in den Containerterminal einlaufenden Schiffe.
- **Skybox** - Skyboxen werden verwendet, um die Umgebung in 3D-Szenen darzustellen. Dazu zählen z.B. Landschaften im Hintergrund sowie Sonne, Himmel und Wolken.
- **Stack** - EIn Teilbereich bzw. eine Unterteilungseinheit eines Yards, auf dem eine bestimmte Anzahl an Containern lagert.
- **TEU** - Twenty-foot equivalent unit. Hierbei handelt es sich um einen Standardcontainer, der auch als 20-Fuß-ISO-Container bezeichnet wird.
- **Vancarrier** - Ein spezielles Umschlaggerät frr ISO-Container, das als Transportfahrzeug auf Containerterminals eingesetzt wird.
- **Virtual Port** - Der Oberbegriff für das Gesamtsystem der Projektgruppe.
- **VPort** - vgl. *Virtual Port*
- **vTOS** - Planungssoftware, die von der Projektgruppe entwickelt wurde.
- **Xith3D** - Für die Visualisierung genutzte 3D-Engine.
- **Yard** - Lagerplatz oder auch Containerdepot (container yard).

B. Seminarausarbeitungen

In diesem Kapitel befinden sich alle im Wintersemester 2008/2009 erstellten Seminararbeiten der Projektgruppenmitglieder. Die Ausarbeitungen stehen in engem thematischen Zusammenhang zu den drei Teilsystemen vTOS, Simulation und Visualisierung und dienten als Einarbeitung in die übergeordnete Thematik des Seeverkehrs und der Seeverkehrswirtschaft.

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

Planungsprobleme in Häfen

vorgelegt von:

Gereon Fössing

Oldenburg, den 27.11.2008

Inhaltsverzeichnis

1 Einleitung

Moderne Container Häfen, im Englischen Container Terminals (CTs), haben eine sehr wichtige Rolle im heutigen Transportnetzwerk. Die Wichtigkeit dieser CTs zur wirtschaftlichen und sozialen Dimension ihrer Region bzw. Landes ist Signifikant [Hen06]. Durch die immer größer werdende Transportmenge im Containerverkehr, steigen auch die logistischen Anforderungen dieser Häfen. Der Bau weiterer Anlagen bzw. die technische Erweiterung bestehender Anlagen zum Erhöhen der Kapazitäten ist sehr teuer. Daher kommt dem Optimieren bzw. Planen der Arbeitsprozesse, um so die Kapazitäten zu steigern, eine immer bedeutendere Aufgabe zu [MS].
Außerdem kann durch das Optimieren der Arbeitsprozesse die *ship turnaround time* gesenkt werden. Das ist die Zeit, die ein Schiff im Hafen verbringen muss. Diese Zeit ist für die Reedereien sehr teuer, eine Senkung führt daher zu hoher Kundenzufriedenheit. Ein höherer Durchsatz von Containern und eine höhere Kundenzufriedenheit, führt zu höheren Gewinnen.

Diese Arbeit wird in Kapitel 2 einen kurzen Überblick über die verschiedenen Planungsprobleme und auch deren Beziehungen zueinander geben. Danach folgt ein kurzer Absatz über den geplanten JadeWeserPort.
In Kapitel 3 wird das Ship-To-Shore Planning Problem behandelt. Hier wird auf die Grundlagen eingegangen und auch ein Lösungsansatz in Teilen vorgestellt.
In Kapitel 4 wird das Crane Scheduling Problem vorgestellt. Dabei werden auch die wichtigsten Aspekte dieses Problems erläutert. Danach wird wieder auf einen Lösungsansatz in Teilen eingegangen.
Zum Schluss folgt ein kurzes Fazit, in dem die Ergebnisse vorgestellt und neue Fragen aufgeworfen werden.

2 Grundlagen

2.1 Einteilung der Planungsprobleme

Es ist sehr schwierig einen optimalen Ablauf der Arbeitsprozesse in einem Hafen zu planen. Sie werden daher auch als Planungsprobleme bezeichnet. Die Planungsprobleme in Häfen werden nach [MS] typischerweise zusammengefasst in:

- Ship-To-Shore Planning Problem (SSP): Das Verplanen von Containerschiffen zu den Liegeplätzen über eine geschätzten Zeitraum.
- Crane Scheduling Problem (CSP): Den Abschnitten bzw. Bays eines Containerschiffs müssen Containerbrücken zugeordnet werden. Außerdem muss eine Ent- und Beladereihenfolge festgelegt werden.

- Transportation Planning Problem (TPP): Regelt den gesamten Verkehr im Hafen, z.B. der Transport zwischen Containerbrücke und Lager durch die Van Carrier.
- Storage Location Problem (SLP): Container müssen auf dem Container Depot optimal gelagert werden. Die Zuteilung des Lagerorts erfolgt anhand bestimmter Bedingungen, z.B. Fälligkeit oder benötigter Stromanschluss.

In der Literatur gibt es auch andere Einteilung der Planungsprobleme, für diese Arbeit soll aber die obere verwendet werden. Des Weiteren wird sich diese Arbeit ausschließlich mit den Planungsproblemen SSP und CSP befassen.

2.2 Abhängigkeiten der Planungsprobleme

Zwischen den einzelnen Planungsproblemen bestehen Abhängigkeiten. Es ist nicht möglich ein Problem vollständig separat zu lösen. Um einzelne Aspekte der Planungsprobleme zu lösen sind Informationen bzw. Ergebnisse der anderen Planungsprobleme notwendig. Der Abhängigkeitsgraph in Abbildung 1 zeigt, zwischen welchen Problemen es Beziehungen gibt.

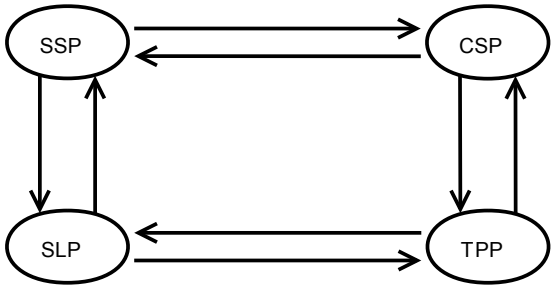


Abbildung 1: Abhängigkeitsgraph der Planungsprobleme

G (V,E)
V {Menge der Knoten, jeder repräsentiert ein Planungsproblem}
E {Menge der Kanten, jede Kante repräsentiert eine Abhängigkeit zwischen zwei Knoten}

Daraus lassen sich folgende Beziehungen in Anlehnung an [MS] ableiten:

- SSP - CSP: Das SSP benötigt Informationen über die Containerbrückenzuordnung, um so die geschätzte Liegezeit zu ermitteln. Dabei wird die durchschnittliche Umschlagsmenge der zugeteilten Containerbrücken auf die gesamte Umschlagsmenge umgerechnet. Das CSP dagegen benötigt Informationen darüber, wo welches Schiff wann liegt, um so die Containerbrücken zuzuordnen.

- CSP - TPP: Das CSP benötigt Informationen darüber wann welches Fahrzeug zur Verfügung steht, um Container von den Containerbrücken zu den Lagerorten zu transportieren. Dagegen Benötigt das TPP Informationen über den Standort der zu bedienenden Containerbrücke.
- TPP - SLP: Das TPP benötigt Informationen über den Lagerort eines zu transportierenden Containers und das SLP benötigt Informationen darüber, wann welches Fahrzeug zum transportieren von Containern zur Verfügung steht.
- SLP - SSP: Das Storage Location Problem benötigt Informationen darüber, wann wo welches Containerschiff liegt, um so die dazugehörigen Container zu verschicken. Im Gegenzug benötigt das SSP Informationen darüber, wo die Container gelagert werden müssen, um so die Schiffe möglichst nahe am Yard zu positionieren.

2.3 Der JadeWeserPort

In diesem Abschnitt sollen kurz einige Grundlagen des JadeWeserPorts, im Bezug auf die in dieser Arbeit behandelten Planungsprobleme, vorgestellten werden. Die Abbildung 2 zeigt den Grundriss des JadeWeserPorts. Die zu verplanende Kailänge beträgt 1725 Meter mit einem Tide unabhängigen Tiefgang von 16,5 Metern. Damit ergibt sich bei einer Gleichverteilung, eine Liegeplatzlänge von 430 Metern. Zum Vergleich, das weltweit größte Containerschiff ist die Emma Maersk. Diese hat eine Länge von 397 und einen Tiefgang von 16 Metern [Gro08]. Desweiteren wird dieser Hafen über 16 Containerbrücken verfügen die von 68 Van Carriern beliefert werden können [KG08].

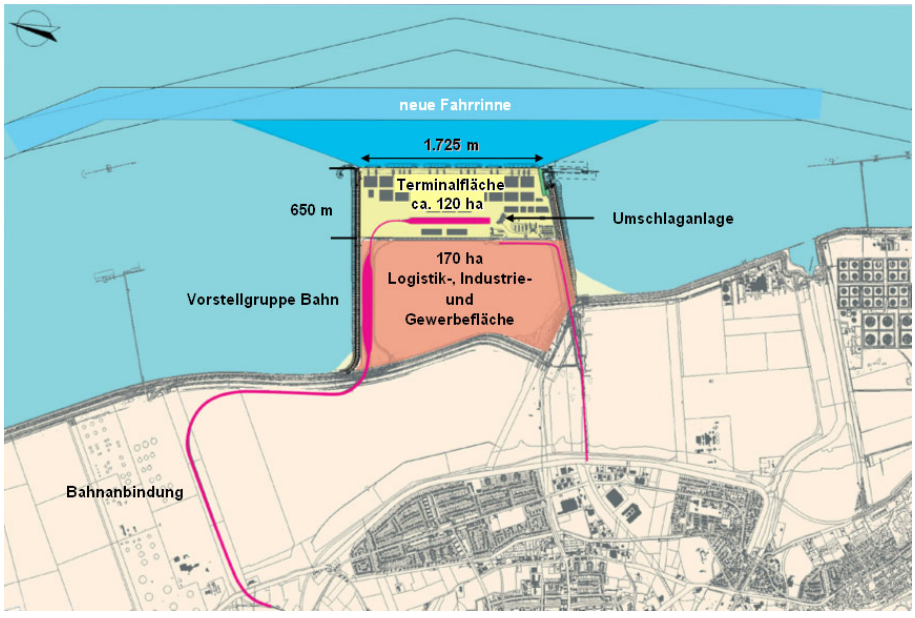


Abbildung 2: Grundriss des JadeWeserPorts [Jad08]

3 Ship-To-Shore Planning Problem

3.1 Problembeschreibung

Beschreibung:

Beim Ship-To-Shore Planning (SSPl) geht es darum, Schiffe zu einem bestimmten Zeitpunkt über einen geschätzten Zeitraum zu verplanen. Dabei sollten Wartezeiten, wenn möglich, vermieden werden [Mat06]. Außerdem darf es nicht zum so genannten *starvation* (Begriff aus der Prozessoptimierung) kommen. D.h., dass kein Schiff verhungern bzw. ewig warten darf. Auch wenn nach [SV07] ein absichtliches Warten lassen einiger Containerschiffe zu einem höherem Durchfluss führen kann. Dies ist jedoch in der Realität keine wirklich gute Option.
Die Verplanung der Containerschiffe findet typischerweise 24 Stunden vor ihrer Ankunft statt. Diese wird eingeleitet durch dessen Anmeldung, mit der auch ein Dokument verschickt wird, in dem die zu entladenen und zu beladenen Container stehen [**?**, Hen06] Dabei muss noch erwähnt werden, dass die Schiffseigner bei der Anmeldung meist schon angeben, ab wann sie Liegezeit fordern. Die geschätzte Liegezeit ergibt sich dann durch die Auswertung von Statistiken mit Erfahrungswerten. Dabei wird die zu ent- und beladene Containermenge, die Containerbrückenauslastung und -einteilung und die Containerbrückengeschwindigkeit betrachtet. Es ist wichtig, dass die Planung relativ exakt ist, da bei großen Kunden häufig eine Garantie darauf gegeben wird, dass sie zu einem bestimmt Termin anlegen können [CTW07].

Ziele:

Zielgrößen sind hier zum einen die Minimierung der Durchflusszeit und zum anderen Kostensenkungen. Durchflusszeit kann in diesem Zusammenhang auch als *ship turnaround time* verstanden werden. Die Minimierung dieser Zeit ist beim Ship-To-Shore Planning nur durch ein Senken der Wartezeit und einer möglichst nahen Positionierung an den Containerstellplätzen möglich. Kostensenkungen hingegen können hier nur durch eine optimale Ressourcennutzung erreicht werden. Die Liegeplätze müssen möglichst optimal verteilt werden, um so nicht nutzbare Kai Fläche zu vermeiden. Diese Fläche entsteht immer genau dann, wenn Schiffe so ungünstig verteilt werden, dass zwischen diesen mehr Platz ist als notwendig, aber zu wenig, um ein weiteres Schiff dort zu positionieren.

Schwierigkeiten:

Probleme entstehen hierbei vor allem dadurch, dass die Containerschiffe auch optimal über die Zeit verteilt werden müssen. D.h., ein momentan guter Plan kann sich innerhalb kürzester Zeit stark verschlechtern. Wenn z.B. einige Containerschiffe sehr ungünstig ablegen und neue dazukommen, kann es passieren, dass die neuen nicht anlegen können, weil

sie länger sind und die Lücken der abgelegten Containerschiffe nicht ausreichen.
Eine weitere Schwierigkeit liegt darin Schiffe, wie oben beschrieben, möglichst nahe zu deren Containerstellplätzen zu positionieren. Dies scheint bei dem JadeWeserPort bei kurzer Betrachtung unwichtig, da dieser kein sehr großer Hafen ist. Bei näherer Betrachtung ist dies jedoch auch bei kleinen Häfen wichtig, denn eine Verkürzung der Distanz zwischen den beiden Punkten von z.B. 100 Metern führt bei 1000 Container schon zu einer Strecke von 100.000 Metern. Wird das auf den gesamten Jahresumsatz hochgerechnet, erreicht die Zahl eine enorme Größe.
Dies beides führt dazu, dass das SSP NP-Vollständig ist, d.h., es muss nicht versucht werden die beste Lösung zu finden, sondern eine für die benötigten Zwecke optimale Lösung.

Formale Problembeschreibung:

Zur Lösung des SSP wird ein Planungssystem bzw. ein Algorithmus benötigt, der ankommende Schiffe möglichst optimal verplant. Abstrakt beschrieben heißt das, der Algorithmus muss in der Lage sein, einen bestimmten Input I zu einem bestimmten Output O zu transformieren und dies unter der Einhaltung von festgelegten Constraints C. Diese sind in Anlehnung an [FFSV05]:

- Input: Anzahl und Länge der Liegeplätze, der Standort freier Containerbrücken, Informationen über das zu verplanende Schiff (vor allem Länge und Anzahl der zu verladenen Container) und der Lagerplatz der Container.
- Output: Liegeplatz und die Liegezeit.
- Constrains: Dies sind Bedingungen die eingehalten werden müssen. Darunter fallen z.B. es muss mit der vorhandenen Menge an Arbeitsmaterial geplant werden, einzuhaltender Abstand zu anderen Schiffen oder einfach der maximal erlaubte Tiefgang.

3.2 Lösungsansatz

In diesem Abschnitt soll ein möglicher Lösungsansatz vorgestellt werden. Zum Speichern der Informationen, wann wo ein Schiff anlegen kann, eignet sich ein Array in Form eines Time-Space-Diagramms hervorragend. Die Abbildung 3 zeigt ein solches Time-Space-Diagramm. Die X-Achse zeigt die Liegeplatz Abschnitte, wobei die Wahl der Größe der Abschnitte hier abhängig ist von der Genauigkeit. Das gleiche gilt für die Y-Achse, die die fortschreitende Zeit abbildet. Somit wäre jeder Zeit und Liegeplatz Abschnitt ein Feld im Array. In den Feldern des Arrays würden sich auch leicht Verweise auf weitere Informationen des jeweiligen Containerschiffs speichern lassen.
Das direkte Verplanen der Containerschiffe soll in diesem Lösungsansatz in zwei Methoden geteilt werden:

- planeSchiff

- optimierePlan

Abbildung 3: Time-Space-Diagram

Diese Untergliederung hat einen ganz bestimmten Zweck. Da zum Zeitpunkt der Anmeldung eines Schiffs zum Zeitpunkt X noch nicht bekannt ist, welche weiteren Schiffe sich zum Zeitpunkt X anmelden, soll nach der Anmeldung erst einmal die Methode *planeSchiff* aufgerufen werden. Diese verplant das Schiff erstmal, ohne dabei großen Wert auf die Effektivität zu legen. Um den Plan effektiv zu machen, soll die Methode *otimierePLan* eingesetzt werden. Diese könnte z.B. optimal aufgerufen werden wenn es zu Wartezeit oder Verspätung kommt, um den Plan dann so umzustellen, dass die Verspätung berücksichtigt wird bzw. es zu keiner Wartezeit kommt. Diese Ausarbeitung beschäftigt sich jedoch lediglich mit der Methode *planeSchiff*. Diese sieht in Pseudocode folgendermaßen aus:

```
Prozedur: planeSchiff
Zweck: Nach Anmeldung eines Schiffs, wird diese Prozedur
       aufgerufen und das Schiff verplant.
Parameter: int Liegeplätze, boolean schiffzuverplanen, int
           Schiffslänge m und int geplante Ankunft t
WHILE schiffZuVerplanen
   WHILE anzahlLiegeplätze = 4 AND schiffZuVerplanen
      wähle Liegeplatz (z.B. nach Random Regel)
      IF m = freieFLächeLiegeplatz(t)
         ermittele Schiffsliegezeit      ->Schnittstelle zu CSP
         IF Array mit ermittelter Schiffsliegezeit frei ist
            plane Schiff zu Liegeplatz(t)
            schiffZuVerplanen = false
         END IF
      END IF
      anzahlLiegeplätze++
   END WHILE
   anzahlLiegeplätze = 1
```

```
    t++
END WHILE
```

Der obige Pseudocode ist zwar sehr stark vereinfacht, trotzdem verdeutlicht er auf recht einfache Weise, wie Schiffe schnell verplant werden können. Mit der ersten While Schleife wird sichergestellt, dass dies solange ausgeführt wird, bis das Schiff verplant ist. Dabei wird bei jedem Schleifendurchlauf die geplante Ankunft um eine Zeiteinheit erhöht. Die zweite While Schleife dient dazu jeden Liegeplatz einmal zu Prüfen. Die Wahl des jeweils zu prüfenden Liegeplatzes könnte nach der Random Regel erfolgen, um so eine Gleichverteilung sicherzustellen. Wobei jedoch dann sichergestellt werden muss, dass jeder Liegeplatz immer nur einmal durch die Random Regel gewählt wird. Mit der If Bedingung wird sichergestellt, dass wenn ein Liegeplatz genug freie Fläche hat (auch über die Zeit), das Schiff hier verplant werden kann. Ist das nicht der Fall wird das Ganze wiederholt.

4 Crane Scheduling Problem

4.1 Problembeschreibung

Beschreibung:

Beim Crane Scheduling Problem (CSP) geht es um zwei zentrale Fragen. Wo und wann kommt welche Containerbrücke zum Einsatz und wie soll be- und entladen werden? Also dieses Problem lässt sich in 2 Teilbereiche spalten.

- Containerbrückenzuordnung
- Be- und Entladevorgang

Containerbrücken sind große, auf Schienen laufende Portalkräne, mit einem see- und landseitigen Ausleger. Dabei ist der seeseitige Ausleger hochklappbar, um so ein- und auslaufende Schiffe nicht zu behindern. Unter dem landseitigen Ausleger befindet sich meist ein Zwischenlager. Hier können z.B. zu beladene Container kurzzeitig gelagert und auch schon vorsortiert werden. Meist werden hier auch die Lukendeckel der Schiffe zwischengelagert [?, Bri05] Das Bild in Abbildung 4 zeigt mehrere Containerbrücken des Unternehmens Eurogate in Hamburg. Diese werden auch am JadeWeserPort eingesetzt.

Ziele:

Zielgrößen sind hier vor allem die optimale Ressourcenausnutzung, die Sicherheit des Schiffs und die be- und entlade Geschwindigkeit. Die optimale Ressourcenausnutzung ist vor allem in der optimalen Verteilung der Containerbrücken begründet. Sicherheit des Schiffs ist die höchste einzuhaltende Zielgröße. Diese kann durch falsches Be- und Entladen gefährdet werden oder dadurch, dass die Gesamtlast am Ende nicht korrekt verteilt

Abbildung 4: Containerbrücken in Hamburg von [Haf08]

wurde. Eine falsche Verteilung der Gesamtlast könnte, z.B. in einem Sturm, verheerende Auswirkungen haben. Was passieren kann, wenn ein Schiff falsch beladen wird, kann man der Abbildung 5 entnehmen. Die Zielgröße Geschwindigkeit steht im direkten Zusammenhang mit der *ship turnaround time* und ist somit ein direktes Maß der Produktivität.

Schwierigkeiten:

Bei der Verplanung der Containerbrücken ist die größte Schwierigkeit, dass es nur ein Gleisbett für die auf Schienen fahrenden Containerbrücken gibt. D.h., Containerbrücken können sich nicht kreuzen. Wenn ein Kran einen anderen passieren möchte, ist dies nicht möglich, stattdessen müssen diese die Aufgaben tauschen. Das Umstellen kostet wertvolle Zeit.
Bei der Be- und Entladung gibt es mehrere Probleme, die gelöst werden müssen. Diese Probleme sind unter anderem:

- Stabilität beim Laden: Bei der Be- und Entladung muss ständig die Stabilität gewährleistet sein. Dies kann z.B. ermöglicht werden, in dem nach einer bestimmten Anzahl Container eine Stabilitätsprüfung stattfindet.
- Stabilität der Gesamtlast: Die Gesamtlast muss so verteilt werden, dass die ma-

ximale Stabilität gewährleistet wird. Hier müssen schon vor dem Laden komplexe Stabilitätsberechnungen stattfinden um diese dann beim Laden berücksichtigen zu können.

- Effektivität beim Laden: Beim Entladen kann es vorkommen, dass es möglich wird gleichzeitig zu Beladen. Beispielsweise wenn ein Stack bzw. Stellplatz frei ist, also dort kein Container mehr entladen werden muss. Es sollte also bei jedem Entladevorgang auch geprüft werden, ob es eine Möglichkeit zum Beladen gibt.

Abbildung 5: Unfälle die beim Ent- und Beladen entstehen können [, Goe08]

Formale Problembeschreibung:

Zur Problemlösung mit Input, Output und Constrains werden folgende Informationen benötigt:

- Input: Liegeplatz bzw. Standort des zu bearbeitenden Schiffs und die Ladeliste mit den entsprechenden Informationen über die zu ent- und beladenen Container.
- Output: Containerbrückenzuordnung, be- und entlade Reihenfolge und die Verteilung der Gesamtlast.
- Constrains: Darunter fallen z.B. Kräne können nicht aneinander vorbeifahren, es kann gleichzeitig be- und entladen werden oder die maximal mögliche Entladegeschwindigkeit kann nicht überschritten werden.

4.2 Lösungsansatz

In diesem Abschnitt soll ein möglicher Lösungsansatz vorgestellt werden. Dieser könnte so aussehen, dass eine Klasse *Terminalplanung* erstellt wird. Diese würde Methoden zur Lösung des gesamten CSP besitzen. Eine mögliche Gliederung der Methoden könnte diese sein:

1. Zuordnen der Kräne zu den Schiffen:
   Wird aufgerufen um eine Verteilung der Kräne zu organisieren. Wobei es auch ein Steuerungssystem geben muss, welches die Kräne laufend überwacht und bei Bedarf dann diese Methode aufruft.

2. Entladen der Schiffe:
   Wahrscheinlich die komplexeste Methode, da es hier dazu kommen kann, das Schiffe während sie entladen werden auch teilweise schon beladen werden können.

3. Beladen der Schiffe:
   Einfaches Beladen der Schiffe. Hier muss lediglich die Reihenfolge festgelegt werden und die Stabilität fortlaufend geprüft werden.

Um eine Vorstellung davon zu bekommen, wie das ganze aussehen könnte, soll an dieser Stelle die Methode zum Entladen der Schiffe in Pseudocode vorgestellt werden:

```
Prozedur: entladeSchiff
Parameter: int Anzahl zu entladener Container und
           int Prioritätsliste (z.B. nach Fälligkeit)
WHILE anzahlContainer
   IF freier Stack vorhanden
                wähle freien Stack (z.B. nach der Random Regel)
      prüfe Stabilität des Schiffs bei Theoretischer Beladung
      IF Stabilitätgewährleistet
         setze eine Belade Anfrage   ->Schnittstelle
      ELSE
         setzte Stack auf belegt
      END IF
   END IF
   wähle Container (z.B. nach Priorität und Random Regel)
   prüfe Stabilität des Schiffs bei Theoretischer Entnahme
   IF Stabilitätgewährleistet
      IF Container zum Beladen da   ->Schnittstelle
         belade Container auf geprüften Stack
      END IF
      entlade Container
      anzahlContainer--
   ELSE
      Priorität des Containers++
   END IF
END WHILE
```

Diese Methode wird von einer Containerbrücke solange ausgeführt wie es zu entladene Container gibt. Jeder Durchlauf dieser Methode entspricht dem Entladen eines Containers. Am

Anfang wird immer geprüft, ob es möglich ist einen Container zu beladen, beispielsweise weil ein Stack bzw. Stellplatz kein Container mehr enthält oder nur noch welche die nicht abgeladen werden müssen. Wenn es eine freie Stelle gibt und es wegen der Stabilität keine Probleme gibt, wird eine Beladeanfrage gestellt. Dies wäre auch eine Schnittstelle zum Lager bzw. Transport, je nachdem welches System dafür zuständig wäre. Da dies nun geklärt wurde, wird nun ganz normal ein Container zum Entladen ausgewählt. Die Wahl könnte durch die Prioritäts und Random Regel erfolgen. Also jedem Container wird eine Priorität zugeordnet, z.B. durch die Fälligkeit des Weitertransports. Gibt es mehrere Container mit gleicher Priorität, könnte nach der Random Regel ausgewählt werden. Nachdem auch hier die Stabilität bei theoretischer Beladung geprüft wurde und diese gewährleistet ist, wird der ausgewählte Container entladen und die Anzahl der zu entladenen Container um eins reduziert. Ansonsten wird abgebrochen und die Priorität des Containers erhöht, um so eine wiederholte Wahl zu vermeiden. Die letzte IF Anweisung ist lediglich dazu da um zu prüfen, ob ein Container da ist, der Beladen werden kann. Wenn dies der Fall ist, wird dieser im nächsten Zyklus beladen.

5 Fazit

Beide hier vorgestellten Planungsprobleme sind NP-Vollständig. Das finden der besten Lösung ist daher meist nicht möglich oder sehr aufwendig. Außerdem können die Probleme nicht getrennt von einander gelöst werden. Dies erhöht den Komplexitätsgrad deutlich. Trotzdem macht es Sinn die Planungsprobleme einzuteilen und wie verbundene Zahnräder miteinander laufen zu lassen.
In dieser Arbeit wurden einige Konzepte zur Lösung der Probleme Ship-To-Shore Planning und Crane Scheduling vorgestellt. Einige mögliche Methoden sind in Pseudocode vorgestellt worden, wobei dann auch einige Schnittstellen zu den jeweils anderen Planungsproblemen aufgezeigt werden konnten. Es wurden aber auch neue Fragen aufgeworfen, z.B. wie die Stabilität und damit die Sicherheit des Schiffs garantiert werden kann oder wie es möglich ist, die Stabilität fortlaufend zu überwachen?

Literatur

[Bri05] B. Brinkmann. *Seehafen: Planung und Entwurf*. Springer Verlag, 2005.

[CTW07] S. T. Christopher, Chung-Piaw Teo, and Kwok Kee Wei. *Supply Chain Analysis: A Handbook on the Interaction of Information, System and Optimization*. Springer Verlag, 2007.

[FFSV05] F. Ferschl, A. Fink, G. Schneidereit, and S. Voß. *Grundlagen der Wirtschaftsinformatik*. Birkhaeuser, 2005.

[Gro08] Maersk Group. http://www.maerskline.com, 2008. Letzter Besuch: 05.12.08.

[Haf08] Hamburger Hafen. http://www.joshushund.com/hh03.html, 2008. Letzter Besuch: 05.12.08.

[Hen06] L. E. Henesey. *Multi-Agent Systems for Container Terminal Management*. Blekinge Institute of Technology, 2006.

[Jad08] JadeWeserPort. Lageplan. http://www.jadeweserport.de/cms/index.php?idcat=25, 2008. Letzter Besuch: 04.12.08.

[KG08] JadeWeserPort Realisierungs GmbH Co. KG. http://www.jadeweserport.de, 2008. Letzter Besuch: 05.12.08.

[Mat06] D. C. Mattfeld. *The Management of Transshipment Terminals: Decision Support for Terminal Operations in Finished Vehicle Supply Chains*. Birkhaeuser, 2006.

[MS] L. Meier and R. Schumann. Coordination of interdependent planning systems. Case Study.

[Sti08] Goerlitz Stiftung. http://www.goerlitz-stiftung.de/bsp06.html, 2008. Letzter Besuch: 05.12.08.

[SV07] R. Stahlbock and S. Voß. *Operations research at container terminals: a literature update*. Springer-Verlag, 2007.

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

Planungsprobleme in Häfen

vorgelegt von:

Bernd Ahlering

Oldenburg, den 27.12.2008

Inhaltsverzeichnis

1 Einleitung

Heutige Containerschiffe haben immer mehr Kapazität und können durch Ihre wachsende Größe nicht mehr an jedem Hafen anlegen. Der Jade Weser Port spielt dabei eine wichtige Rolle. An diesem Tiefwasserhaven können u.A. Container von großen Schiffen auf kleine umgeschlagen werden. Die kleineren Schiffe beliefern dann kleinere Häfen. Das erspart Zeit und dadurch Kosten.
Auch die Häfen wachsen mit den Schiffen. Durch Optimierungen der Be- und Entladeprozesse kann dabei erhebliche Zeit gespart werden. Je kürzer das Schiff am Hafen verbringt (ship turnaround time), desto weniger Kosten fallen für die Reedereien an.
In Kapitel 2 werden die vier Planungsprobleme, die an Häfen auftreten, grob erläutert. Anschließend folgen allgemeine Daten über den Jade Weser Port. In Kapitel 3 wird auf eines der vier vorgestellten Planungsproblemen, das Storage location Problem, näher eingegangen. Von mir selbst entwickelte Lösungsansätze über zur Lösung des Problems beim Jade Weser Port werden in Kapitel 4 beschrieben. Zum Schluss folgt ein Fazit und offene Fragen, die sich aus dieser Ausarbeitung ergeben haben.

2 Grundlagen

2.1 Planungsprobleme in Häfen

In Häfen müssen viele Prozesse geplant und koordiniert werden. Die Planung der Container ist nach [MS06] in vier Kategorien (Planungsprobleme) aufgeteilt:

1. **Ship to Shore Planning Problem:** Die Zuordnung der Liegeplätze, an denen die Containerschiffe anlegen.

2. **Crane Scheduling Problem:** Die Zuordnung der Containerbrücken zum be- und entladen der Containerschiffe

3. **Transportation Planning Problem:** Die Koordination der Transportmittel, um die Container von der Containerbrücke zum Liegeplatz des Containers zu befördern.

4. **Storage Location Problem:** Die Planung des optimalen Lagerplatzes der Container, um bei der Be- und Entladung eines Containerschiffes möglichst wenig Zeit zu verlieren. Auch das Umlagern von Containern soll vermieden werden.

Zwischen diesen Problemen gibt es auch untereinander Abhängigkeiten.
Ship to Shore Planning Problem und **Crane Scheduling Problem**
Um die Anzahl der Kräne, die die Container vom Schiff abladen, planen zu können, werden Informationen über den Liegeplatz und die Anzahl der Container, die umgeschlagen werden, benötigt.
Crane Scheduling Problem und **Transportation Planning Problem**
Um die Transportmittel koordinieren zu können, werden Informationen über den Lagerort

der Container benötigt. Zudem benötigen die Kräne die Information welche Transportmittel Ihnen zur Verfügung stehen.

Transportation Planning Problem und **Storage Location Problem**

Die Transportmittel benötigen Informationen über den Lagerort der Container und um den Lagerort bestimmen zu können, muss man wissen, welche Transportmittel zur Verfügung stehen.

Storage Location Problem und **Ship to Shore Planning Problem**

Um die Be- und Entladezeit zu minimieren, muss man wissen wo die Container stehen, um den Liegeplatz, der am nahesten bei den Containern liegt, zu ermitteln.

2.2 Der Jade Weser Port

Der Jade Weser Port soll ein Tiefwasserhafen in Wilhelmshaven werden und ist seit September 2001 in Planung. 2008 ist der geplante Baubeginn. Der Container Terminal wird eine Größe von 120ha erreichen und eine Kajenlänge von 1725 Meter haben [KGa]. Insgesamt ist eine Umschlagsmenge der Container von 2,7Mio TEU (Twenty Foot Equivalent Unit, Maßeinheit für einen Container [Lex]) geplant.

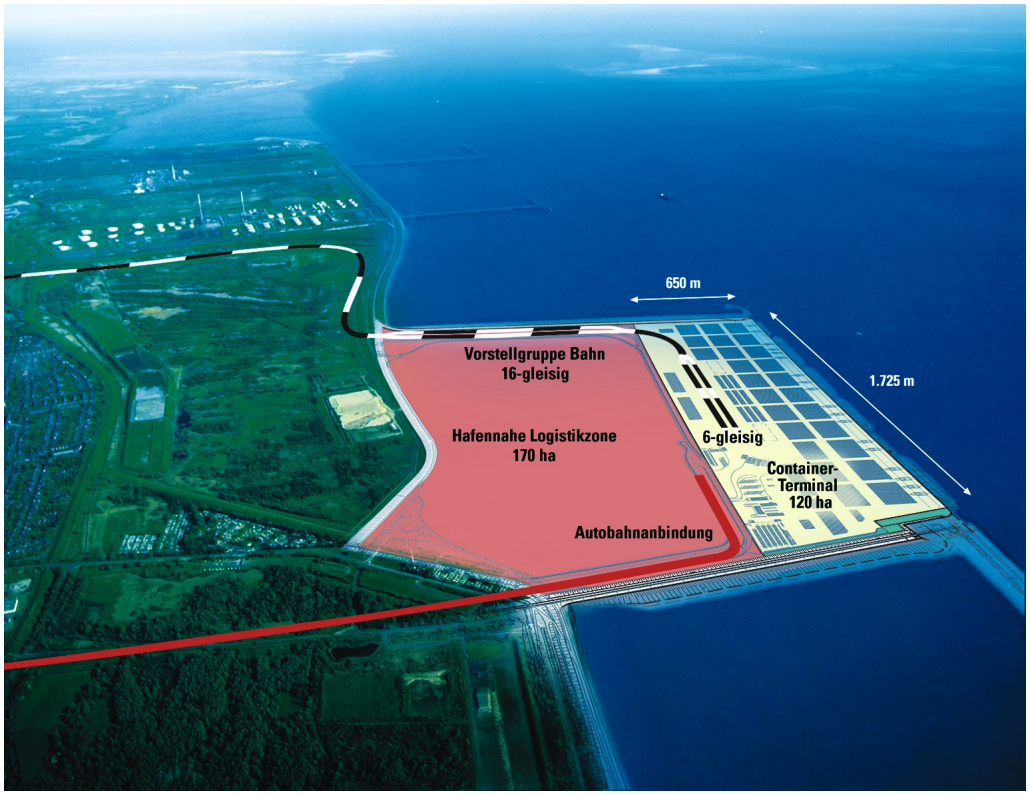


Abbildung 1: Luftaufnahme beschriftet [KGb]

3 Storage location Problem

3.1 Problemstellung

In einem Hafen gibt es viele Abläufe zu optimieren, eines davon ist das Lagern der Container auf dem richtigen Platz (Storage location Problem). Den Lagerplatz der Häfen für die Container nennt man Container Yard [MS06]. Ziel ist es, den Container auf dem Container Yard so zu lagern, dass dieser möglichst wenig umgelagert wird und so nah wie möglich an dem nächsten Transportmittel steht. Um den optimalen Platz für einen Container bestimmen zu können, müssen zwei Faktoren berücksichtigt werden. Zeitlich: Lagerdauer der Container, und örtlich: Standort der Container. Diese beiden Faktoren werden in den nächsten beiden Kapiteln näher erläutert.

3.2 zeitliche Bedingungen

Abbildung 2: Lagerdauer von Containern [Pra]

Jeder Container hat auf dem Hafen eine individuelle Lagerdauer. Auf dem Jade Weser Port sollen später bis zu vier Container übereinander gestapelt werden. Dabei gilt die Lagerdauer des Containers auf dem Hafen zu beachten. Container mit langer Lagerdauer werden nach unten gestellt, und Container mit kurzer Lagerdauer werden auf Container mit längerer Lagerdauer gestellt [Hen06] (siehe Abbildung 2).

3.3 lokale Bedingungen

Jeder Container Yard eines Containerhafens ist anders aufgeteilt. Oft gibt es verschiedene Bereiche. Allgemein kommen folgende Bereiche in Frage:

Ohne Stromanschluss Der größte Bereich eines Container Yards sind für die Standardcontainer ohne Stromanschluss.

Mit Stromanschluss
Da es sehr kostenintensiv und übertrieben ist, jeden möglichen Containerplatz mit einem Stromanschluss auszustatten, gibt es auf dem Container Yard immer einen Bereich mit Stromanschlüssen für Container, die während Ihres Transportes ständig mit Strom versorg werden müssen. Das können z.B. Kühlcontainer für tiefgefrorene Nahrungsmittel sein.

LKW-PLätze
Container, deren nächstes Transportmittel LKW sind, können am besten auf einem seperaten Bereich direkt bei einer Straßenanbindung gelagert werden. Das würde Wartezeiten, die für das Beschaffen des Conatiners aufkommen, einsparen.

Zug-Plätze
Das gleiche gilt auch bei Zug-Plätzen, so können bei Ankunft eines Zuges die Container sofort auf dem Zug geladen werden.

Gefahrengut Plätze
Aus Sicherheitsgründen werden Container mit Gefahrengut auf einem dafür bestimmten Platz gelagert. Dieser Bereich ist zudem Baulich speziell dafür hergerichtet, so dass z.B. auslaufende Öle nicht in das Grundwasser gelagen können.

Feste Plätze für Firmen
Große Firmen bevorzugen es, Ihre Container auf Ihrem eigenen Platz selber zu managen. Es können also auch für solche Fälle Bereiche auf dem Yard einkalkuliert werden.

Plätze für leere Container
Leere Container werden nicht zwischen den vollen Containern gelagert. Das würde die Suche nach einem leeren Container unnötig aufwendig machen. Auch könnte es mit der Statik der aufeinandergestapelten Container probleme geben. Wenn unten leere und oben ein voller Container steht, könnte dieser Turm bei stärkeren Winden leichter umkippen.

Plätze für kaputte Container
Kaputte Container müssen gesondert gelagert werden, da deren Stabilität nicht mehr gewährleistet ist. Zudem muss man für die Reparatur den Container nicht aus dem Lagerplätzen herausholen. Gerade wenn ein Container eine lange Lagerzeit hat, ist dieser meist unter anderen Containern abgestellt. Dann müssten andere Container aufwendig umgestapelt werden, damit der Kaputte repariert werden kann.

3.4 Kombinationen lokaler Bedingungen

Einen Container Yard optimal mit verschiedenen Bereichen zu planen ist sehr kompliziert. Es wäre auch möglich, Bereiche mit verschiedenen Eigenschaften zu kombinieren. So können feste Firmenplätze auch Stromanschlüsse haben. Eine Spedition, die nur über LKW als Transportmittel verfügt, läge optimal an der Straßenanbindung. Container der Deutschen Bahn währen im optimalen Fall direkt bei den Bahngliesen aufgehoben. Man kann jeden Bereich wieder in Unterbereiche einteilen, nur stellt sich sehr schnell die Frage, ob sich der Aufand noch lohnt und ob die Einteilung in so viele kleine Bereiche noch wirtschaftlich ist. Für die digitale Implenentiereung eines Hafens muss entschieden werden, ob man den Container Yard fest nach den Planungen implementiert oder dynamisch, so dass der Benutzer nachträgliche Änderungen in das Softwaresystem eingeben kann.

4 Lösungsanzätze

4.1 Mögliche Bereiche im Jade Weser Port

Eine mögliche Einteilung der Bereiche auf dem Jade Weser Port ist auf Abbildung 3 abgebildet. Die Zugplätze sind direkt in der Nähe der Bahngleise und die LKW-Plätze direkt

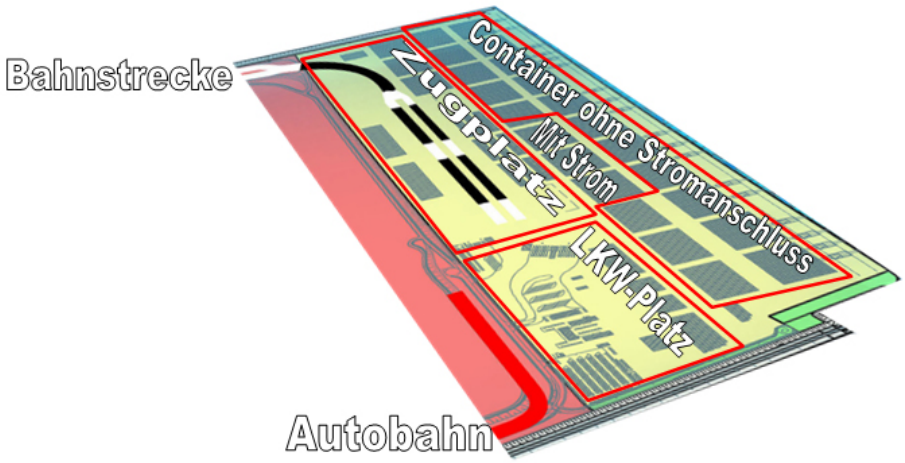


Abbildung 3: mögl. Einteilung der Bereiche [KGb]

bei der Autobahn. Zentral in der Mitte sind Containerplätze mit Stromansschlüssen. So kann man Container aus anderen Bereichen, die einen Stromanschluss benötigen, zentral lagern. Bereiche für Firmen und Gefahrengut wurden auf dieser Abbildung noch nicht berücksichtigt. Ziel ist es, dass man die Software, die in diesem Projekt entwickelt wird, flexibel für mehrere Häfen verwenden kann. Daher empfielt es sich eine Software zu entwickeln, bei der man den Container Yard individuell in Bereiche einteilen kann. Ein Lösungsansatz wird im nächsten Unterkapitel näher erläutert.

4.2 Yard

Der Container Yard jedes Hafens besteht aus drei Dimansionen: Die Länge, Breite und die Höhe (Anzahl der Container übereinander). Es empfielt sich daher ein dreidimensionales Array, in dem jeder einzelne Containerplatz definiert wird, zu erstellen (siehe Abbildung 4).

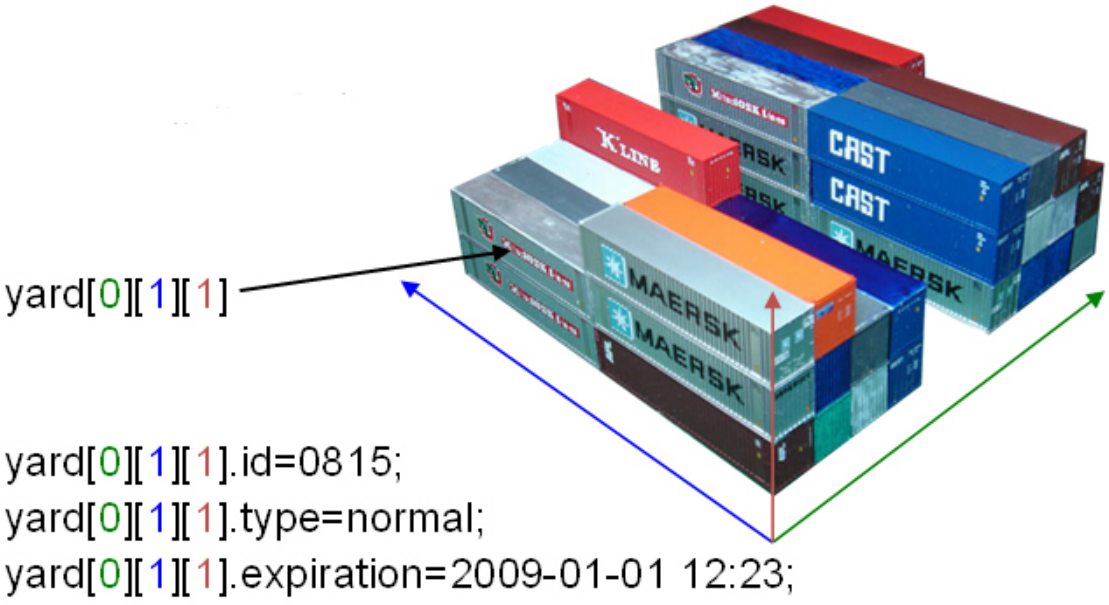


Abbildung 4: Yard 3D [JH]

Jeder Stellplatz benötigt drei Informationen, um eine Planung des optimalen Lagerplatzes eines Containers zu ermöglichen:
ContainerID: Um einen Container auf dem Yard wiederzufinden, wird die ID des Containers gespeichert, steht auf dem Liegeplatz kein Container, bleibt die ID leer.
Typ: In dieser Variablen steht der Typ, für welchen Containertyp dieser Stellplatz geeignet ist.
Abholdatum: Damit der zeitliche Lagerfaktor berücksichtigt werden kann, wird zusätzlich die Information, wann der Container wieder abgeholt wird, gespeichert.

Anhand dieser Daten ist es möglich nur mit den Informationen der Klasse Yard den optimalen Lagerplatz für einen Container zu finden. Zusätzlich benötigt man noch die Informationen, wo der Container abgeladen und wieder aufgeladen wird. Diese Daten müssen jedoch nur für die Berechnung des optimalen Lagerplatzes des eigenen Containers berücksichtigt werden und brauchen nicht extra im Containerlageplan gespeichert werden. Bei kleineren Häfen werden die Weglängen zum Container nicht berücksichtigt. Die Personen, die mit dem Van Carrier die Container zu den Liegeplätzen transportieren, kennen sich auf dem Hafen aus und werden den erfahrungsgemäß schnellsten Weg zu dem Platz benutzen. Die Zeitersparnis wäre daher bei kleineren Häfen minimal.

Da die Anzahl der Container auf einem Yard sehr groß werden kann, ist es wichtig die Datenmenge, die gespeichert werden muss, möglichst gering zu halten. Redundante Informationen würden die Berechnungen unnötig in die Länge treiben.

4.3 Container

Die Container lassen sich ebenfalls als Objekte implementieren. Die wichtigsten Informationen, um einen Container zu verwalten sind:
ID: Die ID dient zur eindeutigen Identifizierung eines Conatainers
Typ: Der Typ des Containers muss bei der Planung mit dem Typ des Containerplatzes auf dem Hafen übereinstimmen.
Abholdatum: Das Abholdatum wird für die Containerplatzplanung benötigt, damit keine Conatiner, die früher abgeholt werden unter dem Container stehen.

Dem Objekt Container können noch zusätzliche Informationen hinzugefügt werden, die nicht wichtig für die Planung des Containerplatzes sind, wie z.B. Das Gewicht, oder der Besitzer des Containers. Diese weiteren Daten kann man jedoch auch in einer zusätzlichen Tabelle in der Datenbank abspeichern.

4.4 Planung

Für das Storage location Problem müssen zwei Methoden implementiert werden. Die aufwändigere Methode, die den Containereingang berechnet und eine weniger aufwändige Methode, die den Containerausgang berechnet.

4.4.1 Containereingang

Die eigentliche Berechnung des optimalen Containerplatzes findet beim Containereingang statt, hier werden viele Informationen benötigt.
Standort des Schiffes: Bei größeren Häfen wird der Standort des Schiffes mit in die Berechnung einbezogen, damit bei der Entladung schon Zeit gespart werden kann, wenn der Weg zum Containerplatz möglichst kurz ist.
Nächstes Transportmittel/Standort des nächsten Transportmittels: Damit der Container möglichst nah am nächsten Transportmittel gelagert werden kann, benötigt man die Information des nächsten Transportmittels und dessen Standort.
Container: Der Container wird als Objekt übergeben und beinhaltet die Informationen ID, Typ und Abholdatum.
Yard Das Objekt Yard beinhaltet den kompletten Lageplan der Container auf dem Yard. In diesem Lageplan sind zudem wichtige Informationen wie die IDs der Container, den Typ der Abstellflächen und das Abholdatum der Container gespeichert.

Anhand dieser Informationen kann nun der optimale Lagerplatz des abzuladenen Containers berechnet werden. In Abbildung 5 ist ein möglicher Ablauf einer Containerplatzbe-

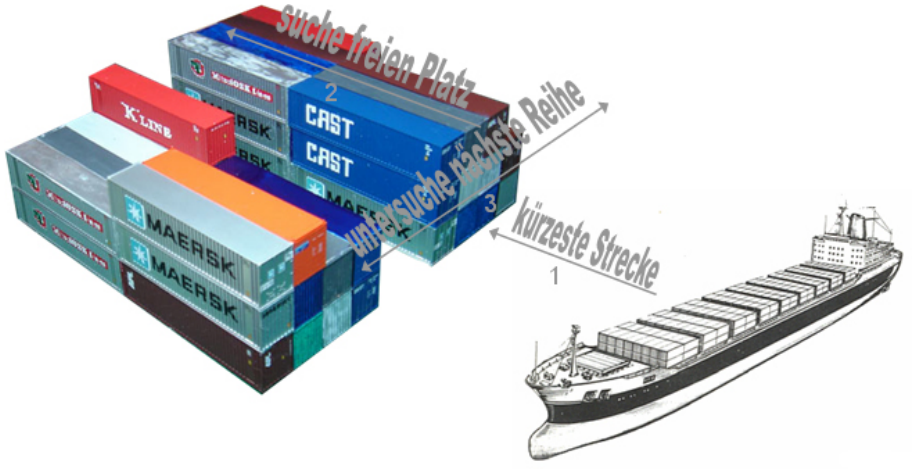


Abbildung 5: mögl. Algorithmus zur Berechnung des Containerplatzes [JH, Hav]

rechnung visualisiert worden. Zuerst wird die Containerreihe, die am nahesten vom Schiff und dem nächsten Transportmittel des Containers liegt, gewählt (1: kürzeste Strecke). Anschließend wird diese Reihe nach freien Containerplätzen durchsucht (2: suche freien PLatz). Ist in dieser Reihe kein geeigneter Platz mehr frei, so wird die gleiche Prozedur mit der benachbarten Reiche durchgeführt (3: untersuche nächste Reihe). Diese Prozedur wird solange duchrgeführt, bis ein freier Containerplatz gefunden wurde. Wenn es keinen Containerplatz mehr geben sollte, muss nochmal durchgerechnet werden, ob man einen Container so umlagern kann, dass die Reihenfolge der zeitlichen Abholungen der Container eingehalten wird. Ist dann immernoch kein Platz gefunden, muss man die zeitlchien Bedingungen ignorieren und den Container auf einen Container, der früher abgeholt wird, stellen. Dieser Container muss dann bei Abholung des darunterliegenden Conatainers umgestapelt werden.
Der Ausgabewert der Methode, die den optimalen Lagerplatz eines Containers berechnet, ist die Platznummer in der Form „3,56,3“. Diese drei Zahlen stehen für die drei Dimensionen. So steht die Zahl für 3. Reihe, 56. Container, 3. Etage. Mit dieser Information kann der Van Carrier den Container zu seinem Platz bringen.

4.4.2 Containerausgang

Da beim Containereingang bereits alle für den optimalen Lagerplatz des Containers relevanten Faktoren berücksichtigt wurden, kann man davon ausgehen, dass der Container auf den optimalen Platz steht. D.h. Die Methode, die den Containerausgang berechnet, muss nur noch eine Abfrage im Objekt Yard (besser vielleicht Abfrage aus der Datenbank) machen, wo der angeforderte Container steht. Als Rückgabewert wird genauso, wie bei der Methode, die den Containereingang berechnet, der Standort des Containers ausgegeben.

5 Fazit

Die Planung des optimalen Lagerplatzes eines Containers ist sehr aufwendig. Daher ist es wichtig, dieses Problem in kleinere Probleme aufzuteilen (siehe 2.1). Das erhöht zudem die Übersichtlichkeit der Berechnungen. Jedoch enstehen dadurch auch Abhängigkeiten, die bei der Planung berücksichtigt werden müssen.
Bei steigenden Umschlagmenegen von Containern wird es immer wichtiger, auch die benötigten Datenmengen möglichst klein zu halten. Redundante Informationen würden die Berechnung nur unnötig verlangsamen. Daher sollte man die Daten auch in mehreren Datenbanktabellen aufteilen, so dass für die Berechnungen nur nötige Informationen geladen werden.
Ob die von mir entwickelten lokalen Lagebedingungen auch im Jade Weser Port berücksichtigt werden, ist schwer zu ermitteln. Jedoch könnte man durch implementieren dieser Bedingungen prüfen, ob man dadurch die Be- und Entladezeiten eines Containerschiffes verürzen kann.

Literatur

[Hav] Lutz Havemann. *http://www.havemann-rrbk.de/page_1170948754093.html*. Letzter Besuch 27.12.08.

[Hen06] L. E. Henesey. *Multi-Agent Systems for Container Terminal Management*. Blekinge Institute of Technology, 2006.

[JH] c/o novuprint Jürgen Hans. *http://www.jaffa1.de/baufortschritt_4.html*. Letzter Besuch 27.12.08.

[KGa] JadeWeserPort Realisierungs GmbH & Co. KG. *http://www.jadeweserport.de/cms/index.php?idcat=25*. Letzter Besuch 27.12.08.

[KGb] JadeWeserPort Realisierungs GmbH & Co. KG. *http://www.jadeweserport.de/cms/upload/downloads/Luftaufnahme_JWP_300dpi.jpg*. Letzter Besuch 27.12.08.

[Lex] Logistik Lexikon. *http://www.logistik-lexikon.de/?main=/ccTiid460*. Letzter Besuch 27.12.08.

[MS06] L. Meier and R. Schumann. *Coordination of independent planning systems*. Birkhaeuser, 2006.

[Pra] Michael Prachensky. *http://www.prachensky.com/michael/projekte/talpino/portal-ortler.php*. Letzter Besuch 27.12.08.

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

Steuerungssysteme am Beispiel des Containerterminal Altenwerder

vorgelegt von:

Uwe Krisch

Oldenburg, den 07.01.2009

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Der Containerterminal Altenwerder (CTA) gehört zu den modernstern der Welt. Er ist einer von drei Terminals in Hamburg und liegt im Stadteil Altenwerder direkt an der Elbe. Neben diesem Terminal gibt es noch die Terminals Burchardkai und Tollerort in Hamburg. Die Hamburg Hafen und Logistik AG hat 74,9 Prozent und die Hapag-Lloyd AG 25,1 Prozent Anteile an den Terminals.
Im Jahr 2007 wurde eine neue Rekordmenge von 6,7 Mio. Standardcontainer (TEU=TwentyfeetEquivalentUnit) umgeschlagen. Geht man davon aus, dass jeder Terminal etwa die gleiche Anzahl Container umgeschlagen hat, ergibt sich für jeden Terminal pro Tag einen Containerumschlag von ca. 6118. Die Container werden i.d.R. immer doppelt verwendet und dies ergibt dann 3059 einzelene Umschlagvorgänge pro Tag und Terminal.

Diese Seminararbeit stellt die Softwarearchitektur des Containerterminal Altenwerder vor und geht dabei insbesondere auf die einzelnen Teilsysteme ein. Dieses Gesamtsystem der Steuerung eines Containerterminals soll verdeutlichen, wie mögliche Lösungen aussehen können und dabei helfen, eigene Lösungen für das Projekt Virtual Port II 2008/09 an der Carl von Ossietzky Universität Oldenburg zu finden. Ein möglicher Lösungsansatz, der aus den Erkenntnissen aus der Softwarearchitektur des Containerterminal Altenwerder entsteht, wird ebefalls vorgestellt.

2 Steuerungssysteme

Der Begriff Steuerungssysteme setzt sich zusammen aus Steuerung und System. Eine Steuerung beeinflusst den Ablauf eines Gerätes oder eines Prozesses. Diese Steuerung erfolgt nach einem bestimmten Plan. Dabei ist die Steuerung von Eingangsgrößen und Zustandsgrößen abhängig.
Eingangsgrößen können folgende sein:

- Anzahl Container
- Zeitpunkt
- Containerkategorie

Wenn der Zeitpunkt z.B. angibt, wann der Container wieder verladen wird, kann dies Einfluss darauf haben an welchen Stellplatz der Container abgestellt wird. Es wäre möglich, einen besonders nahe liegenden Stellplatz zu finden um Wegezeiten zu reduzieren. Die Eingangsgrößen beeinflussen also direkt den Steuerungsprozess.
Eine Ausgangsgrösse beeinflusst ebenfalls den Steuerungsprozess. Ausgangsgrössen können folgende sein:

- Stellplatzbelegung
- Zeitpunkt
- verfügbarer Van-Carrier

Die Steuerung eines Containers kann natürlich nur freie Stellplätze verwenden um einen Container abzustellen und auch nur verfügbarer Van-Carrier verwenden. Neben der Verfügbarkeit wäre aber auch denkbar, Van-Carrier zu verwenden, die sich besonders nahe am zu transportierenden Container befinden, um so wieder Wegezeiten zu reduzieren.

Ein System fasst Elemente zusammen. Diese Elemente sind aufeinander bezogen und haben in einer Weise Wechselwirkungen. Die Zusammenfassung der Elemente ermöglicht es, eine bestimmte Aufgabe zu lösen. Die Elemente werden als eine sinn-oder zweckgebundene Einheite angesehen und grenzen sich gegenüber der sie umgebenden Umwelt ab.
Betrachten wir die einzelnen Komponenten, die zur Softwarearchitektur des Containerterminal Altenwerder gehören, so sind dies die Elemente, die zusammen die Gesamtaufgabe Lösen. Die einzelnen Elemente sind für die einzelnen Steuerungsaufgaben zuständig und insgesamt ergibt sich mit diesen Teilsystem ein Steuerungssystem für den Containerterminal Altenwerder.

3 Architektur des CTA-Softwaresystems

Die Architektur des CTA-Softwaresystems wird im Folgenden verdeutlichen wie eine Gesamtlösung aussehen kann. Das Zusammenwirken der einzelnen Teilsysteme zu einem Gesamtsystem soll Lösungsansätze für das Projekt "Virtual Port II" ermöglichen.
Das Softwaresystem besteht aus mehreren Teilsystemen, die die einzelnen Steuerungsaufgaben realisieren. Kern des Systems ist das Teilsystem für die Terminalsteuerung. Die entwickelten Komponenten wurden in Java umgesetzt.
Eine wesentliche Aufgabe war die Integration der erworbenen Komponenten, sowie der bereits existierenden Komponenten. Für die Geräte und Transportsteuerung wurden Programme entwickelt, die eine Echtzeitoptimierung für alle Transportvorgänge im Terminal erlauben. Dies ermöglicht es auf kurzfriste Planänderungen oder Störungen reagiert zu können und dadurch Zeitverluste möglichst gering zu halten.

3.1 Entwicklungsprojekt

Die Planung des Softwaresystems begann mitte 1999; das war ca. ein Jahr vor Beginn des Tiefbaus. Von Mai bis September des gleichen Jahres wurde eine Architekturstudie aufgestellt, die dazu diente Risiken zu minimieren. Nach Ende der Studie wurde die endgültige Architektur gewählt und die Entwicklung begann Anfang 2000.
Mit allen Beschaffungen, angefangen von den Entwicklungswerkzeugen bis zu einigen fertigen Teilsystemen und der vollständigen Entwicklung aller eigenen Komponenten inkl. Integration und Testen, dauerte die gesamte Entwicklung 2,5 Jahre. Dabei entstanden durch selbstgeschriebene Komponenten ca. 1,5 Mio. Codezeilen. Weitere ca. 1 Mio. Codezeilen hatten die Kaufkomponenten.

Abbildung 1: Containerterminal Altenwerder, Quelle: HPA

Die Abbildung 1 zeigt den Containerterminal Altenwerder mit seinen Komponenten Containerterminal, KLV-Bahnbhof (Bahnhof für den kombinierten Ladungsverkehr), Güterverkehrszentrum, Autobahnanschluss, Bahnanschluss und Vorbahnhof.

3.2 Technologie

Das CTA-Steuerungssystem basiert auf Java-Technologie, d.h. alle entwickelten Komponenten wurden in Java realisiert. Das Container-Basis-System (in der Sprache M geschrieben) wurde übernommen und wie auch bei den Kaufkomponenten eine entsprechende Schnittstelle implementiert. Das zentrale Informationssystem nutzt Java Enterprise Edition Technologien. Das Gerätesteuerungsystem der automatischen Transporte basiert auf asynchronen JMS-Nachrichten (Java Message Service). JMS-Nachrichten bilden auch die Middleware zur Integration der verschiedenen teilweise zugekauften Komponenten. In einem extra Abschnitt wird genauer auf die Funktionsweise des Java Message Service eingegangen.
Die Integration der Umschlaggeräte (Van Carrier, Krane etc.) wurde durch einen Gerätemanager realisiert. Der Gerätemanager übernimmt das Session-Management, Auf- und Abbau der Kommunikation zu den Geräten (OSI-Layer 5). Zur Kommunikation mit JMS-Nachrichten, übersetzt der Gerätemanager die Socket-Diagramme; die Übersetzung läuft in beide Richtungen. Eine weitere Aufgabe ist die gesamte Überwachung der Geräteverbindungen.

3.3 Allgemeine Architekturmerkmale

Die Abbildung 2 zeigt die Komponenten der Architektur sowie die Kommunikation der Komponenten untereinander (JMS-Nachrichten).

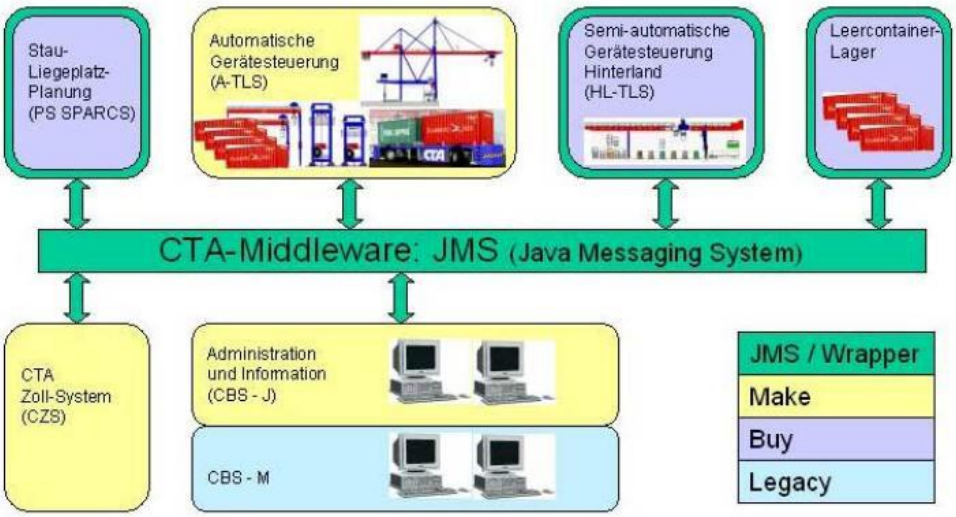


Abbildung 2: Softwarearchitektur des Containerterminal Altenwerder, Quelle: Spindel - HHLA-Präsentation CBS

Die CTA-Middleware stellt die Kommunikationsschnittstelle zwischen den Komponenten dar. Da zur Kommunikation zwischen den Teilsystemen JMS-Nachrichten verwendet werden, mussten Wrapper für die Kaufkomponenten implementiert werden. Damit wurde eine einheitliche Schnittstelle erzeugt.
Die Abbildung 2 zeigt die Komponenten aufgeteilt in selbst entwickelte Komponenten, Kaufkomponenten und Altsystem.

Selbstentwickelte Komponenten sind:

- Automatische Gerätesteuerung (A-TLS)
- CTA-Zoll-System (CZS)
- Administration und Information (CBS-J)

Kaufkomponenten sind:

- Stau-Liegeplatzplanung
- Semiautomatische Gerätesteuerung Hinterland (HL-TLS)
- Leercontainerlager

Das Altsystem besteht aus dem Container-Basis-System (CBS-M), dass in M entwickelt wurde und so in die Softwarearchitektur integriert wurde.

3.4 Teilsysteme

Die einzelnen Teilsysteme der Softwarearchitektur des Containerterminal Altenwerder lösen jeweils eigene Steuerungsproblem und erfüllen zusammen die Gesamtaufgabe der Steuerung des CTA. Im Folgenden werden nun die einzelnen Teilsysteme in ihrer Funktion sowie dem Aufbau vorgestellt.

3.4.1 Container-Basis-System (CBS)

Das Container-Basis-System des CTA ist das zentrale Informationssystem und läuft auf dem zentralen CTA-Server. Es besteht aus dem Altsystem CBS-M und dem Teilsystem CBS-J. Funktional ist das CBS in ausführbare Komponenten und unterstützende Administrations-Komponenten unterteilt.
Die unterstützenden Administrations-Komponenten ermöglichen es Reeder, Spediteure, Behörden etc. auf die für sie jeweiligen relevanten Daten zugreifen zu können bzw. stellt die Daten zur Verfügung. Der Zoll benötigt ebenfalls und regelmäßig Daten vom CBS.

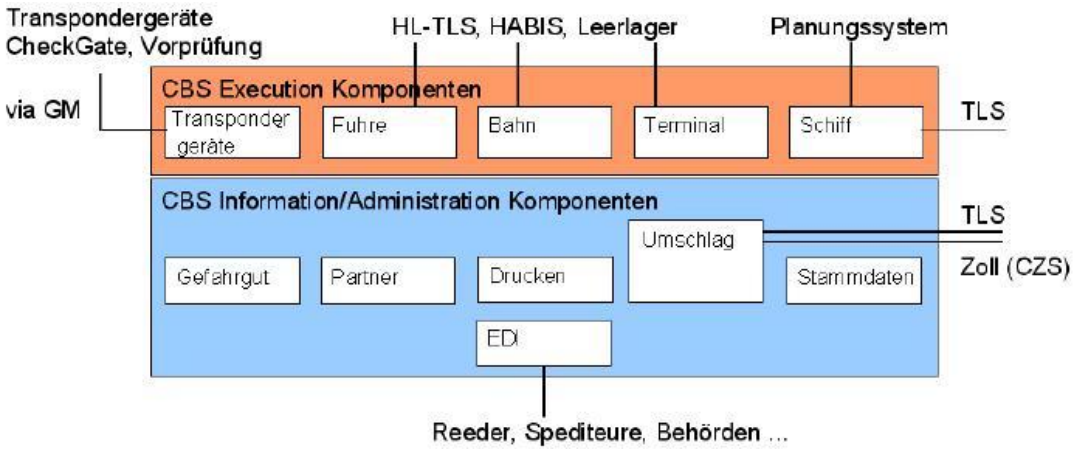


Abbildung 3: Container-Basis-System des CTA, Quelle: Spindel - HHLA-Präsentation CBS

Zu den Informations-/Administrations-Komponenten gehören:

- Gefahrgut Partner
- Drucken
- EDI
- Umschlag
- Stammdaten

Die funktionalen Komponenten sind Systeme, die weitere Steuerungsaufgaben realisieren. Dazu gehören:

- Transponder Geräte
- Fuhre
- Bahn
- Terminal
- Schiff

Das CBS läuft auf dem zentralen CTA- Server.

3.4.2 Automatik-TLS (A-TLS)

Die Aufgabe des Automatik-Teilsystem ist die Überwachung der Transporte. Dazu existieren die zwei abstrakten Objekte Transportvorgang und Transportauftrag. Ein Transportvorgang ist ein kompletter Containertransport wie z.B. der Transport von der Containerbrücke bis zum Lager. Ein Transportauftrag dagegen ist ein Teil des Transports, den eines der Geräte erledigen muss.
Daraus ergibt sich eine Gliederung des A-TLS in Vorgangskomponente und Auftragskomponente. Die Vorgangskomponente legt die verschiedenen globalen Transportvorgänge an. Dazu gehören die Prozesse “start“, “überwachen“ und “optimieren“. Die Auftragskomponent ist bezogen auf ein Gerät zu betrachten.

Funktional lassen sich zwei Optimierungsprozesse unterscheiden. Die globale Optimierung wird von der Vorgangskomponente durchgeführt und die lokale Optimierung von der Auftragskomponente. Bei der globalen Optimierung wird eine Entscheidung zwischen alternativen Lagerorten mit Berücksichtigung der Stellplatzqualität und der Belastung der betroffenen Geräte getroffen. Bei der lokale Optimierung wird im Sinne einer Sequenzierung die optimale Reihenfolge der Transportaufträge für jedes Umschlagsgerät ermittelt.

Weiterhin gehört zum A-TLS der Gerätemanager. Dieser dient den Geräten zum An- und Abmelden sowie der Überwachung im Rahmen der Auftragskomponente. Die Kommunikation mit der Auftragskomponente findet mittels Transformation der Datagramme der Geräte in JMS-Nachrichten statt.

Wie alle Teilsysteme nutzt auch das A-TLS für die Kommunikation JMS-Nachrichten. Um einen gewissen Vorlauf anlegen zu können und auch um unplanmäßige Verzögerungen auffangen zu können wurde ein Warteschlangensystem implementiert. Aktive nebenläufige Objekte tragen Zustände und führen Threads oder Prozesse.
Vorteile dieser Umsetzung des A-TLS ist die Robustheit des Systems, die durch eine lose Kopplung der aktiven Objekte erreicht wird. Die Verwendung von Nachrichten und das Prinzip der Warteschlange sind ebenfalls Faktoren für die Robustheit des Systems. Im Bedarfsfall kann somit ein einzelnes Objekt beendet und neu gestartet werden, ohne dass die Gesamtfunktion beeinflusst wird.

Das Automatik-Teilsystem befindet sich auf dem zentralen CTA-Server.

3.4.3 Hinterland-TLS (H-TLS)

Das Hinterland-TLS ist für den landseitigen Umschlag der Container zuständig. Hier werden die Container gesteuert, die durch Lkw oder Bahn umgeschlagen werden. Das H-TLS steuert zudem die halbautomatischen Bahnkrane und disponiert die Lkw. Dieses Teilsystem ist eine Kaufkomponente und wurde durch Implementierung eines Wrappers, der mit der JMS-Middleware kommuniziert, integriert.
Auch das Hinterland-TLS läuft auf dem zentralen CTA-Server.

3.4.4 Container-Zoll-System (CZS)

Das Container-Zoll-System ist eine selbstentwickelte Komponente, die auf dem zentralen CTA-Server läuft. Diese Komponente kommuniziert mit dem ATLAS-System (Automatisiertes Tarif- und Lokales Zoll-Abwicklungs-System) der Zollverwaltung und wird somit im warehousing-Prozess benötigt.

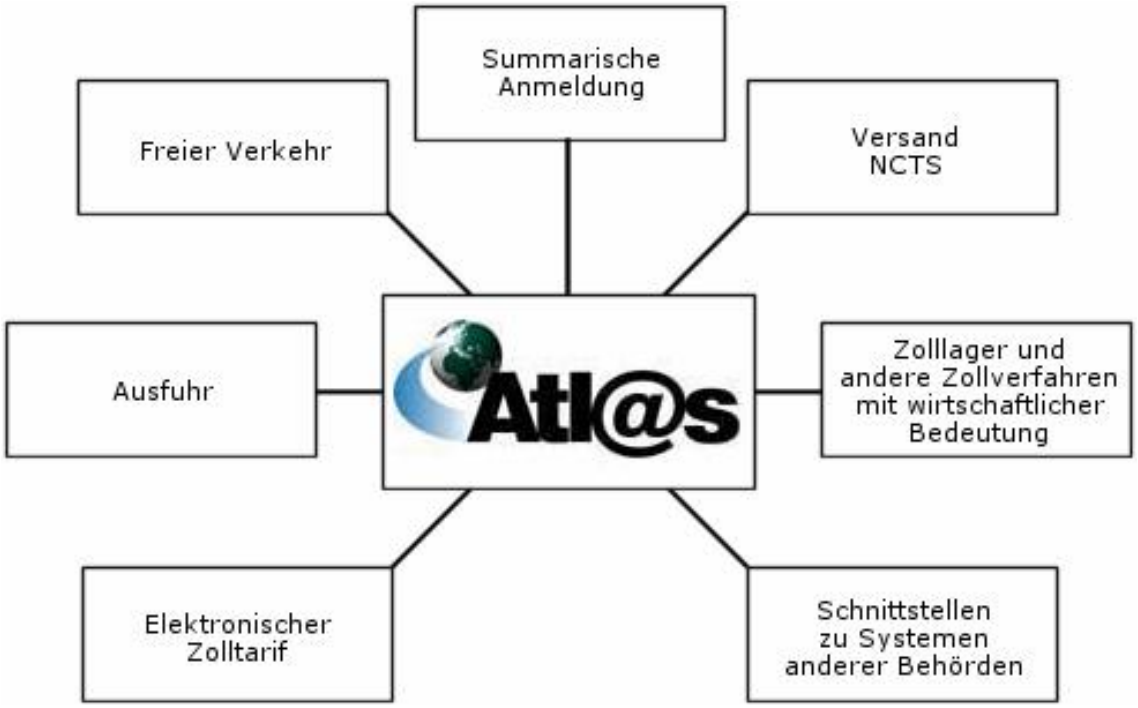


Abbildung 4: ALTAS-System der Zollverwaltung, Quelle: www.zoll.de

Das ATLAS-System automatisiert und beschleunigt die Zollanmeldungen und Verwaltungsakte dadurch, dass die herkömmlichen schriftlichen Vorgänge durch elektronische Vorgänge ersetzt werden.

3.4.5 Planungssystem (SPARCS)

Das Planungssystem SPARCS wurde von der Firma Navis gekauft und kommuniziert mittels einem Wrapper und der JMS-Middleware mit den anderen Teilsystemen. Diese Komponente läuft auf einem speziellen eigenen Windows-Server.
Das Planungssystem ist für die Planung der Schiffsbe- und entladung sowie der Einsatzplanung der wasserseitigen Containerbrücken zuständig.

3.4.6 JMS-Middleware

Als Kommunikationskomponente kommt das Java-Messenger-System zum Einsatz. Alle Teilsysteme kommunizieren über dieses System, das auch auf dem zentralen CTA-Server läuft.
Die verschiedenen Teilsysteme bilden zusammen das Gesamtsystem des Containerterminal Altenwerder. Da nicht alle Komponenten in Java entwickelt wurden, sondern einige als Kaufkomponenten und andere als vorhandene Altsysteme integriert wurden, mussten für diese Komponenten entsprechende Wrapper implementiert werden, damit der Nachrichtenaustausch einheitlich mit dem Java-Messenger-System realisiert werden konnte. Diese Wrapper übersetzen die einzelen Systemkommunikationsschnittstellen nach JMS und zurück.
Das Java-Messanger-System ist eine Message Oriented Middleware, die Punkt-zu-Punkt Verbindungen, Broadcast Verbindungen und eine Zustellgarantie liefert. Die Zustellgarantie benötigt keinen Handshake. Eine Synchronisation sowie das Warten auf den Empfänger sind nicht nötig. Dadurch wird das Lastverhalten der beteiligten Systeme entkoppelt.

4 Infrastruktur

Alle Teilsysteme des CTA-Softwaresystems wurden lediglich auf zwei Server verteilt. Auf dem zentralen CTA-Server, der sich im Rechenzentrum der HHLA in der Hamburger Speicherstadt befindet. Der zweite Server ist ein spezieller Windows-Server auf dem nur das Planungssystem SPARCS läuft.

Durch diese zentrale Installation werden Installation- und Betreuungskosten im Vergleich zu einer dezentralen Installation gering gehalten. Zur Datenübertragung zwischen den Servern und Clients wird ein performates WAN mit 34MBit/s verwendet.
Die Kommunikation in Richtung Containerterminal Altenwerder überträgt Kommandos an die Geräte, Bildschirminhalte der Clients und Statusinformationen über alle Transportvorgänge und aufträge die vom A-TLS übertragen werden. In Richtung des Rechenzentrums werden die Statusmeldungen der Geräte übertragen.

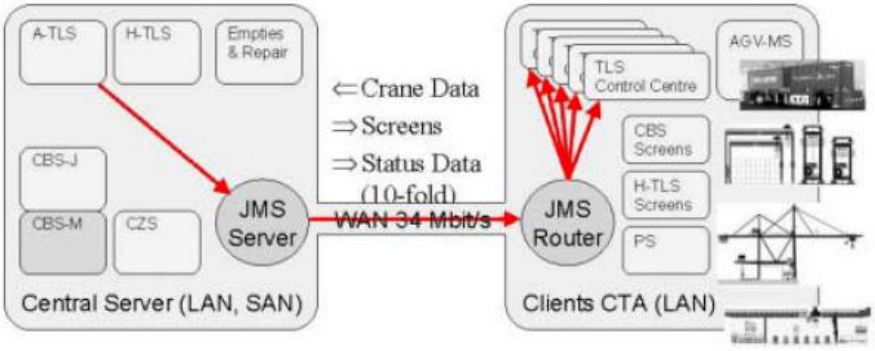


Abbildung 5: Kommunikation des CTA-Softwaresystems, Quelle: Krasemann, Spindel, Softwarearchitektur des CTA

In Richtung CTA überträgt nur das Automatik-TLS die Nachrichten, die zunächst an den JMS-Server auf dem zentralen CTA-Server gesendet werden, an die Clients. Der JMS-Server sendet dann über das 34MBit/s WAN die Daten an den JMS Router, der die Nachrichten an die jeweiligen Clients verteilt.

5 CTA und Virtual Port II

Die Softwarearchitektur des Containerterminal Altenwerder stellt eine Gesamtlösung für eine Containerterminal-Steuerung dar. Für das Projekt Virtual Port II an der Carl von Ossietzky Universität 2008/09 können daraus Lösungsansätze gewonnen werden. Es wird klar, welche Komponenten unbedingt benötigt werden. So werden z.B. benötigt:

- Container
- Container-Stellplätze
- Ladesequenz für Transportaufträge und -vorgänge
- Löschsequenz für Transportaufträge und -vorgänge
- Transportfahrzeuge
- Übergabestellfläche

Mit diesen Komponenten lässt sich ein Tranportprozess realisieren: Ein Container, der durch einen Auftrag über eine Übergabestelle mittels einem Transportmittel an einen Container-Stellplatz gelangt. Die jeweiligen Aufträge können geladen und dann ausgeführt und nach Ausführung gelöscht werden. Aber auch im Sinne einer Echtzeitoptimierung können Aufträge ohne Ausführung gelöscht oder verändert werden.

Das folgende Diagramm zeigt einen Ansatz, der nach dem Referenzmodell von D.Pump 1999 erstellt wurde.

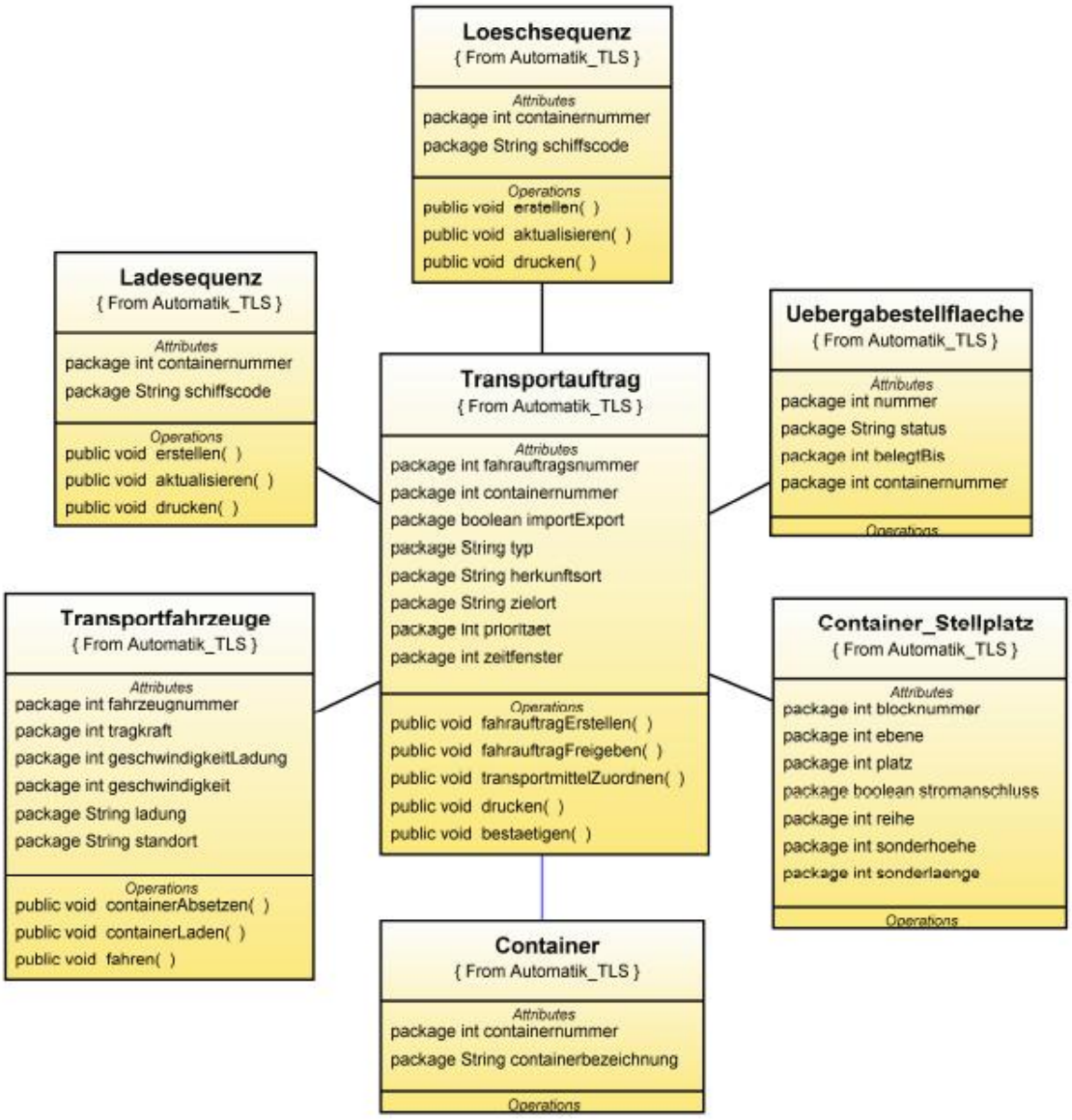


Abbildung 6: Containertransport für den Virtual Port II, Quelle: D. Pump 1999, verändert

Ein Szenario mit dem Ansatz Containertransport könnte wie folgt aussehen:
Nachdem eine Containerliste eingelesen wurde, kann das System die Aufträge erstellen. Alternativ können Containerlisten oder einzelne Container manuell in das Terminal-Operation-System (TOS) eingegeben werden (dazu ist eine entsprechende Anpassung des TOS nötig). Die Steuerungssoftware erstellt dann Transportaufträge und lädt diese in einer optimierten weise(Berechnung unter Berücksichtigung relevanter Zustands- und Ausgangsgrössen zur Erstellung einer Auftragssequenz). Ein Transportauftrag enthält dann alle für den Transport relevanten Daten. Dieser Transportauftrag wird dann in der Datenbank gespeichert.

Die Tranportaufträge werden dann abgearbeitet und danach gelöscht. In der Datenbank werden die Positionen und die transportierten Container gespeichert. Dies ist eine Variante bei der keine Kommunikation zwischen den Teilsystemen stattfindet, so wie das System bereits vorliegt (Kommunikation über die Datenbank). Eine direkte Kommunikation würde bedeuten, dass die Tranportaufträge direkt zu eM-Plant (übernimmt Optimierung der Transportfahrzeuge) geschickt werden und nach Abarbeitung eine Rückmeldung an das TOS erfolgt. Die Transportaufträge brauchen dann nicht mehr in der Datenbank gespeichert zu werden und außerdem wäre dann eine regelmäßige Datenbankabfrage, die den Stand der Transportaufträge abfragt, durch das TOS nicht mehr nötig. Dazu wären dann aber Implementierungen evtl. mehrerer Wrapper nötig (Anzahl berücksichtigter Teilsysteme).

Zu dem vorgestellten Ansatz sind dann noch Implementierungen zur Simulation einer Hinterlandanbindung und evtl. später auch weiterer Teilsysteme möglich, die am Beispiel des Containerterminal Altenwerder vorgestellt wurden.

6 Fazit

Am Beispiel des Containerterminal Altenwerder konnte eine mögliche Softwarearchitektur für einen Containerhafen dargestellt werden. Das moderne System des CTA kann somit dazu verwendet werden, als Orientierung für das Virtual Port II Projekt zu dienen, insbesondere die Aufteilung der Aufgaben in den vorgestellten Teilsystemen und die Kommunikationswege.

Literatur

[JK03] U. Spindel J. Krasemann. *Softwarearchitektur für einen Containerterminal*. T-Systems GEI GmbH, Hamburger Hafen und Lagerhaus AG, Hamburg, 2003.

[Leh06] M. Lehmann. *Einsatzplanung von Fahrerlosen Transportsystemen in Seehafen-Containerterminals*. Technische Universität Berlin, Berlin, 2006.

[Pum99] D. Pumpe. *Ein Referenzmodell zur Planung und Steuerung der Abläufe in Seehafen-Containerterminals*. Berlin, 1999.

[Stu06] A. Stuth. *Software-Architektur für das Steuerungssystem des CTA*. Seminararbeit, Fachhochschule Wedel, 2006.

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

VR-Visualisierung von Produktions- und Logistiksystemen

Autodesk 3ds Max 9

vorgelegt von:

Christoph Dölger

Oldenburg, den 04.01.2009

Inhaltsverzeichnis

1 Einleitung

Diese Seminararbeit wurde im Rahmen der Projektgruppe „VPort II“ vorgelegt und behandelt die Thematik der VR-Visualisierung von Produktions- und Logistiksystemen mit dem Programm Autodesk 3ds Max 9.

Diese Arbeit widmet sich im zweiten Kapitel dem Begriff der Visualisierung und beinhaltet Definitionen und den Nutzen der 3D-Visualisierung.
Kapitel drei beschäftigt sich mit der Modellierung, die einen wichtigen Teil der Visualisierung ausmacht. In dieser werden Definitionen und Methoden der Modellierung aufgezeigt.
Das vierte Kapitel widmet sich dem Programm 3ds Max 9, des Unternehmens Autodesk Das Programm wird beschrieben und eine kurze Historie gegeben, sowie die Funktionen, Outputs und Systemanforderungen dargestellt.
Im fünften Kapitel wird das 3D-Visualisierungsprogramm Autodesk 3ds Max 9, dem in der Projektgruppe „VPort I“ verwendeten Programm, Blender gegenübergestellt.
Aus den vorangegangenen Kapiteln bildet sich, das im sechsten Kapitel aufgeführte Fazit. In diesem wird das Ergebnis dieser Ausarbeitung festgehalten und die Fragestellung beantwortet, ob sich der Wechsel von Blender auf 3ds Max 9 lohnt.

2 Visualisierung

Im Folgenden wird eine Begriffsdefinition für Visualisierung gegeben und durch den schematischen Aufbau unseres Projektes VPort II erklärt.

> *„Visualisierung kann als Begriff für alle Technologien verstanden werden, die mittels visueller Darstellung sowie angepasster Interaktionsmöglichkeit Einblick in Daten ermöglichen. Eine wesentliche Aufgabe der Visualisierung ist die Abbildung von aus Experimenten oder aus Simulationen gewonnenen Daten auf visuell leicht erfassbare Größen.“*
>
> Zitat: [GS08]

Die visuelle Darstellung erfolgt in unserer Projektgruppe durch zwei Beamer, welche die gerenderten Bilder auf eine Leinwand projizieren. Unter angepasster Interaktionsmöglichkeit fällt die Funktionalität, während der Darstellung frei in unserer Hafenwelt herumfliegen zu können, um Einblick in Daten bzw. Funktionsabläufe zu erhalten. Abgebildet werden hierbei aus Simulationen gewonnene Daten.

Der Verband Deutscher Ingenieure hat in seiner Richtlinie 3633 von 1993 folgende Zieldefinition für Visualisierung festgesetzt.

> *„Grundsätzliche Ziele sind in allen Fällen eine hohe Anschaulichkeit des Modells und die Bildung einer gemeinsamen Kommunikationsgrundlage der beteiligten Personen.“*
>
> Zitat: [Ing93]

Diese grundsätzlichen Ziele erreichen wir durch unsere 3D-Visualisierung der Simulationen und Hafenszenarien. Die Darstellung der Abläufe im Hafen in 3D erlaubt uns, auf Grundlage des anschaulichen Modells bzw. Szenarios, mit beteiligten Personen über dieses kommunizieren zu können. Unsere Kunden sind somit in der Lage ein visuelles, interaktives und leicht nachvollziehbares Modell zu begutachten.

Ein weiterer Nutzen einer 3D-Visualisierung ist die Möglichkeit der Überwachung. Anhand des 3D-Visuellen Modells kann man die dargestellten Produktionsabläufe kontrollieren.
Ein weiterer Vorteil von 3D-Visualisierung ist, dass man durch Simulationen bzw. in Experimenten Problemstellungen und -fälle nachbilden und vor dem realen Bau der Hafenanlage Lösungen finden kann. Durch den genauen Einblick in die Abläufe ist es möglich die Effizienz des Systems zu steigern. Schwachstellen können entdeckt und beseitigt werden.
Die 3D-Visualisierung kann uns ebenfalls dabei helfen unser Produkt besser zu verkaufen. Es ist also auch ein gutes Marketingmittel und wertet unsere Präsentationen durch Livedemos auf.

3 Modellierung

Um eine 3D-Visualisierung zu erstellen benötigt man diverse Modelle, welche zuerst einmal erstellt werden müssen.
Ein Modell ist eine Vereinfachung der Wirklichkeit. Es wird eine Geometrie aus Originalen bzw. Referenzdaten erstellt. Das entstandene Modell ist jedoch noch ohne Animationen und Texturen. [SB07]

3.1 Methoden

Ein Objekt wird im Rahmen unserer Projektgruppe komplett manuell, mit Hilfe von Bildern, Konstruktionszeichnungen und aus dem Gedächtnis, nachgebildet.
Im Folgenden werden Modellierungsmethoden aufgeführt.

3.1.1 Boxmodelling

Elementarobjekte werden durch Verformung zu komplexen Objekten modelliert.

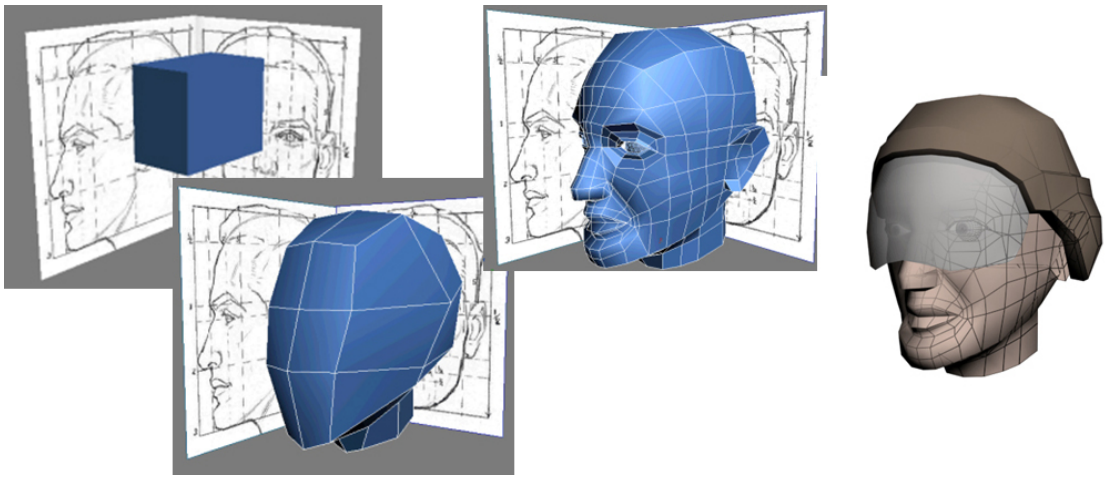


Abbildung 1: Boxmodelling [Aut08b]

Abbildung 1 zeigt den Verformungsprozess vom Elementarobjekt, bis zum fertigen Objekt. Als erstes wird ein Elementarobjekt vor einer Schablone, Blaupause oder technischen Zeichnung erstellt und mit Hilfe von verschiedenen Verformungen verändert. Weitere Elementarobjekte und Verformungen lassen das Objekt immer komplexer erscheinen, bis letztendlich das gewünschte Endergebnis modelliert ist.

3.1.2 Punktwolken

Modelle werden aus Punkten generieren und mit Oberflächen bezogen.

Abbildung 2: Punktwolken [Min08]

Das linke Objekt in Abbildung 2 ist ein aus Punkten erstelltes Modell eines Hasen. Das rechte Objekt zeigt, wie das Punktwolken-Modell mit einer Oberfläche bezogen wurde.

3.1.3 Constructive Solid Geometry (CSG)

Das Constructive Solid Geometry bzw. die Konstruktive Festkörpergeometrie, ist eine Modellierungsmethode mit der komplexe Oberflächen und Körper durch Kombination von Objekten und in Verbindung mit Booleschen Operationen erstellt werden. Vorteil dieser Methode ist der geringe Speicherbedarf. [Hah08a]

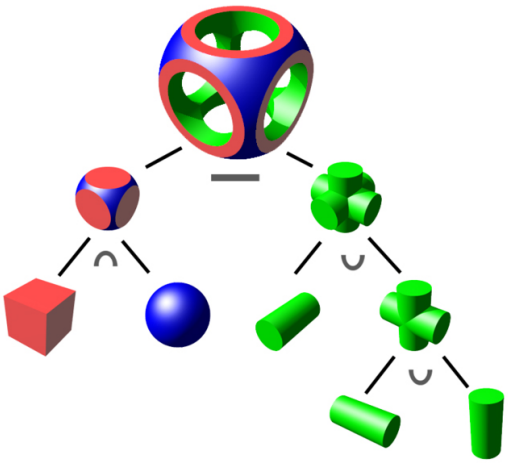


Abbildung 3: CSG-Baum [Zot08]

Abbildung 3 zeigt, dass einfache Körper mit Hilfe von Boolschen Operationen einen komplexen Körper erstellen.

3.1.4 NURBS

Non-Uniform Rational B-Splines sind mathematisch definierte Kurven oder Flächen, die im Computergrafik-Bereich, zur Modellierung beliebiger Formen verwendet werden. Es handelt sich um rationale B-Spline-Kurven mit einem nichtuniformen, also ungleichmäßig verteilten, Knotenvektor. [Hah08b]
Im Prinzip kann jede beliebige technische herstellbare oder natürliche Form mit Hilfe von NURBS dargestellt werden, jedoch ist der Speicheraufwand sehr hoch. [Hah08c]

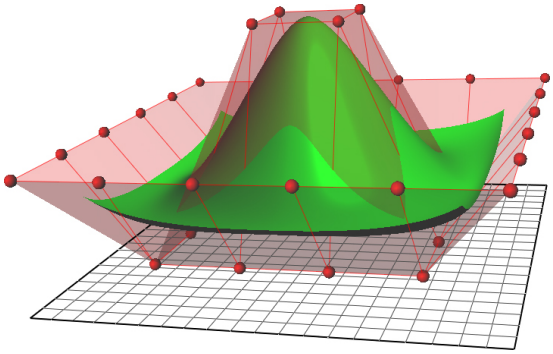


Abbildung 4: NURBS-Surface [Chr08]

Abbildung 4 stellt eine NURBS-Fläche vom Grad vier in grün dar. Sie wird durch 36 rote Kontrollpunkte über einem zweidimensionalen Parametergebiet, das untere Gitter, definiert.

3.1.5 Polygon-Modellierung

Die für uns wichtige Methode ist die Polygon-Modellierung. In der 3D-Computergrafik werden beliebige, zum Beispiel gekrümmte, Oberflächen als Polygonnetz modelliert. Insbesondere Dreiecksnetze eignen sich besonders gut zur schnellen Darstellung von Oberflächen. Aus Effizienzgründen werden vor der Darstellung z.B. die NURBS in Polygon- oder Dreiecksnetze umgewandelt.
Polygon-Modellierung ist die wohl älteste Technik um 3D-Objekte zu gestalten. Das Grundprinzip der Polygon-Modellierung besteht in der Manipulation eines sogenannten Grundkörpers, als Basis dienen meist Würfel, Kugeln oder konvertierte NURBS-Objekte.
Durch gezielte Unterteilung von Kanten und Polygonen (Flächen) werden so die Figuren transferiert. Bei der Polygon-Modellierung liegt der Schwerpunkt in der Formung des Objektes und weniger auf der Konstruktion. [DM08]

Abbildung 5: Nurbsmodell in Polymodell [Sch09]

Abbildung 5 zeigt, dass aus einem Nurbsmodell ein Polygonmodell entsteht, welches verschiedene „Auflösungen“ besitzen kann.

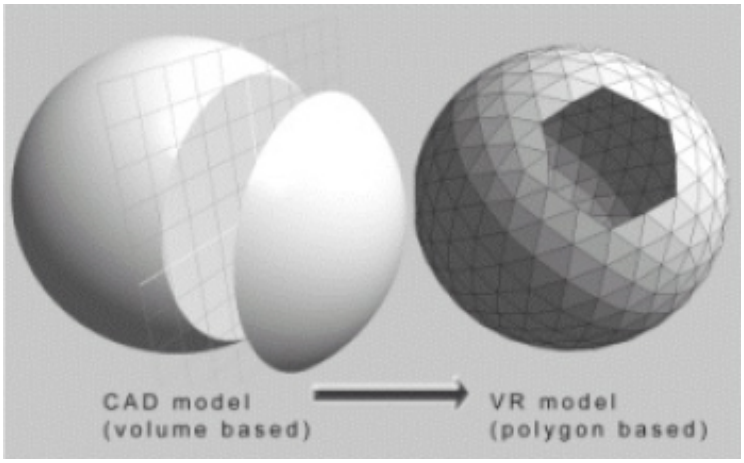


Abbildung 6: CAD-Modell in VR-Modell [RN09]

Abbildung 6 zeigt die Umwandlung von einem CAD-Modell in ein auf Polygone basierendes VR-Modell.

3.2 Export

Das fertige Modell wird in ein Dateiformat exportiert, welches von der Engine gelesen und verwendet werden kann. Tabelle 1 zeigt einige Dateiformate.

Dateiformat	Verwendung
3ds	3D Studio Max
obj	Wavefront Object
vrml	Virtual Reality Modeling Language
dwg	AutoCAD

Tabelle 1: Dateiformate

Das 3ds-Dateiformat wird von dem sehr bekannten Programm 3D Studio Max verwendet und stellt die Source-Datei des Programmes für Modelle dar. Programme, wie zum Beispiel Blender, können ihre erstellten Modelle in dieses Format exportieren und es dann in 3D Studio Max zur Weiterverarbeitung importieren lassen.
Das obj-Dateiformat steht für das Wavefront Object und wird in unserem Projekt VPort genutzt. Alle unserer vorhandenen Modelle bestehen in diesem Format.
Das Dateiformat VRML steht seit der Version 2.0 von 1997 für Virtual Reality Modeling Language. Es wurde ursprünglich als 3D-Standard für das Internet entwickelt und ist für den Menschen lesbar. Die meisten 3D-Modellierungswerkzeuge ermöglichen den Im- und Export von VRML-Dateien, wodurch sich das Dateiformat auch als ein Austauschformat von 3D-Modellen etabliert hat. [Gra09]
Das dwg-Dateiformat wird für technische Zeichnungen genutzt und ist das Dateiformat von AutoCAD.

4 Autodesk 3ds Max 9

Autodesk 3ds Max ist ein 3D-Computergrafik- und Animationsprogramm. Es bietet Lösungen für 3D-Modellierung, 3D-Animation, 3D-Rendering und visuelle Effekte. Eingesetzt wird es im Bereich Computerspiele, Comic, Animationen, Film, als auch in gestalterischen Berufen wie Design oder Architektur. [Aut09]

Früher war das Programm unter dem Namen „3D Studio Max“ bekannt und wurde vom Unternehmen Kinetix für MS-DOS entwickelt.
Die aktuelle Version „3ds Max 2009“ wurde 2008 von Autodesk für 2560 Euro herausgebracht. [Aut09]
Die in dieser Ausarbeitung untersuchte Software von Autodesk ist „3ds Max 9“, welche im Jahre 2006 erschien.

4.1 Outputs

Die Bandbreite reicht vom Output im Comicstil bis hin zu korrekt berechneten Lichtbrechungs- und Beugungseffekten an transparenten Objekten mit indirekter Beleuchtung (Radiosity).

Abbildung 7: Radiosity [Jac08]

Radiosity ist ein Verfahren zur Berechnung der Verteilung von Wärme- oder Lichtstrahlung innerhalb eines virtuellen Modells. Abbildung 7 stellt den Effekt des Lichttransports innerhalb einer Szene dar. In dieser erkennt man, wie das Licht auf dem rechten Bild gebrochen und auf den Sofas dargestellt wird.

4.2 Tutorials

Autodesk 3ds Max 9 bietet eigene Videotutorials an, welche durch einen Willkommensdialog automatisch aufgerufen werden. Der Dialog beinhaltet sieben Videos mit einer Gesamtlaufzeit von 12 Minuten. In diesen Kurzvideos wird ein Überblick über die Benutzerfläche gegeben, die Navigation in Ansichtsfenstern vorgeführt, sowie das Erstellen und Transformieren von Objekten erklärt. Des Weiteren gibt es Tutorialvideos über die Materialienfunktion und das erstellen von Animationen. In dem Willkommensdialog ist außerdem ein Link zur eigenen Tutorialseite „AREA“ angegeben, welche weiterführende Tutorials und Hilfe bei Fragen bietet.

4.3 Funktionen

Das äußerst umfangreiche Softwarepaket Autodesk 3ds Max 9 liefert diverse Funktionalitäten.
Es ist durch Plug-ins erweiterbar und bietet bereits Funktionen für realistische physikalische Simulationen und Charakteranimation, sowie Polygonobjekte und mathematische Flächen (NURBS).
Des Weiteren ermöglicht die Software Mehrfachreflexionen, Beleuchtung nach Geokoordinaten und Jahres- und Tageszeit. Durch Veränderung oder Neuschaffung von Materialien hat man weitgehende Gestaltungsfreiheit. [Aut09]

Autodesk 3ds Max 9 liefert ebenfalls eine eigene Renderfarm, sodass mehrere Clients mit der installierten Software am Videorendering teilnehmen können.

4.4 Systemanforderungen

Die 32-Bit Version von 3ds Max 9 hat folgende Systemanforderungen.

Windows® (XP mit SP 2 wird empfohlen)
DirectX® 9.0c
Prozessor Intel® Pentium® 4 AMD® Athlon® XP oder höher
RAM 512 MB (1 GB empfohlen; Abhängig von der Komplexität der Szene)

Tabelle 2: Systemanforderungen [Aut06]

4.5 Oberfläche

Abbildung 8 zeigt einen Screenshot aus Autodesk 3ds Max 9. Dargestellt wird die Programmoberfläche und geladen ist unser Modell des Vancarriers.

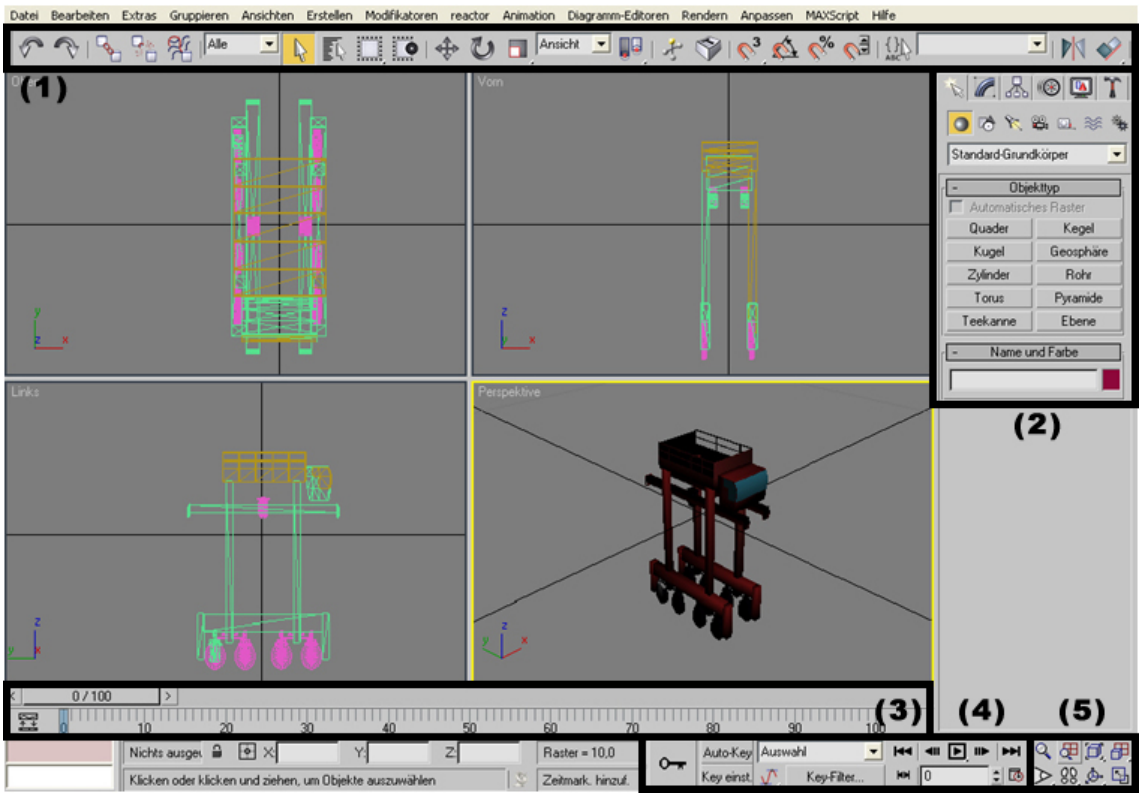


Abbildung 8: Autodesk 3ds Max 9 Oberfläche [Aut08a]

Die vier großen Felder in der Mitte sind die Ansichtsfenster. Oben links ist die Vogelperspektive, oben rechts die Frontalansicht, unten links die Seitenansicht und unten rechts die 3D-Perspektive. Alle Ansichtsfenster lassen sich über die Icons in der unteren Leiste bei Punkt (5) steuern. Mit Hilfe dieser Steuerelemente für Ansichtsfenster kann man unter anderem die Perspektive ändern und ein- und auszoomen.
Die Hauptsymbolleiste ist unter Punkt (1) zu finden. Sie bietet schnellen Zugriff auf die Befehle für die häufigsten Aufgaben, wie zum Beispiel Auswahl- oder Ausrichtbefehle.
Unter Punkt (2) befinden sich die Befehlspaletten für die Objekttypen und Parameter, um Objekte zu erstellen.
Die in Punkt (4) befindlichen Zeitschieber und Spurleiste sind dafür da, um sich die erstellten Animationen anzeigen zu lassen und können mit den in Punkt (5) befindlichen Animation- und Zeitsteuerelementen gesteuert werden.

5 Autodesk 3ds Max vs. Blender

Tabelle 3 stellt einen Vergleich zwischen den 3D-Visualisierungsprogrammen 3ds Max und Blender an.

	3ds Max	Blender
Nutzung	Kommerziell	OpenSource
Mächtigkeit	Sehr mächtig	Nahezu genauso mächtig
Erweiterbarkeit	Ja	Ja
Referenzen	Jurassic Park The Day After Tomorrow Matrix (1+2) Mr. & Mrs. Smith Spiderman 2 Mission: Impossible II Far Cry (PC-Spiel) usw.	Elephants Dream Big Buck Bunny Yo Frankie! (PC-Spiel)

Tabelle 3: 3ds Max vs. Blender

Aus der Tabelle ist ersichtlich, dass 3ds Max kommerziell und Blender hingegen OpenSource ist. Die kommerzielle Software ist, im Gegensatz zu der kostenlosen und für jeden zugänglichen Software Blender, sehr teuer. 3ds Max ist ein sehr mächtiges und bekanntes Tool, wobei man Blender zugute halten muss, dass es nahezu genauso mächtig ist.
Beide Programme sind durch Plug-ins erweiterbar und lediglich bei den Referenzen hat 3ds Max deutlich mehr Marktanteile.

6 Fazit

3ds Max 9 hat einen sehr großen Bekanntheitsgrad und ist durch seine Mächtigkeit und Erweiterbarkeit bei einer Reihe von Filmstudios aber auch bei Entwicklern von Computer- und Videospielen sehr beliebt. Jedoch bietet Blender dieselben Funktionen und steht 3ds Max in nichts nach. 3ds Max 9 liefert zwar eine Renderfarm, jedoch muss dazu das Programm auf jedem Computer installiert sein und setzt somit Windows voraus. Die Projektgruppe VPort I nutzte zum Rendern der Videos 35 Clients, von denen viele ein anderes Betriebssystem als Windows besitzen. Blender hingegen läuft bereits auf allen Clienten und ist an die Prozesskette angepasst und lauffähig.
Außerdem hätte ein Wechsel auf 3ds Max 9 zur Folge, dass nahezu alles nachbearbeitet werden müsste und dieser hohe Zeitaufwand ist ein Argument gegen die Umstellung.
Des Weiteren ist Blender durch seine Shortcut-Funktionalitäten schneller zu bedienen, während bei 3ds Max 9 zu viel mit der Maus gearbeitet werden muss.
Die Projektgruppe VPort II hat sich gegen die Umstellung auf Autodesk 3ds Max 9 entschieden und wird weiterhin Blender verwenden.

Literatur

[Aut06] Autodesk. *Benutzerhandbuch.* 2006.

[Aut08a] Autodesk. *Screenshot aus Autodesk 3ds Max 9.* 16.12.2008.

[Aut08b] Autodesk. *Tutorial.* http://www.autodesk.de, 16.12.2008.

[Aut09] Autodesk. *http://www.autodesk.de.* 04.01.2009.

[Chr08] Chrschn. http://upload.wikimedia.org/wikipedia/de/3/33/NURBS_surface.png, 16.12.2008.

[DM08] 3D-Mediadesign. *http://www.3d-mediadesign.de/3d-visualisierung/3d-modellierung.* 16.12.2008.

[Gra09] Graphcomp. *The Virtual Reality Modeling Language Specification.* http://graphcomp.com/info/specs/sgi/vrml/spec/, 04.01.2009.

[GS08] Geoinformatik-Service. *http://www.geoinformatik.uni-rostock.de/einzel.asp?ID=283231331.* 16.12.2008.

[Hah08a] Prof. Dr.-Ing Alex Hahn. *Vortrag Product Engineering LE1.* 2008.

[Hah08b] Prof. Dr.-Ing Alex Hahn. *Vortrag Product Engineering LE5.* 2008.

[Hah08c] Prof. Dr.-Ing Alex Hahn. *Vortrag Product Engineering LE6.* 2008.

[Ing93] Verband Deutscher Ingenieure. *VDI-Richtlinie 3633, Blatt 1.* 1993.

[Jac08] Jason Jacobs. *Autodesk 3dsmax Tutorial.* http://en.9jcg.com/comm_pages/blog_content-art-54.htm, 16.12.2008.

[Min08] Yugang Min. *Point Based Graphics, modeling and rendering.* http://www.cs.ucf.edu/∼ymin/, 16.12.2008.

[RN09] S. Kolbig T. Polzin R. Neugebauer, D. Weidlich. *Perspektiven von Virtual-Reality-Technologien in der Produktionstechnik- VRAx.* http://www.iwu.fraunhofer.de/vrax/images/photos/Lit_CPK_2004_perspektiven.pdf, 04.01.2009.

[SB07] Sören Schweigert and Jan C. Busch. *Vortrag Produktionsprozess Modellierung.* 2007.

[Sch09] Studio Thomas Schneider. http://www.studio-schneider.com/images/STUDIO_SCHNEIDER_3D_Visualisierung.pdf, 04.01.2009.

[Zot08] Zottie. *Demo of constructive solid geometry tree.* http://upload.wikimedia.org/wikipedia/commons/8/8b/Csg_tree.png, 16.12.2008.

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

ISL - Institut für Seeverkehrswirtschaft und Logistik

vorgelegt von:

Stephan Schulten

Oldenburg, den 07.01.2009

Inhaltsverzeichnis

1 Einleitung

Der im Zuge der Globalisierung wachsende Welthandel findet immer mehr auf dem Seeweg statt, was weitreichende Konsequenzen für den Import und Export von Überseecontainern mit sich bringt. Nicht nur, dass vorhandene Containerumschlagsbetriebe an ihre Grenzen gelangen und Reedereien immer größere Schiffe einsetzen müssen, sondern der Containerterminal wird zunehmend zu einem komplexeren Gebilde, zwischen dessen Komponenten eine empfindliche Wechselwirkung besteht. Nach Angaben des Statistischen Bundesamtes hat sich allein in den ersten neun Monaten des Jahres 2007 der Containerumschlag der deutschen Seehäfen um mehr als 12% gegenüber dem Vorjahreszeitraum erhöht [Deu07]. Die kapitalintensiven Investitionen, die für geeignete Infrastrukturmaßnahmen mit entsprechender Technologie benötigt werden, sind zur Gewährleistung der Konkurrenzfähigkeit im internationalen Vergleich unverzichtbar. Vorraussetzung für eine erfolgreiche Inbetriebnahme sind eine exakte Planung und Projektdurchführung, deren Vorstufe die Erstellung eines Modells sowie dessen Simulation ist. Die Ergebnisse können beispielsweise Aufschluss über die Kapazitätsauslastung der Stellflächen und Transportmittel oder über den maximalen Containerumschlag in einem bestimmten Zeitraum geben.

Auf Grund der immer größer werdenden Bedeutung der Schifffahrt entstanden zahlreiche Institute und Forschungseinrichtungen, die sich mit der Schifffahrt und der Seeverkehrswirtschaft wissenschaftlich auseinandersetzen und außerdem Beratungsangebote zu ihren Dienstleistungen zählen. Im Rahmen dieser Seminararbeit der Projektgruppe Virtual Port 2 des Departments Informatik der Carl von Ossietzky Universität Oldenburg soll das Institut für Seeverkehrswirtschaft und Logistik (ISL) sowie dessen Forschungsschwerpunkte und exemplarisch einige Projekte vorgestellt werden. Abschließend erfolgt ein Resümee, inwieweit die Projekte und das Institut allgemein eine Hilfe für die Projektgruppe darstellen und die Umsetzung der Ziele von Virtual Port 2 unterstützen können.

2 Institut für Seeverkehrswirtschaft und Logistik (ISL)

In diesem Abschnitt wird das ISL mit seiner hinter sich liegenden Geschichte, seinen Abteilungen und deren Tätigkeitsbereichen behandelt. Um einen Überblick über die praktische Arbeit der Forschungseinrichtung zu bekommen, widmet sich der letze Abschnitt einigen Beispielprojekten.

2.1 Profil

Das Institut für Seeverkehrswirtschaft und Logistik wurde 1954 als unabhängige gemeinnützige Stiftung privaten Rechts in Bremen gegründet. Der Gründer und erste Direktor war Dr. Gustav Adolf Theel, der eine umfangreiche Schiffahrtsbibliothek aufbaute, die heute noch als wesentliches Markenzeichen des Instituts bekannt ist. Mit dem Zweck wissenschaftliche Schifffahrtsforschung betreiben und fördern zu wollen, zählt das Institut in den Bereichen maritime Forschung und Beratung zu den europaweit führenden Instituten. Das ursprüngliche Institut für Schifffahrtsforschung erfuhr 1967 eine Namensänderung in Institut für Seeverkehrswirtschaft, womit die Zuordnung zum Senator für Bildung, Wissenschaft und Kunst einherging. Nachdem bereits im Jahr 1980 erstmals logistische Systeme sowie die Erfassung der Transportketten in die Aufgabenbereiche einbezogen wurden, erweiterte das Institut sein Tätigkeitsfeld vier Jahre später um die Bereiche Telematik und Logistik. In Folge dieser Ausweitung des Tätigkeitsfeldes nennt sich das Institut fortan Institut für Seeverkehrswirtschaft und Logistik.

Finanzielle Unterstützung erhält das ISL durch die 1982 gegründete Förderkreis Stiftung Institut für Seeverkehrswirtschaft und Logistik e.V., zu deren Mitgliedern Schifffahrtsunternehmen, Speditionen und Banken gehören. Die Stiftung verfolgt das Ziel, wissenschaftliche, seeverkehrswirtschaftliche und logistische Forschungen sowohl in geistiger als auch in materieller Hinsicht zu fördern. Zu den Kooperationspartnern der ISL zählen verschiedene Organisationen, Institute und Unternehmen aus dem In- und Ausland wie beispielsweise die Datenbank Bremische Häfen Logistics IT AG[1]. Das Institut ist Mitglied unterschiedlicher Institutionen wie zum Beispiel der Bundesvereinigung für Logistik e.V. oder der Association for Automatic Identification and Mobility (AIM - Deutschland e.V.).

Weitere Geschäftsstellen der ISL befinden sich in Lübeck und Bremerhaven. Die ISL Baltic Consult in Lübeck versteht sich als ein auf den Ostseeraum spezialisiertes Planungs- und Beratungsunternehmen, das neben Strategieberatung auch Hafen- und Terminalplanung, Projektmanagement sowie weitere Dienstleistungen[2] zu ihren Geschäftsfeldern zählt. Der Standort Bremerhaven ist durch die Kompetenzbereiche Auto-ID und Sicherheit im Containerverkehr sowie durch Optimierung und Simulation gekennzeichnet, mit deren Erforschung sich die ISL-Abteilung Informationslogistik auseinandersetzt. Nachfolgend sollen

[1] http://www.dbh.de
[2] http://www.isl-bc.com/index.php?id=5\&L=0

die einzelnen Kompetenzbereiche und Abteilungen innerhalb des Instituts vorgestellt und deren Forschungsschwerpunkte dargelegt werden.

2.2 Abteilungen

Seit dem Jahr 2007 teilt sich das Institut in die Abteilungen Logistische Systeme, Maritime Wirtschaft und Verkehr sowie nach der Bündelung der IT-bezogenen Forschungs- und Entwicklungsaktivitäten in die erweiterte Abteilung Informationslogistik, unter die seither die Geschäftsbereiche Auto-ID und Sicherheit im Containerverkehr sowie Optimierung und Simulation fallen.

2.2.1 Logistische Systeme

Die Abteilung Logistische Systeme beschäftigt sich mit kooperativen Systemen in und zwischen Logistik und Produktion, logistischen Standorten und Knoten sowie betriebswirtschftlichen Unternehmenskonzepten. Das ISL unterscheidet die Projektbereiche Intermodalität, Standortlogistik, nachhaltige Systeme in Produktion und Logistik sowie Wissensmanagement. Intermodalität steht im Kontext zu Transportketten, in denen unterschiedliche Verkehrsmittel zum Einsatz kommen und hierdurch eine effiziente Prozesskette entsteht. Hiermit eng verwoben sind Aspekte der allgemeinen Netzwerk- und Distributionspolitik, die in logistischen multimodalen Ketten eine wichtige Rolle spielen. Eine weitere Kernkompetenz dieser Abteilung wird in der prozessorientierten Gestaltung nachhaltiger Geschäftsmodelle gesehen. Zur langfristigen Sicherung der Wertschöpfung im Bereich der Hafenwirtschaft ist es wichtig, dass sämtliche Wertschöpfungsprozesse nicht auf Kosten kommender Generationen stattfinden, sondern eine nachhaltige Entwicklung Einzug erhält, die ökonomisch, ökologisch und gesellschaftlich verträglich ist. Entlang der logistischen Kette kann dies bedeuten, dass neue Systeme energieeffizienter gestaltet werden.

2.2.2 Maritime Wirtschaft und Verkehr

Zum Aufgabenfeld der Abteilung Maritime Wirtschaft und Verkehr gehören Marktanalysen und Prognosen zur wirtschaftlichen Entwicklung sowie zur Schifffahrtsentwicklung. Das Erstellen von Gutachten zu Marktchancen definierter Schiffstypen und -größen oder die Beratung von Politik und Wirtschaft zählen ebnso zum Tätigkeitsspektrum. Für Beratungszwecke kann die Abteilung auf Marktinformationen und über die Jahre gesammelte Daten zurückgreifen, die eine umfassende und adäquate Beratung ermöglichen. Ein weiterer Tätigkeitsbereich ist die Verkehrsplanung und Modellierung, die sich mit der weltweiten Simulation von Transport- und Logistiknetzen beschäftigt. Zu den Dienstleistungen zählen neben Machbarkeitsstudien zur Hafen- und Schifffahrtsentwicklung auch Prognosen zur Hafenumschlags- und Hinterlandverkehrsentwicklung.

Für das Projekt Jade Weser Port in Wilhelmshaven erstellte die ISL eine Analyse der Umschlagspotentiale für einen Container- und Mehrzweckhafen. Auf der Basis einer Ist-

Analyse der Umschlagsmengen in der Nordrange wurde eine Prognose für das Gesamtvolumen erstellt und auf deren Grundlage die möglichen Marktanteile für den Jade Weser Port beziffert [Ver07]. Das ISL war an der Machbarkeitsstudie für den Container- und Mehrzweckhafen in Wilhelmshaven beteiligt und stellte Annahmen zur Wirtschaftlichkeit auf. Die Abteilung war ebenso in die Entwicklung und Simulation der Suprastruktur[3] des Containerterminal involviert [Jad07].

2.2.3 Informationslogistik

Die erweiterte Abteilung Informationslogistik befasst sich mit eBusiness und Vernetzung sowie mit IT-Systemen für die Transportwirtschaft. Dienstleistungen und Produkte im Bereich der Informationstechnologien in der Transportwirtschaft bilden zusammen mit dem Wissensmangement innherhalb der Bereiche Transport und Logistik einen Schwerpunkt des Tätigkeitsfeldes. Zur Kernkompetenz der Informationslogistik zählt ebenso das Wissen um die Geschäftsprozesse mit neuen Technologien nutzbar zu machen und dabei aktuelles IT-Know-How miteinfließen zu lassen. Die Implementierung von Tools für die Vernetzung von EDV-Systemen mit externen Geschäftspartnern per EDI[4] hat dank des Wachstumsanstiegs im Containerumschlag Hochkonjunktur, da Geschäftsprozesse automatisiert werden und deshalb schneller ablaufen können, was einen höheren Durchsatz begünstigt. Zum Anbgebot gehören die professionelle Software-Entwicklung sowie Support und die Einrichtung und Betreuung von Firmenetzwerken. Es bestehen Beratungsangebote und die Möglichkeit von Schulungen im Bereich E-Commerce/M-Commerce oder zum Wissensmanagement in der Logistik.

Der Abteilung Informationslogistik wohnen zwei Kompetenzzentren inne, die erst im Jahre 2007 am Standort Bremerhaven angesiedelt wurden. Innerhalb des Kompetenzzentrums Optimierung und Simulation hat die Entwicklung von Spezialsoftware zur Unterstützung von Planungsaufgaben einen hohen Stellenwert gewonnen, die in einem nachfolgenden Abschnitt dieser Ausarbeitung an einigen Beispielen näher gebracht werden wird. Darüber hinaus umfassen das Angebot Monitoring- und Steuerungssysteme, die Abläufe und Prozesse optimieren können, sowie Konzepte zur Verbesserung von logistischen Abläufen. Das zweite Kompetenzzentrum Auto-ID und Sicherheit im Containerverkehr befasst sich mit automatischen Identifizierungstechniken und deren Einbindung in Geschäfts- und IT-Prozesse. Das ISL sieht sich im Bereich Auto-ID bzw. RFID[5] als Mittler zwischen Anwendern in der Transportwirtschaft, Technologieanbietern sowie Politik und Gesetzgeber. Der Fokus liegt hierbei auf dem sinnvollen Einsatz der jeweils passenden Technologie und deren optimaler Integration in die unternehmensinternen und -übergreifenden Geschäftsprozesse der Anwender [ISL07].

[3] alle auf der Infrastruktur errichteten Anlagen, Gebäude und Einrichtungen
[4] Elektronischer Datenaustausch
[5] Radio Frequency Identification

2.3 Tools des ISL

An dieser Stelle soll eine Auswahl der von der ISL realisierten Projekte vorgestellt werden, die von unmittelbaren Interesse für die Problemstellungen der Projektgruppe Virtual Port 2 sind.

2.3.1 SCUSY - Simulation of Container Unit Handling Systems

Die Motivation für eine Simulation besteht darin, dass unterschiedliche Szenarien nachgebildet werden können, die bei realer Umsetzung hohe Kosten und hohen Zeitaufwand bedeuten würden. Änderungen können ohne großen Zeit- und Kapitalverlust vorgenommen und wiederum simuliert werden. Diesen Gedanken greift das Planungstool SCUSY auf, mit dem sich neue Terminals planen oder existierende Terminals unter Einbeziehung von Layoutveränderungen reorganisieren lassen. Das Tool erlaubt die Erstellung eines Terminallayouts, auf dem alle benötigten Komponenten wie beispielsweise Gebäude oder Containerbrücken frei angeordnet werden könnnen, wie Abbildung 1 veranschaulicht. Es unterstützt die Simulation von Terminals mit unterschiedlicher Ausstattung und der damit verbundenen Entwicklung eines optimal einsatzfähigen Systems. Auch Veränderungen in der Geräteausstattung des Terminals und die damit verbundenen Auswirkungen auf die Umschlagsdauer und die entstehenden Kosten lassen sich untersuchen. SCUSY ermöglicht es alternative Handlungsstrategien beispielsweise in Bezug auf Umstau- oder Vorstauarbeiten zu überprüfen.

Neben dem Layout des Terminals erfasst das Tool fixe und variable Kosten sowie die Zuweisung von Transport- und Stapelgeräten. Darüber hinaus können Zeitpläne beliebiger Transportmittel am Terminal berücksichtigt und im Gesamtkontext die Wechselwirkung zwischen den verschiedenen Komponenten veranschaulicht und auf Optimierungsbedarf überprüft werden.

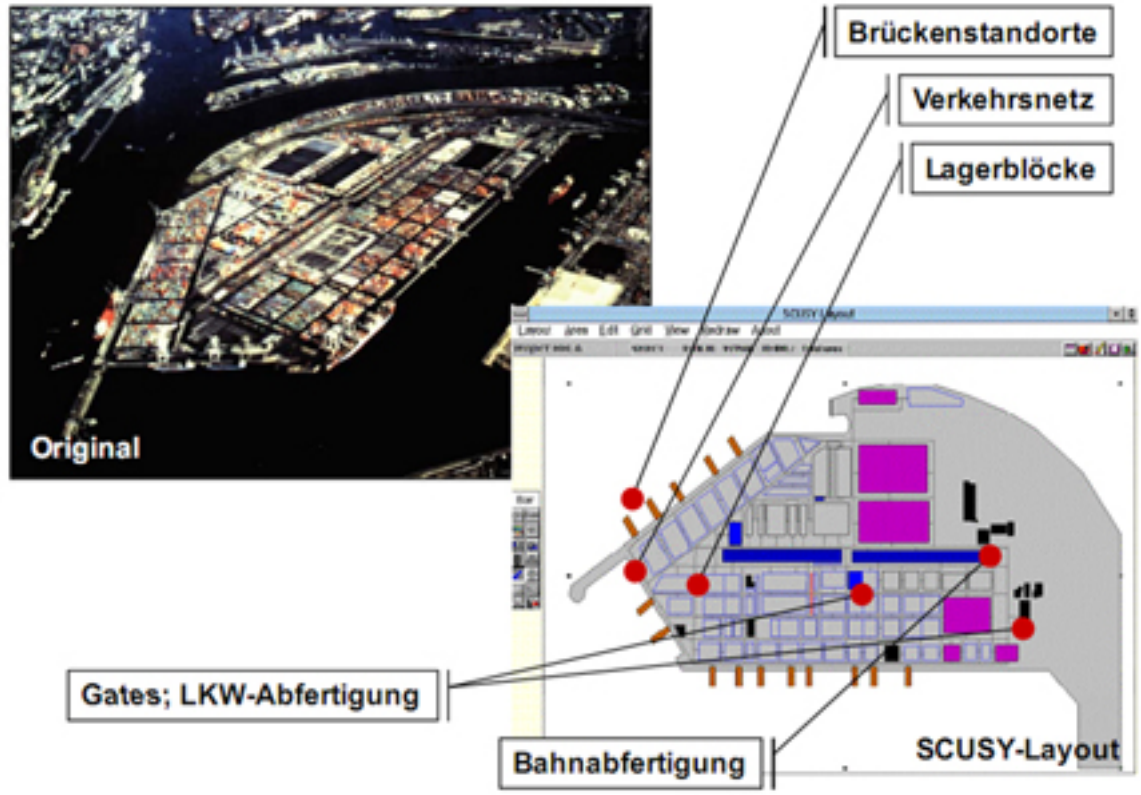


Abbildung 1: Gestaltung eines Terminallayouts mit SCUSY

Das Tool gibt Auskunft über die Kapazitätsauslastung aller Lagerflächen und unterstützt die Auswertung der Abfertigung des Zugverkehrs sowie der LKW-Servicezeiten. Eine detaillierte Kostenanalyse hilft zusammen mit der Möglichkeit der Einzelauswertung oder auf der Basis aller aggregierten Daten Verbesserungspotentiale aufzudecken und die Produktivität zu erhöhen. Zu den weiteren Leistungsmerkmalen gehören die Anwendbarkeit auf beliebige Terminalkonstellationen und die graphische Benutzeroberfläche. Der modulare Aufbau des Werkzeugs wird in Abbildung 2 dargestellt. Zu den Referenzen des Projekts zählen neben der HHLA Hamburg und Eurogate Bremerhaven auch der Jade Weser Port in Wilhelmshaven[6].

[6] vgl. http://www.isl.org/products_services/scusy/downloads/scusy_deu_presentation.pdf

Abbildung 2: Module des ISL-Tools SCUSY

2.3.2 CAPS - Capacity Planning System

Das Jahresumschlagsvolumen eines Terminals hängt von vielen unterschiedlichen Faktoren ab, deren Wechselwirkung den Umschlag beeinflusst. Hierzu zählen die Auslastung der verfügbaren Kapazitäten in Bezug auf Lagerflächen, Transportmittel und Kajenbeschaffenheit. Bei der Bestimmung dieser Kapazitäten soll das Tool CAPS helfen, das ebenso in der Lage ist, die maximale und operative Gesamtkapazität eines vorhandenen oder geplanten Terminals zu bestimmen. CAPS gibt Auskunft darüber, ob die vorhandene Fläche oder die Kaje bezogen auf die Umschlagsmenge einen Engpass darstellen. Mit Hilfe unterschiedlicher Modelle und deren Simulation lässt sich ermitteln, inwieweit eine Veränderung der genannten Parameter Auswirkungen auf die prozentuale Auslastung der Gesamtstellfläche hat und für welches Umschlagsvolumen sich der betrachtete Terminal eignet.

Für die Simulationsszenarien werden unterschiedliche Kajenlängen sowie die Anzahl der Containerbrücken und deren Verteilung auf die einzelnen Kajenabschnitte herangezogen. Die Szenarien berücksichtigen ebenso verschiedene Schiffstypen und Schiffsfahrpläne, die das Jahresumschlagsvolumen und die Verteilung beeinflussen. Bei der Simulation können für die Terminallagerfläche unterschiedliche Containertypen, wie Standard-, Kühl-, Gefahrgut- und Leer-Container, sowie die verfügbaren Stellplätze pro Containertyp miteinfließen. Von Bedeutung ist in diesem Zusammenhang auch die Verweildauer der einzelnen Containertypen.

Zu den Auswertungsergebnissen des Tools gehören die Schiffsabfertigungszeiten und Wartezeiten nach Schiffstypen. Informationen über die Auslastung der Lagerflächen und Containerbrücken geben zusammen mit einer detaillierten Auskunft über die Kajenbelegung Hinweise auf bestehende Engpässe.

2.3.3 COSMA - Container Operation System for Management and Administration

Eine entscheidende Bedeutung für einen reibungslosen Ablauf im Containerterminal spielt die Verfügbarkeit relevanter Informationen, wie Art, Bestimmungsort und Inhalt eines Containers, mit deren Hilfe ein effizienter Planungsablauf bereits vor Ankunft realisiert werden kann. Das Tool COSMA bietet Unterstützung im Containermanagement und im Umgang sowie bei der Verwaltung wichtiger Informationen. Auf diese Weise soll eine größere Transparenz der gesamten Bearbeitungsprozesse erreicht werden. Zu den Motivationsschwerpunkten für die Anwendung gehört die Unterstützung bei der Planung von Transporten und der Auswertung der Lagerbelegung. Die Verwaltung der Container wird durch Buchen von Containerbewegungen und die Anzeige von Containerinformationen realisiert, auf deren Informationsgrundlage Analysen der Containerbewegungen, deren Erfassung mit COSMA Abbildung 3 veranschaulicht, erstellt werden können. Eine virtuelle Lagerabbildung erlaubt die Konfiguration von Stellplätzen und eine realistische Anzeige des Lagers, in der jeder Container ausgewählt oder anhand identifizierender Kriterien gesucht werden kann.

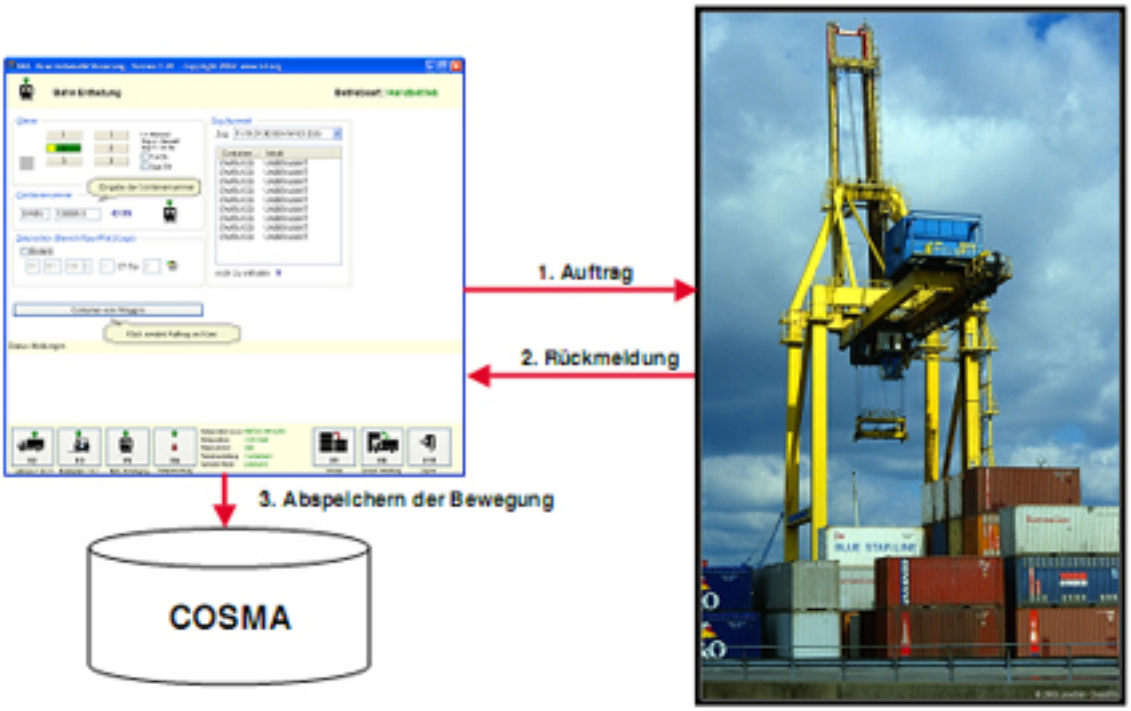


Abbildung 3: Erfassung der Containerbewegungen mit dem ISL-Tool COSMA

Wie diese Anforderungen in ein Systemmodell umsetzbar sind, veranschaulicht Abbildung 4. Um bei sich verändernden Anforderungen flexibel reagieren zu können, liegt dem Modell ein modularer Aufbau zugrunde, in den sich Änderungen oder Ergänzungen besser integrieren lassen. COSMA besteht aus 4 Modulen, die bestimmte Aufgaben erfüllen. Die Basis stellt grundlegende Funktionalitäten wie eine Stammdatenverwaltung zur Verfügung, mit Hilfe welcher Informationen zu beispielsweise Containern, Transportmitteln oder Gütern

Abbildung 4: Modularer Aufbau des ISL-Tools COSMA

vorgehalten werden können. Zu den weiteren Grundfunktionalitäten gehört ein Buchungsmodul, mit dem die Buchungs- und Bewegungshistorie verfolgbar ist. Ein Reportingmodul erlaubt eine frei bestimmbare Anordnung von Reportingelementen, die auf der Grundlage aggregierter Daten die gewünschte Ansicht präsentieren. Das Modul der Bestandsverwaltung vereinigt die Transportplanung sowie das Buchen von Ein- und Ausgängen. Das Tool verfügt ebenso über eine auswertungsrelevante Funktionalität, die Auskunft darüber gibt, innerhalb welcher Zeiträume und mit welchen Transportmitteln Containerbewegungen stattgefunden haben. Weiter lassen sich Informationen zur Buchungs- und Bewegungshistorie eines jeden Containers sowie die Lagerbelegung anzeigen. Mit dem Modul der Lagerverwaltung können einzelne Stellplätze gesperrt oder wieder freigegeben werden.

3 Fazit

Das Institut für Seeverkehrswirtschaft und Logistik stellt ein umfangreiches Angebot an Dienstleistungen im Bereich Schifffahrt, Transportwirtschaft und Logistik zur Verfügung. Der kundenindividuellen Software-Entwicklung zur Überwachung und Unterstützung von Geschäftsprozessen wird in komplexer werdenden logstischen Ketten ein immer höherer Stellenwert beigemessen. Auf das fachspezifische Wissen des Instituts greifen Politik und Wirtschaft zurück, die die Beratungsangebote in Anspruch nehmen. Auf ihrem Gebiet hat sich das ISL einen internationalen Namen gemacht, der für qualitativ hochwertige und zukunftsweisende Forschung und Entwicklung bekannt ist.

Die Tatsache, dass die ISL bereits für das Projekt Jade Weser Port tätig gewesen ist, impliziert, dass sich das Institut mit den Anforderungen und Besonderheiten des Vorhabens beschäftigt haben muss. Inwieweit jedoch Fragen zur Containerstellplatzverwaltung oder zur Einteilung der Containerbeförderungsmittel beantwortet werden können, das heißt welche Strategien bei der Planung verfolgt werden, konnte aus den zur Verfügung stehenden Materialien nicht in Erfahrung gebracht werden. Zum Teil bieten die Tools eine Orientierung, welche Informationen für die Simulation des Containerumschlags am Terminal wichtig sind, die innerhalb der Projektgruppe für die Weiterentwicklung des Terminal Operating System (TOS) von Bedeutung sein könnten. Die Fragen, nach welchen Kriterien die einzelnen Containertypen auf dem Yard angeordnet werden oder ob Speditionen eigene Stellplätze für Gefahrgutcontainer und solche mit Stromanschluss besitzen, bleiben noch unbeantwortet. Es ist zu bedenken, dass kein vorhandenes System kopiert werden soll, sondern auf der Grundlage eigener Überlegungen und Strategien zu den spezifischen Planungsproblemen eine Hafensoftware sowie eine Simulation realisiert werden soll.

Literatur

[Deu07] Statistisches Bundesamt Deutschland. Pressemitteilung Nr. 527 vom 27.12.2007. Website, 2007. http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2007/12/PD07__527__463.psml; Letzter Zugriff: 05.01.2009.

[ISL07] ISL. Institut für Seeverkehrswirtschaft und Logistik. Website, 2007. http://www.isl.org/profile/rfid/index.php?lang=de; Letzter Zugriff: 05.01.2009.

[ISL09] ISL. Institut für Seeverkehrswirtschaft und Logistik. Website, 2009. http://www.isl.org; Letzter Zugriff: 05.01.2009.

[Jad07] Jadeweserport. Feasability-Studie für einen Container- und Mehrzweckhafen in Wilhelmshaven. Website, 2007. http://www.isl.org/projects/project.php?lang=de\&proj_num=2242; Letzter Zugriff: 05.01.2009.

[Ver07] Abteilung Maritime Wirtschaft Verkehr. Analyse der Umschlagspotentiale für einen Container- und Mehrzweckhafen in Wilhelmshaven. Website, 2007. http://www.isl.org/projects/project.php?lang=de\&proj_num=2207; Letzter Zugriff: 05.01.2009.

Department für Informatik
Abteilung Wirtschafsinformatik

Seminararbeit

Planung im Hafen

vorgelegt von:

Daniel Probst

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Der stetig zunehmende Umschlag von Containern und der durch die Globalisierung weltweit gestiegene Handelsverkehr macht es unverzichtbar, Transportketten auf Effizienz und Wirtschaftlichkeit zu untersuchen. Gegenstand der nachfolgenden Ausführungen sind die Aufgaben der Planung im Hafen sowie exemplarisch einige Prozessabläufe verschiedener Verkehrsträger, die im Rahmen der Transportkette die Schnittstelle zum Terminal bilden. Ferner ist heutzutage der Informationsaustausch mittels moderner Techniken in Containerterminals unverzichtbar, da beispielsweise Reedereien immer kürzere Liegezeiten ihrer Schiffe erwarten.

In Kapitel 2 geht es um die Aufgaben der Schiffsabfertigung, zu denen die Liegeplatzplanung (vgl. Abbildung 1, *grau dargestellt*), die Brückeneinsatzplanung (*blau dargestellt*) und die Stauplanung auf dem Schiff (*rot dargestellt*) gehören. Das darauffolgende Kapitel 3 umfasst den Import und Export der Bahnabfertigung, wie sie in einem Containerterminal vorkommen. Am Ende dieser Seminararbeit werden die Beteiligten eines Datenaustausches identifiziert, und es wird auf die Bedeutung des elektronischen Datenverkehrs eingegangen.

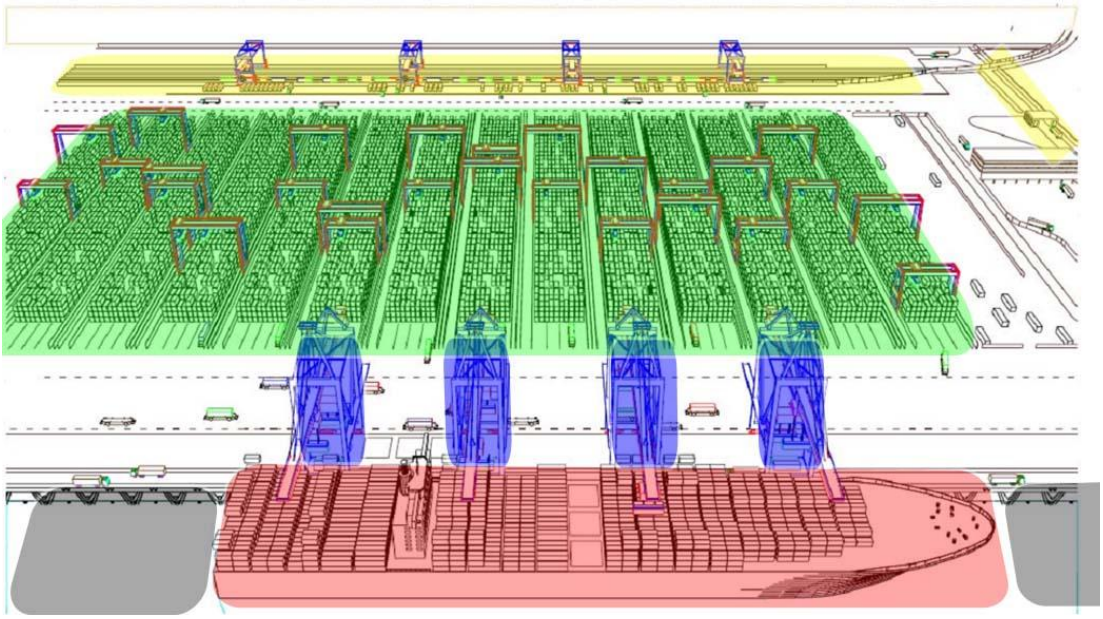


Abbildung 1: Aufgaben im Hafen - Überblick

2 Schiffsabfertigung

Die Hauptaufgabe eines Containerterminals ist das Löschen und Laden von Schiffen. Dabei ist der Containerterminal die Schnittstelle des Containerumschlags vom Hinterland zur See, von der See zum Hinterland und von See zur See. Es findet entweder ein Weitertransport per Schiff, LKW oder Bahn statt, oder die Container werden einem Stellplatz auf dem Yard zugeordnet und bis zum Weitertransport eingelagert. Containerbewegungen vom Überseecontainerschiff zum Hinterland (Import) und Containerbewegungen zum Überseecontainerschiff (Export) müssen bei der Stellplatzverwaltung in Abhängigkeit von der Verweildauer berücksichtigt werden.

Die zentrale Aufgabe der Schiffsabfertigung ist ein reibungsloser Ablauf der Lösch- und Ladevorgänge, wobei in möglichst kurzer Zeit eine möglichst große Menge an Containern umgeschlagen werden soll, um so die Fahrpläne der Containerschiffe sicherzustellen. Darüberhinaus müssen die Vorschriften der Statik und des Gefahrenguts eingehalten werden. Die verschiedenen Aufgaben sind in Abbildung 2 dargestellt und lassen sich in drei Bereiche aufteilen: die Planung der Liegeplätze, die Brückeneinsatzplanung und die Planung der Stellplatzverwaltung auf dem Schiff.

Der Vorgang beginnt mit der Anmeldung des anzulegenden Schiffes. Zuerst muss der Liegeplatz ermittelt werden, an dem das Schiff anlegen soll. Danach kann dem Kapitän der Platz an der Kaimauer zugewiesen werden, die nötigen Yardbereiche für die Import- und Exportcontainer reserviert werden und der Brückeneinsatz geplant werden. Anschließend wird der Bedarf der Van Carrier[1] ermittelt.

Um unnötigen Wartezeiten und Umsatzausfällen entgegenzuwirken, unterliegt das Be- und Entladen von Transportern aus dem Hinterland einer genauen Planung, durch welche die Terminalproduktivität optimal zu beeinflussen versucht wird.

[1] Auch Portalhubwagen genannt. Wird als Transportfahrzeug auf dem Containerterminal in Häfen eingesetzt.

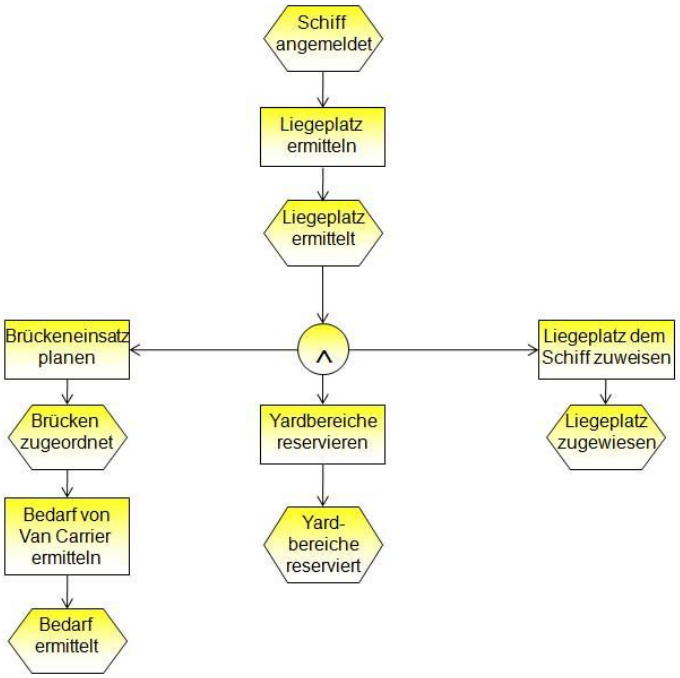


Abbildung 2: Ereignisgesteuerte Prozesskette der Schiffsabfertigung[2]

Im Folgenden werden die drei Hauptaufgaben der Schiffsabfertigung näher betrachtet.

2.1 Liegeplatzplanung

Ein wichtiger Bestandteil der Schiffsabfertigung ist die Liegeplatzplanung, da sie eng im Zusammenhang mit nachgefolgerten Planungsproblemen stehen. Dabei beginnt die Planung mit dem Erhalt der Fahrpläne der Schiffe von den Reedereien. Diese Pläne machen Aussagen über die Verweildauer von Schiffen an den Containerterminals, die wiederrum abhängig von der Anzahl der zu löschenden und zu beladenen Container ist.[3] Da die Liegezeiten der Schiffe ein großer Kostenfaktor für die Reedereien darstellt, ist es wichtig diese zu minimieren.

Die Länge Kaimauer des JadeWeserPorts in Wilhelmshaven[4] beträgt 1.725 m. Grundsätzlich kann jedes Schiff eine beliebige Position einnehmen. Die Planung der Plätze wird nach den Kriterien der Länge und der Verweildauer gemacht. Dabei orientieren sich die Pläne der Feeder- und Binnenschiffe an denen der Überseeschiffe, da die kleineren Schiffe aufgrund der höheren Manövrierbarkeit ohne größere Probleme den Liegeplatz wechseln können. Weiterhin wird

[2] vgl. Pumpe, S. 75
[3] vgl. Pumpe, S. 92
[4] vgl. http://www.jadeweserport.de

berücksichtigt, dass die Größe der Schiffe Auswirkungen auf den Liegeplatz haben, da die Containerbrücken des Terminals über die gesamte Schiffsbreite reichen müssen, um so einen reibungslose Löschung und Beladung zu garantieren. Die Anordnung und Verfahrbarkeit der Containerbrücken zum einen, und die Länge der Kaimauer zum anderen, bestimmen den Grad der Komplexität der Liegeplatzplanung.[5]

In großen Häfen werden computergestützte Softwaresysteme zur Liegeplatzplanung eingesetzt. Dabei besteht bei allen Systemen die Gemeinsamkeit, dass auf der X-Achse die Zeit und auf der Y-Achse die Liegeplatznummer bzw. die Länge der Kaimauer dargestellt werden. Die Länge eines Schiffes kann in der Höhe eines Balkens, und die Anlegedauer in der Länge eines Balkens abgelesen werden. (vgl. Abbildung 3).

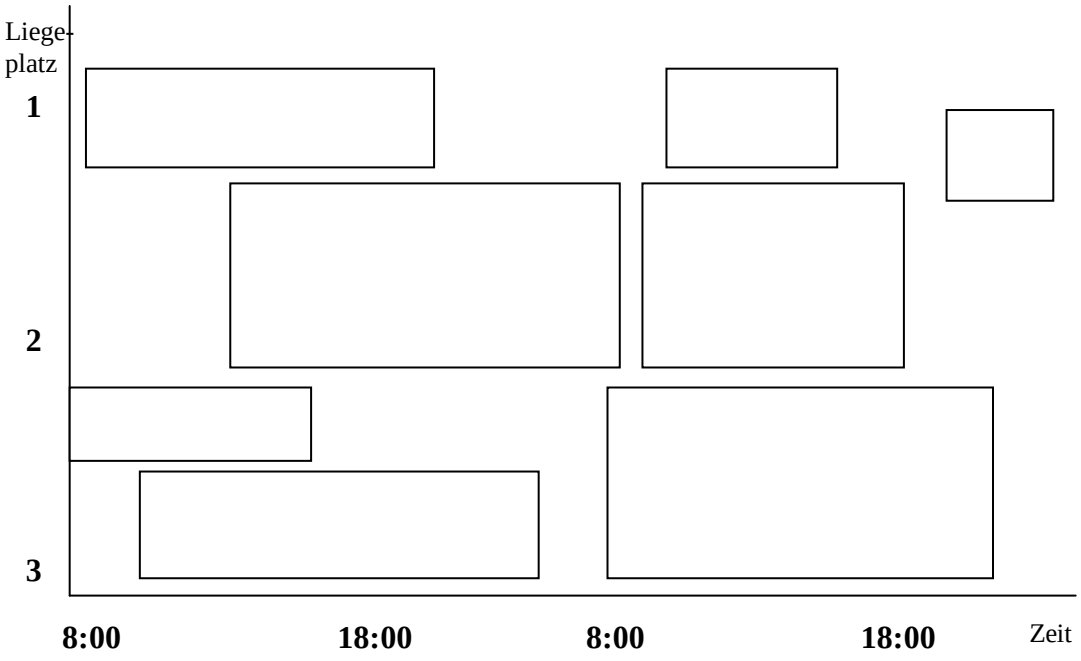


Abbildung 3: Ausschnitt einer Liegeplatzplanung[6]

[5] vgl. Pumpe, S. 93
[6] vgl. Pumpe, S. 95

2.2 Brückeneinsatzplanung

Die Brückeneinsatzplanung ordnet die Brücken eines Containerterminals den abzufertigenden Schiffen zu und legt fest, welche Brücke an welchem Schiffsabschnitt in welcher Reihenfolge arbeitet.[7] Sie ist abhängig von der Liegeplatzplanung der Schiffe, da die Anzahl der benötigten Brücken von der Schiffsgröße und der zu löschenden und zu ladenden Container abhängig ist. Außerdem bestimmen der Lade-Ist-Zustand vor dem Einlaufen, der Lade-Soll-Zustand nach dem Auslaufen und die Aufenthaltsdauer des Schiffes die Anzahl der benötigten Brücken.

Der Brückeneinsatzplan bestimmt das Programm jeder einzelnen Brücke. Es wird definiert, in welcher Reihenfolge Container gelöscht oder geladen werden. Ferner hat jede Brücke die Wahl, ob sie an der Kai- oder Wasserseite beginnt, ob sie vertikal, horizontal oder alternierend von der Mitte aus arbeitet. Nachdem diese Brückeneinsatzstrategie festgelegt wurde, können die Lade- und Löschsequenzen der einzelnen Container definiert werden.

Generell besteht die Optimierungsaufgabe zum einen darin, die vorhandenen Brücken den Schiffen so zuzuordnen, dass die Lade- und Löschzeit möglichst kurz ist (Minimumprinzip) bzw. möglichst viele Schiffe in einer angegebenen Zeitspanne abgefertigt werden (Maximumprinzip).[8]

Die Containerbrücken sind schienengebundene Förderfahrzeuge, die parallel zur Kaimauer arbeiten. Aufgrund ihrer großen Bauform und ihres hohen Eigengewichts ist die Flexibilität entsprechend gering. Prinzipiell können sich die Brücken nicht aneinander vorbei bewegen, weshalb eine Fehlplanung zu einem hohen Zeitverlust und zu einer daraus resultierenden, möglichen Neuplanung führen kann.

2.3 Stauplanung auf dem Schiff

Die zentrale Aufgabe der Stauplanung auf dem Schiff ist die Verstauung der Container, wobei die genaue Position eines jeden Container auf dem Schiff festgelegt wird. Die Komplexität dieser Aufgabe erhöht sich mit Anzahl der Container. Es ergeben sich theoretisch n! Anordnungsmöglichkeiten, wenn n die Anzahl der zu verladenen Container bezeichnet.[9] Rahmenbedingungen in Form von Art und Typen

[7] vgl. Steenken, S. 7
[8] vgl. Pumpe, S. 98
[9] vgl. Pumpe, S. 85

der Container und technischer Gegebenheiten der Terminals schränken diese Anzahl zudem weiter ein. Die Reederei erstellt die Fahrpläne der Schiffe, weshalb diese auch für den Ladezustand über die gesamte Rundreise verantwortlich ist.

Die Planung ist ein zweistufiger Prozess, der aus der Fein- und Grobplanung besteht. Dabei nehmen die Reedereien die Grobplanung vor, die Containergruppen Stellplatzgruppen zuordnet. Es wird für jeden Container festgelegt, an welchem Hafen er abgeliefert oder aufgeladen wird. Die Feinplanung übernimmt der jeweilige Containerterminal, der vom Schiff angefahren wird. Hierbei wird eine konkrete Planung der Container zu den einzelnen Stellplätzen vorgenommen. Dabei müssen folgende Bedingungen berücksichtigt werden:

- Stapelbarkeit und Größe der Container
- Gefahrgutvorschriften
- Verteilung der Vorrats- und Ballastwassermenge

Man unterscheidet zwischen 20‘ und 40‘ Containern. Das Stapeln der Container unterliegt einigen Einschränkungen wie Abbildung 4 zeigt. Dabei kann beispielsweise ein 40‘ Container auf zwei 20‘ Containern gestapelt werden, der umgekehrte Fall ist aber nicht zulässig, da es in der Mitte der größeren Container keine Aufliegevorrichtungen gibt.

Unter Beachtung der Gefahrengutvorschriften wird die Einhaltung der korrekten Lagerung von Gefahrgutcontainern verstanden. Es gibt Vorschriften nach denen manche Gefahrgüter nicht unmittelbar nebeneinander gelagert werden dürfen oder im Falle eines Brandes der schnelle Zugang erreicht werden muss.

Außerdem orientiert sich die Feinplanung an dem Ziel der optimalen Verteilung der Vorrats- und Ballastwassermenge. Um eine gleichmäßige Beladung zu gewährleisten, haben Containerschiffe Ballastwassertanks, die gefüllt werden können.

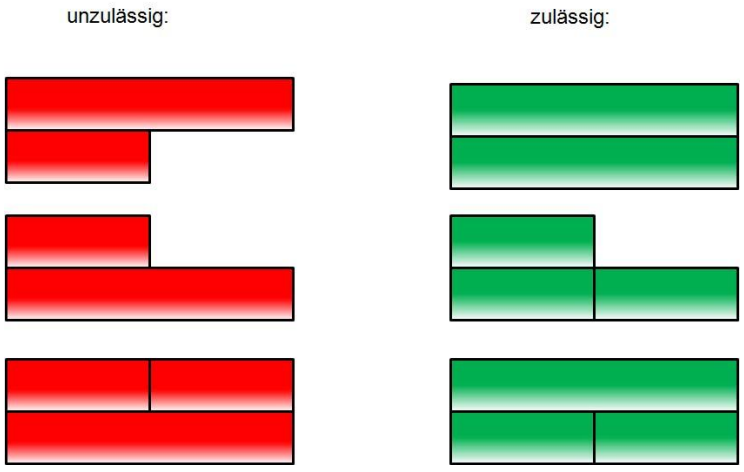


Abbildung 4: Stauvarianten

3 Bahnabfertigung

Die Abfertigung eines Containerterminals zur Bahn stellt verschiedene Ansprüche an den Terminal. Je nach Größe und Auslastung der Importe und Exporte von Containern mittels der Bahn, bedarf es seitens des Terminals entsprechender Umschlaggeräte. Trainstainer bieten sich an, wenn mehrere Gleise nebeneinander versorgt werden müssen oder die Bahnwagen als Ganzzug auf dem Terminal be- und entladen werden, da die Wege für die VanCarrier zu lang und somit nicht mehr wirtschaftlich wären.[10]

Der Export, also die Anlieferung vom Hinterland zum Terminal, gestaltet sich relativ einfach. Nach der Anmeldung wird der Bahn mitgeteilt an welches Gleis sie zu fahren hat. Die Container werden daraufhin auf Korrektheit und Beschädigungen geprüft. Anschließend werden die Container abgeladen und mittels VanCarrier zu ihren Stellplätzen auf dem Yard gebracht.

Der Import, also die Anlieferung vom Terminal zum Hinterland, erfolgt nach Absprache der Speditionen, der Bahn und dem Terminal und stellt eine Herausforderung dar, da beim Entladen der Schiffe nicht eindeutig bestimmt werden kann, über welchen weiteren Transport der Container verladen wird. Dies ist dadurch zu begründen, dass sich durch kurzfristige Nachfrage von bestimmten Gütern der Transportweg ändert.

Abbildung 5 zeigt den Ablauf eines Container – Imports. Zuerst wird der Transport mit dessen Kapazitäten bei der Bahn angemeldet, woraufhin die Größe der Wagen ermittelt und bereitgestellt wird. Die zugehörigen Containerplätze auf dem Yard werden dann ermittelt und zugehörige Transportaufträge für die VanCarrier erstellt. Sofern der Zoll die Container für den Import freigegeben hat, kann das beladen der Wagen beginnen. Abschließend wird dem Zugführer die Fertigstellung gemeldet und der Zug kann abfahren.

[10] vgl. Pumpe, S. 77

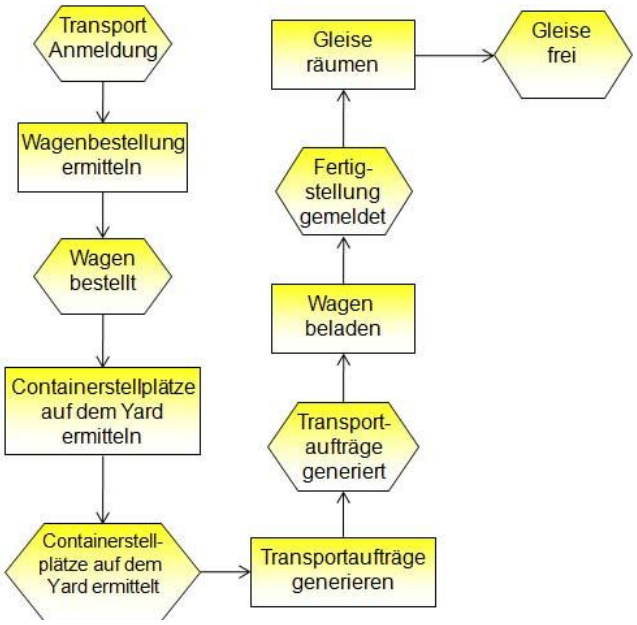


Abbildung 5: Ereignisgesteuerte Prozesskette der Bahnabfertigung (Import)[11]

4 Datenaustausch

Voraussetzung für einen reibungslosen Prozessablauf an einem Containerterminal ist die Verfügbarkeit der relevanten Informationen, wie beispielsweise die Anzahl und die Art der eingehenden Container, um geeignete Containerstellplätze reservieren und die Belegung der Containerbeförderungsmittel, sowie den Weitertransport ins Hinterland planen zu können. Dabei besteht eine besondere Herausforderung in der dynamischen Reaktion auf Verzögerungen oder Störungen des Prozessablaufs im Containerterminal. Um dieser Probleme Herr zu werden, setzen heute moderne Containerterminals komplexe Logistiksysteme ein, um eine hohe, kontinuierliche Produktivität in der Abfertigung von Schiffen, Zügen und LKWs zu erreichen.[12]

Durch den weltweit wachsenden Containerumschlag werden immer mehr innovative IT – Systeme entwickelt. Die Erwartungen der Reedereien auf immer kürzere Liegezeiten von Schiffen in Häfen, erzwingen den Einsatz innovativer Technologien, wie zum Beispiel automatisierte Steuerungssysteme. Dementsprechend besteht zwischen dem Terminal und allen am Transport Beteiligten ein bezüglich Menge und Typenvielfalt reichhaltiger Datenaustausch (vgl. Abbildung 6).

[11] vgl. Pumpe, S. 80
[12] vgl. Steenken, S. 1

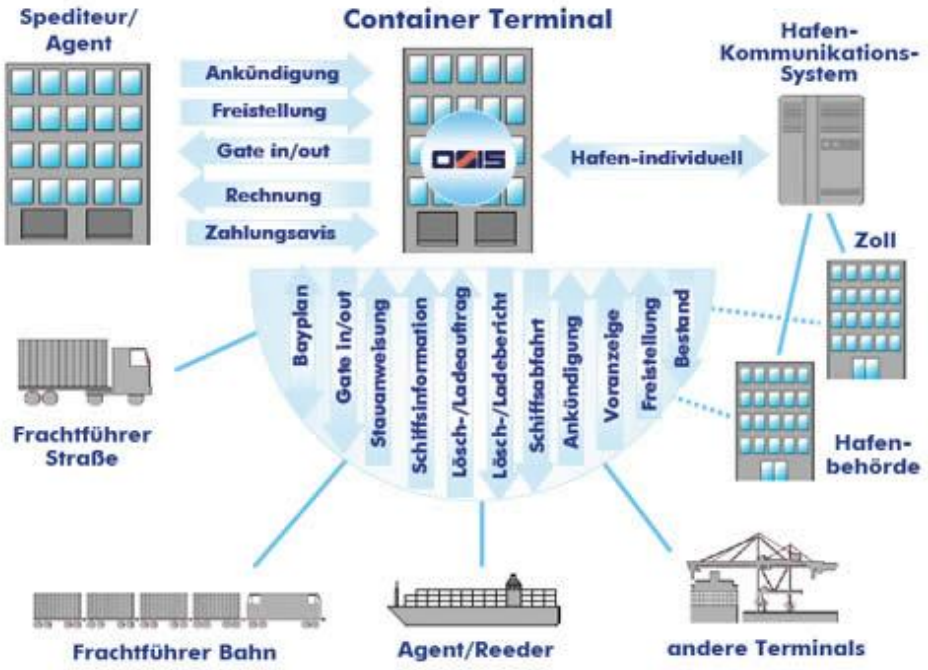


Abbildung 6: Datenaustausch eines Containerterminals[13]

Die Beteiligten des Informationsaustausches sind beispielsweise Spediteure, Agenten, Reeder, Frachtführer Straße und Bahn, andere Terminals, Hafenbehörden und der Zoll. Dabei werden EDIFACT- Nachrichtentypen von Containerterminals bevorzugt eingesetzt, da existierende Lösungen für den Containerverkehr und die Schifffahrt, wie zum Beispiel INVOIC, BAPLIE, CALINF, COARRI, CODECO, bestehen.[14]

[13] vgl. http://www.lsgmbh.de/deutsch/loesung6_bild.htm
[14] vgl. LS GmbH

5 Fazit

In der vorliegenden Arbeit wurden die Aufgaben eines Containerterminals betrachtet und einige Planungsprobleme näher erläutert und analysiert. Dabei stellte sich heraus, dass die Planungsaufgaben im Hafen nicht nur komplex, sondern auch untereinander enge Abhängigkeiten aufweisen. Um den Ablauf der Lade- und Löschsequenzen möglichst exakt planen zu können, bedarf es ein modernes Kommunikationssystem, welches den Datenaustausch aller Beteiligten regelt.

Für die Projektgruppe „Virtual Port 2“ ist es wichtig, die Planungsaufgaben eines Containerterminals zu verstehen und sich mit ihnen auseinanderzusetzen, um diese realitätsgetreu in dem zu entwickelten Modell umzusetzen und darzustellen.

Literaturverzeichnis

Fink, Andreas, Voß, Stefan und Schneidereit, Ulrike. 2005. *Grundlagen der Wirtschaftsinformatik.* Hamburg : s.n., 2005.

LS GmbH. OSIS im EDI Szenario. [Online] [Letzter Zugriff am: 05. Januar 2009.] http://www.lsgmbh.de/deutsch/loesung6.htm.

Pumpe, Dieter. 1999. *Ein Referenzmodell zur Planung und Steuerung der Abläufe in Seehafen-Containerterminals.* Berlin : s.n., 1999.

Steenken, Dirk. Online Optimierung in der Container Terminal Logistik . [Online] [Letzter Zugriff am: 5. Januar 2009.] http://www.dmkn-beta.de/downloads/39/b2/FILE239/b2-5_steenken.pdf/.

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

Planungsprobleme in Häfen

vorgelegt von:
Fabrizio Nencini

Inhaltsverzeichnis

1 Einleitung

In den letzten vier Jahrzehnten , hat der Container einen großen Erfolg im internationalen Seefrachttransport gewonnen. Mit der zunehmenden Containerisierung letzten Jahren hat die Zahl von Seehafen-Containerterminals und das Wettbewerb beträchtlich zugenommen. Einer der Erfolg-Faktoren eines Terminals ist mit der Zeit im Hafen für Containerschiffe verbunden und die Umladehafengebuhren die von Schiff-Maschinenbediener bezahlt werden müssen. In diesem Papier wird geschrieben die Hauptlogistik-Prozesse in Seehafen-Containerterminals und vorliegenden Methoden für ihre Optimierung. Wir konzentrieren uns auf den Prozess des Containertransports durch Gantry-Kräne und Straddle-Carriers zwischen dem Containerschiff und dem Containerdepot. Das primäre Ziel ist die Reduktion der Beladungszeit der Schiffe im Hafen durch die Maximierung der Gantry-Kräne Leistungfähigkeit. Wir untersuchen verschiedene Strategien um die Arbeit von Straddle Carries zu Gantry Kränen zu verteilen und zeigen das Potenzial von Entwicklungsalgorithmen, um die Lösungen zu verbessern.

Abbildung 1

Quelle:Fisch -n- ships home page

2 Grundlagen

Container kamen für die internationale Beförderung der Seefracht am Anfang der sechziger Jahre auf den Markt. Sie setzen fort und haben die Anerkennung gewonnen auf Grund dessen Multimodalität . Container sind relativ gleichförmige Kästen, deren Inhalt muss nicht an jedem Punkt der Übertragung ausgepackt werden. Außer den Vorteilen für die Entladung und die Beladungsprozess , stellt die Standardisierung von Metallkästen viele Vorteile für die Kunden zur Verfügung dar, da Schutz gegen Wetter und Diebstahl gibt. Besonders, wenn zwei oder verschiedene Mittel des Transports in eine einzelne Transportkette integriert worden sind und die Umladung des Container erfolgen müss , genaue Informationen über ihre Position, Dimension , Gewicht ermittelt werden sollen.

Der erste Containerdienst begann 1961 mit einem internationalen Containerdienst zwischen der US-Ostküste und die Karibik, Zentralen und Südamerika.Während der ersten decade des internationalen Seefrachttransport ist den Containerisierungsaktienmarkt leicht gestiegen . Eine echte Containerization kann nur erreicht werden wenn es eine Kapitalinvestition in entsprechende Schiffe und Seaport mit dazugehörigen Equipment gibt und die Verfügbarkeit von Container muss im voraus zur Verfügung gestellt werden. Im Moment über 60% der Seeschiffracht sind mit dem Container hergestellt worden ,auch wenn in solche Strecken (besonders zwischen entwickelten Ländern) 100% wird mit dem Container abgedeckt .
Ein internationaler Marktanalyse zeigt, dass im jahr 1995 9.2 Millionen TEU (Twenty foot Equivalent Units, die die Länge einen Container beschreiben), im Umlauf waren. Die Containerflotte hat sich in zehn Jahren von einer Größe von 4.9 Millionen TEU 1985 fast verdoppelt. Wegen der positiven Vorhersage für die Containerfracht transportation eine ähnliche Entwicklung kann in der Zukunft erwartet werden.

Die steigende Zahl von Containersendungen verursacht höhere Anforderungen auf den Seehafen-Containerterminals, Container Logistik, und Management, sowie auf der technischen Ausrüstung. Deshalb eine vergrößerte Konkurrenz zwischen Seehafen ,die geografisch nahe sind, ein Ergebnis dieser Entwicklung ist. Die Seehafen bewerben sich hauptsächlich für die Ozeantransportunternehmenskundschaft so wie für landgestützten Lastwagen- und Industriebahn-Dienstleistungen. Die Wettbewerbsfähigkeit eines Containerseehafens wird am meisten mit der Zeit im Hafen für Schiffe (Umladungszeit) und mit niedrigen Gebühren fur die Beladung und Entladung gekennzeichnet. Deshalb ist ein entscheidender Wettbewerbsvorteil der schnelle Umsatz der Container, der auf die Verminderung der Zeit im Hafen der Containerschiffe, und von den Kosten des Umladungsprozesses selbst verantwortlich ist. Hinsichtlich der beschriebenen Situation von Containertransportdienstleistungen wird mit diesem Papier eine Annährung um die Optimierung der Zeit im Hafen und den waterside Umladungsprozess eines Containerhafens darstellen zu können.

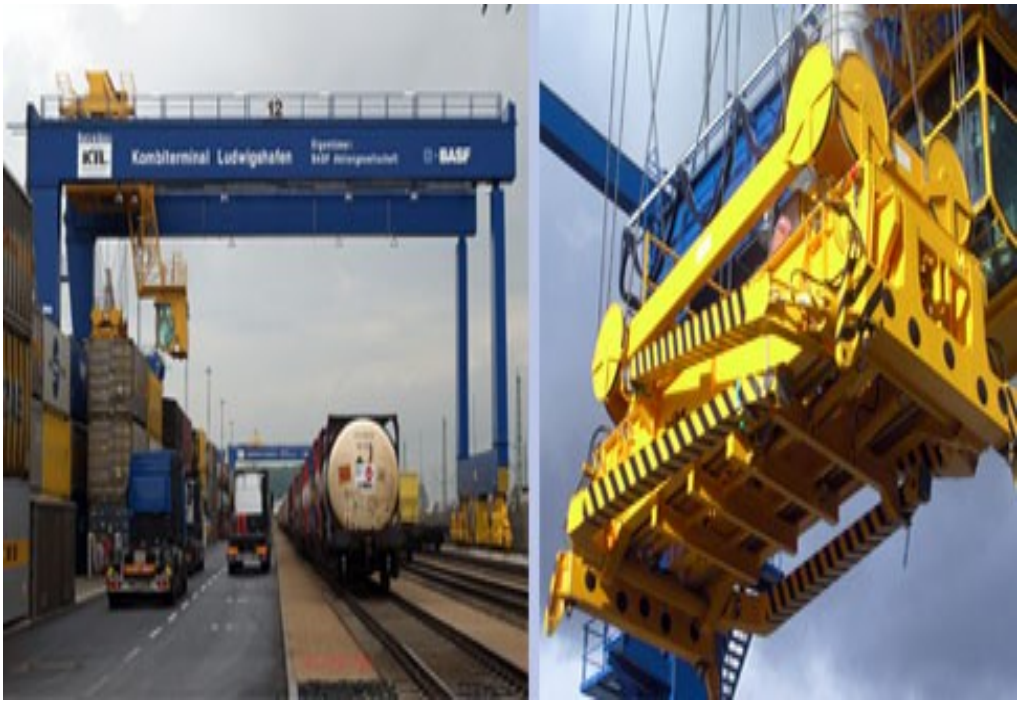
Unser Ziel ist, die Produktivität der Gantry-Kräne(die für das Containerladen und die Entladung zwischen Schiffen und Kai tätig sind) und von den Fahrzeugen die Container auf dem Hof umstellen zu optimieren. Zwei Annäherungen können vorgesehen werden, weil es organizational Änderungen sowie algorithmische Verbesserungen gibt. Während für den ersten Fall haben wir uns auf einige Ideen konzentriert ,die das Arbeitsgebiet von Straddle Carries zu Brücken ändern, im second Fall werden moderne heuristische Suchkonzepte betractet, die von evolutionären und genetischen Algorithmen Gebrauch machen.

Das Modell in diesem Papier beruht auf den Lageplan des Terminals Burchardkai in Hamburg, das durch den Hamburger Hafen- und-Lagerhaus AG AG (HHLA) geführt wird. Der folgende Abschnitt gibt eine allgemeine Übersicht der Funktionalitäten eines Containerseehafen-Terminal mit einem Fokus auf physischen Containerbewegungen.

3 Die Arbeitsweise eines Seehafen-Containersterminals und vorhandene Optimierungsannäherungen

Ein Seehafen-Containerterminal ein kompliziertes System von mehreren miteinander verbundenen Logistik-Prozessen ist. Abbildung 2 zeigt typische Operationsgebiete eines Containersterminal :Containerschiffe , Lastwagen und Züge Arbeitsprozess, und Lagergebiet . Nachher wird eine Beschreibung der illustrierten Bereiche der Aktivität sowie der vorhandenen gegenseitigen Abhängigkeiten gegeben.
Ein Containerschiff geht in den Hafen ein und wird einer passenden mit speziellen Containerkränen ausgestatteten Kaigebühr zugeteilt (auch genannt Gantry-Kräne). Alle für die Umladung zugeteilten Container werden entladen oder von Gantry-Kränen geladen. Sie müssen durch einen beschränkten Bereich unter dem Gantry-Kräne gehen,einer der engpässe im Küstenumladungsprozess wegen des beschränkten Raums. Die Containerbewegungen zu und von der Kaigebühr, die für das Laden und die Entladungsprozesses erforderlich sind, werden durch bewegliche Fahrzeuge oder Kran-Installationen ausgeführt . Abhängig von speziellen Voraussetzungen der Containersfracht (Container, die eine elektrische Verbindung benötigen oder gefährliche Güter) können an bestimmte Stelle bewegt werden. Die Mehrheit der Importcontainer wird zu Positionen in vorreservierten Bereichen in der Nähe vom Platz transportiert, wo sie beladen als nächstes sein werden. Nur eine kleine Zahl von ihnen könnte ohne Zwischenstufen auf landgestützten Mitteln des Transports umgeladen werden. Allgemein wird die Containerumladung sowie die Containerbewegung zu und vom Kai mit dem Küstenumladungsprozess verbunden (WTP), in der folgenden Abschnitte wird ausführlich darüber besprochen. Container ,die durch die Straße oder Eisenbahn am Terminal ankommen, werden innerhalb der zur Verfügung gestellten Lastwagen- und Zug Tätigkeitsbereich abgeräumt. Die Container werden zu verschiedenen Lager gelagert, die notwendigen Voraussetzungen entsprechen. Die Landseite Umladungsprozess (LTP) wird durch besondere gebaute Fahrzeuge oder Gantry-Kräne ausgeführt . Außerdem gibt es mehrere innere Bewegungen zwischen den verschiedenen Gebieten im Hinterland .
Die Bewegungen umfassen Containertransports vom Leercontainerslager bis zum Aufmachungszentrum und gepackteContainer zum Containerlager . Sie werden als einen Teil der Hinterland-Transport-Prozesse (HTP) betrachtet.

Abbildung 2



Quelle: Künz Homepage

Die Kapazitätsplanung für technische Ressource und das räumliche oder funktionelle Aufbau von Terminals ist neben der Arbeitsverteilung und der Ablaufplanungsstrategie ausgewertet. Simulationsergebnisse bieten wertvolle Entscheidungshilfe für die Terminalplaner und Betriebsleiter an.

Hinsichtlich einer passenden Automation sowie Gebrauchs von computerisierten Entscheidungshilfe-Systemen , wurden einige Prozesse in der Vergangenheit analysiert , die auf mehrere viel versprechende algorithmische Annäherungen hinauslaufen. Das Ziel ist eine intelligente Zuteilung der Technical Ausrüstung (z.B. Gantry-Kräne und Straddle -Carrier) in den verschiedenen Terminalbereiche , sowie eine effizient Job-Verteilung der verwenderten Ressourcen. Diese Algorithmen werden größtenteils mit passenden Simulierungsmodellen verbunden um die Einflüsse auf den logistischen Prozessen zu analysieren . In diesem Zusammenhang ein anderes Forschungsgebiet betrifft Ankerplatz und Lagerplanung durch der Zuteilung von Schiffen zu Kai-Positionen und Containers zu Hof-Positionen . Wegen der großen gegenseitigen Abhängigkeit werden Ankerplatz und Lagerplanung in einem gemeinsamen Optimierungsmodell oft betrachtet.

Insgesamt wird im Seehafenhinterlandverkehr zwischen der Schiffsentladung und dem eigentlichen Hinterlandtransport unterschieden. Der Hinterlandtransport wird auf der zweiten Konkretisierungsstufe jeweils für Bahn- und LKW-Transport beschrieben. Auf der dritten Konkretisierungsstufe wird jeweils ein Teilprozess der zweiten Konkretisierungsstufe beschrieben.

3.1 Schiffsentladung

Der Teil Schiffsentladung beschreibt die Vorgänge mit denen der Container vom Stauplatz auf dem Schiff zum Lager im Hafenterminal verbracht wird. Der nur selten vorkommende Direktumschlag auf ein nachfolgendes Verkehrsmittel wird an dieser Stelle nicht betrachtet.

Konkretisierungsstufe 1
Auf der ersten Konkretisierungsstufe ist die Schiffentladung ein Ortswechsel. Dabei sind folgende Aspekte zu beachten und zu koordinieren:
 Personal für die Containerbrücken, Lasher und Transportfahrzeugfahrer
 Anzahl Containerbrücken und Anzahl und Arten Transportfahrzeuge
 Dokumentation welche Container entladen werden sollen und wo sich diese im Schiff befinden
Die Anzahl der verwendeten Ressourcen richtet sich nach der erwarteten Liegezeit des Schiffes im Hafen, der Anzahl der zu entladenden Container und der Anzahl der zur Verfügung stehenden Ressourcen unter Beachtung konkurrierender Ressourcenbedarfe (parallel Be- und Entladung anderer Schiffe).

Konkretisierungsstufe 2
Auf der zweiten Konkretisierungsstufe wird die Schiffsentladung in die Prozesse
 (Be- und) Entladungsplanung,
 Kranen Schiff - Kai,
 Transport im Terminal zwischen Kai und Lager,
 Absetzen des Containers im Hafenlager und
 Dokumentation des Lagerstandortes
unterteilt.
Ausgangspunkt ist der Container auf seinem Stellplatz auf dem Schiff. Bei der Be- und Entladungsplanung werden die oben genannten Ressourcen koordiniert. Es handelt sich um einen Prozess der Behandlung. Zwar wird nicht der einzelne Container physisch verändert, jedoch wird dieser mit zusätzlichen Informationen ausgestattet (wann durch wen entladen).
Das Kranen des Containers vom Schiff auf den Kai bzw. auf ein am Kai bereitstehendes Transportfahrzeug ist ein Ortswechsel. Gleiches gilt für den Transport zwischen Kai und Lager und der Einlagerung des Containers im Hafenlager.
Zum Schluss muss der Lagerstandort des Containers erfasst werden, damit dieser im ITSystem später lokalisiert werden kann bzw. ein bereits vor geplanter Lagerort bestätigt wird.
Hierbei handelt es sich wieder um eine Behandlung.

Konkretisierungsstufe 3
Auf der dritten Konkretisierungsstufe wird der Prozess des Kranens zwischen Schiff und Kai genauer beschrieben.
Zunächst müssen auf dem Schiff die Twistlocks, die die Container zur Ladungssicherung untereinander verbinden, entriegelt werden (zu diesem Prozess gehört auch das Lösen und Entfernen der Lashstangen). Hierbei handelt es sich um eine Behandlung. Anschließend erfolgt das Kranen des Containers vom Schiff auf den Kai bzw. auf eine spezielle Plattform (Ortwechsel), wo anschließend die Twistlocks entfernt werden (Behandeln). Anschließend wird der Container auf dem Boden oder ein Transportfahrzeug abgesetzt (Ortswechsel), von wo aus der Transport in das Hafenlager erfolgt.

3.2 Hinterlandtransport

Hier wird der Hinterlandtransport am LKW- und am Bahntransport beschrieben.

Konkretisierungsstufe 1

Bei beiden Transportmitteln ist auf der ersten Konkretisierungsstufe der Hinterlandtransport angesiedelt, der im Ganzen einen Ortswechsel darstellt.
Es wird im Folgenden nicht auf die Planungsschritte (v. a. die Ressourceneinsatzplanung) der Verkehrsträger eingegangen. Lediglich die informatorischen Prozesse bezüglich des einzelnen Containers werden genannt.
Die Konkretisierungsstufen zwei und drei werden für beide Verkehrsträger getrennt beschrieben,
da hier wesentliche Unterschiede bestehen, die dennoch durch die Basisprozesse beschrieben werden können. Im Folgenden werden zunächst die Konkretisierungsstufen 2 und 3 für den Schienentransport und anschließend für den Straßentransport beschrieben.

Konkretisierungsstufe 2 - Schiene

Auf der zweiten Konkretisierungsstufe werden beim Schienentransport folgende Prozesse durchlaufen

- Verladungsplanung,
- Bahnverladung,
- Bahntransport,
- Bahnentladung und
- Dokumentation Verbleib.

Die Verladungsplanung legt fest, welche Container auf welchen Zug verladen werden. Sie legt dabei auch fest, wann die Container aus dem Lager geholt und auf den entsprechenden Zug auf einen definierten Tragwagen verladen werden. Dabei ist zu beachten, dass die Container rechtzeitig am richtigen Zielterminal sind, also auf einen entsprechenden Zug gebucht werden. Der Prozess Verladungsplanung ist eine Behandlung und muss an dem ITSystem kommuniziert werden.. Bei der anschließenden
Bahnverladung werden die Container aus dem Hafenlager zum Terminal bzw. Zug gebracht und auf einen entsprechenden Tragwagen verladen. Bei diesem Prozess handelt es sich um einen Ortswechsel. Anschließend erfolgt der Bahntransport, der ebenfalls einen Ortswechsel darstellt. Am Zielbahnhof werden die Container vom Zug entladen und entweder in ein Lager gebracht oder direkt auf weiterführende Transportmittel (meistens LKW) verladen. Auch hier handelt es sich um einen Ortswechsel. Zum Schluss muss dokumentiert werden, wo der Container nach dem Bahntransport verbleibt. Dieses ist entweder das Containerlager des Terminals zwecks späteren Weitertransports oder bei Direktumschlag die Kennung (bspw. LKW-Kennzeichen und Eigentümer) des weiterführenden Verkehrsmittels. Bei dieser Dokumentation handelt es sich wiederum um einen Behandlungsprozess.

Konkretisierungsstufe 3 - Schiene

Auf der dritten Konkretisierungsstufe wird der Bahntransport näher beschrieben.
Aus dem vorprozess steht der Container auf dem Tragwagen. Es erfolgt zunächst die Ladungssicherung, was eine Behandlung darstellt. Anschließend erfolgt eine Kontrolle des Zuges. Dabei geschieht an den Containern nichts, wodurch für diese ein Liegen entsteht. Nach der Zugkontrolle (ggf. nach Beseitigen von Mängeln) verlässt der Zug das Terminal.

Dieses ist genau so wie die Zugfahrt und die Einfahrt in das Zielterminal eine Ortsveränderung. Bei der Ausfahrt des Zuges aus dem Hafenterminal und bei der Einfahrt in das Zielterminal erfolgen jeweils parallel Dokumentationen des Containeraus- bzw. Eingangs. Beides sind Behandlungen. Im Zielterminal wird die Lokomotive abgesetzt. Dieser Prozess ist bezogen auf die Container ein Liegen (wobei u. U. die Entladung (Ortwechsel) des Zuges parallel beginnen kann). Am Ende der Prozesskette sind die Container auf dem Zug im Zielterminal.

Nachdem die Schienentransportprozesse beschrieben wurden, werden jetzt die Konkretisierungsstufen2 und 3 für den Straßentransport beschrieben.

Konkretisierungsstufe 2 - Straße
Hinterlandtransport - Straße
Auf der zweiten Konkretisierungsstufe sind folgende Prozesse zu behandeln:
- LKW-Anmeldung,
- LKW-Verladung,
- LKW-Transport und
- LKW-Entladung.

Die Prozesskette beginnt mit der Anmeldung des LKW am Terminaleingang. Dieses ist eine Behandlung, da die Information erfasst und weiter bearbeitet wird. Der Container wird anschließend auf den LKW verladen, was einen Ortswechsel darstellt. Der anschließende LKW-Transport und die LKW-Entladung sind Ortswechsel. Parallel zum LKW-Transport (genauer bei der Ausfahrt des LKW aus dem Hafenterminal) erfolgt eine Dokumentation des Hafenausgangs. Dieses stellt eine Behandlung dar. Am Ende der Prozesskette ist der Container (in der Regel) an seinem Zielort.

Konkretisierungsstufe 3 – Straße
Auf der dritten Konkretisierungsstufe wird die LKW-Verladung detaillierter beschrieben. Nach der Anmeldung des LKW an der Hafeneinfahrt erfolgt ein Verladungsauftrag an das Containerlager (bzw. die hierfür zuständige Organisationseinheit). Hierbei handelt es sich um eine Behandlung. Es erfolgt daraufhin die Auslagerung des Containers aus dem Lager, der Transport zum LKW und die Beladung des LKW. Diese Prozesse sind jeweils Ortswechsel.

4 Problemformulierung

Der primäre Fokus dieses Papiers ist die Diskussion von Ideen, die Zeit im Hafen von Containerschiffe reduzieren. Hauptsziel ist die Optimierung der Produktivität der angestellten Gantry Kräne . In Bezug auf dem Ziel, wir konzentrieren auf der Betrachtung des Küstenumladungsprozess WTP, insbesondere auf den Containerbewegungen zu und von der Kaigebühr,die von den Straddle Carriers ausgefuhrt wird. Die Maximierung der Gantry-Krane-Leistungsfähigkeit kann durch der effizient Ablaufplanung/Einplanung der gegebenen Straddle -Carrier erreicht werden. Die statische Strategie verwendet eine bestimmte Anzahl von Straddle-Carriers , die zu einer Gantry-Krane während einer Arbeitsschicht statisch zugeteilt werden. In mehreren Fällen ist diese Anweisungsstrategie für waterside Containertransporte ein Versorgungsengpäss wegen des kleinen Laderaum unter den Gantry-Kränen.

In diesem Papier setzen wir die Größe des Laderaum zu zwei Container (basiert auf den Aufbau von Gantry-Kränen) und die Zahl von Straddle -Carrier zu drei. Im Bezug auf die Entladungsprozess , muss eine Gantry-Krane ihre Arbeit unterbrechen, wenn die entsprechende Straddle-Carrier nicht in einem bestimmten Zeitfenster die Container vom Laderaum entfernen kann, das vom Durchlauf des Gantry-Krans abhängt. Analog hält der ladende Prozess an, wenn es keinen Container gibt, der für die Umladung zur Lagerung verfügbar ist, so dass der Gantry-Kran ununterbrochen arbeiten kann, um eine hohe Leistungfähigkeit zu erreichen.

Nehmen wir zunächst tg als durchschnittliche Konstantewertszeit, g Є {1,…….G} für jede G Gantry Crane zuständig für die Aufgabenfolge jg , i die mit befestigte Zeit ausgeführt wird , während i Є {1,……,ng } wo ng die Anzahl der Aufgabenfolge dargestellt.

J stellt die Umstellung von Container von einem Platz zu einen anderen dar. Im Entladungsprozess eines Schiffes wird von der Gantry Crane jede tg Sekunde ein Container runtergestellt damit dann verfügbar für die Aufgabe der Straddle Carriers sein kann Tbirth g,i (ready Time). Im Beladungsprozess das „ready Time“ stimmt mit dem Moment uberein in dem das Container in seiner Lagerungsposition und für die Beladungsprozess bereit ist. Außerdem ist die Anzahl der Straddle Carrier auf Grund der Kosten begrenzt .

Tbirth g ,i= {t initial g +(i* tg) Entladung prozess

Tbirth g ,i= {t initial g +(i - 1)* tg) – ttrans g,i Beladung Prozess

t initial =wenn eine Gantry Crane fangt ihre Arbeit auf dem Schiff

ttrans g,i=Transportzeit eines Container um die Aufgabe i durchzuführen durch eine Gantry Crane g

tg =als durchschnittliche Umladungszeit Konstantswert

Jede Straddle Carrier c wo c Є {1,…..,C} hat eine Aufgabenfolge vc in Auftrag gegeben. Die Ende der Arbeit eine Straddle Carrier tdeath c,k hängt von :

der Ende der vorstehende Job tdeath c,k 1 ,

von der umbeladene Betriebszeit tempty c,k von eine Straddle carrier um an der Aufnehmerspositionanzukommen

der Transportszeit ttrans k

der Wartezeit twait c,k ist =0 wenn die Straddle Carrier trifft die Aufnehmerspositionvorzeitig ein (bevor das „birth time“ einer Aufgabe).Diese Deathzeit wird so berechnet:

tdeath c,k= tdeath c,k 1 + tempty c,k + ttrans k+ twait c,k

wo k=1,……,vc sind die Aufgabenfolge

Das primäre Ziel ist die Reduktion alle Verzögerungen(Verzögerung wenn die Straddle Carrier kommen an der Aufnehmersposition nach dem" birthtime") , die die Gantry-Kräne Leistungfähigkeit minimieren.

$$\text{Min} \sum_{g=1}^{G} \sum_{i=1}^{J_g} \text{Max} \left(\left(t_{c,k-1}^{death} + t_{c,k}^{empty} \right) - t_{g,i}^{birth}, 0 \right)$$

In einer echte Welt verschiedene Stochasticheinflüsse konnen das WTP beeinflussen (z.B verschiedene Geschwindigkeit von Straddle -Carrier, Misserfolg der technischen Ausrüstung, usw.) . Diese Auswirkungen werden erstmal ignoriert und müssen mit einem Simulierungsmodell zusammen mit den Effekten untersucht werden, die sich aus der beschriebenen Situationen (Lagerungspuffer unter den Gantry-Kränen) ergeben. Um die Engpässe zu vermeiden, müssen wir Arbeitsreihenfolge für die verwendeten Straddle-Carrier ausrechnen, wo die Summe von der Anwesenheitszeit des Container minimal ist.Die Planung in Burchardkai ,kann so beschreibt werden :Die Disposition der Straddle Carrier beruht auf einer festzuordnung von drei Carrier zu einer Gantry Crane.
Jede Straddle Carrier wird normalerweise an genau eine Gantry Crane für die gesamte Arbeitsschicht zugeordnet .

5 Evolutionäre Algorithmen

Evolutionsstrategien, genetische Algorithmen und Evolutionärprogrammierung sind drei verschiedene Strömungen im Forschungsbereich des evolutionäre Algorithmen beziehungsweise des evolution programs. Den Ansätzen ist gemeinsam, dass sie Algorithmen darstellen, die auf Prinzipien der natural evolution basieren, aber sie unterscheiden sich in den zu Grunde liegenden Strategien.
Verglichen mit klassischen Methoden, besitzen evolutionäe Algorithmen einige Eigenschaften, die sie für bestimmte schwierige Probleme interessant machen, z.B.:

•Die Existenz einer Population, die eine Anzahl an Lösungsalternativen beinhaltet, was eine parallele Exploration des Lösungsraums erlaubt..

•Der Transfer von Prinzipien (und Terminologien) von der natural evolution, die ein großes Potenzial darstellt, um gute Lösungen innerhalb ihrer Umwelt zu schaffen. Jedoch sollte man beachten, dass derzeitige Durchführungen (und die Theorie selbst) weit von einer reinen Imitation der Natur (was generell überhaupt nicht angestrebt wird) entfernt sind, weil einige genetische Mechanismen, wie sexuelle Ausbreitung/Weitergabe, lokale Populationsentwicklung nicht beachten werden.

Vorteile der evolutionäre Algorithmen sind die Generalität für die Anwendung auf Optimierungsprobleme. Neben einer gemeinsamen Schnittstelle in der Lösungsdarstellung, ist der Algorithmus unabhängig von dem problem specifica und „lernt" die Struktur des Lösungsraums während des Optimierungsprozesses.

Evolutionäre Algorithmen lassen die effiziente Feststellung von Bereichen nahe der (lokalen) Optima, aber sind nicht gut für die Feineinstellungsstrukturen geeignet, welche nahe an den optimalen Lösungen liegen. Daher wird die Integration lokaler Suchstrategien (hybride Methoden) durchgeführt, um Individuen für die Exploration dieser lokalen Optima zu verbessern. Ohne mehr Charakteristiken des evolutionary algorithms und evolution programming offenzulegen, beschränken wir uns in diesem Aufsatz auf genetic algorithms. Diese Entscheidung beruht auf der Problem- (integer optimization problem) und Lösungsrepräsentation (permutation vector).

5.1 Grundstruktur der genetischen Algorithmen

Zuerst muss die Startpopulation (Sammlung von Individuen) mit einer vorzugsweise hohen Variabilität für die Gene (Attribute oder Individuen) der Chromosomen (Repräsentation eines Individuums oder eines Lösungsvektors) angelegt werden. Die ersuchte Variabilität kann durch die Nutzung nicht-deterministischer Methoden (willkürliche Generation von Individuen, graduell-willkürliche Variationen eines gegebenen Individuums) erreicht werden oder, wenn Spezialwissen des Problems vorhanden ist, durch die Nutzung von Startheuristiken (z.B. günstigste Einfügung, nächstbestes für sequencing Probleme), und die willkürliche Variation dieser Lösungen.
Die Population entwickelt sich über einen Zeitraum hinweg, indem sie gewisse Mechanismen nutzt, mit dem Ziel, die Fitness des Individuums, daher der Population, zu verbessern. Als erstes muss eine Evaluationsfunktion definiert werden, um die Fitness des Individuums zu messen. Diese Fitness kann für die Selektion von Individuen für die nächste Population genutzt werden, gemäß der Natur, wo die fitteren Individuen Vorteile bzgl. des Überlebens und der Reproduktion durch das Anziehen anderer haben („Überleben des Tauglichsten“).

Selektion Die Selektion von Individuen für die nächste Population basiert auf der Fitness, wobei entweder eine deterministische oder willkürliche (um eine größere Variabilität zu erreichen) Selektion benutzt wird. Individuen mit einer besseren Fitness werden mit höherer Wahrscheinlichkeit für die nächste Generation oder die Produktion von Nachkommen ausgewählt. Verschiedene Mechanismen können benutzt werden, wie fitness-proportional Selektion, Rang basierende Selektion oder Wettkampf-Selektion.
Ein Selektionsmechanismus, der nicht auf dem Genotyp basiert (die Attribute, die im Chromosom kodiert sind), wendet einen Trainingsablauf auf die Individuen an, um bessere Fitness zu erreichen. . Die Selektion ist die Intensivierung innerhalb des Suchraums, indem vielversprechende Individuen für die nächste Population behalten werden.

Rekombination Rekombination ist ein Operator, um die Attribute zwei oder mehrerer Individuen zu assozieren und zu einem oder mehreren neuen Individuen zu verbinden. Die Rekombination ermöglicht die Diversifikation der Evolution durch die Schaffung neuer Individuen, während vielversprechende Fragmente des Chromosoms beibehalten werden. In der Literatur kann eine große Anzahl von vorgeschlagenen Rekombinationsoperatoren gefunden werden. Die meisten von ihnen sind vom Problem und der der Repräsentation des Individuums abhängig. Der bekannteste Rekombinationsoperator ist die Kreuzung, welche der Kreuzung in der Biologie analog ist. Zwei oder mehr Individuen werden mit einer gewissen Wahrscheinlichkeit für die Beteiligung an einer Kreuzung ausgewählt. Willkürlich wird eine oder mehre Stellen für die Kreuzung (Kreuzungspunkt) gewählt, wo die Individuen getrennt werden. Die Stelle ist für jedes Individuum die gleiche. Die entstandenen Fragmente werden unter den Individuen ausgetauscht und mit neuen neu kombiniert. Dieser universale Operator erwägt gewisse Problemspezifikationen nicht und deshalb kann die Rekombination undurchführbare Individuen produzieren. In der Natur würden diese Individuen möglicherweise sterben (letale Mutation), wenn immer der verfügbare Reparationsmechanismus die die Konflikte nicht lösen kann. Der Korrespondent für den evolutionary algorithm ist ein problemspezifischer Reparationsoperator, der auf jedes Individuum angewendet wird, das aus der Kreuzung entstand.
Mutation Die Variabilität der Population wird durch eine Mutation (Variation) der Individuen erhöht, die mit einer gewissen Wahrscheinlichkeit auftritt. Die Variation verursacht eine Veränderung der Gene im Chromosom (z.B. Vertauschen von Genen, Löschen von Genen). Wenn das Individuum nach der Mutation undurchführbar ist, muss ein Reparationsprozess angewendet werden um die Umsetzbarkeit wiederherzustellen. Die Mutation ist ein Mechanismus, der Diversifikation, jedoch mit weniger Einfluss als die Kreuzung, bereitstellt.

5.2 Rapräsentation von Einzeilteil

Die Containerbewegungen auf dem Kai sind in einem Permutationsvector programmiert , wo eine Aufgabe von einer Gene dargestellt wird.
Die große der Grundgesamtheit ist einen Algorithmusparameter, der im allgemein nicht im Voraus bestimmt werden kann und hängt von verschiedenen Faktoren ab .
Wir nehmen ein Konstantmaß der Grundgesamtheit im jeder unseren Experiment an, aber prüfen wir verschiedene Grundgesamtheitengroße um die Parameterauswirkung zu analysieren.

Das folgende Java-Code beschreibt wie man die Zielfunktion berechnen kann.

```
s                                  (containing a permutation of the solution)
i ← 1                              (order in the solution sequence)
T ← 14400                          (length of a 4 hour shift in seconds)
C ← 1                              (number of straddle carrier)
T_C ← 0                            (time used by the straddle carrier N_c)
D_C ← 0                            (delay time of the straddle carrier N_c)
W_C ← 0                            (waiting time of the straddle carrier N_c)
while i ≤ n do                     (Go through the solution vector)
    (Duration, Delay, Wait) = getTime (s_i)
    T_C ← T_C + Duration
    D_C ← D_C + Delay
    W_C ← W_C + Wait
    if (T_C ≥ T) ∨ (s_i ≤ s_i−1 executed by the same straddle carrier) then
      C ← C + 1
      T_C ← 0; D_C ← 0; W_C ← 0
    else
      i ← i + 1
    endif
end
```

Quelle:Vehicle Dispatching at Seaport Container Terminals using Evolutionary Algorithms (2000)

Literatur

Torsten Reiners, Dirk Steenken , Stefan Vo : **Vehicle Dispatching at Seaport Container Terminals using Evolutionary Algorithms (2000)**

Hafenbau Einleger 6-seitig_04
*www.inroslackner.de/index.php/de/content/download/378/2350/version/3/file/**Hafen**logistik_&_**Hafen**bau.pdf*

Ausrüstung im Hinterland
www.chemion.com/web/de/chemion/Presse/Chemion-in-der-Presse/.../01/.../Gefaehrliche%20Ladung-Containerlogistik-*Sep2006.pdf*

Arns, M.; Bause, F.; Beilner, H.; Fischer, M.; Völker, M.**:** ***Beispielmodellierung*** *von Behälterkreisläufen im B1-Paradigma – Analyse*. Interner Bericht – Sonderforschungsbereich
559 „Modellierung großer Netze in der Logistik“ 00013,
2000.

Arns, M.; Beilner, H.; Fischer, M.; Kemper, P.; Völker, M.: ***Beispielmodellierung*** *von Beschaffungskanälen im B1-Paradigma – Analyse*. Interner
Bericht – Sonderforschungsbereich 559 „Modellierung großer Netze in der Logistik“ 00014, 2000.

Bernhard, J.; Hömberg, K.; Jodin, D**.:** ***Standardprozesse als Grundlage für die Informationsbedarfsanalyse zur Modellierung von Großen Netzen der Logistik*.**
In: Magdeburger Schriftenreihe zur Logistik – Logistikprozesse entwerfen, führen,
bewerten. Wissenschaftliche Themenhefte des Lehrstuhls für Logistik der Universität Magdeburg; Heft 21, 2005, S. 3-14. ISSN 1436-9109.

Kay Hömberg, Jan Hustadt, Dirk Jodin, Joachim Kochsiek,
Lars Nagel, Iwo Riha **:Basisprozesse für die Modellierung in großen Netzen der Logistik**

CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung Wirtschaftsinformatik

Seminararbeit

Simulation

vorgelegt von:

Jörg Oelerink

Oldenburg, den 27.11.2008

Inhaltsverzeichnis

1 Einleitung

Diese Seminararbeit befasst sich mit dem Thema Simulation hinsichtlich der Anforderungen an einen simulierten Containerhafen. Aufbauend auf der Projektgruppe Virtual Port 1 betrifft diese Arbeit die nachfolger Projektgruppe Virtual Port 2.

In dem Kapitel Grundlagen wird der Begriff der Simulation allgemein geklärt, zudem wird auf den allgemeinen Simulationsaufbau, sowie verschiedene Simulationsarten eingegangen. In dem darauffolgendem Kapitel Probblemstellung wird eine Beschreibung der Simulationsgrundlage von Virtual Port 2 gegeben. Des weiteren werden Anforderungen definiert, welche die Simulation aus Sicht der Simulation des Jade Weser Ports in Virtual Port 2 erfüllt werden sollen. Anschließend wird in dem Kapitel Simulationswerkzeuge auf die konkreten Anforderung an ein Simulationswerkzeug eingegangen. Die vorgestellen in Frage kommenden kommerziellen Simulationswerkzeug werden in einem Vergleich gegenübergestellt. Die Fragestellung, welches Simulationswerkzeug in Virtual Port 2 am sinnvollsten genutzt werden kann, wird abschließend in dem letztem Kapitel beantwortet.

2 Grundlagen

2.1 Definition

In diesem Abschnitt möchten wir den Begriff Simulation klären. Hierzu gibt es folgende Definitionen:

Gemäß der VDI-Richtlinie 3633 ist:

„Simulation (...) das Nachbilden eines Systems mit seinen dynamischen Prozessen in einem experimentierfähigen Modell, um zu Erkenntnissen zu gelangen, die auf die Wirklichkeit übertragbar sind. Im weiteren Sinne wird unter Simulation das Vorbereiten, Durchführen und Auswerten gezielter Experimente mit einem Simulationsmodell verstanden.“

Eine allgemeinere Definition besagt:

Simulation ist die Nachbildung realer Prozesse mittels Computern auf Grundlage mathematischer Modelle.

Allgemein kann man also sagen, dass Simulation die Abstraktion eines realen Systems auf ein simulierfähiges und dadurch reduziertes Modell bedeutet.

2.2 Simulationsaufbau

Ein allgemeiner Simulationsaufbau ist in Abbildung 1 dargestellt. Am Anfang steht das zu simulierende System. Dies wird in der Modellierungsphase modelliert. Das Modell wird auf die für die Simulation wesentliche Dinge reduziert. Anschließend wird das Simulationsexperiment durchgeführt. Stellt man hierbei etwaige Fehler im Modell fest, muss das Modell überarbeitet werden. Dargestellt durch den ruckläufigen Pfeil zur Modellierung. Ist das Simulationsexperiment erfolgreich ausgeführt worden, werden die Ergebnisse analysiert und eventuelle Optimierungsmöglichkeiten ausgelotet. Diese Versuche werden so lange wiederholt, bis man eine Optimalkonfiguration des Modells erhält und diese dann auf das reale System abbildet.

2.3 Simulationsarten

Allgemein lassen sich zwei Arten nach [Ste94] von Simulationen unterscheiden:

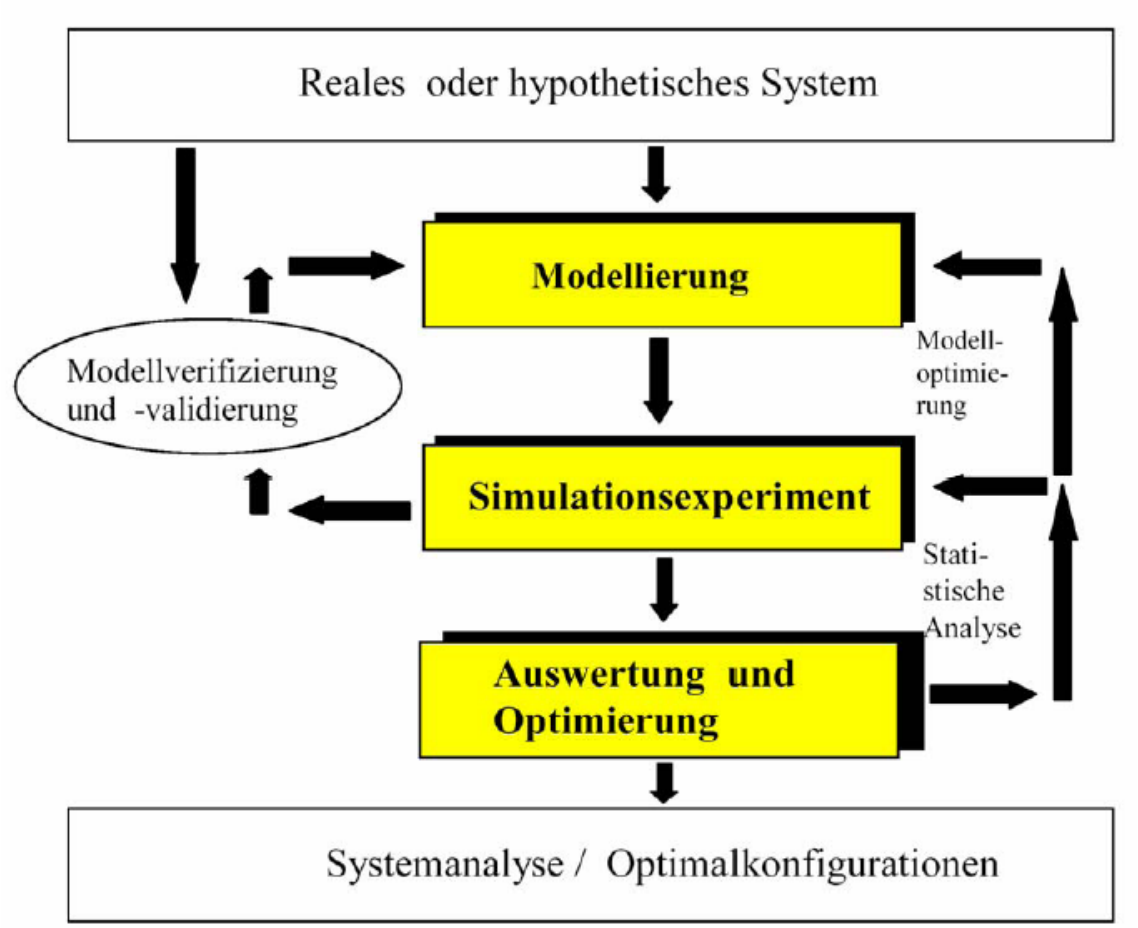


Abbildung 1: Simulationsaufbau

- Unterrichtssimulation
- Forschungssimulation

In einer Unterrichtssimulation werden komplexe Systeme trainiert. D.h. der Proband kann eine Situation simuliert erleben ohne das die Folgen tatsächliche Schäden verursachen. Ein sehr bekanntes Beispiel hierfür ist der Einsatz von Flugsimulatoren. Diese Art der Simulation ermöglicht ein großes Spektrum an Situationen in einer Art Zeitraffer zu erleben, wofür man in der Realität sehr lange brauchen wurde.

Die Forschungssimulation wird nochmal in drei Aufgaben unterteilt:

- Pfadanalyse
- Optimierung
- Stabilisierung und Regelung

Diese Simulation werden vor allem mit mathematischen Modellen beschrieben [Bos94] [Bos04]. Hierbei ist es unerheblich, ob es das untersuchte Modell ein technische, wirtschaftliches oder ein biologisches System beschreibt.
Die Pfadanalyse beschreibt einen Simulationsvorgang, der bei gleichbleibenden Startparametern unterschiedlich simuliert wird. Ein Beispiel hierfür ist die Entwicklung der Geburtenrate bei unterschiedlichen Methoden zur Geburtenkontrolle.
Die Optimierung hingegen hat den Sinn die besten Startparameter für ein, nach den vorgebenen Kriterien, optimales Systemverhalten zu finden.
Unter Stabilisierung und Regelung werden Systeme untersucht, welche die Notwendigkeit von Strukturänderungen aufgrund von bestimmten Schwingungsneigungen zu finden.

Für alle Simulationsarten gibt es noch die Unterscheidung nach einer deterministischen und einer stochastischen Simulation. Bei einer derterministischen Simulation sind alle benötigten Daten und Regeln zur Erstellung des Modells bekannt und eindeutig. Die Wechselwirkungen der Objekte innerhalb eines Modells, werden als konstant angenommen. D.h. bei gleichbleibenden Startparametern ist das Verhalten des Modells zu jedem Zeitpunkt vorhersehbar. Insbesondere bei komplexen Simulation sind diese Annahmen eher nicht gegeben, da die exakten Zusammenhänge der Objekte im Modell nicht bekannt sind, kommt es zu unbekannten stochastischen Störgrößen.
Bei der stochastischen Simulation geht man davon aus, dass die Eigenschaften so wie der Relation im System vom Zufall abhängen. Diese Problem tritt z.b. bei der Simulation eines Supermarktes auf. Das Kaufverhalten eines einzelnen Kunden ist hier eher zufälliger Natur, als exakt vorhersehbar. Durch solche vom Zufall abhängigen Eingangswerte im System ist kein eindeutiger Modellzustand vorhersehbar. Daher werden Wahrscheinlichkeitsverteilung

angenommen, diese können nur duch eine repräsentative Anzahl von Simulation ermittelt werden.

3 Probblemstellung

3.1 Beschreibung

In der Projektgruppe Virtual Port 2 soll der noch im Bau befindliche Jade Weser Port simuliert werden. Es wird also ein Simulationswekrzeug benötigt, welches die Simulation einzelner Objekte in einem Gesamtsystem ermöglicht. Das Gesamtsystem umfasst den Hafen mit Lagern, Andockplätze für die Schiffe usw. . Die zu simulierenden Objekte sind die Kräne, Schiffe, Container, Züge, LKWs und weitere.

3.2 Anforderung

Für unser Projekt kommen nur ereignisgesteuerte Simulatoren in Frage. Des weiteren muss eine parallele Simulation einzelner Objekte im System möglich sein. Außerdem muss der Simulator die Möglichkeit zur Echtzeitsimulation bereitstellen.

4 Simulationswerkzeuge

Nach den oben genannten Anforderungen kommen die folgenden drei Simulationswerkzeuge in die engere Auswahl:

- Witness
- eM-Plant
- Arena

Hierbei ist zu erwähnen, dass die Vorgruppe Virtual Port 1 EmPlant als Simulationswerkzeug genutzt hat.

4.1 Vergleich

In diesem Abschnitt werden die verschiedenen Simulationswerkzeuge verglichen. In der Tabelle sieht man, dass sich die Kosten für die einzelnen Produkte doch stark unterscheiden. Acuh aufgrund der Kosten war es dem Autor nicht möglich, alle Simulationswerkzeuge selber zu testen. Der hohe unterschied bei den Kosten ist vermutlich auf den Support zurückzuführen. Die Fähigkeit das simulierte Modell in Echtzeit darzustellen besitzen alle

drei Werkzeuge in einer 2D Simulation. Bei einer 3D Simulation ist das Arena Simulationswerkzeug den anderen Beiden im Nachteil, da diese Simulation erst im Nachhinein gerendert werden kann. Die Verbreitung der Tools wurde anhand von Googletreffern ermittelt. Die entsprechenden Suchanfragen wurden am 8.12.2008 auf google.de durchgeführt. Demnach ist das Arenatool am weitesten verbreitet, gefolgt von eM-Plant, welches insbesonder viele deutschsprachige Seiten als Ergebnis hatte. Das Witness Simulationswerkzeug ist demnach noch so verbreitet, wie die anderen beiden. Die Bedienung kann hier vom Autor nur anhand der Testversion des jeweiligen Simulationswerkzeuges beurteilt werden. Hierbei erwiesen sich alle Werkzeuge als nicht gerade Einsteigerfreundlich, ein richtiger Debugger ist bei allen nicht gegeben und man muss auch für das jeweilige Werkzeug ein eigene Syntax erlenen. Diese ist zumeist an der Programmiersprache C angelehnt. Lediglich eM-Plant lieferte ein umfangreiches Turorial mit, welches die Einarbeitung ein wenig vereinfachte. Eine nicht vorhandene möglichkeit zur Datenbankanbindung, wäre ein Ausschlusskriterieum gewesen, da die einzelnen Simulation über eine Datenbank geschaltet werden sollen. Diese Anforderung erfüllen alle drei.

	Arena [Are]	**eM-Plant**	**Witness**
Anbieter	Rockwell Automation [Aut]	Tecnomatix [Tec]	Lanner [Lan]
Kosten	795$ pro Arbeitsplatz	50000	Ab 8925
2D Simulation	Ja	Ja	Ja
3D Simulation	Rendered	Echtzeit	Echtzeit
Verbreitung	+++	++(in Deutschland)	+
Bedienung	+	+	+
Datenbankanbindung	Ja	Ja	Ja

5 Zusammenfassung und Fazit

Zusammenfassend läßt sich eine Empfehlung für das eM-Plant Simulationswerkzeug aussprechen. Vor allem die Kosten können durch den weiteren Einsatz dieses Werkzeuges gering gehalten werden, da die Lizenz bereits im Besitz der Projektgruppe ist. Ein weiterer Vorteil liegt in der Verbreitung, da vor allem viele deutsche Universitäten eM-Plant einsetzen ist so von einem gutem Feedback in deutschen Foren auszugehen. Der Supportvorteil von eM-Plant besteht insbesondere auch in der Vorgruppe Virtual Port 1, da diese das Werkzeug bereits ein Jahr eingesetzt haben, wird man dort auf unkompliziertem Weg Hilfe bekommen können.

Literatur

[Are] Arena. *http://www.arenasimulation.com/*. 8

[Aut] Rockwell Automation. *http://www.rockwellautomation.com/*. 8

[Bos94] H. Bossel. *Modellbildung und Simulation*. vieweg-Verlag, 1994. 6

[Bos04] H. Bossel. *Systeme, Dynamik, Simulation: Modellbildung, Analyse und Simulation komplexer Systeme*. Books on Demand Gmbh, 2004. 6

[Lan] Lanner. *http://www.lanner.com/*. 8

[Ste94] D. Steinhausen. *Simulationstechniken*. Oldenbourg-Verlag, 1994. 4

[Tec] Tecnomatix. *http://www.emplant.de/*. 8

Literaturverzeichnis

[1] Deutschland, Statistisches Bundesamt: *Pressemitteilung Nr. 527 vom 27.12.2007*. Website, 2007. http://www.destatis.de/jetspeed/portal/cms/Sites/destatis/Internet/DE/Presse/pm/2007/12/PD07_527_463.psml;
Letzter Zugriff: 05.01.2009.

[2] GmbH, JadeWeserPort Realisierungs: *Begr*.

[3] Hans-Otto Günther, Kap Hwan Kim und: *Container Terminals and Cargo Systems*. Springer Verlag, Berlin Heidelberg, 2007.

[4] Jürg Kuster, Eugen Huber, Robert Lippmann: *Handbuch Projektmanagement*. Springer, Berlin, Heidelberg, 2008.

[5] Meier, Leif Hendrik: *Koordination interdependenter Planungssysteme in der Logistik*. Gabler Edition Wissenschaft, Göttingen, 2008.

[6] Pichler, Roman: *Agiles Projektmanagement: Eine Einf*.

[7] Pumpe, Dieter: *Ein Referenzmodell zur Planung und Steuerung der Abläufe in Seehafen-Containerterminals*. Mensch und Buch Verlag, Berlin, 2000.

[8] Schott, Rainer: *Stauplanung für Containerschiffe*. Vandenhoeck & Ruprecht, Göttingen, 1989.