Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

André Bürger

Konzeption und Realisierung eines Dock- und Yard-Management-Systems

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

André Bürger

Konzeption und Realisierung eines Dock- und Yard-Management-Systems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Angewandte Informatik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Stefan Sarstedt

Abgegeben am 08.12.2011

André Bürger

Thema der Bachelorarbeit
Konzeption und Realisierung eines Dock- und Yard-Management-Systems
Stichworte
Logistik, Hoflogistik, J2EE, Google Web Toolkit, Cross Docking, Zeitfenster
Kurzzusammenfassung
Neben dem Warenfluss beschäftigen sich logistische Unternehmen mittlerweile vermehrt mit dem Informationsfluss. Informationen zu logistischen Prozessen dienen nicht nur der eigenen Verbesserung der logistischen Prozessabläufe, sondern können zudem Kunden zur Verfügung gestellt werden zur Steigerung deren Prozesseffizienz. Dock und Yard Management Systeme unterstützen im Bereich der Lagerlogistik bei der Koordination und dienen als Informationsquelle bei der Entscheidungsfindung zur Prozessverbesserung. Das Ziel dieser Arbeit ist der Entwurf und die Entwicklung eines Dock und Yard Management Systems, welches die Anforderungen an ein modernes System erfüllt. Zur Ermittlung dieser Anforderungen sind Interviews mit zwei Lagerlogistikunternehmen durchgeführt worden. Ein daraufhin konzipierter und realisierter Prototyp soll dann als erster Schritt für ein marktreifes Dock und Yard Management System dienen.

André Bürger

Title of the paper
Design and realization of an Dock- and Yard-Management-System
Keywords
Logistic, Dock- and Yard-Management, J2EE, Google Web Toolkit, Cross Docking, Time-Slot
Abstract
For logistic companies the flow of information becomes more vital additional to the flow of goods. The own logistical processes could be improved with the encouragement of information. Furthermore the logistical companies could make information available for their customers to use the message for increasing their efficiency. Dock and Yard Management Systems support the logistical warehouse by coordination and act as information source to find decisions for increasing processes. The intention of this bachelor thesis is the design and the development of a Dock and Yard Management System which complied the requirements of a modern system. To determine the requirements interviews have been realized with two logistic companies. Based on these requirements a prototype is designed and developed. This prototype can be regarded as a first step on the way to a market-ready system.

Danksagung

Ich bedanke mich bei Herrn Prof. Dr. Olaf Zukunft für die hervorragende Betreuung dieser Arbeit. Weiterhin bedanke ich mich bei Herrn Prof. Dr. Stefan Sarstedt, der sich bereit erklärt hat, die Zweitkorrektur dieser Arbeit zu übernehmen.

Einen besonderen Dank richte ich an meine Eltern, meiner Schwester und meinem Bruder, die mich in der Zeit, während diese Arbeit entstand, unterstützt haben und mir immer zur Seite standen. Meinen Eltern danke ich zudem noch, dass sie mir meine Ausbildung ermöglicht haben.

Ein weiterer besonderer Dank gilt meinem Freund Marc Bothur, der mir den Rücken frei gehalten hat, so dass ich mich ganz auf diese Arbeit konzentrieren konnte.

Ich danke den Interviewteilnehmern der Firmen Cargo Service Nord GmbH, UeTG GmbH und Semmelhaack Logistik GmbH für ihre Zeit und Mühe.

André Bürger, Dezember 2011

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Algorithmenverzeichnis

Kapitel 1

Einleitung

Die Nutzung der Informationstechnologie in der Logistik steigt in den letzten Jahren stetig. In der Logistik spielen Informationen eine grosse Rolle. Der Informationsaustausch, die Informationsgewinnung und die Informationsverarbeitung gewinnen immer mehr an Bedeutung, daher bietet die Kombination von Logistik und Informatik ein grosses Potential für neue Ideen und Entwicklungen.

1.1 Motivation

Durch die stetige Nachfrage an Gütern steigt die Menge der Waren, die täglich in Logistiklägern angeliefert und abgeholt wird. Die Informationsmenge, die diese Anlieferungen und Abholungen generieren, wächst entsprechend. Für den Menschen wird es immer schwieriger noch den Überblick über die Informationen zu behalten und die wichtigen Informationen herauszufiltern. Zudem muss dies auch noch in immer kürzer werdender Zeit geschehen, da durch die Zunahme der Umschlagsmenge die Zeitspannen zwischen den Ladevorgängen abnimmt. Ein Dock und Yard Management System unterstützt hier den Menschen, indem es die Daten sammelt, verarbeitet, aufbereitet und übersichtlich darstellt. Auf dieser Grundlage können Entscheidungen schneller getroffen werden, Prozesse besser koordiniert und analysiert werden.

Die Cargo Service Nord GmbH betreibt ein solches Logistiklager, welches in der letzten Zeit eine stetige Zunahme des Umschlages verzeichnete und in Zukunft noch weiter verzeichnen wird. Das System, welches die Cargo Service Nord GmbH in diesem Lager zur Verwaltung der Rampen einsetzt, ist nicht weiterentwickelt worden, um die über die Zeit gestiegenen Anforderungen zu erfüllen. Es ist also notwendig, das alte System zu ersetzen. Ein Besuch der Logistikfachmesse „Logimat” im März diesen Jahres in Stuttgart und weitere Marktrecherchen zeigten, dass es im deutschsprachigen Raum nur wenige Anbieter von Dock und Yard Management Systemen speziell für den mittelständichen Logistikbereich gibt.

1.2 Ziel der Arbeit

Ziel dieser Arbeit ist ein Konzept für ein modernes Dock und Yard Management System. Durch Analysen und Interviews soll eine moderne Systemarchitektur entworfen werden, die anschliessend in einem Prototypen realisiert wird, um die technische Machbarkeit zu beweisen. Zum Schluss wird der Prototyp noch getestet, um seine Qualität zu bestimmen.

Das in dieser Arbeit zu entwickelnde System soll in der Lage sein, das alte System abzulösen. Dabei sollen natürlich Schwachstellen des alten Systems besser gelöst werden. Zusätzlich soll das neue System in der Lage sein, die heutigen Anforderungen zu erfüllen, und durch Weiterentwicklung zukünftige Anforderungen umzusetzen.

1.3 Gliederung der Arbeit

Das Kapitel 2 „Grundlagen" befasst sich mit den Grundlagen dieser Arbeit. Zunächst wird der Begriff „Logistik" kurz erläutert zum besseren Verständnis und Einordnung eines allgemeinen Dock und Yard Management Systems, welches im Anschluss beschrieben wird. Danach werden die Techniken erklärt, mit dessen Hilfe der Prototyp realisiert worden ist.

Im Kapitel 3 „Analyse" wird zunächst das Altsystem einer Analyse unterzogen. Aus den Ergebnissen dieser Analyse zusammen mit den Interviewergebnissen werden die Anforderungen an das Neusystem spezifiziert.

Das Kapitel 4 „Systemarchitektur" beschreibt die Architektur des neuen Systems. Dabei wird die Systemarchitektur in drei Architekturen aufgeschlüsselt. Die A-Architektur beschreibt die fachlichen Komponenten, die T-Architektur die technischen Komponenten und die TI-Architektur die Techniken des Systems.

Im Kapitel 5 „Prototyp" erfolgt die Vorstellung des Prototypen. Zunächst werden ausgewählte Techniken des Prototypen vorgestellt. Danach erfolgen Beschreibungen von ausgesuchten Komponenten. Zum Schluss wird noch auf Probleme eingegangen, die sich während der Entwicklung herausstellten.

Im Kapitel 6 „Test" wird ein Testkonzept für den Prototypen erstellt und anschliessend durchgeführt.

Das Kapitel 7 „Schlussbetrachtung" zieht ein kurzes Fazit der Arbeit und gibt einen Ausblick auf die Weiterentwicklung des Systems.

Kapitel 2

Grundlagen

In diesem Kapitel werden zur Erlangung eines allgemeinen Verständnisses grundlegende Begriffe erläutert. Zuerst wird die Logistik im allgemeinen erklärt. Danach folgt dann eine Definition von Dock und Yard Management Systemen. Und zum Schluss werden die in dieser Arbeit genutzten Technologien näher erklärt.

2.1 Logistik

Das Thema Logistik ist sehr umfangreich, was sich auch an der zur Verfügung stehenden Literatur erkennen lässt. Diese kurze Einführung in das Themengebiet Logistik ist bei weitem nicht vollständig und soll auch nur dem besseren Verständnis bzgl. des Einsatzumfeldes von Dock und Yard Management Systemen dienen.

In dieser Arbeit sollen die Logistiksysteme beschrieben werden, die bei Industrie- und Handelsunternehmen sowie bei Dienstleistungsunternehmen, deren Kernaufgabe die Logistik ist, zum Einsatz kommen.

2.1.1 Begriffe der Logistik

Der Begriff „Logistik" stammt ursprünglich vom Militär. Die Logistik beschrieb dort Vorgänge wie die Nachschubregelung der kämpfenden Einheiten, die Unterbringung der Truppen oder deren Bewegung von einem Ort zum anderen. Erst Mitte der 1950er Jahre wurde die „Logistik" erstmals in zivilen wirtschaftlichen Bereichen in den USA übernommen(Sch09b; Pfo10; Wik11d; Möl07).

In der Literatur ist keine eindeutige Definition zum Begriff Logistik zu finden, daher soll an dieser Stelle die Definition von Schulte genannt werden, der die Logistik versteht als *„marktorientierte, integrierte Planung, Gestaltung, Abwicklung und Kontrolle des gesamten Material- und dazugehörigen Informationsflusses zwischen einem Unternehmen und seinen*

Lieferanten, innerhalb eines Unternehmens sowie zwischen einem Unternehmen und seinen Kunden."(Sch09b).

Eine andere aber etwas direktere Definition von Logistik stammt von E. Grosvenor Plowman, der die Logistik mit seinen 7 R's der Logistik (Seven-Rights) beschreibt:

Es gilt,

- das richtige Gut,
- in der richtigen Menge,
- im richtigen Zustand (Qualität),
- am richtigen Ort,
- zur richtigen Zeit,
- für den richtigen Kunden,
- zu den richtigen Kosten

zur Verfügung zu stellen.

Die Aufgabe ist es also, die Güter bei Bedarf zwischen Orten zu transportieren, umzuschlagen oder zu lagern. Interne Orte können bspw. das Lager und ein Produktionsbereich sein. Externe Orte sind bspw. die Unternehmen eines Lieferanten und eines Kunden. Für die Durchführung, die Planung, die Steuerung und die Kontrolle der logistischen Aufgaben werden Informationen benötigt. Die Informationen können nach einer Planung eines Güterflusses einen Transportauftrag auslösen. Während des Transportauftrages können die Informationen diesen kontrollieren, ob das Gut auch rechtzeitig am richtigen Ort sein wird oder ggf. ihn steuern. Dies zeigt sehr gut, dass die Informationen ein wichtiger Bestandteil der Logistik sind und somit auch die Informationssysteme, die im Abschnitt 2.1.3 näher betrachtet werden.

Supply Chain

Der Begriff „Supply Chain" ist eines der Schlagwörter in der Logistik, daher soll es hier kurz erläutert werden. Als Supply Chain bezeichnet man den von der Logistik betrachteten erweiterten Güter- und Informationsfluss. So kommen gem. Schulte (vgl. (Sch09b, S.14/15)) noch zusätzlich die Betrachtung der Lieferanten von den Lieferanten (Rohstofflieferanten, Teilelieferanten, Komponentenlieferanten) und bei den Kunden die Kunden (Großhandel, Einzelhandel, Endkunde) dazu.

2.1.2 Teilbereiche der Logistik

Der Güter- und Informationsfluss beginnt beim Lieferanten, läuft über das Unternehmen bis zum Verbraucher. Diesen Fluss kann man auch in Phasen aufteilen (Pfo10; Möl07).

- Beschaffungslogistik

  kümmert sich um den Güter- und Informationsfluss vom Lieferanten (Beschaffungsmarkt) bis zum Eingangslager des Unternehmens.

- Produktionslogistik

  kümmert sich um den Güter- und Informationsfluss vom Eingangslager über die Produktion bis zum Ausgangslager.

- Distributionslogistik

  kümmert sich um den Güter- und Informationsfluss vom Ausgangslager bis zum Kunden.

- Entsorgungslogistik

  Rückfluss vom Verbraucher, bspw. Leergut oder Recyclingware.

Abbildung 2.1 zeigt die Teilbereiche der Logistik eines Unternehmens und die Einordnung eines Dock und Yard Management Systems. Daneben gibt es noch weitere Teilbereiche, die hier aber nicht weiter erläutert werden sollen.

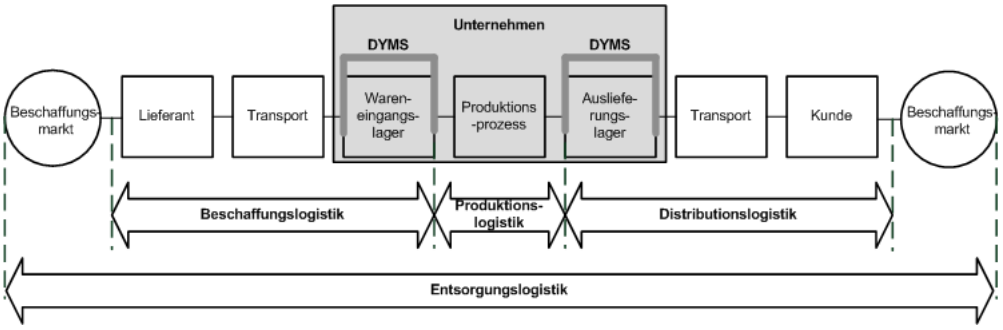

Abbildung 2.1: Teilbereiche der Logistik und Einsatzumfeld DYMS (angelehnt an (Pfo10; Joc04))

2.1.3 IT in der Logistik

Informationen sind ein wichtiger Bestandteil in der Logistik, wie der Abschnitt 2.1.1 gezeigt hat. Ohne sie sind die komplexen Güterflüsse heutzutage kaum zu beherrschen. Man kann

hier auch von der „Informationslogistik" sprechen, die analog zur Definition der Logistik (siehe Abschnitt 2.1.1) die Aufgabe hat, die richtige Information, zur richtigen Zeit, im richtigen Zustand, am richtigen Ort zur Verfügung zu stellen. In der heutigen Zeit übernehmen Informationssysteme die Aufgabe, diese Informationen schnell und korrekt zu übermitteln.

Einen groben Überblick über die Aufgaben von IT-Systemen in der Logistik, oder auch Logistik-IT genannt, soll die Abbildung 2.2 geben. Die Abbildung 2.2 zeigt, dass logistische Informationssysteme in drei Aufgabenbereiche (Lenkungsaufgaben, Basisaufgaben, Gestaltungsaufgaben) geteilt werden können.

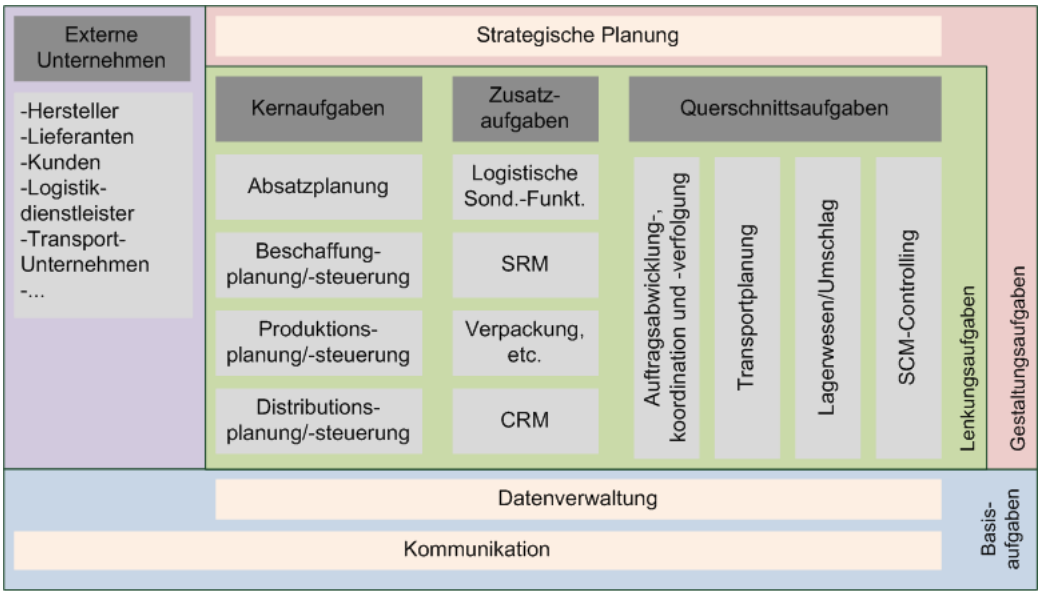

Abbildung 2.2: Geänderte und ergänzte Darstellung Aufgabenmodell Logistik-Software (KPW10, S.21)

Lenkungsaufgaben

Die Unterstützung eines oder mehrerer logistischer Teilbereiche (siehe Abschnitt 2.1.2) gehört zu den Kernaufgaben der Logistik-IT. Zusätzlich können noch weitere Aufgaben unterstützt werden, wie bspw. Verpackungs- oder Signierungsaufgaben, oder Aufgaben, die z.T. in den SRM oder CRM-Bereich fallen. Die Unterstützung bei Logistikprozessen (Transport, Lagerung, Umschlag) bilden den dritten Aufgabenbereich bei den Lenkungsaufgaben der Logistik-IT. Dabei gilt zu beachten, dass die Logistikprozesse, und somit auch die Logistik-IT, die diese unterstützen, über alle logistischen Teilbereiche zum Einsatz kommen. Diese

Vielzahl an Aufgaben, die zu bewältigen sind, hat oft zur Folge, dass eine Fülle von Systemen eingesetzt werden.

Basisaufgaben

Durch den Einsatz von verschiedenen Systemen kann es dazu kommen, dass jedes System seine eigene Datenbasis besitzt. Dies führt zu dem Problem, dass Daten mehrfach erfasst werden müssen und es somit zu Dateninkonsistenz zwischen den Systemen kommen kann. Daher sollten die logistischen Systeme idealerweise auf einer gemeinsamen Datenbasis arbeiten.

Eine weitere wichtige Grundlage ist die Kommunikation zwischen den Systemen, sowohl zwischen den internen Systemen untereinander, als auch unternehmensübergreifend mit externen Systemen. Für den reibungslosen Datenaustausch (EDI[1]) bedarf es einer Standardisierung. Einer dieser Standards ist der EDIFACT[2]-Standard. Der Sinn der Kommunikation ist unteranderem, dass die Informationen den Güterfluss begleiten oder optimaler, ihm voraus eilen. Damit hat ein Lagermeister bspw. die Möglichkeit durch vorauseilende Informationen den Wareneingang reibungslos und effizient zu gestalten, da er im Vorwege entsprechende Mittel und Ressourcen bereithalten kann.

Eine verbesserte Kommunikation zwischen den Systemen ist auch z.T. durch das sich stetig verbreiternde Web erst möglich. Weiterhin helfen Konzepte wie EAI[3] oder SOA[4] auch bei der Verbesserung der Kommunikation.

Gestaltungsaufgaben

Die Datenflut aus der Vielzahl der Informationssysteme muss aufbereitet und dargestellt werden. Sie dienen dann als Grundlage für das Management, um die zukünftigen strategischen Ziele des Unternehmens zu definieren.

Die Logistik-IT hat mit der Problematik zu kämpfen, dass sie in den meisten Betrieben nur als Kostenfaktor gesehen wird, der notwendig ist, um die Güterflüsse zu planen und zu optimieren. Hier findet langsam ein Umdenken dahingehend statt, dass die Logistik-IT auch als „Business Enabler" dem Unternehmen beim Gewinnen neuer Kunden und Geschäftsfelder dienen kann. Denn das blosse Erledigen von Transportleistungen unterscheidet ein Logistikunternehmen nicht mehr von einem anderen. Der Kunde möchte einen Mehrwert haben, und

[1] EDI = Electronic Data Interchange

[2] EDIFACT = Electronic Data Interchange For Administration, Commerce and Transport, siehe (Sta11b)

[3] EAI = Enterprise Application Integration, Verbindung der verschiedenen Geschäfts-Anwendungssysteme mittels eines Business Bus. Weiterführende Literatur (Kai02).

[4] SOA = Service Oriented Architecture, Businesskomponenten eines Unternehmens, deren Services über ein Netzwerk erreichbar sind.

dies kann ihm die Logistik-IT in Form von wichtigen und schnellen Informationen liefern. Das Umdenken kann am Beispiel von Kühne + Nagel gezeigt werden, welche fast ihre gesamten Unternehmensapplikationen selbst entwickeln (Lix09). Zum einen hat es einen Kostenvorteil und zum anderen erhöht es die Flexibilität bspw. schnell auf Kundenbedürfnisse eingehen zu können, um besagten Mehrwert zu liefern.

2.2 Dock und Yard Management Systeme

Dock und Yard Management Systeme(DYMS)[5] sind logistische Informationssysteme, die im Kontext von Warenverteilzentren, Lagerhäusern und Speditionsanlagen zu finden sind. Diese Gebäudekomplexe bestehen i.d.R. aus zwei Komponenten. Dem eigentlichen Gebäude und dem dazugehörigen Betriebshof (Yard). Der Betriebshof ist mit dem Gebäude über Schnittstellen, in Form von Rampen (Docks), verbunden. Im Gebäude übernehmen Lagerverwaltungssysteme (LVS) die Tätigkeit, die Lageraufgaben (Wareneinlagerung, Warenauslagerung, Kommissionierung, Bestandspflege, Wegeoptimierungen usw.) zu unterstützen. LVS beginnen bzw. enden ab dem Punkt, an dem die Ware sich auf der Rampe befindet. Die Dock und Yard Management Systeme hingegen verwalten, planen, steuern und protokollieren zum einen die Rampen (Dock Management) und zum anderen den Betriebshof (Yard Management).

An dieser Stelle soll auf die Diplomarbeit von Thorsten Jochheim hingewiesen werden, der auf Grundlage von einigen ausgewählten DYMS, eine allgemeine Einführung in die Thematik Dock und Yard Management Systeme gibt (Joc04).

2.2.1 Funktionsumfang

DYMS gibt es in verschiedenen Ausprägungen. So gibt es kleine Systeme, die die notwendigsten Daten aufnehmen, und deren Steuerung komplett manuell erfolgt. Sie dienen lediglich der übersichtlichen Unterstützung und Protokollierung. Dann gibt es Systeme, die vollautomatisch die Steuerung des Betriebshofes und der Rampen übernehmen. Gemeinsam haben aber alle Systeme, dass sie die Daten in Echtzeit verarbeiten und darstellen müssen, da die Logistik ein ständiger Güter- und Informationsfluss ist, der nicht ins Stocken geraten darf.

DYMS kann man in die zwei Funktionsbereiche „Speichern von Informationen” und „Planung und Steuerung” unterteilen. Beide werden nachstehend erläutert.

[5] Als Synonym wird auch oft verwendet nur das Yard Management, die Hoflogistik oder das Hof-Management. Bei allen ist gemeinsam, dass die Rampenverwaltung zwar inkludiert ist, sie aber nicht explizit erwähnt wird.

Speichern von Informationen

Bei den Informationen unterscheidet man zwischen statischen und dynamischen. Die statischen Informationen umfassen sämtliche festen und langfristigen Daten aller Prozessteilnehmer, wie bspw. Rampenanzahl, Rampenhöhe, Parkplatzanzahl auf dem Betriebshof, Nutzlast der Rangierfahrzeuge usw.

Durch die stetigen Aktivitäten, Be-/Entladen, Rangieren, usw., ändert sich der Zustand der Prozessbeteiligten ständig. Dadurch fallen permanent dynamische Informationen an, wie bspw. der Rampenstatus, Informationen über die Fahrzeuge auf dem Betriebshof oder deren Position usw.

Planung und Steuerung

Dieser Funktionsbereich ist das eigentliche Herzstück der DYMS. Auf Basis der statischen und dynamischen Informationen plant und steuert das DYMS die Aktivitäten. Zur Verdeutlichung sind im Folgenden vier Aktivitäten erläutert:

- Nach der Anmeldung wird dem LKW entsprechend seines Auftrages (Be-/Entladen) eine Wartefläche oder direkt eine Rampe zugewiesen. Auf der Wartefläche wird der Fahrer bspw. mittels einer Anzeigentafel über seine Rampenzuweisung informiert.
- Den Rangierfahrzeugen werden Fahraufträge zugewiesen, um die Ladungsträger zur Be-/Entladung an die Rampen zu setzen.
- Kontrolle der Parameter der Verkehrsträger. Warnmeldungen können bspw. vom DYMS erfolgen, wenn:
  - einem Ladungsträger vom Speditionssystem eine Ladung zugewiesen wird, die dessen Nutzlast übersteigt.
  - Oder ein Anliefertermin gefährdet ist, da der LKW sich schon zu lange in Warteposition befindet, und durch das Überschreiten der Lenk- bzw. Schichtzeit des Fahrers eine termingerechte Zustellung nicht mehr möglich sein sollte.
- Mittels Kombination von Informationen aus dem Lagerverwaltungssystem, bspw. über die Anzahl der noch offenen Kommissionieraufträge, und seinen eigenen Informationen, kann das DYMS Prognosen erstellen, ob es in bestimmter Zeit zu Engpässen im Ein-/Auslagerprozess kommen kann.

Zu beachten ist außerdem noch, ob das DYMS Werkverkehr bzw. eigene Fahrzeuge, oder hauptsächlich Fremdfahrzeuge verwalten muss. Beim Einsatz von eigenen Fahrzeugen oder Werkverkehr sind die Möglichkeiten des Informationsflusses zwischen dem Fahrzeug und

dem DYMS wesentlich höher als bei Fremdfahrzeugen, da bspw. bei den eigenen Fahrzeugen auf das Telematiksystem zugegriffen werden kann und somit ein exakteres Planen möglich ist.

Cross Docking

Um die Funktionsweise eines DYMS etwas näher zu erläutern, soll der Einsatz eines DYMS in einem typisch logistischen Einsatzszenario, dem Cross Docking, beschrieben werden.

Beim Cross Docking liefern LKW vorkommissionierte Paletten von verschiedenen Lieferanten an ein Umschlagslager an. Im Umschlagslager werden alle Paletten nach Empfängern sortiert, dann schnellstmöglich und unverändert auf LKW im Warenausgang verladen. Somit werden die Empfänger nicht direkt von jedem Lieferanten angefahren, sondern pro Empfänger erfolgt nur eine Anlieferung (Sch09b; Pfo10). Die Abbildung 2.3 veranschaulicht das Cross Docking Konzept.

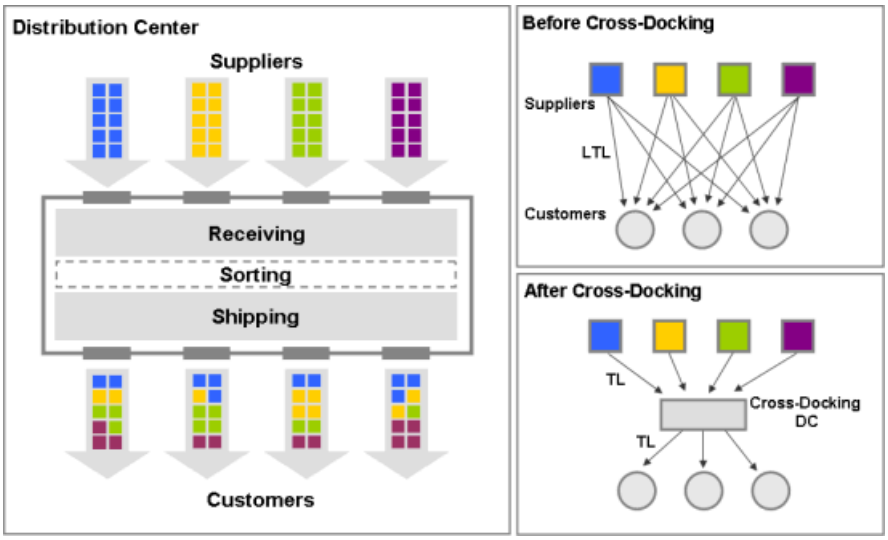

Abbildung 2.3: Cross Docking Konzept Quelle: (Wik10)

Beim Cross Docking werden hauptsächlich mit den Informationen der anliefernden bzw. abholenden LKW und den Rampen gearbeitet. Das Lager dient hier lediglich als Bindeglied zwischen den Rampen. Da alle diese Informationen im DYMS verarbeitet werden, ist das DYMS für die Aufgabe der Koordination beim Cross Docking prädestiniert. Es weist den anliefernden und den abholdenden LKW die optimalsten Rampen zu, um die Umladewege zwischen den entsprechenden LKW zu minimieren. Weiterhin kann es die Belegung und die Reservierung der Rampen steuern, um somit ein effizientes Cross Docking zu erreichen.

2.2.2 Komponenten

Eine wichtige Hauptkomponente ist der Leitstand. Der Leitstand stellt meistens grafisch den Grundriss des Betriebsgeländes mit der Halle und die Fahrzeuge als Symbole dar. Diese grafische Darstellung bietet dem Benutzer eine schnellere und intuitivere Übersicht als eine rein tabellarische Darstellungsform. Je genauer die grafische Darstellung der örtlichen Gegebenheiten, bspw. der Grundriss der Lagerhalle und die Lage der Parkplätze um die Halle, umso schneller und effektiver kann der menschliche Benutzer mit dem System arbeiten, da es dem Menschen einfacher fällt, die Daten zu den Örtlichkeiten zuzuordnen. Daher ist eine Standardisierung von DYMS nicht ganz trivial, denn jede Speditionsanlage ist mit dazugehörigen Betriebshof sehr individuell.

Die beiden wichtigsten Interaktionskomponenten werden nun kurz erläutert.

Benutzer-Schnittstelle

DYMS haben verschiedene Benutzungsoberflächen:

- Gate

  Am Gate sollte es eine Benutzungsoberfläche geben, die zwei Aufgaben erfüllt: Anmelden und Abmelden der Fahrzeuge.

- Leitstand

  Der Leitstand ist die Kernkomponente des Systems. Es ist die zentrale Planungs- und Steuereinheit. In dieser Benutzungsoberfläche fliessen alle Informationen über den Hof und die Rampen zusammen. Vom Leitstand aus steuert der Lagermeister mit Hilfe von Aufträgen die Be- und Entladevorgänge.

- Kommunikation mit den Auftragsausführenden

  Die Kommunikation mit den Auftragsausführenden kann mittels Staplerterminals, Fahrzeugterminals in den Rangierfahrzeugen, Anzeigetafeln, Pagern oder Mobilfunk erfolgen.

Fremdsystem-Schnittstelle

Das DYMS sollte Schnittstellen zu anderen Systemen einbinden. Dies könnten bspw. Schnittstellen sein zu einem LVS oder einem Speditionssystem, um Informationen von diesen abzurufen oder für diese bereitzustellen. Im Speditionssystem könnten die Informationen aus dem DYMS ins Tracking und Tracing-Modul fliessen, um dieses System aussagekräftiger und detaillierter zu machen. Eine Schnittstelle zu einem von der Spedition genutzen Telematiksystem kann bei der Planung im DYMS helfen, wenn z.B. die Ankunft eines LKW per Geofencing im Leitstand zeitlich vorausgesagt wird. Ideal wäre, wenn das DYMS die gleiche

Datenbasis benutzt wie die Fremdsysteme. Weiterhin kann es Schnittstellen einbinden, um bspw. mit den Kundensystemen zu kommunizieren, um hier den Kunden Informationen über ihre Lieferaufträge zur Verfügung zu stellen.

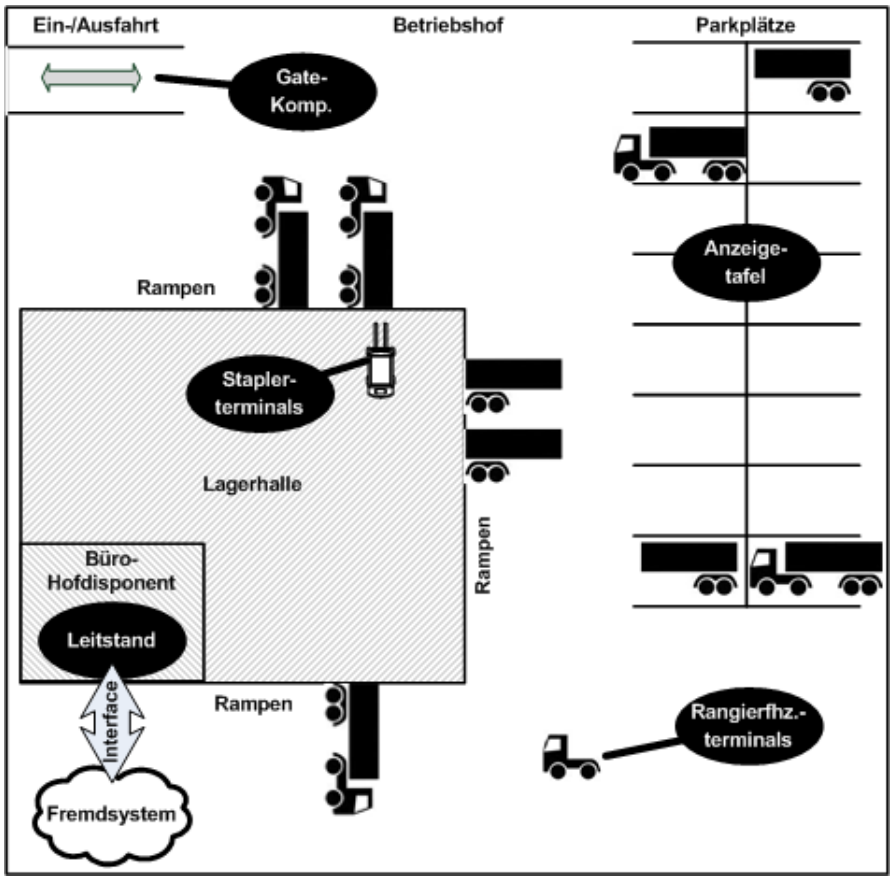

Abbildung 2.4: Schnittstellen eines DYMS

2.2.3 Wirtschaftlichkeit

Die Aberdeen Group hat im Jahr 2007 eine Untersuchung auf Wirtschaftlichkeit und Nutzen von Yard Management Systemen (YMS) auf dem amerikanischen Markt vorgenommen (Hob07). Da im deutschsprachigen Raum die DYMS nicht so stark verbreitet sind wie im amerikanischsprachigen Raum, können die Werte bzgl. des Einsatzes von YMS für den deutschen Markt als sehr optimistisch betrachtet werden. Die Untersuchung ergab folgendes:

- 58% verwalteten die Fahrzeuge manuell
- 24% nutzten ein YMS ohne Integration ins LVS

- 12% hatten ein vollintegriertes DYMS im Einsatz

Weiterhin wurde die Aufenthaltszeit der Fahrzeuge auf dem Betriebsgelände zwischen Ein- und Ausfahrt gemessen. Das Ergebnis war, dass sich bei den Unternehmen, die ein vollintegriertes DYMS einsetzten, sich die Fahrzeuge nicht länger als 3 Stunden aufhielten. Die Unternehmen mit manueller Verwaltung hatten Zeiten von mehr als 6 Stunden (siehe Abbildung 2.5).

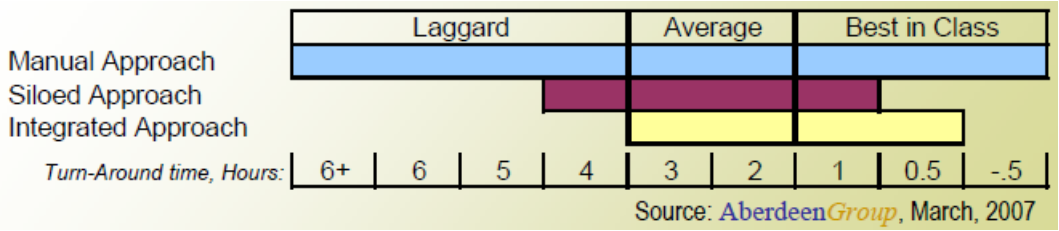

Abbildung 2.5: Benötigte Zeit auf dem Betriebshof in Abhängigkeit der Nutzung von YMS. Quelle: (Hob07)

Die Untersuchung der Aberdeen Group hat gezeigt, dass DYMS helfen, effizient zu arbeiten.

Das Potenzial von DYMS ist eine lange Liste. So können bspw. unnötige Kosten, wie Standgelder vermieden werden (Standgelder sind gesetzlich geregelt nach § 412 Abs.2,3 HGB). Oder es können Leerlaufzeiten im Lager durch eine effiziente Rampenbelegung verkürzt oder im Idealfall auf Null reduziert werden. Weiterhin können durch vorauseilende Informationen Kommissioniervorgänge im Lager optimiert werden, so dass Ware für ein Fahrzeug Just-In-Time bereitgestellt werden kann. Dadurch würde die Kommissionierfläche verkleinert werden, was den Effekt hat, dass diese Fläche für Lagerware zur Verfügung steht und somit zur Erhöhung des Gewinns beiträgt.

2.3 Genutzte Technologien für diese Unternehmensanwendung

An dieser Stelle nun werden die beiden wichtigsten Techniken, die in dem Prototypen verwendet werden sollen, näher erläutert. Dies ist zum einen Java EE und zum anderen Ajax unter Verwendung des GWT.

2.3.1 Java EE

Java EE ist die Abkürzung von Java Enterprise Edition und definiert einen Standard. Es ist kein fertiges Stück Software, sondern eine Ansammlung von Spezifikationen von Diens-

ten, die von einem Server angeboten werden (IHHK07; Roz07; Wik11b). Java EE ist eine Erweiterung von Java SE (Java Standard Edition)(IHHK07; Roz07). Java SE ist für die Entwicklung von Java-basierenden Desktop Applikationen gedacht, die oftmals nur auf einem PC lokal ausgeführt werden und nur für einen oder nur wenige Benutzer ausgelegt sind. Mit Hilfe von Java EE lassen sich grosse Unternehmensanwendungen entwickeln und betreiben. Der Grundgedanke bei der Entwicklung mit dem Java EE-Framework liegt dabei auf zwei grundlegenden Konzepten(Sta09):

- Client-Server Modell

  Unternehmensanwendungen sollten nach dem Client/Server-Modell aufgebaut werden, bei dem die Server Dienste anbieten, die von den Clients über ein Netzwerk genutzt werden (Han96; Som07; Bal00). Durch die Verteilung der Anwendung auf Clients und Server spricht man auch von einer verteilten Anwendung. Java EE unterstützt das C/S-Modell.

- Mehrschichtenanwendung und Java EE-Technologien

  Die Client/Server-Architektur wird in logischen Schichten betrachtet, die untereinander mittels Schnittstellen kommunizieren. Dies macht es möglich, die Schichten einzeln weiter zu entwickeln ohne Nebeneffekte zu erzeugen, vorausgesetzt, die Schnittstellen bleiben erhalten.

  Die gängigste Schichtenarchitektur ist die 3-Schichten-Architektur (Fis06; Som07; Bal00; Han96), die sich aus folgenden 3 Schichten zusammensetzt:

  Darstellungsschicht Ist das Frontend für die Interaktion mit dem Benutzer und wird von Java EE umgesetzt mit (Sta09):

  – Den JavaServer Pages (JSP), verbindet statische HTML-Seiten mit Java-Elementen zu dynamischen Dokumenten, deren Inhalt und Aussehen erst zur Laufzeit bestimmt wird.
  – Den Servlets, diese laufen auf dem Server in einem Webserver und beantworten Anfragen von Clients mittels http.
  – Den JavaServer Faces (JSF), erleichtert das Entwickeln von Benutzerschnittstellen in Webseiten.

  Anwendungsschicht Diese enthält die Geschäftslogik und wird von Java EE umgesetzt mit (Sta09):

  – Den Enterprise JavaBeans (EJB), dient der Entwicklung der Geschäftslogik. Wird im Folgenden näher erläutert.

  - Dem Java Message Service (JMS), ermöglicht asynchronen Nachrichtenaustausch.
  - Dem Java Naming and Directory Interface (JNDI), Namens- und Verzeichnisdienst über den sich die Dienste finden lassen.

Datenhaltungsschicht Hier findet die Persistierung der Daten statt, bspw. in einer Datenbank, und wird von Java EE umgesetzt mit (Sta09):

  - Der Java Persistenz API, handelt den Austausch von Daten zwischen einer relationalen Datenbank und Objekten.

Daneben gibt es noch bspw. die 2-Schichten-Architektur, bei der die Anwendungsschicht entweder mit im Client liegt (Fat-Client) oder zusammen mit der Datenhaltung auf dem Server (Thin-Client). Weiterhin können noch weitere Schichten eingefügt werden. Bspw. kann zwischen dem Server für die Anwendungsschicht und der Datenbank ein weiterer Server eingesetzt werden, wenn z.B. auf mehrere Datenbanken zugegriffen werden muss. Diese Architektur wird dann ”Mehrschichten-Architektur” genannt (Rei09; Som07).

Seitens des Unternehmensmanagement besteht bei vielen Unternehmen die Forderung an die Unternehmensanwendung, dass der Grad der Wiederverwendung von Komponenten erhöht wird. Damit kann eine schnellere Entwicklung erreicht werden, da neue Anwendungen bereits existierende Komponenten bzw. Services nutzen können. Dies spart Zeit und somit Geld (Fis06; Rei09).

Die 3-Schichten-Architektur dient unter anderem der Forderung nach Wiederverwendung. Denn die Komponenten der Anwendungsschicht, die in der sog. Middleware-Plattform physisch entwickelt werden, können für unterschiedliche Realisierungen der Darstellungsschicht zur Verfügung stehen und somit wiederverwendet werden.

Die technische Umsetzung der Java EE-Spezifikationen findet in einem Anwendungsserver statt. Dieser enthält die Implementationen der spezifizierten Dienste und Technologien. Der Anwendungsserver übernimmt auch die Erfüllung der nicht-funktionalen Anforderungen von Unternehmensanwendungen, wie z.B. Verfügbarkeit, Transaktionalität, hohe Anzahl konkurrierender Zugriffe und Skalierbarkeit (IHHK07). Beispiele für diese Anwendungsserver sind:

- JBoss (Com11)
- Glassfish (Gla11)
- IBM WebSphere (Gmb11a)

Als nächstes wird EJB, eine der Hauptkomponenten von Java EE, erläutert.

2.3.1.1 EJB

Enterprise JavaBeans sind normale JavaBeans oder Java-Klassen, die durch Annotationen zu EJB-Komponenten werden. Annotationen sind Erweiterungen der Javasprache, die direkt im Java-Quellcode eingefügt werden. Annotationen markieren eine Java-Klasse, ein Attribut oder eine Methode, dadurch steuern sie deren Laufzeitverhalten im EJB-Container. Kenntlich gemacht werden die Annotationen durch ein vorangestelltes @-Zeichen. Als Beispiel siehe Listing 2.1 und 2.2.

EJB-Container Die EJB-Komponenten benötigen eine Laufzeitumgebung. Diese Laufzeitumgebung bietet der EJB-Container. Der EJB-Container ist ein Element des bereits erwähnten Anwendungsservers. Die wichtigsten Aufgaben eines EJB-Containers sind (IHHK07):

- Überwachung des Lebenszyklus von EJB:
  Der EJB-Container übernimmt die vollständige Kontrolle der EJB-Komponenten von der Erzeugung bis zum Ende. Der EJB-Container erzeugt eigenständig Instanzen von EJB-Komponenten. Dieses Verhalten kann durch Parameter gesteuert werden.

- Instanzen-Pools:
  EJB-Container erzeugen eine bestimmte Menge an EJB-Komponenten auf Vorrat, einen sogenannten Pool. Benötigt nun ein Client eine Instanz, kann ihm diese sofort zugeteilt werden, ohne erst das Erstellen der Instanz abwarten zu müssen.

- Transaktionshandling:
  Transaktionen sind für Unternehmensanwendungen ein wichtiger Bestandteil (Han96; IHHK07). Es kann entweder die Transaktionssteuerung automatisch durch den EJB-Container für die EJB-Komponenten erfolgen, oder es kann im Quellcode mit Hilfe der Java Transaction API (JTA) durch den Programmierer festgelegt werden. Bei der Transaktionssteuerung durch den EJB-Container nutzt dieser das Transaktionsmanagement, welches ihm vom Anwendungsserver zur Verfügung gestellt wird.

- Konkurrierende Zugriffe:
  Beim konkurrierenden Zugriff auf eine EJB-Komponente durch mehrere Clients sorgt der EJB-Container dafür, dass die Daten der EJB-Komponente konsistent bleiben.

- Zugriffsrechte auf Methoden der EJB-Komponenten:
  Für EJB-Komponenten können Zugriffsrechte für Benutzer durch Annotationen festgelegt werden. Der EJB-Container wertet die Zugriffrechte zur Laufzeit mit Hilfe des Java Authentication and Authorization Services (JAAS) aus.

EJB-Typen Hier werden nun die EJB-Komponenten erläutert. Die EJB-Spezifikation, die ab Java EE 5 in der Version 3.0 vorliegt, teilt die EJB-Komponenten in drei Typen:

- Session Beans
- Message Driven Beans
- Entity Beans

Diese werden nun etwas näher erläutert.

Session Beans Session Beans sind das eigentliche Herzstück, denn in ihnen wird die Geschäftslogik implementiert. Sie stellen diese Logik zur Nutzung in Form von Services bereit. Um an Daten aus einer Datenbank zu gelangen, nutzen sie die Entity Beans oder neu ab Version 3.0 die Java Persistenz API. Die Session Beans gibt es in zwei Varianten:

- Stateless Session Beans:
  Diese Beans sind zustandslos. Sie arbeiten mit dem Client immer genau für einen Methodenaufruf zusammen. Danach vergessen sie den Client wieder. Beim gleichen Methodenaufruf hintereinander kann es somit sein, dass der Client für jeden Methodenaufruf eine andere Instanz zugewiesen bekommt. Beispiel Listing 2.1
- Stateful Session Beans:
  Diese Beans sind zustandsbehaftet. Die Bean bleibt solange mit dem Client verbunden, bis dieser die Bean wieder freigibt, auch über mehrere Methodenaufrufe hinweg. Solange die Bean an den Client gebunden ist, kann sie von keinem anderen Client benutzt werden. Beispiel Listing 2.2

Algorithm 2.1 Beispiel einer zustandslosen Session Bean. Die Annotation @Stateless sorgt dafür.

```
1 package my.ejb;
2 import javax.ejb.Stateless;
3 @Stateless
4 public class AddiereBean implements AddiereRemote{
5         public int addiere(int a, int b){
6                 return (a + b);
7         }
8 }
```

Algorithm 2.2 Beispiel einer zustandsbehafteten Session Bean. Markiert durch die Annotation @Stateful.

```
1  package my.ejb;
2  import javax.ejb.Stateful;
3  @Stateful
4  public class HelloBean implements HelloRemote{
5          private String name = "";
6          public void setName(String n){
7                  this.name = n;
8          }
9          public String hello(){
10                 return this.name;
11         }
12 }
```

Message Driven Beans Message Driven Beans (MDB) dienen der asynchronen Kommunikation. Sie können ausschliesslich über Nachrichten angesprochen werden. Den Austausch der Nachrichten übernimmt dabei der Java Message Service (JMS). Die MDB hat, bedingt durch ihre Unsichtbarkeit für den Client, keinen clientbezogenen Zustand. MDB machen dann Sinn, wenn die Berechnung eines Geschäftsvorfalles viel Zeit in Anspruch nimmt, da durch die asynchrone Kommunikation der Client während der Berechnung nicht blockiert wird.

Entity Beans Die Entity Beans stellen die Schnittstelle zur Datenbank dar. Dabei handelt es sich um normale Java-Klassen mit getter- und setter-Methoden. Mittels Annotationen werden die Klassen auf Datenbanktabellen und die Methoden auf die Tabellenattribute abgebildet. Die Aufgabe der Synchronisation der Entity Bean mit der Datenbank übernimmt der EntityManager, der vom Anwendungsserver gestellt wird.

Seit der EJB-Spezifikation 3.0 übernimmt die neue Java Persistenz API diese Aufgabe, so dass die Entity Beans immer mehr an Bedeutung verlieren.

EJB-entfernter Aufruf Entfernte Clients greifen bzw. nutzen die durch EJB-Komponenten angebotenen Services bspw. durch entfernte Methodenaufrufe (Remote Method Invocation, Abk.: RMI).

Mittels des JNDI-Namensdienstes fragt der Client beim Anwendungsserver nach einem bestimmten Service, also einer EJB-Komponente. Als Antwort erhält der Client eine Instanz einer Klasse, die das Interface der EJB-Komponente implementiert, das sog. „Stub". An diese richtet der Client seinen Methodenaufruf. Für den Client scheint die Methodenausführung lokal zu sein. Die Stub-Klasse leitet im Hintergrund den Methodenaufruf an ihr Gegenstück,

die Skeleton-Klasse, welche sich auf dem Server befindet, weiter. Die Skeleton-Klasse leitet dann die Methodenausführung an die EJB weiter. Die Antwort geht über die Skeleton-Klasse zur Stub-Klasse an den Client zurück.

2.3.1.2 Alternative zum EJB-Container

Mit den Grundkonzepten, den spezifizierten Technologien und den Anwendungsservern unterstützt Java EE bei der Entwicklung von verteilten Unternehmensanwendungen. Die Anwendungsserver bestehen aus einer Vielzahl von Komponenten, wie bspw. dem EJB-Container, Servlet-Container oder Webservern (IHHK07). Dies macht sie sehr komplex und umfangreich, was den Vorteil bietet, dass bei grossen Unternehmensanwendungen der Entwicklungsaufwand abnimmt. Allerdings ist diese Komplexität auch ein Nachteil, denn der Umgang mit dem Anwendungsserver muss erst erlernt werden, und bei Bedarf wird immer ein grosser Anwendungsserver mit allen Containern installiert, obwohl bspw. nur ein Container für die Geschäftslogik benötigt wird.

Hier hilft das Spring-Framework. Es handelt sich dabei um ein Open-Source-Framework für die Java-Plattform. Das Ziel des Spring-Frameworks ist es, die Entwicklung der Anwendungslogik durch die Verwendung von POJOs (Plain Old Java Objects) zu vereinfachen (Fis06; IHHK07; Wik11h; Sch09a). POJOs sind einfache Java-Klassen. Die Verwendung der POJOs ermöglicht ein Konzept namens *Dependency Injection*. Bei Dependency Injection werden den Java-Klassen, also den POJOs, die Abhängigkeiten zu anderen Klassen und Ressourcen zugewiesen, d.h., sie müssen sie nicht selbst holen. Es gibt drei Arten der Dependency Injection, also die Arten, wie Informationen injiziert werden können (Fow04), allerdings werden in der Literatur im Zusammenhang mit dem Spring-Framework nur die ersten beiden Arten (Constructor, Setter) genannt (JHA+08; Fis06; Sch09a):

- Constructor Injection:
  Beim Erstellen werden die Abhängigkeiten oder Ressourcen über den Konstruktor der Klasse an die Bean übergeben.

- Setter Injection:
  Die Bean definiert setter-Methoden, über die Abhängigkeiten und Ressourcen übergeben werden.

- Interface Injection:
  Hierbei werden Interfaces genutzt, die die Injektionsmethoden definieren. Die Klasse, die Abhängigkeiten und Ressourcen vom Framework übergeben bekommen möchte, hat dieses Interface zu implementieren.

Die Verantwortlichkeit dieser Zuweisungen liegen beim Framework. Diese Eigenschaft des Frameworks nennt man *Inversion of Control*.

2.3.1.3 Alternativen zu Java-EE

Neben Java EE gibt es noch weitere Standards und Spezifikationen bzw. Produkte, um die Entwicklung von Unternehmensanwendungen zu erleichtern. In der Literatur wird in diesem Zusammenhang die .NET-Plattform und CORBA genannt (Fis06; Bal00; Som07).

CORBA

CORBA steht für *Common Object Request Broker Architecture* und ist eine Spezifikation, dessen Umsetzung es ermöglicht zwischen verteilten, heterogenen Systemen zu kommunizieren. Diese Kommunikation ermöglicht der Object Request Broker (ORB) (Bal00; Som07; Röm06). CORBA wurde Anfang der 1991-Jahre definiert und war die erste Architektur, die diese Art der Kommunikation ermöglichte (Gmb02). Mittlerweile gibt es modernere Techniken, daher soll hier nicht weiter auf CORBA eingegangen werden.

.NET-Plattform

.NET ist eine Software-Plattform von Microsoft für die Entwicklung von verteilten Systemen. .NET ist ein Produkt und keine Spezifikation. Die .NET Platform ist sehr umfangreich, daher sind die hier erläuterten Komponenten nicht vollständig.
Herzstück der .NET Platform ist das .NET Framework für die Entwicklung, welches aus den zwei wesentlichen Bestandteilen besteht:

- aus einer Laufzeitumgebung:
  Der Quellcode der von .NET unterstützten Sprachen, wie z.B. C#, VB.NET, Eiffel#, wird nicht direkt in Maschinencode übersetzt, sondern in die sog. *Common Intermediate Language* (CIL). Das .NET-Framework beinhaltet die Laufzeitumgebung *Common Language Runtime* (CLR), die den Zwischencode, die CIL, interpretiert und in den Maschinencode für den jeweiligen Prozessor übersetzt. Dies ähnelt der Java-Technik, die auch einen Interpreter einsetzt. Die CLR übernimmt Aufgaben, wie z.B. Überwachung der Speichernutzung oder Zugriffe auf Dienste.

- und einer umfangreichen Klassenbibliothek:
  In der Version 3.5 sind ca. 11.400 Klassen enthalten. Als Beispiele für eine Klassensammlung zur Erledigung einer bestimmten Aufgabe seien hier genannt ASP.NET, für die Entwicklung von Web-Anwendungen, und ADO.NET, welche den Zugriff auf relationale Datenbanken ermöglicht.

Die Geschäftslogik wird in sog. .NET managed components abgebildet, die von der Laufzeitumgebung CLR überwacht werden.
Als weitere Komponenten gibt es noch die .NET Services unter dem Namen „Hailstorm" bekannt. Ein Beispiel für einen .NET Service ist „Passport", ein Service, mit dem sich Benutzer

mit denselben Logindaten an verschiedenen Diensten anmelden können. (Fis06; Wik11g; VR01)

Es gibt in der Literatur zwar Vergleiche zwischen bspw. Java EE und CORBA (Bal00, S.936,937) oder Java EE und .NET (VR01), aber die drei können auch ergänzend im Unternehmen eingesetzt werden (Fis06).

2.3.2 Ajax

Die Benutzungsoberflächen des Prototypen werden die Ajax-Technologie anwenden. Ajax steht für „Asynchronous JavaScript and XML" und wird im Folgenden kurz erläutert. Das Hauptaugenmerk bezüglich der Benutzungsoberflächen liegt allerdings in dieser Arbeit auf dem GWT, mit dessen Hilfe die Ajax-Applikation entwickelt werden wird. Das GWT wird im Anschluss daher detaillierter erläutert.

Die Kommunikation zwischen einem Browser und einem Webserver über HTTP ist zustandslos. Dies bedeutet, dass jede Anfrage des Browsers beim Webserver zur Folge hat, dass dieser immer eine komplette Webseite zum Browser zurückschicken muss. Durch Ajax ist es nun möglich, dass nur noch die Teile einer Webseite aktualisiert werden müssen, bei denen Änderungen aufgetreten sind, d.h., der Webserver schickt auch nur noch die Informationen, die sich geändert haben, zum Browser. Um dies zu ermöglichen Bedarf es eines Kommunikationsobjektes (XMLHttpRequest), welches die Aufgabe hat mit dem Webserver zu kommunizieren. Wird nun im Browser ein Ereignis ausgelöst, so erzeugt dieses ein Kommunikationsobjekt mit den Anforderungen. Das Kommunikationsobjekt kommuniziert nun asynchron mit dem Webserver und erhält die angeforderten Informationen, welche dann zur Aktualisierung der entsprechenden Bereiche auf der Webseite benutzt werden (Ste07).

2.3.2.1 GWT

Das GWT steht für „Google Web Toolkit", und ist ein Entwicklungs-Framework bzw. eine Ansammlung von Tools, um Ajax-Applikationen komfortabel zu entwickeln (Ste07). Es wurde am 17. Mai 2006 von Google veröffentlicht. Im Dezember 2006 wurde es dann unter die Apache-Lizenz 2.0 gestellt und somit vollständig zur Open-Source-Software (Ste07, S.22). Das Herzstück des GWT ist ein Java-to-JavaScript-Compiler. Dieser ermöglicht es, die Anwendung komplett in Java zu entwickeln. Der GWT-Compiler übersetzt dann den Java-Quellcode in JavaScript-, HTML- und CSS-Dateien, also neben XML die Grundtechnologien einer Ajax-Applikation.

2.3.2.2 GWT-Komponenten

Im Folgenden werden nun die Komponenten, aus denen das GWT sich zusammensetzt, erläutert. Neben dem GWT-Compiler gibt es noch weitere Komponenten, die z. T. direkt mit dem Compiler verknüpft sind oder die als Java-Bibliotheken die GWT-API bilden (HT07):

Kernstück

- Java-to-JavaScript-Compiler
  übersetzt den Java-Code in JavaScript-Code

direkt mit Compiler verknüpft

- JSNI
  JavaScript Native Interface: Gibt die Möglichkeit natives JavaScript im Java-Code einzubetten.
- JRE-Emulation
  Ermöglicht die nicht vollständige Verwendung der JRE-Klassen java.lang.* und java.util.*.

GWT-API

- GWT- Web-UI-Klassenbibliothek
  Diese stellt verschiedene Widgets und Panels zur Verfügung, um GUI's zu erstellen.
- Internationalisierungs- und Konfigurationstools
  In einer Eigenschaftendatei können Konfigurationsparamter bzw. Übersetzungen abgelegt werden, die dann aus der GWT-Anwendung heraus aufgerufen und benutzt werden können.
- RPC
  „Remote Procedure Call" stellt im Zusammenhang dieser Arbeit ein wichtiges Bauteil dar. Mittels des RPC kann eine asynchrone Kommunikation unter Verwendung von serialisierbaren Java-Objekten mit einem auf einem Server angebotenen GWT-Servlet erfolgen. Das GWT-Servlet wiederum dient als Client für den Aufruf und die Verwendung von EJB (siehe Abschnitt 2.3.1.1). Dabei stellt der Client eine Anforderung an ein Proxy-Objekt. Dieses serialisiert das Objekt des Client und schickt es zum Server. Das GWT-Servlet auf dem Server empfängt, deserialisiert das Objekt und schickt die Anforderung an einen EJB-Dienst. Die Antwort wird dann wieder serialisiert vom GWT-Servlet an das Proxy-Objekt auf dem Client zurückgeschickt. Vom Proxy-Objekt wird die Antwort dann wieder in ein Java-Objekt deserialisiert und kann im Client verwendet werden.

- XML-Parser
  Kommunikation mit Hilfe des XML-Formates wird immer beliebter, daher bietet das GWT einen XML-Parser, der auf dem DOM (Document Object Model) basiert, an.

- Verlaufsverwaltung
  Mittels der Schnittstelle *HistoryListener*, die vom GWT zur Verfügung gestellt wird, hat man die Möglichkeit auf den „Zurück-Button" des Browsers zu reagieren.

- JUnit-Integration
  Das bekannte Testframework JUnit für automatisierte Softwaretests kann auch für GWT verwendet werden.

2.3.2.3 GUI-Entwicklung

Das GWT kommt in der Darstellungsschicht, wenn man vom 3-Schichten-Modell ausgeht, zum Einsatz. Es hat somit die Hauptaufgabe, die Schnittstelle zwischen der Anwendung und dem Benutzer zu bilden. Dies erfolgt über GUI's (Graphical User Interface), somit bildet die GUI-Entwicklung beim GWT einen Schwerpunkt. Zum Erstellen von GUI's gibt es bei GWT verschiedene Möglichkeiten:

- interne GWT-UI-Klassenbibliothek
  wie bereits unter den GWT-Komponenten erwähnt, stellt das GWT eine eigene Bibliothek an Widgets und Panels zur Verfügung. Die Anzahl ist allerdings sehr klein. Ein Showcase mit denen zur Verfügung stehenden Widgets und Panels ist hier zu finden (Goo11b).

- externe Frameworks
  Weiterhin kann das GWT um externe Frameworks erweitert werden, die eine wesentlich größere Menge an Widgets und Panels zur Verfügung stellen. Die beiden Meistgenannten sind:

  - smartGWT, und
  - Ext GWT oder auch GXT

Die beiden genannten Frameworks werden nun etwas näher vorgestellt.

smartGWT: Das smartGWT ist eine API, die das JavaScript-Framework smartClient mit dem GWT verbindet(Vog09). Das SmartClient-Framework bietet eine grosse Anzahl von Widgets und Panels. SmartClient und smartGWT sind ein Produkt der Firma Isomorphic Software Inc. . Folgende Funktionen bietet das smartGWT unter anderem an:

- Eine große Menge an Widgets, Panels und Funktionen, wie z.B. Drag&Drop-Funktionalität, Tabellen mit Gruppenfuss zum Summieren von Zwischenwerten, Kalender zum Eintragen von Terminen, Slider, Progessbar und weitere. Für einen umfangreichen Überblick über die Widgets und Panels vom smartGWT sei hier auf das Showcase von smartGWT hingeweisen (Sof11a).
- Data Binding, welches die Datenbindung an die Widgets nach dem Model-View-Controller-Prinzip ermöglicht.
- Das Aussehen der Widgets und Panels kann mittels CSS-Dateien angepasst werden.
- Server Features, dies sind eine Sammlung von jar-Dateien, servlets, jsp, tags und tools, die in einem J2EE-Applikationserver integriert werden können, und diesen um weitere API's und Services erweitern. Bspw. um einen Excel Export (Sof11b).

Das smartGWT wird in 4 verschiedenen Editionen angeboten: Eine kostenfreie LGPL Edition, und die drei kostenpflichtigen Editionen: Professional, Power und Enterprise. Die kostenfreie Edition, welche für den Prototypen verwendet werden wird, besitzt keine Server Features.

ExtGWT: ExtGWT ist ein Internet Application Framework für das GWT. Es ist ein Produkt der Firma Sencha Inc. . An Funktionen sollen hier unteranderem genannt werden:

- Eine große Menge an Widgets und Panels. Für einen Überblick soll hier auf die Samples&Demos auf der Homepage der Firma Sencha Inc. verwiesen werden (Inc11c). Darüber hinaus behauptet die Firma Sencha Inc. allerdings, dass ihre Komponenten besonders performant sein sollen (Inc11b).
- Wie das smartGWT, so bietet auch das ExtGWT die Funktionalität des Data Bindings und die Möglichkeit, das Aussehen der Komponenten mittels CSS-Dateien anzupassen.

Das ExtGWT-Framework gibt es sowohl als kostenpflichtige als auch als Open Source-Edition, die unter der GNU GPL license v3 steht. Dies bedeutet, dass die Anwendung, in der ExtGWT als Open Source verwendet wird, selber als Open Source weitergeben werden muss (Inc11a).

Da beide Frameworks das GWT erweitern, bleibt die Möglichkeit, das GWT-RPC zu nutzen bestehen.

2.3.2.4 Vor-und Nachteile von GWT

Vorteile:

- Durch die Entwicklung komplett in Java, kann auch von Java-spezifischen Eigenschaften Gebrauch gemacht werden, wie z.B. feste Datentypen, Exception-Handling, Objektorientierung usw.
- Erleichterter Umgang mit Remote Procedure Calls (RPC).
- Das Ajax-Problem: Durch die asynchrone Kommunikation mit dem Webserver verliert der „Zurück"-Button seine Wirkung. Dieses Problem wird durch GWT gelöst (Ste07, S.26).
- Bei JavaScript kommt es bei unterschiedlichen Browsern oft zu Problemen. Das GWT sorgt für eine Unterstützung der gängisten Browser: Internet Explorer, Firefox, Safari, Opera. Das GWT löst dieses Problem durch Permutationen, es erstellt für jeden Browser spezielle JavaScript-Dateien.
- Das GWT stellt ein Internationalisierungstool zur Verfügung, mit dem es einfach ist, die Applikation mehrsprachig zu entwickeln.

Nachteile:

- Integration in eine bestehende Infrastruktur, wie z.B. Struts ist schwierig, da die GWT-RPC-Technik Besonderheiten bezüglich der Servlets aufweist (Sta07).
- Das Herzstück des GWT, der GWT-Compiler, ist nicht Open-Source. Zur Zeit wird er als „Black-Box" von Google zur Verfügung gestellt. Dies könnte sich natürlich ändern, wenn Google den Compiler kostenpflichtig macht (Sta07).
- Da die GWT-Anwendungen nur aus JavaScript bestehen und keinem HTML, sind diese für Suchmaschinen nicht sichtbar (Wüt10).

Trotz der Nachteile ist das Google Web Toolkit für die Entwicklung von Webapplikationen gerade für Entwickler mit geringer Erfahrung in diesem Bereich besonders hilfreich. Durch die bekannte Entwicklung in der Sprache Java, braucht der Entwickler sich nicht erst in JavaScript einzuarbeiten. Dies spart Zeit und Geld. Durch das umfassende Angebot an Widgets und Panels, auch durch die Nutzbarkeit z.B. des smartClient-Frameworks, können schnell Ergebnisse bei der GUI-Entwicklung erzielt werden. Allerdings bleibt der Gedanke im Hinterkopf, dass Google irgendwann den GWT-Compiler nicht mehr kostenlos zur Verfügung stellen könnte.

Kapitel 3

Analyse

Die Anlayse dient der Ermittlung von Anforderungen an das Dock und Yard Management System. Dazu wird zunächst das Altsystem untersucht und bewertet, um hierbei erste Anforderungen abzuleiten. Danach werden in Interviews weitere Anforderungen definiert. Aus den beiden Analysen werden dann am Schluss des Kapitels die Anforderungen an das neue Dock und Yard Management System spezifiziert.

3.1 Das Altsystem

Die Cargo Service Nord GmbH hat vom vorherigen Lagerhalter ein System zur Verwaltung der LKW und der Rampen übernommen. Für den Entwickler des Systems und den vorherigen Lagerhalter war es das erste System dieser Art. Da noch keine Erfahrungen bestanden wurden nur Mindestanforderungen umgesetzt.

3.1.1 Funktionsumfang

Um einen Eindruck des Funktionsumfanges des Systems zu erhalten wurde ein Interview mit dem Entwickler geführt. Weiterhin stellte der Entwickler das System in einer Testversion zur Verfügung. Aus dem Gespräch konnte ein Anwendungsfalldiagramm erstellt werden, welches in der Abbildung 3.1 zu sehen ist.

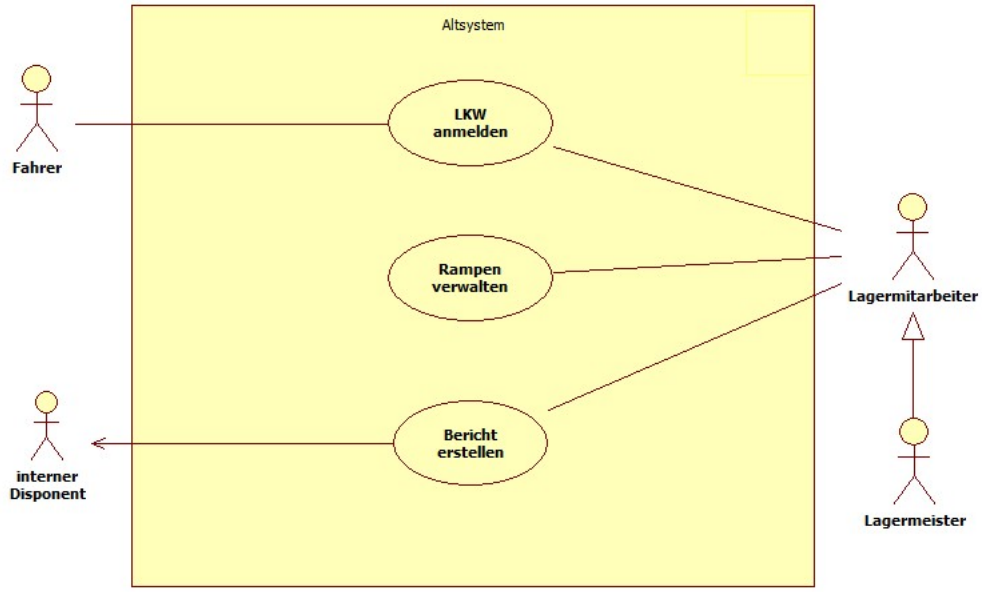

Abbildung 3.1: Anwendungsfalldiagramm des Altsystems

Als nächstes werden nun die Anwendungsfälle näher beschrieben. Insgesamt gibt es vier Akteure: Fahrer, interner Disponent, Lagermitarbeiter und Lagermeister. Der Lagermeister ist der Vorgesetzte der Lagermitarbeiter. Der Lagermeister ist eine Spezialisierung des Lagermitarbeiters. In dem System kann allerdings nicht zwischen dem Lagermitarbeiter und dem Lagermeister unterschieden werden, so dass beide die gleichen Anwendungsfälle bedienen dürfen. Daher wird auch im Folgenden nicht zwischen Lagermitarbeiter und Lagermeister unterschieden, es wird nur der Lagermitarbeiter genannt.

Der Anwendungsfall „LKW anmelden" beschreibt den Vorgang, dass ein LKW-Fahrer, der auf das Betriebsgelände fährt, sich anmeldet. Die Anmeldung erfolgt mit einer Benutzungsoberfläche auf einem Anmeldeterminal, in dem der Fahrer Daten, wie Firmenname, Mobilnummer, Entladeort, Lieferscheinnummer und Auftraggeber angeben muss. In der Datenbank wird ein Zeitstempel für die Anmeldung gesetzt. Möchte der Fahrer für mehrere Auftraggeber laden, so muss er sich für jeden Auftraggeber einzeln anmelden. Nachdem der Fahrer angemeldet ist sucht er sich mit seinem LKW eine freie Stellfläche und parkt dort. Hier wartet er dann auf den Anruf des Lagermitarbeiters.

Beim Anwendungsfall „Rampen verwalten" hat der Lagermitarbeiter die Möglichkeit, LKW den Rampen zuzuweisen, Rampen freizugeben und diese zu sperren. Beim Rampen zuweisen sucht der Lagermitarbeiter einen angemeldeten LKW aus einer Tabelle aus und weist diesem eine Rampe zu, damit der LKW be-,entladen oder beides werden kann. Das Aussuchen des nächsten LKW wird dem Lagermitarbeiter erleichtert, indem die Tabelle mit den angemeldeten LKW nach dem Kriterium „Anmeldezeit" sortiert ist. Hat der Fahrer sich mehrfach

angemeldet, weil er bspw. für mehrere Auftraggeber laden soll, so kann der LKW mehreren Rampen zugewiesen werden. Mittels der Mobilnummer, die bei der Anmeldung angegeben worden ist, informiert der Lagermitarbeiter den Fahrer über die Rampennummer, an die der Fahrer anzudocken hat. Das Zuweisen der Rampe wird auch mit einem Zeitstempel in der Datenbank protokolliert. Weiterhin hat der Lagermitarbeiter die Möglichkeit eine Rampe zu sperren, so dass diese nicht mehr verwendet werden kann. Dies kann bspw. erforderlich sein, wenn eine Rampe defekt ist. Ist die Aktivität des LKW an der Rampe abgeschlossen, so gibt der Lagermitarbeiter die Rampe im System wieder frei. Bei mehrfacher Rampenzuweisung muss jede Rampe freigegeben werden. Durch diese Freigabe wird ein weiterer Zeitstempel in der Datenbank des Systems gesetzt.

Der Anwendungsfall „Bericht erstellen" beschreibt den Vorgang, in dem ein Bericht erstellt werden kann, der die drei protokollierten Zeitstempel mit den dazugehörigen LKW-Daten enthält. Der Lagermitarbeiter wählt einen Zeitraum aus, für den der Bericht erstellt werden soll. Das System liefert ihm dann eine Excelliste, die er dann dem internen Disponenten schicken kann oder für eigene Kontrollzwecke nutzt. Die gerichtete Assoziation zwischen dem Anwendungsfall „Bericht erstellen" und dem internen Disponenten zeigt, dass dieser keine Möglichkeit hat selbst Berichte zu erstellen.

3.1.2 Architektur

Für das Altsystem gibt es keine Dokumentation. Diese wurde mit Hilfe von zwei Softwaretools rekonstruiert. Das Tool „Enterprise Architect 9.0" (Gmb11b) bietet die Möglichkeit aus Quellcode ein Klassendiagramm zu erstellen. Der Entwickler stellte hierfür seinen Quellcode in der Sprache Visual Basic zur Verfügung, so dass hieraus ein Klassendiagramm erstellt werden konnte (siehe Abbildung 3.2). Das Tool „Enterprise Architect" wurde für diesen Zweck in der Trial Version eingesetzt. Mit dem Tool „DBDesigner 4.0" (fab11) wurde aus der verwendeten MS-Access mdb-Datei mittels der Funktion „Reverse Engineering" ein Entity-Relationship-Diagramm erstellt (siehe Abbildung 3.5).

Klassendiagramm

Das Klassendiagramm (siehe Abbildung 3.2) zeigt, dass das System aus drei Forms besteht. Die Forms repräsentieren in Visual Basic GUI's. Die Klasse „formUSEPA_Elmshorn" stellt die GUI für die Verwaltung der Rampen dar (siehe Abbildung 3.4), die Klasse „formAnmeldung_USEPA_Horst" zeigt die GUI für die Anmeldung (siehe Abbildung 3.3), und die dritte Form-Klasse „formReportUSEPA_Elmshorn" zeigt eine GUI, welches eine Datumsauswahl bietet. Diese Datumsauswahl dient als Eingrenzungszeitraum für den Bericht. Weiterhin gibt es zwei Klassen, die mit dem Stereotyp „module" gekennzeichnet sind. Zur Form-Klasse „formUSEPA_Elmshorn" gehört die Modul-Klasse „modDeklarationUSEPA_Elmshorn", sowie zur Form-Klasse „fomAnmeldung_USEPA_Horst" die Modul-Klasse „modAnmeldung".

Diese Modul-Klassen enthalten Attribute und Operationen, die nicht direkt für die Darstellung der GUI's benötigt werden. Die letzte Klasse ist durch den Stereotyp „struct" gekennzeichnet. Diese Klasse beschreibt eine Datenstruktur zur Speicherung von Rampeninformationen. In der Form-Klasse für die Verwaltung wird diese Datenstruktur verwendet, um mehrere Rampen in einem Array „arrRampe" zu speichern.

Anhand des Klassendiagramms lässt sich erkennen, dass die Programmierung in prozeduraler Art erfolgte, denn es gibt keine Klassen, die Modelle der realen Umwelt sind. Denkbar wären hier bspw. ein Modell für LKW, die Rampen oder die Fahrer. Weiterhin gibt es keine Assoziationen zwischen Klassen, dies bedeutet, dass die Klassen nicht miteinander interagieren durch gegenseitigen Aufruf von bereitgestellten Methoden, was unteranderem die objektorientierte Programmierung ausmacht.

Fast alle Methoden und Attribute, die für das System notwendig sind, sind in den Form-Klassen eingebaut, was bedeutet, dass in diesen Klassen die Präsentationslogik und die Geschäftslogik enthalten sind. Es gibt also keine Trennung nach dem Model-View-Controller-Muster. In den beiden Modul-Klassen ist jeweils eine Methode „DBÖffnen", die eine direkte Verbindung zur Datenbank herstellt. Somit ist das System nach der 2-Schichten-Architektur aufgebaut.

Im Klassendiagramm ist zu erkennen, dass der Pfad zur Datenbank hart im Quellcode eingetragen ist (siehe Modul-Klassen „modDeklarationUSEPA_Elmshorn" und „modAnmeldung" jeweils Attribut „DBPfad"). Weiterhin ist die Anschrift des Lagers auch fest im Quellcode eingetragen (siehe Modul-Klasse „modAnmeldung" Attribut „anschrift").

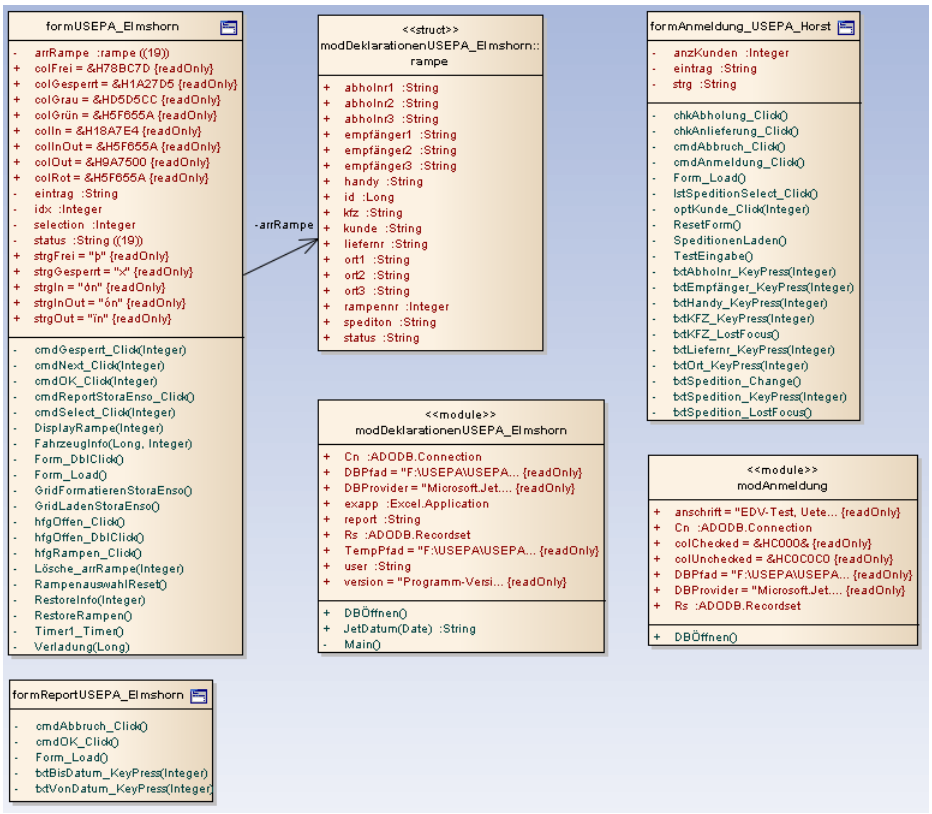

Abbildung 3.2: Klassendiagramm des Altsystems

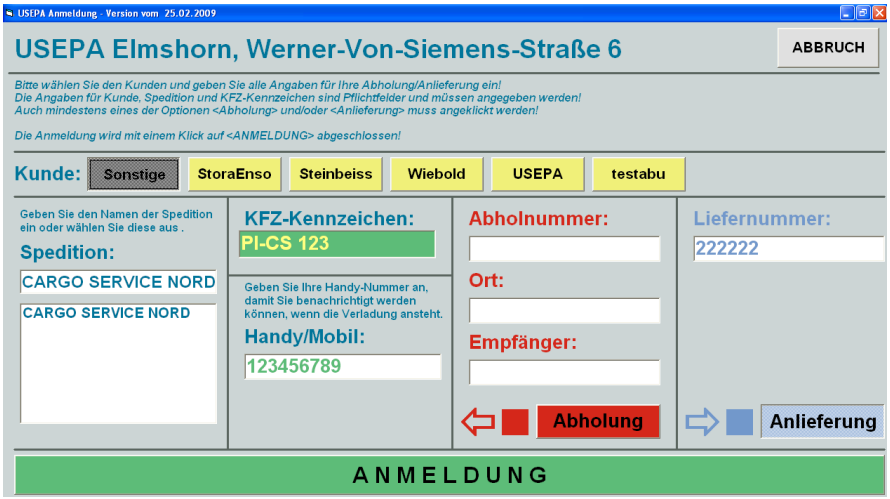

Abbildung 3.3: Anmeldemaske am Anmeldeterminal (falscher Firmenname, da zur Verfügung gestellte Testversion)

Abbildung 3.4: Verwaltung für Lagermitarbeiter

Entity-Relationship-Diagramm

Das Entity-Relationship-Diagramm (siehe Abbildung 3.5) zeigt, dass die Datenbank für das System aus insgesamt drei Tabellen besteht:

- tblFahrzeuge:

  Diese Tabelle enthält alle Daten, die bei der Anmeldung des Fahrers erhoben werden. Weiterhin enthält sie die drei Zeitstempel. Einer für die Anmeldezeit, einer für die Rampenzuteilung und einer bei der Abmeldung des Fahrzeuges. Das Attribut „Id" ist indiziert.

- tblRestoreRampen:

  Diese Tabelle enthält die aktuelle Zuordnung von Fahrzeugen zu den Rampen. Dafür werden die Attributwerte „FahrzeugId" und „KFZ" aus der Tabelle „tblFahrzeuge" in diese Tabelle geschrieben. Die Attributwerte „FahrzeugId" und „Id" sind indiziert.

- tblKunden:

  Diese Tabelle ist eine Konfigurationstabelle, mit dessen Hilfe die Kunden-Button's auf der Anmelde-Benutzungsoberfläche konfiguriert werden.

Alle drei Tabellen enthalten einen Primärschlüssel, der numerisch und nicht Attributabhängig ist. Zwischen den Tabellen existieren keine Fremdschlüsselbeziehungen. Die Tabelle „tblFahrzeuge" enthält Redundanzen, da die Attribute „Spedition" und „Empfänger" im-

mer vollständige Werte enthalten und keine Schlüsselwerte. Weiterhin liegt keine Normalisierung vor, denn gegen die 1. Normalform spricht, dass die Attribute „Empfänger, LieferNr, AbholNr, Ort" nicht atomar sind. Liegen bei der Anmeldung mehrere Werte vor, so müssen diese komplett in dem jeweiligen Attribut eingetragen werden, so dass bspw. das Attribut „Empfänger" mehrere Empfänger enthalten kann.

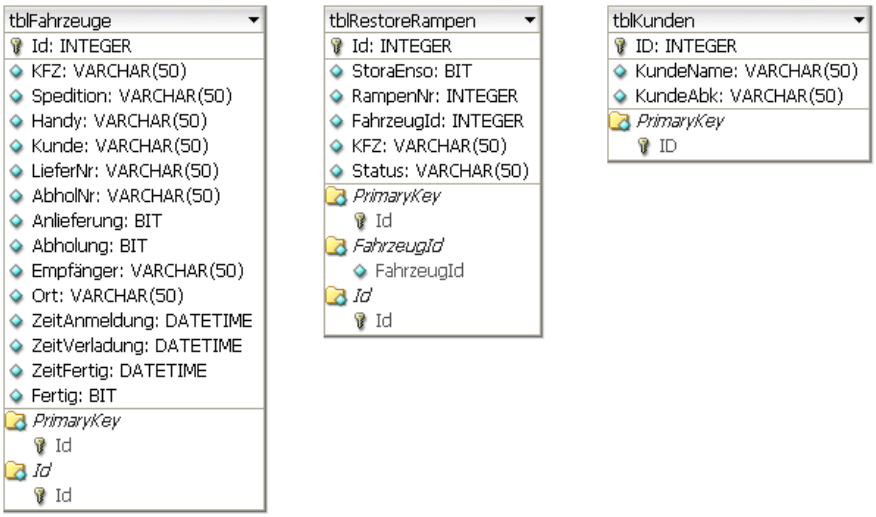

Abbildung 3.5: Entity-Relationship-Diagramm des Altsystems

3.1.3 Bewertung

Die Bewertung des Altsystems soll nach dem Standard ISO/IEC 9126[6] erfolgen. Dieser Standard ist von der International Standard Organization (ISO) und der International Electrical Technical Commission (IEC) entworfen worden, um die Qualität von Software beurteilen zu können. Der ISO/IEC 9126 Standard nennt sechs Merkmale: Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Wartbarkeit und Übertragbarkeit (DSD11; Wik11a). Das Altsystem wird nun im Folgenden anhand dieser sechs Merkmale bewertet.

Funktionalität

Vom vorherigen Lagerhalter sind zwei Anforderungen an das System gestellt worden. Die erste Anforderung war die Verwaltung der LKW, die sich auf dem Betriebsgelände befinden. Es sollte für die Lagermitarbeiter eine Übersicht geschaffen werden, welche LKW warten noch auf dem Betriebsgelände und welche Rampen sind belegt, frei oder gesperrt. Dadurch laufen die Lagermitarbeiter nicht Gefahr eine Rampe doppelt zu belegen oder einen

[6]Die ISO/IEC 9126 ist durch die ISO/IEC 25000 abgelöst (Sta11a). Leider wurde nur kostenpflichtiger Zugang zu diesem Dokument gefunden, daher wird sich in dieser Arbeit auf die alte Norm bezogen.

LKW zu vergessen. Diese Anforderung ist durch die beiden Komponenten „Anmeldung" und „Verwaltung" erfüllt. Die zweite Anforderung war eine Protokollierung der Aktivität des LKW zu erhalten. Diese sollte zum einen der Effizienzsteigerung dienen und zum anderen als Nachweis der Standzeit bei eventuellen Standgeldforderungen. Diese Anforderung ist durch die drei Zeitstempel sowie den Bericht erfüllt. Zur Sicherheit ist zu sagen, dass es keinen Autorisierungs- oder Authentifizierungsmechanismus gibt. Auf die MS Access-Datenbank kann mit jeder MS-Access-Anwendung ohne Passwort zugegriffen werden. Dies ermöglicht auch eine Manipulation der Daten. Durch die fehlenden Fremdschlüssel in der Datenbank kann es zu Dateninkonsistenz kommen. Bspw. wird ein Attributwert geändert oder gelöscht, auf dem es in einer anderen Tabelle eine Beziehung gibt, so wird dieses von der Datenbank akzeptiert.

Zuverlässigkeit

Zur Zuverlässigkeit ist zu sagen, dass das System korrekt arbeitet. Eingaben und Werte werden nicht verfälscht oder vom System geändert. Des Weiteren ist das System zu jeder Zeit verfügbar, vorausgesetzt eine Verbindung zur Datenbank besteht. Falsche Eingaben führen nicht zu einer unerwarteten Reaktion. Die Bedienung durch den Benutzer lässt das System auch nicht in einen Zustand überführen, welcher undefiniert oder aus dem kein anderer Zustand erreicht werden kann. Eine Fehlertoleranz ist in Bezug auf die Datenbankverbindung nicht vorhanden. Will das System auf die Datenbank zugreifen, welche aber nicht erreichbar ist, so wird eine Fehlermeldung ausgegeben. Nach Bestätigung der Fehlermeldung schliesst sich die Anwendung. Dies führt gerade beim Anmeldeterminal, an dem sich externe Fahrer anmelden, zu Problemen. Denn das Anmeldeterminal steht nicht unter ständiger Beobachtung von internen Mitarbeitern und die externen Fahrer haben oftmals keine Kenntnis über das Funktionsprinzip eines Neustarts der Anwendung. Schliesst sich die Anwendung wegen einer nicht vorhandenen Datenbankverbindung, sind die Daten, die in der Benutzungsoberfläche erfasst worden sind, nicht wiederherstellbar.

Benutzbarkeit

Wie in den Abbildungen 3.3 und 3.4 zu sehen, sind die Oberflächen nicht mit Widgets oder Funktionen überfrachtet. Die Oberflächen sind übersichtlich, so dass sich mit einem Blick das Wesentliche erfassen lässt. Dies wird auch durch den gezielten Einsatz von Farbe erreicht. Das System ist intuitiv aufgebaut und weist auch Beschreibungen zur Bedienung direkt auf der Oberfläche auf, z.B. bei der Anmeldung (siehe Abbildung 3.3). Somit ist das System schnell erlernbar und gut zu bedienen.

Effizienz

Zur Zeit befindet sich die Access-Datenbank auf einem normalen Office-PC, der mit einer 2,33 GHz CPU und 2 GB RAM ausgestattet ist. Als Betriebssystem dient ein Microsoft Windows XP Professional. Dieser Datenbank-PC steht physikalisch am Lagerstandort und ist mit den Client-PC's über ein Local Area Network (LAN) verbunden, dessen Leistung 100 Mbit/s beträgt. Beim Zugriff der Clients auf die Access-Datenbank ist kein spürbarer Leistungseinbruch des PC's zu merken. Auch die Antwortzeiten auf den Clients liegen im Millisekundenbereich. Zu Testzwecken wurde ein Client ausserhalb des Lagerstandortes mit dem Datenbank-PC verbunden. Die Anbindung erfolgte hierbei mittels eines VPN-Tunnels, der physisch über eine SDSL-Leitung mit einer Leistung von 2Mbit/s aufgebaut worden ist. Der Test zeigte, dass die Antwortzeiten des Clients im 2-stelligen Sekundenbereich lagen. Somit ist ein Arbeiten ausserhalb des LAN kaum möglich.

Wartbarkeit

Die Analyse des Klassendiagramms hat gezeigt, dass einige Werte im Quellcode hart codiert sind. Dies führt dazu, dass bei Änderung dieser Werte der Quellcode geändert, neu compiliert und ausgerollt werden muss. Dies bedeutet hohen Aufwand, bspw. beim Ändern des Datenbanknamens oder des Speicherortes der Datenbank. Die Analyse der Datenbank hat gezeigt, dass die Tabelle „tblKunden" für die Anzeige der Kunden-Button's auf der Benutzungsoberfläche zuständig ist. Die Tabelle kann unbegrenzt erweitert werden, jedoch ist der Platz auf der Benutzungsoberfläche nur begrenzt, so dass ab einer gewissen Anzahl von Kunden nicht mehr alle Kunden-Button's angezeigt werden können. Die Modifizierbarkeit ist somit in Bezug auf die Kunden-Button's nur eingeschränkt möglich. Die 2-Schichten-Architektur erschwert die Erweiterung des Systems, bspw. das Anbinden anderer Systeme. Eine Wartung des Systems wird bei zunehmender Funktionsmenge immer schwieriger werden, bedingt durch die Nichtverwendung von objektorientierter Programmierung, das Fehlen des Model-View-Controller-Prinzips und die fehlende Normalisierung der Datenbank.

Übertragbarkeit

Das System ist mit der Programmiersprache Visual Basic entwickelt, was den Effekt hat, dass das System nahezu auf allen Windows-Plattformen lauffähig ist. Auch ein Austausch von Versionen innerhalb des MS-Access Produktes ist ohne weiteres möglich. Allerdings, bedingt durch die 2-Schichten-Architektur und die damit verbundene enge Kopplung zwischen Präsentation und Datenhaltung, ist ein Austausch der Datenbank, bspw. der Einsatz eines RDBMS wie Oracle, nur durch Anpassung der Präsentationsschicht möglich.

Zusammengefasst kann man sagen, dass das System die gestellten Anforderungen erfüllt hat und bereits seit Jahren erfolgreich im Einsatz ist bei einer Einsatzzeit von 24 Stunden am Tag in 6 Tagen die Woche. Die Qualitätsmerkmale Wartbarkeit, Übertragbarkeit und die fehlende Datensicherheit zeigen jedoch, dass eine Erweiterung des Systems nicht sinnvoll wäre.

3.2 Das Neusystem

Die Bewertung des Altsystems zeigt, dass zwar einige Qualitätskriterien erfüllt werden, jedoch nicht alle. Weiterhin fehlen ihm viele Anforderungen moderner Unternehmensanwendungen und moderner Dock und Yard Management Systeme. In vielen Bereichen sind Verbesserungen möglich. Das neue System sollte es nicht erlauben Daten direkt in der Datenbank zu manipulieren, da dies zu Fehlinformationen und somit zu Fehlentscheidungen führen kann. Es sollte als Informationsportal für alle beteiligten Mitarbeiter des Logistikunternehmens werden. Der Informationsfluss muss in Echtzeit standortübergreifend gewährleistet werden. Das System sollte so aufgebaut werden, dass es den schnell ändernden Anforderungen eines Logistikunternehmens folgen kann. Da in der Logistik der Austausch von Informationen wichtig ist, sollte das neue System auch Schnittstellen anbieten, um mit anderen Systemen kommunizieren zu können.

Ein weiterer wichtiger Punkt ist das Zeitfenster-Management. In Dock und Yard Management Systemen, die sich am Markt befinden, im Altsystem und in der allgemeinen Beschreibung eines DYMS von Thorsten Jochheim (Joc04) findet man nichts über Zeitfenster-Management. In der jüngsten Zeit sind einige Fachartikel erschienen (Küm11; Joh11; Tra11; tbu11), die eine Notwendigkeit für Zeitfenster-Management beschreiben, da hierdurch eine Optimierung des Lagerablaufs durch Verbesserung der Planbarkeit der Ressourcen möglich ist. Die beiden Interviewpartner (die Firma Cargo Service Nord GmbH und die Firma Semmelhaack-Logistik GmbH) sehen auch eine Notwendigkeit für Zeitfenster-Management. Daher ist das Zeitfenster-Management eine weitere wichtige Anforderung an das neue System.

3.2.1 Anforderungsanalyse

Um die Anforderungen an das neue System zu spezifizieren wurde sich in dieser Arbeit an das inkrementelle Modell der Softwareentwicklung gehalten. Beim inkrementellen Modell werden die Anforderungen an das System nach Möglichkeit vollständig erfasst. Im ersten Schritt wird dann jedoch nur ein Teil der Anforderungen implementiert, ein sog. Inkrement. Das Inkrement wird dann um weitere Inkremente erweitert, bis das System vollständig ist. Die fast vollständige Erfassung der Anforderungen jedoch bewirkt, dass nicht umgesetzte Anforderungen beim Design mit berücksichtigt werden. Das erste Inkrement wird an den

Kunden ausgeliefert, damit dieser es testen und ggf. noch Anforderungen anpassen oder ergänzen kann. Das inkrementelle Vorgehensmodell gehört somit zu den iterativen Prozess-Modellen, da zusammen mit dem Kunden das System iterativ entwickelt wird.(Bal00; Som07) Um die Anforderungen zum Anfang möglichst vollständig zu erfassen, werden in dieser Arbeit sog. Mock-ups verwendet, die kurz erläutert werden sollen.

Mock-up

Mock-up kommt aus dem Englischen und wird unter anderem übersetzt mit „die Attrappe". Ursprünglich stammt der Begriff aus der Luftfahrtindustrie. Dort werden Attrappen zu verschiedenen Testzwecken eingesetzt, um bspw. Tests an der Innenaussattung durchzuführen oder als Ausstellungsstück auf Messen. Die Attrappen enthalten entweder gar keine Funktionalität oder nur in sehr begrenzter Menge. In der Softwareentwicklung bezeichnen Mock-ups Benutzungsoberflächen ohne oder nur mit sehr geringer Funktionalität (Wik11f). Vorteile der Mock-ups sind eine Verbesserung der Kommunikation zwischen Entwickler und Benutzer, da beide nicht abstrakt über eine Benutzungsoberfläche reden müssen. Weiterhin können Mock-ups die spätere Akzeptanz einer Anwendung fördern, da der Benutzer bereits in der frühen Analysephase einen Eindruck von der neuen Anwendung gewinnt. In dieser Arbeit kam das kostenpflichtige Mock-up-Tool der Firma Balsamiq Studios (BS11) zum Einsatz.

Vorgehensweise

Um eine Anforderungsanalyse zu erstellen wurden Interviews mit Mitarbeitern der Firma Cargo Service Nord GmbH und der Firma Semmelhaack-Logistik GmbH durchgeführt. Vor den Interviews wurde anhand von Vorgesprächen mit den zukünftigen Benutzern Mock-ups für das gesamte System (siehe Anhang B.1) und ein Aktivitätsdiagramm für die Anmeldung (siehe Abbildung 3.6) entworfen. Die Mock-ups und das Aktivitätsdiagramm dienten dann in den Interviews als Gesprächsgrundlage. Nach den Interviews sind bereits erste konkrete Vorstellungen an das neue System entstanden (siehe Protokolle Anhang C.1). Anhand dieser Vorstellungen sind die Mock-ups (siehe Anhang B.2) und das Aktivitätsdiagramm (siehe Abbildung 3.7) weiterentwickelt worden.

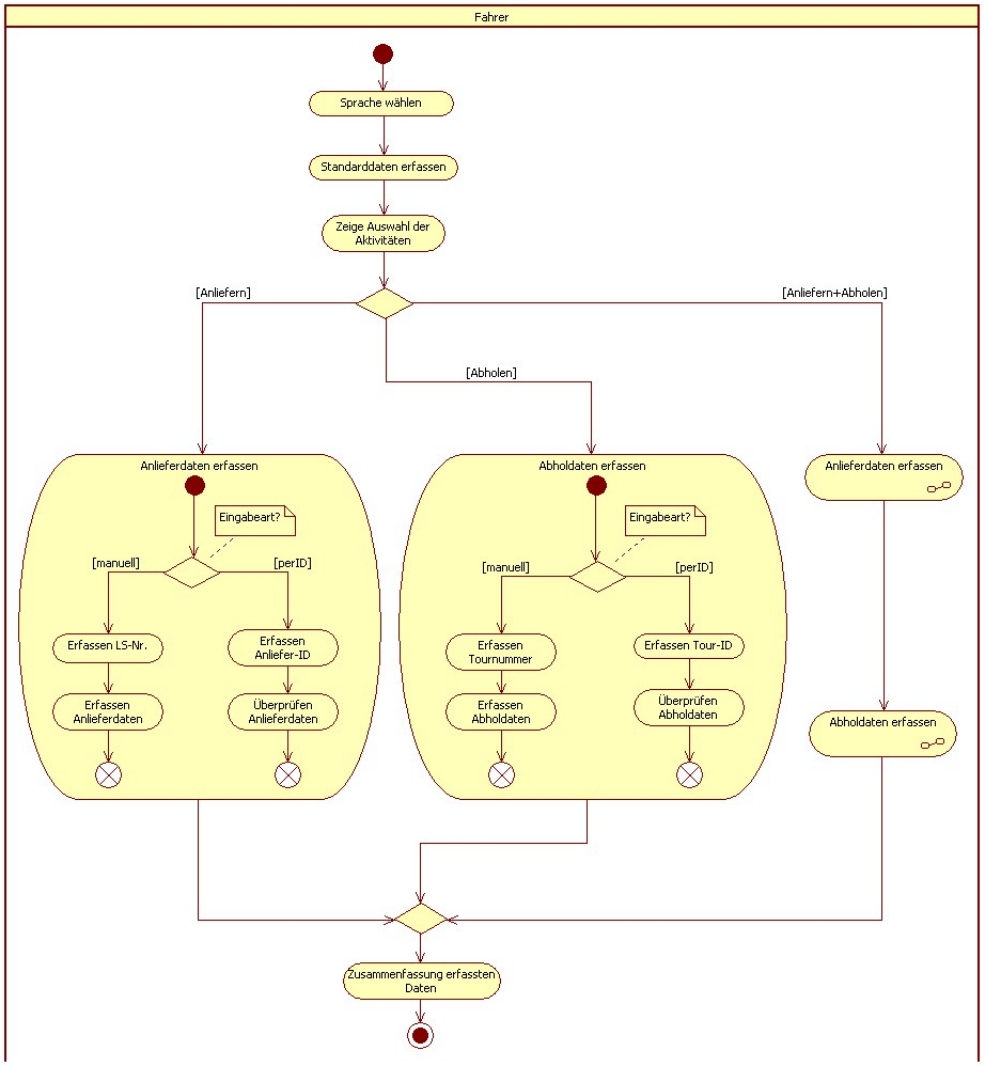

Abbildung 3.6: Aktivitätsdiagramm Workflow der Anmeldung aus Vorgespräch

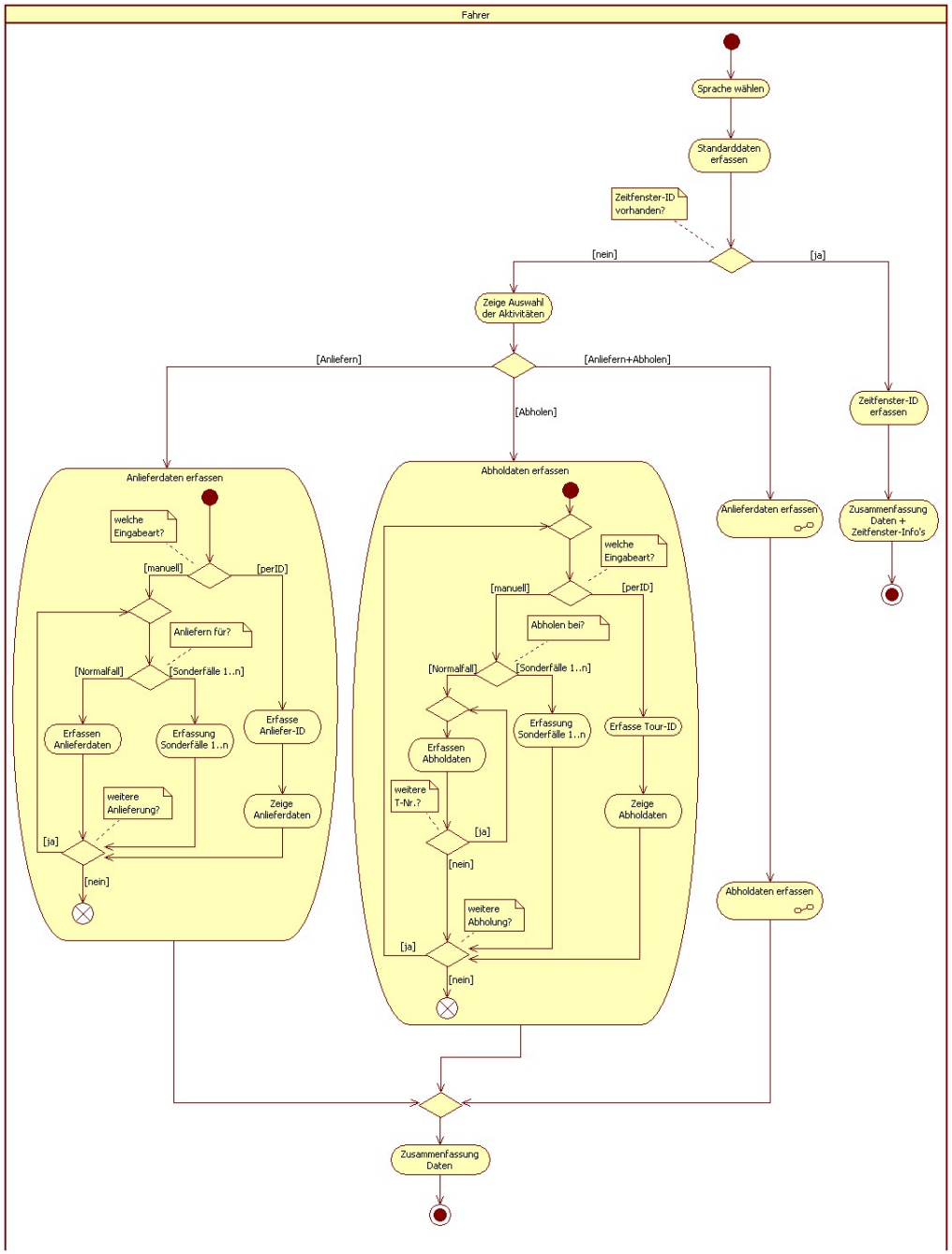

Abbildung 3.7: Aktivitätsdiagramm Workflow der Anmeldung nach Interviews

3.2.2 Anforderungsspezifikation

Aus den Interviews, den Erfahrungen mit dem Altsystem und dem allgemeinen Funktionsumfang von DYMS sind folgende Anforderungen an das neue System spezifiziert worden. Die Anforderungen gliedern sich in funktionale Anforderungen und nichtfunktionale Anforderungen. Die funktionalen Anforderungen beschreiben dabei die Tätigkeiten, die das System erfüllen soll. Die nichtfunktionalen Anforderungen beschreiben die Eigenschaften des Systems.

3.2.2.1 Funktionale Anforderungen

Die funktionalen Anforderungen werden in Anwendungfällen beschrieben. Um zunächst eine klare Übersicht über die Anforderungen zu erhalten, sind diese als Anwendungsfälle in einem Anwendungsfalldiagramm in Abbildung 3.8 dargestellt. Exemplarisch sind danach drei Anwendungsfälle detailliert beschrieben. Eine vollständige Liste der detaillierten Anwendungsfallbeschreibungen ist im Anhang D zu finden. Für eine Beschreibung Bedarf es vorab einer Definition einiger Ausdrücke. Mit fahrer- und fahrzeugspezifischen Daten sind Daten gemeint wie der Name des Fahrers, der Name der Firma für die der Fahrer tätig ist, die Mobilrufnummer des Fahrers, das amtliche Kennzeichen des Fahrzeuges, ob es sich bei dem Fahrzeug um einen Sattelauflieger oder um einen Motorwagen handelt. Mit ladungsspezifischen Daten sind Daten gemeint wie Empfangsorte bei Abholungen oder Absender und Empfangslagerkunden bei Anlieferungen. Weiterhin sind dazu anzugeben die Warenart, Menge und Gewichte.

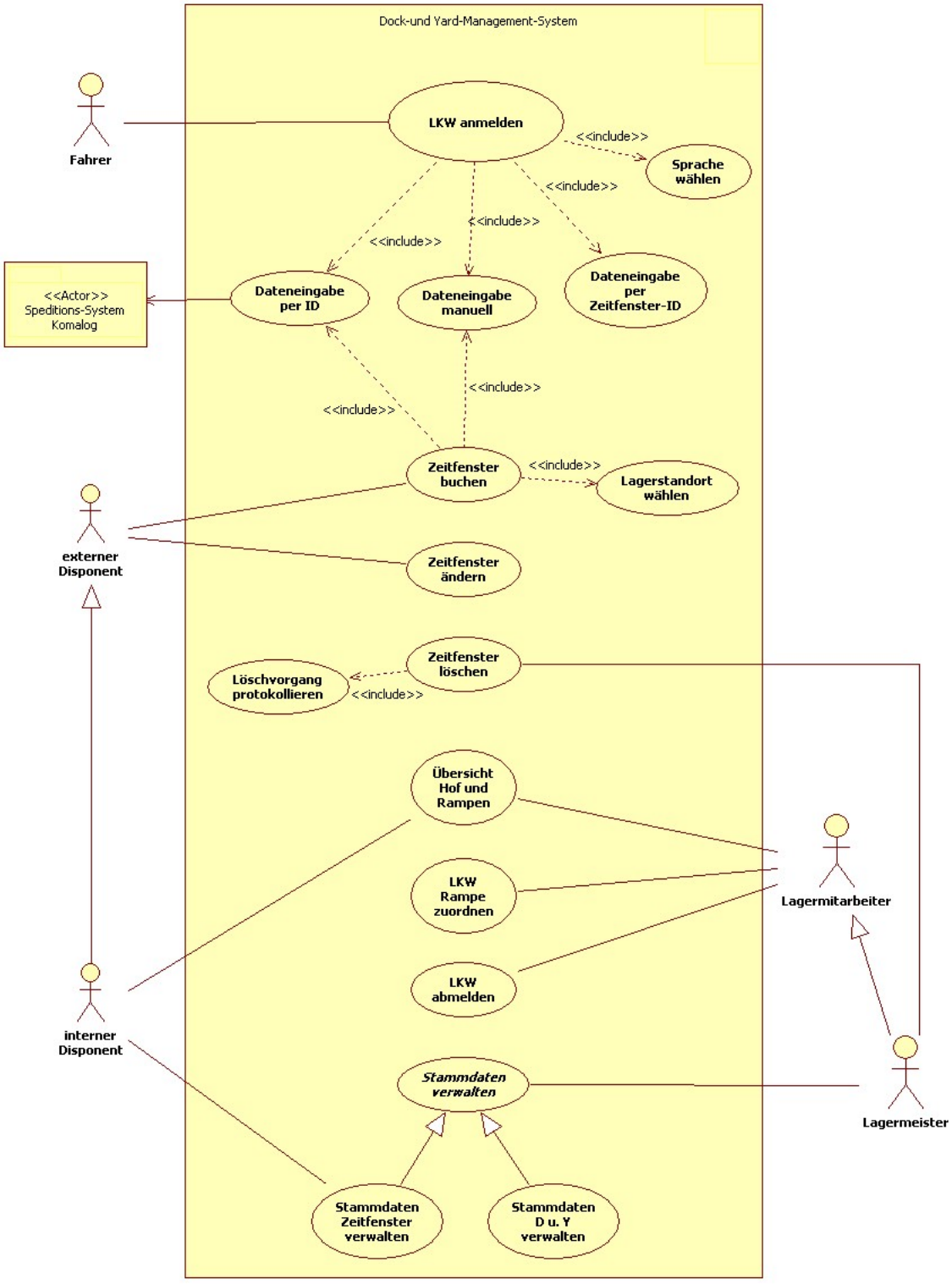

Abbildung 3.8: Anwendungsfalldiagramm

Nr.:	6
Anwendungsfall:	Übersicht Hof und Rampen
Ziel:	Den aktuellen Zustand der Rampen und des Hofes sehen
Kategorie:	must have
Vorbedingung:	Die Übersicht muss von verschiedenen örtlich getrennten Standorten einsehbar sein
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, Lagermeister, Lagermitarbeiter
Auslösendes Ereignis:	-
Beschreibung:	Nach dem Einloggen am System ist eine Übersicht, bspw. in tabellenform, zu sehen. Die Übersicht zeigt aktuell, welche Fahrzeuge angemeldet auf dem Hof stehen und welche Rampen mit welchen Fahrzeugen belegt sind. Dazu gibt es noch weitere Informationen, wie bspw. wie lange sind die Fahrzeuge schon auf dem Hof oder an einer Rampe, was sollen sie laden und noch weitere Details. Zudem müssen sich wichtige Informationen, wie bspw. die Wartezeit, automatisch aktualisieren, damit der Lagermeister diese zeitkritischen Informationen immer aktuell zur Verfügung hat.

Tabelle 3.1: Anwendungsfall: Übersicht Hof und Rampen

Nr.:	7
Anwendungsfall:	LKW Rampen zuordnen
Ziel:	einem LKW ist 1 bis n Rampen zugewiesen
Kategorie:	must have
Vorbedingung:	LKW ist korrekt angemeldet und die zuzuweisenden Rampen sind belegbar
Nachbedingung Erfolg:	Das Fahrzeug ist 1 oder n Rampen zugewiesen
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister, Lagermitarbeiter
Auslösendes Ereignis:	Ein Zeitfenster wird fällig oder ein Fahrzeug und die entsprechende Ladung sind bereit
Beschreibung:	Einer der Akteure prüft die angemeldeten Fahrzeuge auf dem Hof und die Verfügbarkeit der Rampen Nach Kriterien, wie bspw. Wartezeit, gebuchtes Zeitfenster oder Relation, wählt er ein Fahrzeug aus und weist es einer Rampe zu. Handelt es sich bei dem Fahrzeug z.B. um einen Gliederzug, so kann er diesem Fahrzeug auch 2 Rampen gleichzeitg zuweisen. Durch die Auswahl des Lagerkunden kann der Akteur erkennen, ob das Fahrzeug eventuell an mehreren Rampen laden muss. Der Akteur kann dann dem Fahrzeug die weiteren Rampen zuweisen. Durch das Attribut Teilladung oder Komplettladung kann der Akteur erkenn, ob ein Fahrzeug eventuell vorgezogen werden kann, da es nur eine Teilladung bekommt. Ist eine Rampe belegt, so kann trotzdem ein weiteres Fahrzeug dieser Rampe zugeordnet werden. Allerdings muss erkennbar sein, welches Fahrzeug aktiv an der Rampe tätig ist, und welches Fahrzeug auf das Freiwerden der Rampe wartet.

Tabelle 3.2: Anwendungsfall: LKW Rampen zuordnen

Nr.:	12
Anwendungsfall:	Zeitfenster buchen
Ziel:	Ein Zeitfenster für eine Aktivität (Laden oder Entladen) gebucht zu bekommen
Kategorie:	must have
Vorbedingung:	Der Akteur muss ein Account haben und das zu buchende Zeitfenster muss eine Mindestzeitspanne in die Zukunft haben
Nachbedingung Erfolg:	Der Akteur erhält eine Zeitfenster-ID. Diese kann später der Fahrer zur Anmeldung am System nutzen
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, externer Disponent
Auslösendes Ereignis:	-
Beschreibung:	1. Der Akteur loggt sich ein 2. Der Akteur gibt ladungsspezifische Daten in das System ein, dabei kann er zwischen der manuellen Eingabe oder der Eingabe per ID wählen 3. In Abhängigkeit der ladungsspezifischen Daten schaltet das System die möglichen Zeitfenster frei 4. Der Akteur wählt ein Zeitfenster 5. Der Akteur kontrolliert die Daten und bestätigt diese, um das Zeitfenster erfolgreich zu buchen

Tabelle 3.3: Anwendungsfall: Zeitfenster buchen

3.2.2.2 Nichtfunktionale Anforderungen

Das System sollte zudem folgende nichtfunktionale Anforderungen erfüllen. Einige davon sind auch in der Literatur allgemein als nichtfunktionale Anforderungen für Unternehmensanwendungen definiert.

Mehrbenutzerfähigkeit(Bal00; lHHK07)

Eine hohe Anzahl konkurrierender Zugriffe sollte möglich sein, und die Benutzer einer Verwaltung unterliegen. Das System soll den Zweck erfüllen, dass gerade im Tagesgeschäft

mehrere Disponenten, Lagerarbeiter, Lagermeister parallelen Zugriff auf das System haben werden. Dazu kommen beim Zeitfenster-Management eine unbekannte Zahl an externen Disponenten und Benutzern, die Zeitfenster parallel buchen möchten.

Skalierbarkeit(Bal00; Fis06; IHHK07)

Die Anwendung sollte sich an Lastveränderungen anpassen können, bspw. bei einer Erhöhung der Benutzerzahl. Wird das Zeitfenster-Management gut von den externen Disponenten angenommen, so kann sich die Anzahl der Benutzer drastisch und unbekannt erhöhen. Das System darf zu Spitzenzeiten keine deutliche Erhöhung der Antwortzeiten verursachen.

Verfügbarkeit(Bal00; Fis06; IHHK07)

Die Verfügbarkeit sollte hoch sein, bspw. sollte beim Ausfall eines Teilsystems nicht die gesamte Anwendung ausfallen und somit die Geschäftstätigkeit des Unternehmens gefährden. Auch bei Fehleingaben darf das System in keinen unerwarteten Zustand gelangen und schlimmer, aus dem es dann nicht mehr hinaus kommt. Besonders schwerwiegend wäre dieses Fehlverhalten bei der Anmeldung, die weitestgehend unüberwacht von den Lagermitarbeitern läuft.

Transaktionalität(Han96; IHHK07)

Die Kommunikation zwischen den Clients und dem Server sollten auf Transaktionen basieren. Eine Transkation fasst mehrere zusammenhängende Aktionen zu einem Geschäftsvorfall zusammen. Dies verhindert Dateninkonsistenz, da der Geschäftsvorfall entweder ganz oder, im Fehlerfall einer Aktion, gar nicht ausgeführt wird.

Datensicherheit

Die Daten des Systems dürfen nicht direkt bspw. in der Datenbank manipuliert werden. Eine Manipulation darf nur über die Anwendung erfolgen, da diese auch die Manipulationen protokolliert und die Authorisierung vornimmt.

Bedienbarkeit

Das System sollte leicht bedienbar sein. Das allgemeine Glossar im Bereich der Logistik sollte sich in der Anwendung wiederfinden. Weiterhin sollte gerade bei der „Anmeldung" keine Fremdwörter oder ein Vermischen von deutscher und englischer Sprache stattfinden. Zudem sollte der Benutzer mittels Masken in der „Anmeldung" durch einen Fragenkatalog geführt werden, um so eine intuitive Bedienung zu ermöglichen.

Weitere Anforderungen

Das System sollte ohne Installationsaufwand in allen Standorten und Lägern verfügbar sein.

Kapitel 4

Systemarchitektur

In diesem Kapitel wird der Architekturentwurf des neu zu entwickelnden Systems beschrieben. Als Grundlage für den Entwurf dient die Anforderungsspezifikation aus Abschnitt 3.2.2. In dieser Arbeit wird die Architektur mit Hilfe der Quasar-Richtlinien entworfen. Quasar steht für Qualitätssoftwarearchitektur und wurde von der Firma sd&m 1998 ins Leben gerufen. Quasar liefert einen Richtlinien-Bauplan, der es ermöglichen soll qualitativ hochwertige Software zu bauen (Sie04).

Die Systemarchitektur definiert sich aus drei Architekturen:

- die A-Architektur
- die T-Architektur
- die TI-Architektur

Im Folgenden wird für das neue System zu jeder Architektur ein Entwurf vorgestellt, um eine vollständige Systemarchitektur zu erhalten.

4.1 A-Architektur

Die A-Architektur beschreibt alle Komponenten aus rein fachlicher Sicht ohne technische Bezüge. Sie ist speziell auf die fachliche Aufgabe, was das System leisten soll, ausgelegt, und daher so gut wie nicht wiederverwendbar. Die A-Architektur sollte Fachbegriffe verwenden, dadurch hat der Auftraggeber einen gewissen Wiedererkennungswert seiner fachlichen Anforderung. Die A-Architektur enthält mehrere A-Komponenten. Alle A-Komponenten zusammen ergeben eine endliche Menge und bilden somit eine übergeordnete A-Komponente, den sog. Anwendungskern.

4.1.1 A-Komponenten

Für das DYMS ergeben sich nach Analyse der Anforderungen folgende A-Komponenten:

- Anmeldedaten
- Aktivitäten
- Hof
- Rampen
- Zeitfenster
- Stammdaten

Die Abbildung 4.1 zeigt eine grafische Übersicht auf die vorhandenen A-Komponenten und ihre Beziehungen zueinander sowie die Schnittstellen zu den T-Komponenten. Im Folgenden werden die A-Komponenten näher erläutert.

Alle fahrzeugspezifischen und fahrerspezifischen Daten bilden die Anmeldedaten. Dies können bspw. Fahrername, Fahrzeugkennzeichen usw. sein. Diese Daten müssen zu jedem Fahrzeug, welches sich auf dem Betriebshof anmeldet, erhoben werden, um Fahrer und Fahrzeug im System an jeder Stelle identifizieren zu können. Die Anmeldedaten sind im DYMS von zentraler Bedeutung, den im Workflow eines Fahrzeuges mit einer Aktivität auf dem Betriebshof, durchläuft dieses nacheinander die Stationen Hof, Rampe und Verlassen des Hofes.

Weiter besitzt jeder Anmeldedatensatz Aktivitäten. Aktivitäten beschreiben die Tätigkeit, weshalb sich ein Fahrzeug auf dem Betriebsgelände aufhält. Ein Fahrzeug kann Waren anliefern und/oder abholen. Weiter gibt es noch spezielle Sonderfälle, die bei jedem Unternehmen unterschiedlich ausfallen können. Im DYMS wurde als Sonderfallaktivität noch die Behandlung von leeren Europaletten bzw. im allgemeinen leerer Ladehilfsmittel, mit ins Konzept aufgenommen. Die Aktivitäten geben dem Lagermeister bzw. Lagermitarbeiter die Informationen, wie sie das Fahrzeug zu behandeln haben. Bspw. besagen die Daten der Anlieferaktivität, dass ein Fahrzeug dringend benötigte Ware anliefern möchte. Dann sollte die Behandlung des Fahrzeuges mit höherer Priorität erfolgen.

Der Hof beschreibt fachlich den Betriebshof. Diese Komponente verwaltet die Fahrzeuge auf dem Betriebshof. Es werden Informationen verwaltet wie aktuell befindliche Fahrzeuge auf dem Hof, deren Wartezeit und ob bspw. ein Zeitfenster vorliegt.

Die Rampen verwalten alle Rampen der Lagerhalle. Anwendungsfälle sind bspw. einem Fahrzeug, welches sich auf dem Hof befindet, einer Rampe zuweisen, ein zugewiesenes Fahrzeug von einer Rampe abmelden oder den Status einer Rampe ändern. Als Rampenstatus gibt es frei, gesperrt oder reserviert. Weiterhin können Daten zu den Rampen verwaltet werden, wie bspw., ob eine Rampe für eine Zugmaschine mit Sattelauflieger geeignet ist.

Dazu werden die Anmeldedaten benötigt, da in diesen als fahrzeugspezifische Daten erfasst wird, vom welchem Ladungsträgertyp ein Fahrzeug ist. Sprich handelt es sich um einen Sattelauflieger oder einen Motorwagen mit Anhänger. Wie die Komponente „Hof" verwaltet die Komponente „Rampen" den Zeitpunkt, wann ein Fahrzeug einer Rampe zugewiesen worden ist. Anhand dieser Zeitpunkte wird dann eine Wartezeit errechnet.

In der Komponente Zeitfenster können externe Disponenten von Fremdfirmen sowie interne Disponenten Zeitfenster für ihre Fahrzeuge buchen. Die Komponente Zeitfenster bietet die Anwendungsfälle: „Zeitfenster buchen", „Zeitfenster löschen" und „Zeitfenster ändern" an. Nicht alle Anwendungsfälle sind für jeden Benutzer ausführbar. Dies regelt die Autorisierungskomponente. Ein Zeitfenster besteht aus einer eindeutigen Nummer, einer ID und einer Zeitspanne. Beim Buchen eines Zeitfensters werden die Anmeldedaten und die dazugehörigen Aktivitäten erfasst. Somit stehen diese Daten im System bereits zur Verfügung, um bspw. eine kürzere und schnellere Anmeldung zu erreichen.

Die Komponente Stammdaten verwaltet alle Daten des Systems, die für den Ablauf eines Workflows zwar benötigt werden, die aber nicht von diesem geändert werden. Die Persistierung der Stammdaten ist unabdingbar, damit diese im System permanent zur Verfügung stehen. Als Beispiel für Stammdaten sind in diesem System Benutzerdaten und Adressdaten. Adressen werden bei den Aktivitäten benötigt, um bspw. die Absenderadresse zu bestimmen oder den Lagerkunden für die Anlieferung. Weiterhin wird den Benutzern auch Firmenadressen zugeordnet, um diesen die korrekten Zugriffsrechte in der Zeitfensterkomponente einzuräumen. So dürfen alle Benutzer der gleichen Firma alle firmenspezifischen Daten sehen. Benutzer werden für die Autorisierungskomponente benötigt, da diese auf Benutzergruppen-Basis arbeitet.

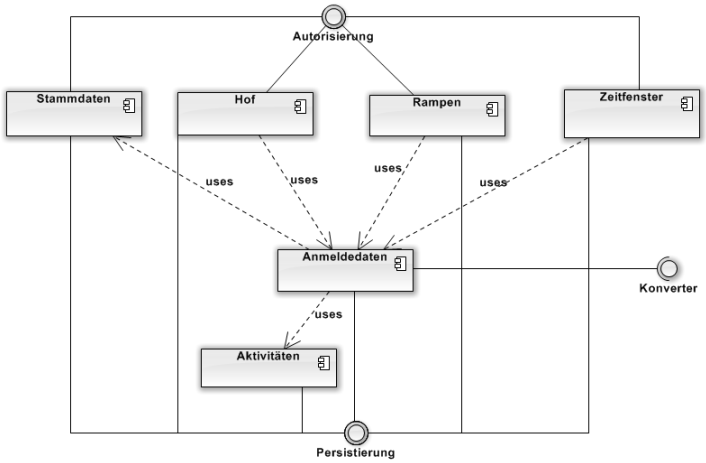

Abbildung 4.1: A-Architektur

4.1.2 Datenmodell zur A-Architektur

Die Abbildung 4.2 zeigt eine schematische Darstellung des Datenmodells wie es in der Datenbank für das DYMS angelegt ist. Bei dem Diagramm handelt es sich um ein Entity-Relationship-Diagramm in der sog. Martin-Notation(Wik11e). Die vier Hauptentitäten sind Anmeldedaten, Hof, Rampen und Zeitfenster. Gut zu erkennen ist, dass die Entität Anmeldedaten eine zentrale Rolle einnimmt. Zu den Anmeldedaten gehören Aktivitäten, die hier durch die Entitäten Anlieferdaten, Abholdaten und Leerepaletten repräsentiert werden. Eine Aktivität muss immer genau ein Anmeldedatensatz referenzieren. Ein Anmeldedatensatz muss aber nicht jede Aktivität referenzieren, wenn bspw. nur etwas abgeholt werden soll, dann werden nur die Entität Abholdaten benötigt. Der Aufbau der Aktivitäten im Datenmodell ermöglicht es, beliebig viele Aktivitäten hinzuzufügen. Denkbar wären noch Aktivitäten wie Werkstattbesuch oder Ad-Blue tanken. Somit ist das DYMS erweiterbar, um individuelle Fahrzeugtätigkeiten auf einem Betriebshof zu protokollieren. Wie bereits in der A-Komponente Anmeldedaten beschrieben, wird die Entität Anmeldedaten in den anderen drei Hauptentitäten Hof, Rampen und Zeitfenster je nach Fortschritt des Workflows referenziert. Ein Anmeldedatensatz kann entstehen, indem er über ein Anmeldeportal erfasst wird oder aber beim Buchen eines Zeitfensters. Beim Anmeldeportal wird der Anmeldedatensatz sofort einem Hof zugewiesen, da er sich auf dem Betriebshof befindet. Beim Zeitfensterbuchen wird er zunächst nur vom Zeitfenster referenziert, da ein Zeitfenster vor der Ankunft des Fahrzeuges auf dem Betriebshof gebucht wird. Meldet der Fahrer sich dann mit der Zeitfenster-ID im Anmeldeportal an, so wird die Referenz zusammen mit einem Ankunftszeitstempel in die Entität Hof eingetragen. Wird das Fahrzeug dann einer Rampe zugewiesen, so erfolgt ein Eintrag der Anmeldedatenreferenz in die Entität Rampen. Ein Anmeldedatensatz muss nicht von einer Rampe referenziert werden, wenn bspw. nur die Aufenthaltsdauer des Fahrzeuges auf dem Betriebshof protokolliert werden soll. Bei den Entitäten Adressen, Benutzer und Rampendaten handelt es sich um Stammdaten.

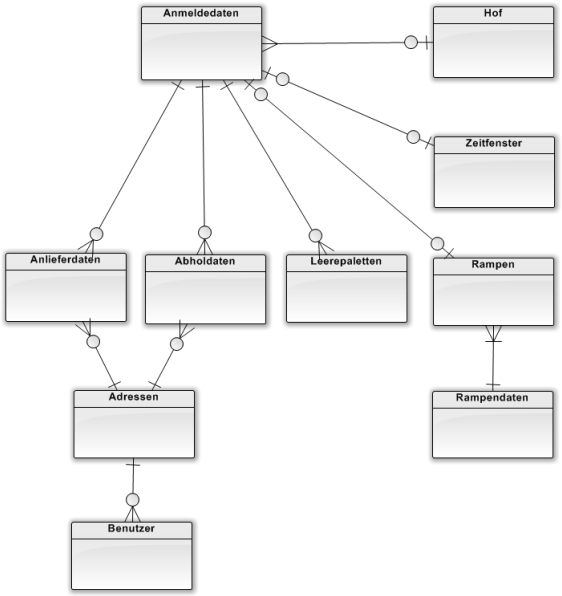

Abbildung 4.2: Datenmodell

4.2 T-Architektur

Die T-Architektur beschreibt die technischen Komponenten des Systems. Diese T-Komponenten sind nicht anwendungsspezifisch. Sie sind allgemeingültig und sollten die Möglichkeit bieten, auch in anderen Systemen wiederverwendet zu werden. Die T-Architektur umschliesst die A-Architektur. Die T-Komponenten erfüllen die Aufgaben, die neben den fachlichen Aufgaben der A-Komponenten notwendig sind, um ein System zu betreiben. Als Beispiel soll hier die Persistierung der Daten dienen.

Die Abbildung 4.3 zeigt die T-Architektur des Dock und Yard Management Systems. Das Konzept dieses Systems sieht eine 4-Schichten-Architektur vor. Wie bereits im Abschnitt 2.3.1 erläutert resultieren aus dem Einsatz von Schichten mehrere Vorteile. Die 4-Schichten-Architektur dieses Systems sieht folgende Schichten vor:

- Client-Schicht
- Darstellungs-Schicht
- Anwendungs-Schicht
- Datenhaltungs-Schicht

Das DYMS wird ein webbasiertes System, da es ohne aufwendige Installationen von jedem Standort aus zu erreichen sein soll. Die Client-Schicht besteht somit aus einem Browser.

Dieser muss JavaScript fähig sein, da der GWT-Compiler neben HTML- und CSS-Dateien noch JavaScript-Dateien erzeugt. Um die Anwendungsdaten aus dem EJB-Server darzustellen, benutzt der Browser die Remotedienste eines GWT-Servers. Eine Beschreibung der Funktionsweise des GWT-RPC-Mechanismus ist im Abschnitt 2.3.2.2 erläutert.

Ein Web-Server bildet die Darstellungs-Schicht. Zum einen stellt er die HTML-, JavaScript- und CSS-Dateien für den Browser zur Verfügung und zum anderen beinhaltet sein Servlet-Container die GWT-Servlets. Die GWT-Servlets definieren den GWT-Server, als den Remotedienst, der vom Browser als GWT-RPC-Client aus aufgerufen werden kann. Die GWT-Servlets kommunizieren mit dem EJB-Server.

Die Anwendungs-Schicht enthält den EJB-Server sowie eine Konverter-Komponente. Der EJB-Server enthält neben dem Anwendungskern noch eine Autorisierungs-Komponente und eine Persistenz-Komponente.

Die vierte Schicht ist die Datenhaltungs-Schicht. Diese enthält den Datenbankserver. Um der Anforderung der Datensicherheit zu genügen, sollte es sich hierbei um ein RDBMS handeln, welches nicht ohne Autorisierung eine Manipulation oder Einsicht in die Daten erlaubt.

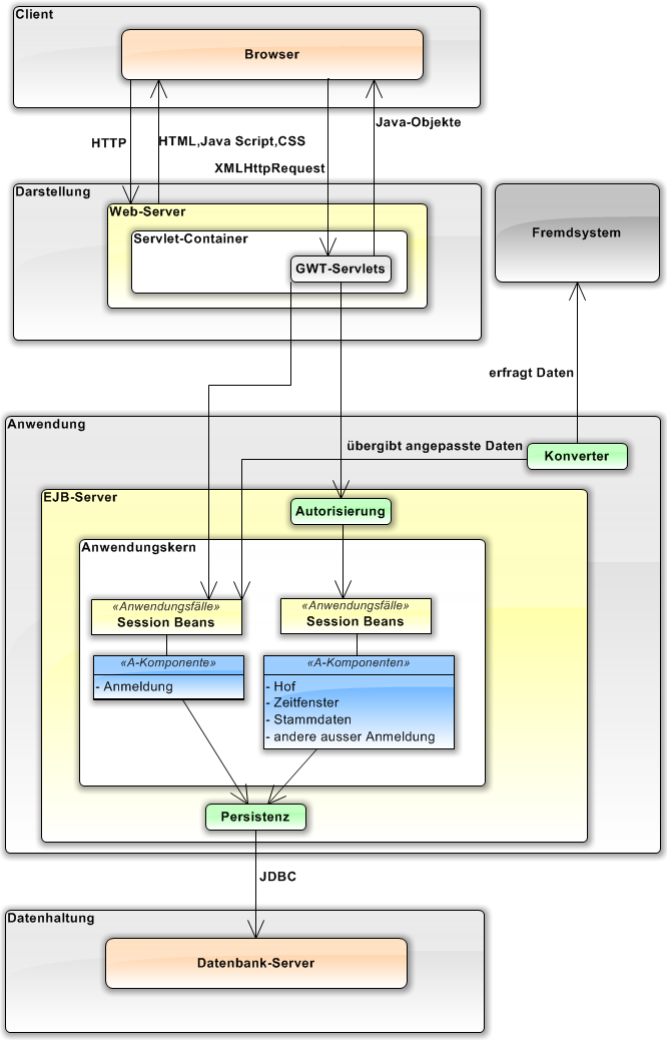

Abbildung 4.3: T-Architektur

4.2.1 T-Komponenten

Die im Abschnitt 4.2 angesprochenen T-Komponenten sollen nun näher erläutert werden.

Die Konverter-Komponente stellt individuelle Schnittstellen zu Fremdsystemen zur Verfügung. Das DYMS kann über den Konverter Daten bei den Fremdsystemen anfordern. Der Konverter sorgt dabei für die Konvertierung der Daten aus dem Fremdsystem in Formate, die für das Datenmodell des DYMS passend sind. Da die Schnittstellen individuell für jedes Fremdsystem entwickelt werden, kann nahezu jedes Fremdsystem angebunden werden. Dadurch wird der Anwendungsfall „Dateneingabe per ID" (siehe Abbildung 3.8) umgesetzt. Hierbei sollen Daten aus dem Speditionssystem Komalog in der Anmeldung zur Verfügung

stehen. Der Benutzer gibt eine ID ein. Das DYMS fragt Daten mit dieser ID beim Konverter ab. Der Konverter fragt sein Fremdsystem nach dieser ID und erhält im Erfolgsfall entsprechende Daten. Diese passt er dann an das DYMS-Datenmodell an und schickt sie als Resultat an die Anmeldung zurück. Dort dienen die Daten dann einer fehlerfreieren und schnelleren Anmeldung.

Die Autorisierungs-Komponente sorgt für die Zuteilung von Zugriffsrechten. Im DYMS müssen folgende Benutzergruppen existieren: Lagermitarbeiter, Lagermeister, interner Disponent und externer Disponent. Eine Benutzergruppe „Fahrer" bedarf es nicht, da diese Gruppe nur mit der Anmeldungs-Komponente arbeiten wird, und diese Komponente keinerlei Möglichkeiten hat, bestehende Daten zu ändern oder zu löschen. Die Benutzergruppe „Lagermitarbeiter" darf eine Übersicht über die Rampen und den Hof sehen, Fahrzeuge den Rampen zuordnen und wieder abmelden. Die Benutzergruppe „Lagermeister" erbt die Rechte der Benutzergruppe „Lagermitarbeiter" und darf zusätzlich noch alle Stammdaten verwalten und Zeitfenster löschen. Die Benutzergruppe „externer Disponent" darf Zeitfenster anlegen und ändern. Die Benutzergruppe „interner Disponent" erbt die Rechte der Benutzergruppe „externer Disponent" und darf zusätzlich noch eine Übersicht über den Hof und die Rampen sehen und die Stammdaten der Zeitfenster verwalten.

Die Persistenz-Komponente sorgt für die Kommunikation zwischen dem Anwendungskern und der internen Datenbank. Der Zugriff auf die interne Datenbank erfolgt hier mit JDBC. Der Anwendungskern soll nichts von der Datenbank wissen. Er soll lediglich mit Objekten arbeiten, die er liest, ändert, beschreibt oder löscht. Wie die Daten vom Objekt persitiert werden in der Datenbank ist Aufgabe der Persistenz-Komponente. Damit ist es möglich, dass der Anwendungskern mit jeder Datenbank arbeiten kann.

4.2.2 Paketdiagramm

Die Abbildung 4.4 zeigt die Aufteilung des Systems in Pakete mit den entsprechenden Klassen und Interfaces sowie die Beziehungen zueinander.

Das DYMS setzt sich in erster Ebene aus einem EJB-Paket und einem GWT-Paket zusammen.

Im EJB-Paket sind die Klassen und Interfaces enthalten, die für den Anwendungsserver notwendig sind. Das EJB-Paket besteht aus drei Subpaketen: dem EJB.Server, dem EJB.Server.Entitybeans und dem EJB.Server.SessionBeans. Das Paket „EJB.Server.SessionBeans" beinhaltet die Stateless- und Stateful Session Bean-Klassen. Im Paket „EJB.Server.Entitybeans" sind die EntityBean-Klassen enthalten, die für die Persistierung benötigt werden. Diese EntityBean-Klassen bilden Objekte, die die permanenten Daten der Anwendung enthalten. Diese Objekte stehen im gesamten System zur Verfügung und werden bspw. im GWT-Client dem Benutzer graphisch dargestellt. Das Paket „EJB.Server" definiert Interfaces mit dessen Hilfe ausserhalb des EJB-Paketes auf die Methoden der SessionBeans zugegriffen werden kann.

Das GWT-Paket definiert die Klassen und Interfaces, die für die Darstellung benötigt werden. Dieses Paket untergliedert sich wiederum in die zwei Hauptpakete „GWT.Server" und „GWT.Client". Das Paket „SmartGWT" ist ein externes Paket, welches die Bibliotheken enthält, um die SmartGWT-Features zu nutzen. „SmartGWT" wird vom Paket „GWT.Client" importiert, da es nur hier benötigt wird. Das Paket „GWT.Client" besteht aus der Klasse „DockYardGUI". Diese Klasse implementiert das Interface „EntryPoint", welches vom GWT zur Verfügung gestellt wird. Dieses Interface schreibt die Methode „onModuleLoad" vor, welches den Einstiegspunkt in die Applikation darstellt. Diese Methode ist vergleichbar mit der „main"-Methode einer Java-Applikation. Weiter enthält das Paket „GWT.Client" Interfaces, die Methoden definieren, die vom GWT-Server implementiert werden müssen. Diese Interfaces stellen die Kommunikationsbasis zwischen GWT-Client und GWT-Server dar. Die Implementierungen der GWT-Server-Methoden liegen im Paket „GWT.Server". Die Namensdefinition für die beiden GWT-Pakete „GWT.Client" und „GWT.Server" ist nicht beliebig, denn das GWT schreibt vor, dass Client-Klassen in einem Paket enthalten sein müssen, welches auf „client" endet, genauso, wie die GWT-Server-Klassen in einem Paket mit der Endung „server". Die Methoden im Paket „GWT.Server" nutzen die im Paket „EJB-Server" angebotenen Interfaces, um auf die SessionBeans zuzugreifen.

Eine exemplarische Erläuterung zur Nutzung eines EntityBean-Objektes erfolgt im Abschnitt 5.3.

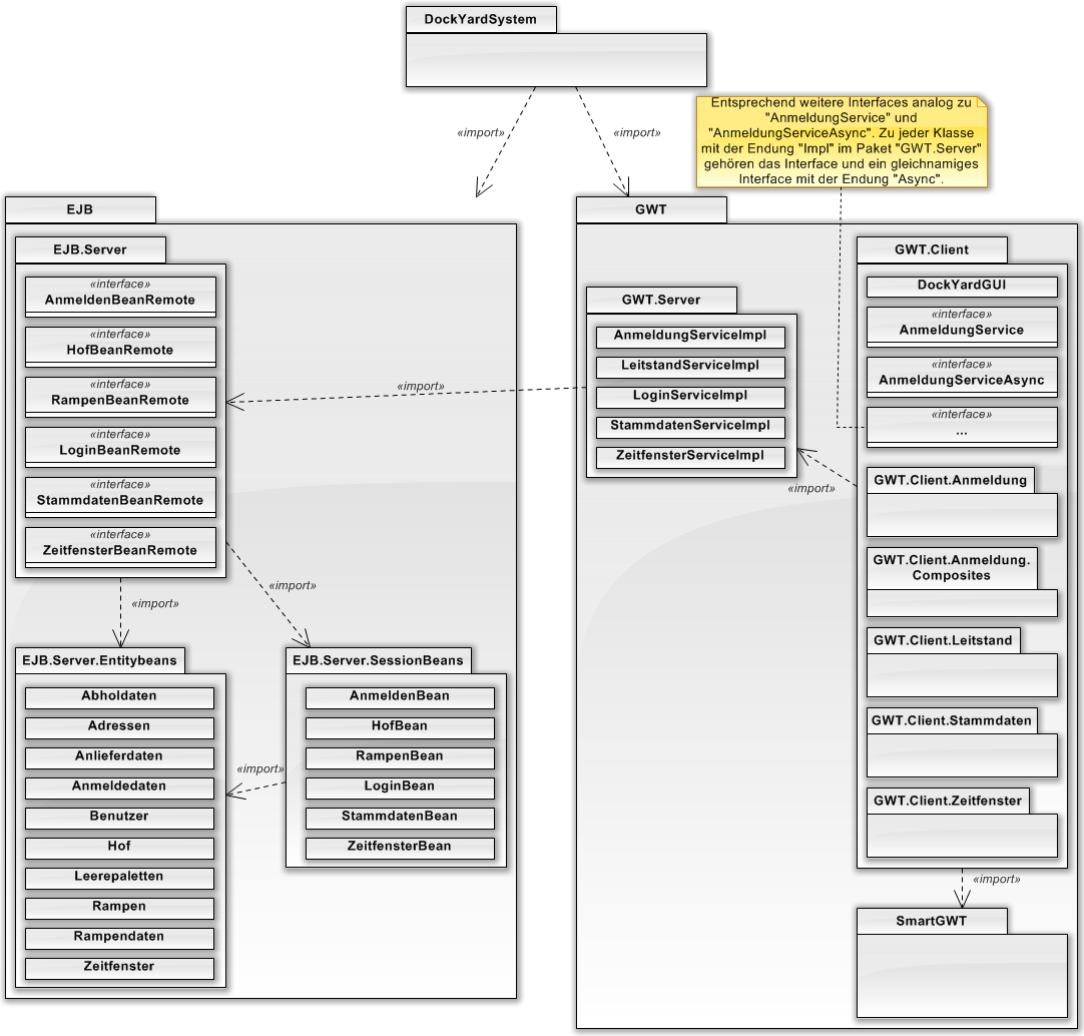

Abbildung 4.4: Paketdiagramm

4.3 TI-Architektur

Die TI-Architektur beschreibt die Architektur der technischen Infrastruktur. Hier werden die genutzten Techniken (siehe Kapitel 2.3) in die Architektur des Systems eingeordnet. Weiterhin wird die Kommunikation zwischen den Techniken beschrieben. Abbildung 4.5 zeigt die TI-Architektur mit den dazugehörigen Techniken.

Um die fachliche Anforderung der standortübergreifenden Nutzung zu realisieren, ist das System als Webapplikation konzipiert. Für die Realisierung des Prototypen werden folgende Technologien mit den jeweils angegebene Entwicklungsständen genutzt. Für die Erstellung der Benutzungsoberfläche kommt GWT in der Version 2.2.0 zum Einsatz. Zur Erhöhung der

Features der Weboberfläche wird GWT mit smartGWT in der Version 2.5 ergänzt. Als Web-Server und Servlet-Container ist ein Apache Tomcat in der Version 7.0.19 zum Einsatz gekommen. Als Anwendungsserver wird das Produkt JBoss der Firma Red Hat in der Version 6.0.0 eingesetzt. Für die Persistenz wird eine Datenbank der PostgrsSQL Global Development Group in der Version 9.0 zuständig sein. Um als Benutzer mit dem System zu arbeiten sind folgende Browser geeignet: Internet Explorer, Chrome, Firefox, Safari und Opera. Die Einschränkung auf bestimmte Browser wird durch das GWT bestimmt. Der GWT-Compiler erstellt JavaScript und HTML Code für jeden dieser Browser mit seinen speziellen Eigenheiten. Bei anderen Browsern besteht keine Aussage bezüglich einer fehlerfreien Ausführung des Codes. Weiterhin muss die Ausführung von JavaScript erlaubt sein.

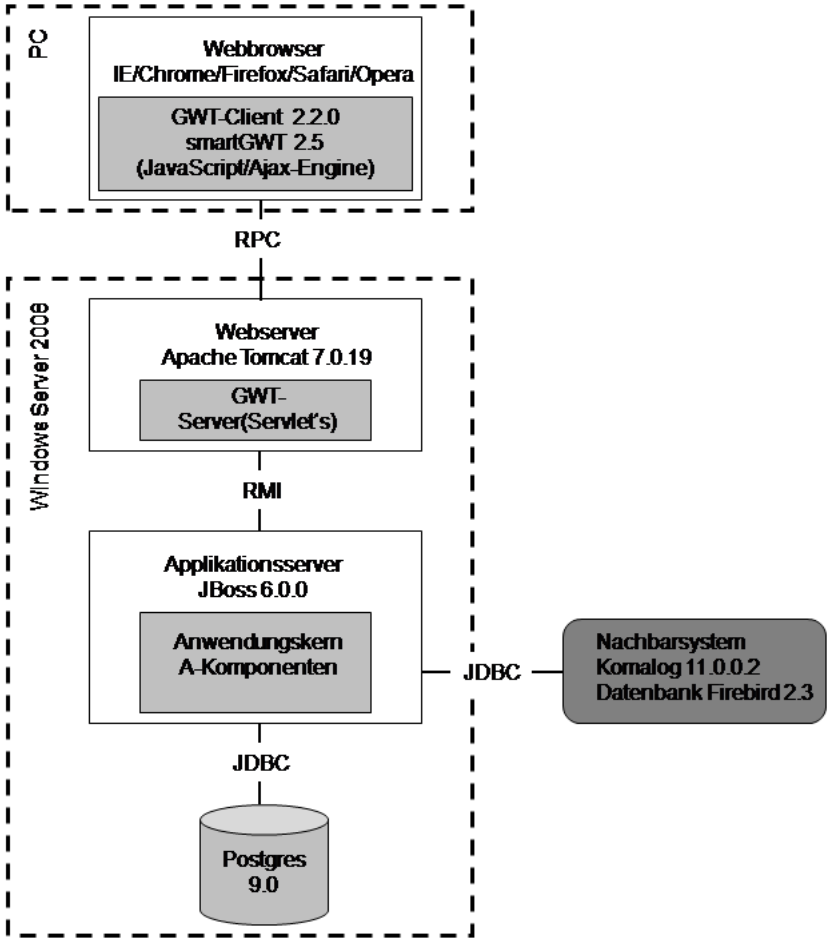

Abbildung 4.5: TI-Architektur

Kapitel 5

Prototyp

Die in Kapitel 4 entworfene Systemarchitektur soll nun in einem Prototypen umgesetzt werden. Dabei werden nur die Funktionalitäten realisiert, die die wichtigsten Anforderungen an das Neusystem erfüllen. Dies dient dann dem Beweis der Machbarkeit eines Dock und Yard Management Systems mit den entsprechenden Anforderungen.

5.1 Ziel

Ziel des Prototypen ist zu prüfen, ob zum einen sich die fachlichen Anforderungen erfüllen lassen und zum anderen, ob die erwählten Techniken sich dafür eignen. Weiterhin sollen die eklatantesten Kritikpunkte des Altsystems sowohl in fachlicher Sicht als auch in technischer Sicht versucht werden zu elemenieren. Weiterhin soll fachlich geprüft werden, ob es eventuelle Fehler im Konzept gibt, und ob diese beseitigt werden können. Technisch wird geprüft, ob die Techniken praktisch so zusammenarbeiten, wie es theoretisch erdacht worden ist. Damit der Benutzer sich einen Eindruck über die Leistungsfähigkeit des Systems verschaffen kann, wird der Prototyp soweit entwickelt, dass ein kompletter Workflow von der Anmeldung bis zur Abmeldung vorgenommen werden kann. Die Anmeldung ist hierbei von Bedeutung, da hier Daten erfasst werden können, mit denen dann im weiteren System gearbeitet werden kann.

Für den Prototypen ist nicht an allen notwendigen Stellen ein Exceptionhandling vorgenommen worden. Auch werden Benutzereingaben nicht verifiziert.

5.2 GWT-Module

Zur Konfiguration der GWT-Applikation bzw. des GWT-Compilers existieren sog. Module. Dies sind Dateien im xml-Format. In dieser Datei wird unteranderem der Einstiegspunkt der Applikation, also die Klasse, die das EntryPoint-Interface implementiert, oder Informationen

über vererbte Module angegeben. Diese Informationen geben dem GWT-Compiler die Anweisungen, welche Module noch zusätzlich in die Applikation einzubinden sind.

Für den Prototypen existiert das Modul „DockYardGUI.gwt.xml". Die Datei enthält die Definition der EntryPoint-Klasse sowie einige Angaben zu vererbten Modulen. Zeile 4 im Modul „DockYardGUI.gwt.xml" (siehe Algorithmus 5.1) enthält bspw. die Angabe, an welcher Stelle die Klassen für SmartGWT zu finden sind.

Besonders soll allerdings auf die Zeile 5 im Modul „DockYardGUI.gwt.xml" hingewiesen werden. An dieser Stelle werden Angaben zu einem Modul namens „Core.gwt.xml" definiert, welches sich im EJB-Paket des Projektes befindet. Der Inhalt dieses Moduls ist im Algorithmus 5.2 zu entnehmen. Das Modul „Core.gwt.xml" definiert dabei von seinem Standort im Projektverzeichnis („EJB.Server") einen relativen Pfad zum Paket „Entitybeans" („EJB.Server.Entitybeans"). In diesem Paket sind die Entitybeans der Applikation zu finden. Mit dieser Angabe ist es dem GWT-Compiler nun möglich, die Klassen der EntityBeans zu berücksichtigen. Würde diese Angabe fehlen, so könnten die EntityBeans nicht in der GWT-Applikation verwendet werden.

Algorithm 5.1 Modul DockYardGUI.gwt.xml

```
1 <module>
2         <inherits name="com.google.gwt.user.User"/>
3         <inherits name="com.google.gwt.user.theme.standard.Standard"/>
4         <inherits name="com.smartgwt.SmartGwt"/>
5         <inherits name="com.dockyard.ejb.server.Core"/>
6         <entry-point class="com.dockyard.gwt.client.DockYardGUI"/>
7 </module>
```

Algorithm 5.2 Modul Core.gwt.xml

```
1 <module rename-to="com.dockyard.ejb.server.Core">
2         <source path="entitybeans"/>
3 </module>
```

5.3 Nutzung von EntityBeans im GWT-Client

Im vorherigen Abschnitt (5.2) ist eine Voraussetzung erläutert worden, um EntityBeans in GWT nutzen zu können. Die Objekte, die zwischen dem GWT-Server und dem GWT-Client ausgetauscht werden, müssen serialisierbar sein. Eine funktionale Beschreibung des GWT-RPC ist im Abschnitt 2.3.2.2 zu finden. Somit ist eine weitere Voraussetzung, dass die EntityBeans die Schnittstelle „Serializable" implementieren. Unabhängig vom GWT wird dies emp-

fohlen, damit sich die EntityBeans besser vom EntityManager verwalten lassen. Im Prototypen implementieren alle EntityBeans diese Schnittstelle „Serializable". In diesem Abschnitt sollen nun die nötigen Interfaces und Klassen exemplarisch erläutert werden, die benötigt werden, um ein Objekt einer EntityBean vom Anwendungsserver bis in den GWT-Client zu transportieren, um dort genutzt zu werden.

Als Beispiel soll hier die Implementierung der T-Komponente für die Autorisierung dienen.

Zunächst findet auf der GWT-Seite eine Kommunikation zwischen dem GWT-Client und dem GWT-Server statt, das sog. GWT-RPC. Für die Implementierung sind zwei Interfaces und eine Klasse notwendig. Der Algortihmus 5.3 zeigt den Quellcode für das Interface „LoginService" und der Algorithmus 5.4 zeigt den Quellcode für das Interface „LoginServiceAsync". Beide Interfaces werden auf der Client-Seite benötigt. Gem. GWT Vorgaben müssen sich diese Interfaces in einem Projektpaket mit der Endung „client" befinden. Der Algorithmus 5.5 zeigt die Klasse „LoginServiceImpl", die auf der Server-Seite eingesetzt wird. Diese Klasse muss sich in einem Projektpaket mit der Endung „server" befinden.

Das Interface „LoginService" definiert drei Methoden. Die Methode „login" prüft die Logindaten und meldet den Benutzer in Abhängigkeit des Prüfergebnisses an oder nicht. Als Parameter erwartet diese Methode einen Benutzernamen und ein Passwort als String. Als Rückgabewert liefert sie eine HashMap mit einem Status-String und einem Objekt der EntityBean-Klasse „Benutzer". Die Mehtode „getLogin" liefert den aktuell angemeldeten Benutzer auch als Objekt der EntityBean-Klasse „Benutzer". Die dritte Methode meldet den aktuellen Benutzer vom System ab.

Das Interface „LoginServiceAsync" erweitert die drei Methoden des Interfaces „LoginService" jeweils um einen Parameter mit dem Interface „AsyncCallback". Die Kommunikation zwischen GWT-Client und GWT-Server verläuft asynchron, so dass der Client nach Aufruf einer Server-Methode nicht blockiert wird. Dieses Interface definiert zwei Methoden, sog. Callback-Methoden, „onFailure" und „onSuccess". Nach Abschluss der Verarbeitung der Anfrage im Server, ruft dieser dann im Erfolgsfall die „onSuccess"-Methode auf, bzw. die „onFailure"-Methode, wenn etwas während der Verarbeitung im Server fehlgeschlagen ist.

Die Klasse „LoginServiceImpl" ist ein GWT-Servlet, und wird auf dem GWT-Server ausgeführt. Es implementiert die Methoden des Interfaces „LoginService". Exemplarisch soll hier die Methode „login" näher erläutert werden. Diese Methode verbindet sich mit dem Anwendungsserver und ruft auf diesem mit Hilfe des Interfaces „LoginBeanRemote" die Session Bean „LoginBean" auf (Algorithmus 5.5 Zeile 21-31). Diese Session Bean bietet über sein Interface auch eine login-Methode an. Die Methode im GWT-Servlet ruft nun die login-Methode in der Session Bean auf und übergibt die Parameterdaten (Algorithmus 5.5 Zeile 34). In der LoginBean findet nun die Prüfung des Benutzernamens und dem Passwort statt. Als Rückgabewert liefert das LoginBean eine HashMap mit einem Statusstring als Schlüssel und einem Objekt der EntityBean-Klasse „Benutzer". Der Statusstring beinhaltet das Ergebnis der Prüfung. Das GWT-Servlet erhält die HashMap und liefert diese wiederum als Rückgabewert zurück.

Das LoginBean ist eine StatefulSessionBean, da sie die Benutzerinformationen für die gesamte Dauer der Sitzung speichern muss. Damit das GWT-Servlet bei jedem Aufruf der LoginBean immer das zu seiner Sitzung gehörende nutzt, hat es die Möglichkeit, die Instanz der LoginBean in seiner Session zu speichern. Dies funktioniert, weil das GWT-Servlet „LoginServiceImpl" das RemoteServiceServlet erweitert, welches seinerseits wiederum das HTTPServlet erweitert, welches wiederum den Zugriff auf die HTTPSession erlaubt. Im Algorithmus 5.5 Zeile 32 wird das LoginBean in der Session gespeichert, in Zeile 17 wird es aus der Session gelesen.

Die Algorithmen 5.6 und 5.7 zeigen den Aufruf der vom GWT-Server angebotenen Methoden. Beide implementieren die vom AsyncCallback-Interface vorgeschriebenen Callback-Methoden „onSuccess" und „onFailure". Der Algorithmus 5.6 zeigt den Aufruf der login-Methode, die als Beispiel in diesem Abschnitt diente. Ist während der Verarbeitung im GWT-Server kein Fehler aufgetreten, so wird von diesem die „onSuccess"-Methode aufgerufen. Diese prüft nun im Client den Rückgabewert, speziell den Statusstring, der vom LoginBean gesetzt worden ist, und verfährt entsprechend des Wertes. Im Algorithmus 5.7 ist zu sehen, dass der GWT-Client in der „onSuccess"-Methode ein Objekt der EntityBean-Klasse verarbeitet. Dieses Objekt ist vom Anwendungsserver mit Daten gefüllt worden und steht nun dem GWT-Client zur Verfügung, der den Vor- und Nachnamen des Benutzers sowie den Firmennamen ausliest und in der Benutzungsoberfläche darstellt.

Algorithm 5.3 Interface „LoginService”

```
1  @RemoteServiceRelativePath("LoginService")
2  public interface LoginService extends RemoteService
3  {
4          public HashMap<String,Benutzer> login(String benutzer, String passwort);
5          public Benutzer getLogin();
6          public void abmelden();
7          /**
8           * Utility class for simplifying access to the instance
9           * of async service.
10          */
11         public static class Util {
12                 private static LoginServiceAsync instance;
13                 public static LoginServiceAsync getInstance(){
14                         if (instance == null) {
15                                 instance = GWT.create(LoginService.class);
16                         }
17                         return instance;
18                 }
19         }
20 }
```

Algorithm 5.4 Interface LoginServiceAsync

```
1  public interface LoginServiceAsync
2  {
3          public void login(String benutzer, String passwort,
4          AsyncCallback<HashMap<String,Benutzer>> callback);
5          public void getLogin(AsyncCallback<Benutzer> callback);
6          public void abmelden(AsyncCallback<Void> callback);
7  }
```

Algorithm 5.5 Klasse LoginServiceImpl

```
1  public class LoginServiceImpl extends RemoteServiceServlet
2          implements LoginService
3  {
4    private static final long serialVersionUID = 1L;
5    /**
6     * Returns the current session
7     *
8     * @return  The current Session
9     */
10   private HttpSession getSession() {
11    // Get the current request and then return its session
12    return this.getThreadLocalRequest().getSession();
13   }
14   @Override
15   public HashMap<String,Benutzer> login(String benutzer, String passwort) {
16     try{
17       LoginBeanRemote lbr = (LoginBeanRemote)this.getSession().
18                                                   getAttribute("LoginBean");
19
20       if (lbr == null){
21         Properties p = new Properties();
22         p.put(Context.INITIAL_CONTEXT_FACTORY,
23               "org.jnp.interfaces.NamingContextFactory");
24         p.put(Context.URL_PKG_PREFIXES,
25               "org.jboss.naming:org.jnp.interfaces");
26         p.put(Context.PROVIDER_URL,
27               "jnp://localhost:1099");
28         Context ctx = new InitialContext(p);
29         Object ref = ctx.lookup("LoginBean/remote");
30         lbr = (LoginBeanRemote) PortableRemoteObject.
31                                 narrow(ref, LoginBeanRemote.class);
32         this.getSession().setAttribute("LoginBean", lbr);
33       }
34       return lbr.login(benutzer, passwort);
35     } catch (BenutzerException e) {
36       e.printStackTrace();
37     } catch (NamingException e) {
38       e.printStackTrace();
39     }
40     return null;
41   }
42 ...
43 }
```

Algorithm 5.6 Benutzung der login-Methode im Client

```
1  LoginService.Util.getInstance().login(e.getBenutzer(),e.getPasswort(),
2  new AsyncCallback<HashMap<String,Benutzer>>()
3  {
4    @Override
5    public void onFailure(Throwable caught) {
6      Window.alert("Error: login");
7    }
8    @Override
9    public void onSuccess(HashMap<String,Benutzer> result) {
10     if (result.containsKey("ok")){
11       leitstandGui = getLeitstandGui();
12       rootPanel.add(leitstandGui);
13       rootPanel.remove(loginGui);
14     }else{
15       Window.alert("Benutzername oder Passwort falsch!!!");
16     }
17   }
18 });
```

Algorithm 5.7 Benutzung der getLogin-Methode im Client

```
1  LoginService.Util.getInstance().getLogin(
2  new AsyncCallback<Benutzer>()
3  {
4    @Override
5    public void onFailure(Throwable caught) {
6      SC.warn("Error: getLogin");
7    }
8    @Override
9    public void onSuccess(Benutzer result) {
10     if (result != null){
11       lbl_LoginFirma.setContents(result.getFirma().getName());
12       lbl_LoginUser.setContents(result.getVorname()+" "+result.getNachname());
13     }else{
14       lbl_LoginFirma.setContents("-");
15       lbl_LoginUser.setContents("-");
16     }
17   }
18 });
```

5.4 Ausgewählte Komponentenimplementierungen

In diesem Abschnitt werden diejenigen Komponenten beschrieben, die für einen kompletten Workflow nötig sind. So ist die T-Komponente „Autorisierung" nicht vollständig implementiert. Es können Benutzerdaten angelegt, gelöscht oder geändert werden, um sich mit diesen anzumelden. Eine Gruppierung in verschiedene Benutzerrechtestufen mit entsprechender Freischaltung von Komponenten ist nicht erfolgt, da dies für den explorativen Workflow nicht relevant ist.

5.4.1 A-Komponente Stammdaten

Im Prototypen sind als Stammdaten realisiert Benutzerdaten, Adressendaten und Rampendaten. Für die Benutzer- und Adressendaten sind die Anwendungsfälle „Daten neu anlegen", „Daten ändern" und „Daten löschen" implementiert worden. Dazu dient im Anwendungsserver die Klasse „StammdatenBean", die die entsprechenden Methoden anbietet.

Benutzungsoberflächen

Exemplarisch ist hier die Adressenbenutzungsoberfläche (siehe Abbildung 5.1) dargestellt. Der Benutzer hat hier die Möglichkeit sog. CRUD[7]-Operationen direkt auf die Adressendaten auszuführen. Die Benutzerbenutzungsoberfläche ist analog zur Adressenbenutzungsoberfläche aufgebaut. Um ein wenig die Möglichkeiten einer Rich Internet Application zu demonstrieren, wurde im Prototypen eine Verwendungsmöglichkeit der Adressdaten in den Benutzerdaten geschaffen, die sonst üblicherweise aus reinen Desktopapplikationen bekannt sind. Legt man bspw. einen neuen Benutzer an, so hat man die Möglichkeit auf einen Button „Adressen" zu klicken. Dieses Ereignis öffnet ein modales Dialogfenster (siehe Abbildung 5.2), in welchem der Benutzer nicht nur eine Adresse auswählen kann, sondern er hat die Möglichkeit auch in diesem Dialogfenster alle CRUD-Operationen auf den Adressdaten auszuführen.

[7] CRUD steht für Create Read Update Delete.

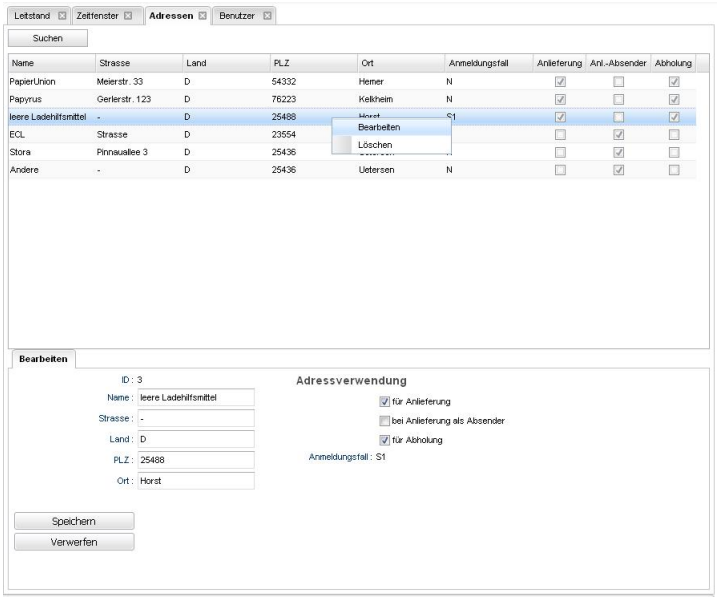

Abbildung 5.1: Adressenbenutzungsoberfläche

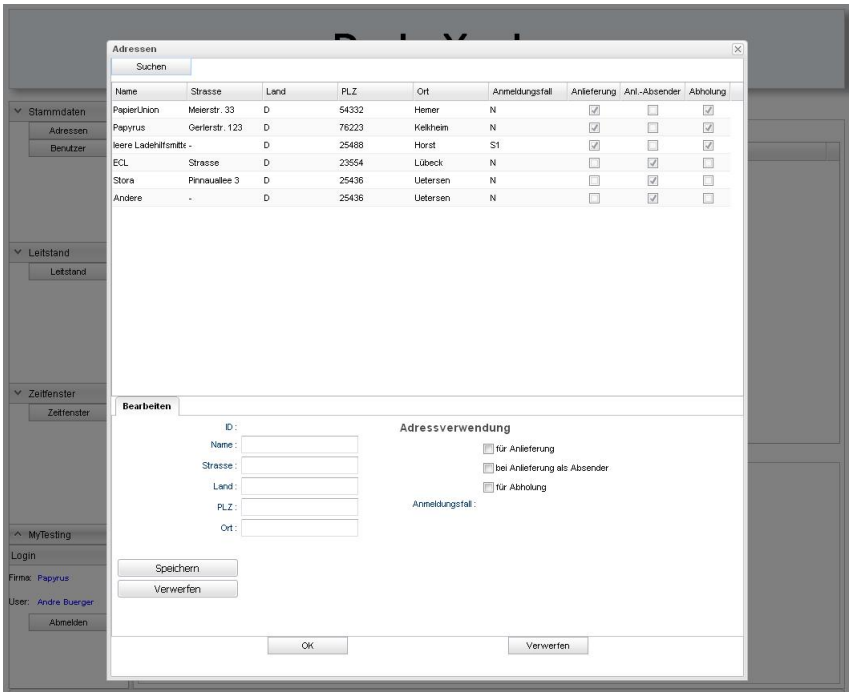

Abbildung 5.2: Benutzerbenutzungsoberfläche mit geöffnetem Unterfenster für Adressenauswahl

5.4.2 A-Komponenten Anmeldedaten und Aktivitäten

Die Anmeldedatenkomponente und die Aktivitätenkomponente sind in der Stateless Session Bean „AnmeldenBean" umgesetzt. Diese bietet Methoden an, um ein Objekt der EntityBean-Klasse „Anmeldedaten" zu speichern. Des weiteren können Anmeldedaten abgerufen werden, sowie zu jedem Anmeldedatensatz die jeweiligen Aktivitäten. Methoden zum Speichern von Aktivitäten sind nicht vorhanden, da die EntityBean-Klasse „Anmeldedaten" die Aktivitäten als Listen beinhaltet. Über die Annotation „@OneToMany" werden die Aktivitäten dann beim Persistieren der Anmeldedaten vom EntityManager in die Datenbank geschrieben, bzw. gelesen. Da das Füllen der Anmeldedaten und der dazugehörigen Aktivitäten in mehreren Schritten erfolgt (Fragenkatalog), muss der jeweilige Stand des Objektes nach jedem Schritt gesichert werden. Dazu bietet sich neben der Stateful Session Bean die Möglichkeit an, diese Sicherung im GWT in einer globalen Klassenvariablen, die sich in einer Klasse des GWT-Clients befindet, vorzunehmen. Diese Klassenvariable ist vom EntityBean-Klassentyp „Anmeldedaten". Bei jedem Schritt, nachdem Daten vom Benutzer eingegeben worden sind, werden diese Daten nicht bis in den Anwendungsserver transportiert, um dort in einer Stateful Session Bean gespeichert zu werden, sondern die Daten verbleiben im Paket des GWT-Client und werden dort in der globalen Klassenvariablen gespeichert. Erst nachdem alle Daten in der Klassenvariablen vorhanden sind, wird das Objekt aus der Klassenvariablen an den Anwendungsserver übergeben, um persistiert zu werden.

Das GWT bietet die Möglichkeit dieser globalen Klassenvariablen, und ein Prototyp eignet sich gut, um alternative Techniken zu testen. Der Vorteil könnte sich in der Performance ausmachen, da die Daten hier nicht erst für den Transport mittels eines Remote-Interfaces verpackt werden müssen, um zum Anwendungsserver geschickt zu werden. Dieser Performancevorteil könnte sich aber wieder relativieren, wenn es sich bei dem Interface über ein Lokales Interface handelt, denn dann werden die Daten nicht extra für den Transport verpackt. Daher bietet unteranderem EJB diese Lokale Interface Technik. Es wurden allerdings keine Performancetests durchgeführt. Daher kann hierüber keine Aussage getroffen werden. Es sollte lediglich getestet werden, ob es möglich ist im GWT mit globalen Klassenvariablen zu arbeiten. Der Nachteil ist, dass gem. der 4-Schichten-Architektur hier eine Vermischung zwischen Darstellungsschicht und Anwendungsschicht erfolgt. Dadurch verliert das System an Wartbarkeit. Fazit ist also, dass die Nutzung einer globalen Klassenvariablen zwar Performancevorteile bringen könnte, dadurch aber die Architektur leidet, da hier Logik in die Darstellungsschicht gelangt.

Benutzungsoberflächen

Mit der Benutzungsoberfläche für die Anmeldedaten- und Aktivitätenkomponente werden zum grössten Teil Fahrer arbeiten, um sich anzumelden. Bei diesen Fahrern ist widerum ein bestimmter Anteil, die das erste Mal mit dem System arbeiten. Daher ist eine Anforderung

an diese Oberfläche, dass sie möglichst einfach und intuitiv zu bedienen sein muss. Diese Anforderung wurde realisiert, indem der Benutzer durch einen Fragenkatalog geführt wird, bei dem sich pro Seite so wenig Fragen wie möglich befinden, auf die er sich konzentrieren kann. Dadurch soll erreicht werden, dass auch neue Benutzer ohne Einführung sofort mit dem System arbeiten können. Im Prototypen sind dafür Composites verwendet worden, wobei jedes Composite eine Seite im Fragenkatalog darstellt.

Für die Gestaltung der Composites dienten die für die Interviews entworfenen Mock-Ups. Ein Aktivitätsdiagramm stellte die Aufrufreihenfolge der Composites dar. Bei ersten Tests der Benutzer mit dem Prototypen zeigte sich jedoch dann ein Fehler im Konzept. Das erste Konzept sah im Aktivitätsdiagramm bei der Auswahl der Aktivität eine ternäre Entscheidung vor (siehe Abbildung 3.7). Der Benutzer musste sich entscheiden, ob er Anliefern, Abholen oder beides machen möchte. Benutzer, die nicht an den Interviews beteiligt waren, hatten bei dieser Entscheidung oft Fragen. Dies bedeutete für das Konzept, dass diese tenäre Fragestellung nicht intuitiv war. Somit wurde das Aktivitätsdiagramm und der Prototyp geändert. Aus der tenären Frage wurden zwei binäre Fragen. Die Abbildung 5.3 zeigt das geänderte Aktivitätsdiagramm, welches nur noch binäre Fragestellungen enthält.

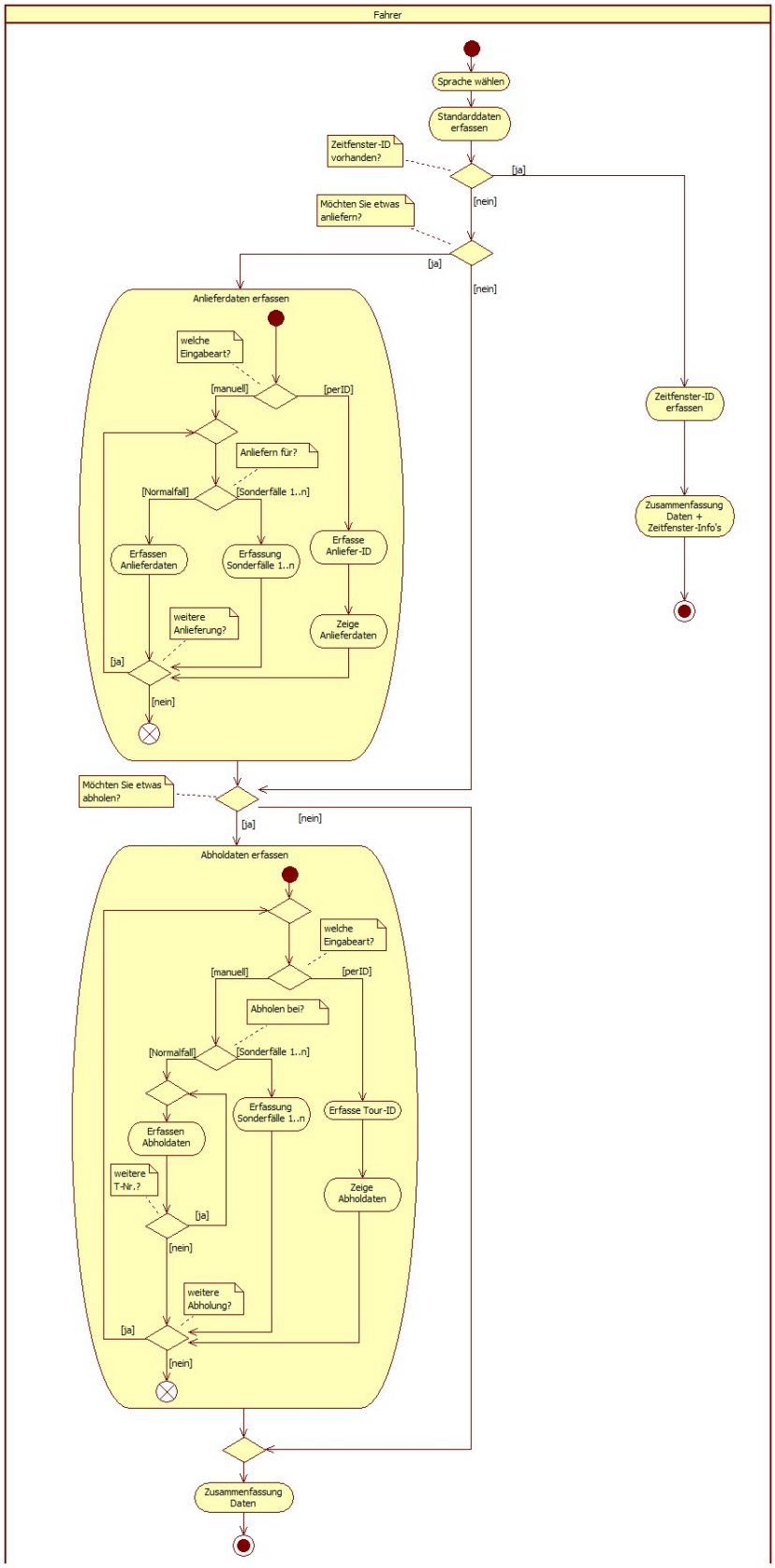

Abbildung 5.3: Geändertes Aktivitätsdiagramm in reine binäre Fragestellung

Die Abbildungen 5.4 und 5.5 zeigen exemplarisch zwei Composites, dabei ist das Composite auf der Abbildung 5.5 exemplarisch für die binäre Fragestellungen.

Abbildung 5.4: Anmeldebenutzungsoberfläche: Allgemeine Anmeldedaten

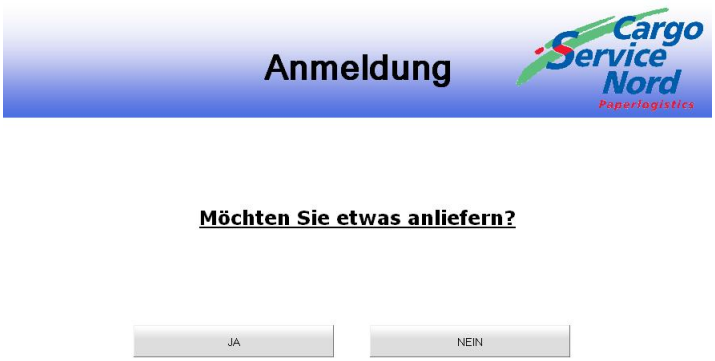

Abbildung 5.5: Anmeldebenutzungsoberfläche: Bsp. binäre Auswahl

5.4.3 A-Komponenten Hof und Rampe

Die Hofkomponente verwaltet alle Fahrzeuge, die sich aktuell auf dem Hof befinden und die sich in der Vergangenheit auf dem Hof befunden haben. Meldet sich ein Fahrzeug erfolgreich an, wird es automatisch der Hofkomponente zugeordnet, da es sich auf dem Hof befindet. Beim Abmelden des Fahrzeuges wird dieses nicht aus der Hofkomponente gelöscht, sondern erhält einen Abmeldezeitstempel. Im Prototypen ist die Funktionalität zum Abrufen aller Fahrzeuge, die sich auf dem Hof befinden, realisiert. Weiterhin wird in der Hofkomponente zu jedem Fahrzeug eine Wartezeit und, ob für das Fahrzeug ein Zeitfenster gebucht worden ist, verwaltet. Die Wartezeit errechnet sich aus dem Zeitstempel der Anmeldung und der aktuellen Zeit. Die Realisierung ist komplett mit Funktionen aus der Datenbank umgesetzt worden. Die Wartezeitberechnung wird vom System automatisch in bestimmten Intervallen ausgeführt, so dass ohne Benutzerinteraktion immer ein aktueller Stand der Wartezeiten im System vorliegt und angezeigt werden kann.

Die Rampenkomponente verwaltet die Rampen. Eine Rampe hat immer einen definierten Zustand, der aber geändert werden kann. Der Zustand kann „frei", „gesperrt" oder „reserviert" sein. Weiterhin ist die Rampe in die zwei Bereiche „Aktiv" und „Wartezone" unterteilt. Ein Fahrzeug im Aktivbereich wird gerade an der Rampe abgefertigt. Das Fahrzeug in der Wartezone steht als nächstes für diese Rampe zur Abfertigung an, sobald das Fahrzeug aus dem Aktivbereich die Rampe verlässt. Es können also immer zwei Fahrzeuge einer Rampe zugewiesen werden. Wobei die Zuweisung in die Bereiche systemseitig gesteuert wird, in Abhängigkeit der Zuweisungsreihenfolge. Ist der Aktivbereich einer Rampe frei, so wird das zugewiesene Fahrzeug dem Aktivbereich zugeordnet. Wird dieser Rampe nun ein weiteres Fahrzeug zugewiesen, so wird dieses Fahrzeug automatisch in den Wartezonenbereich eingeordnet. Wird nun das Fahrzeug aus dem Aktivbereich abgemeldet, so wird systemseitig automatisch das Fahrzeug aus dem Wartezonenbereich in den Aktivbereich verschoben. Eine Funktionalität zur Berechnung der Wartezeit ist analog zur Wartezeitberechnung der Hofkomponente auch in dieser Komponete umgesetzt. Hier berechnet sich die Wartezeit aus dem Zeitpunkt, an dem das Fahrzeug der Rampe zugewiesen worden ist, unabhängig ob in den Aktivbereich oder die Wartezone.

Die genannten Funktionalitäten beider Komponenten sind im Prototypen realisiert worden, um den Informationsgehalt des Systems zu zeigen und die Unterstützung des Benutzers durch eine gewisse Systemintelligenz. Gerade der Punkt der Wartezeitberechnung ist in einem DYMS von Bedeutung, da es eine fundamentale Aufgabe eines solchen Systems ist, die Wartezeiten zu optimieren.

Benutzungsoberfläche

Die Abbildung 5.6 zeigt die Benutzungsoberfläche für die beiden Komponenten Hof und Rampe. Die gesamte Benutzungsoberfläche wurde so gestaltet, dass sie sich bei Vollbild

automatisch der Bildschirmgrösse anpasst. Weiterhin sind die einzelnen Bereiche in ihrer Grösse anpassbar, bzw. können ausgeblendet werden. Somit kann bei grossen Lagerhallen mit vielen Rampen ein Monitor mit grösserer Auflösung mehr Übersichlichkeit schaffen. In der Tabelle „Parkplatz” wie auch in der Tabelle „Rampen” wurde die sog. „Nested Grid”-Technik verwendet. Dies bedeutet, jede Tabellenzeile lässt sich aufklappen und enthält weitere Untertabellen, die dann ausführliche Detailinformationen zu dem jeweiligen Fahrzeug enthalten, wie bspw. Abholdaten oder Anlieferdaten. Weiterhin ist in den beiden Tabellen die Wartezeit, in Abhängigkeit der Grösse, farblich hinterlegt. Ist die Wartezeit kleiner 60 Minuten gilt die Farbe grün, zwischen 61 und 180 Minuten die Farbe orange, und mehr als 180 Minuten die Farbe rot. Die Verwaltung des Zeitfensters wird auch farblich kenntlich gemacht, indem zum einen das Feld zum Aufklappen der eingebetteten Tabellen rot unterlegt wird, und zum anderen eine Information zum gebuchten Zeitraum dargestellt wird. Das Zuweisen von Fahrzeugen einer Rampe ist im Prototypen mittels der „Drag and Drop”-Technik realisiert worden, indem das Fahrzeug aus der „Parkplatz”-Tabelle auf eine freie Rampe mit der Maus gezogen wird.

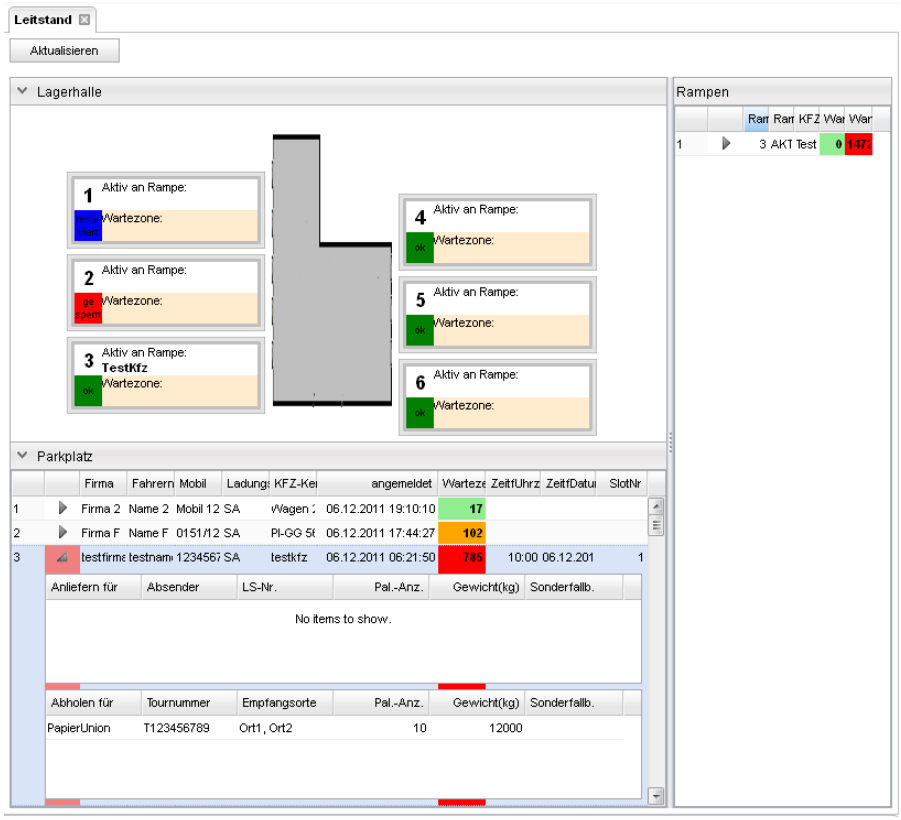

Abbildung 5.6: Leitstandbenutzungsoberfläche

5.4.4 A-Komponente Zeitfenster

Die Zeitfensterkomponente verwaltet Zeitfenster. Im Prototypen können Zeitfenster gebucht und gelöscht werden. Ein Zeitfenster setzt sich aus einer Zeitfensternummer und einem Zeitbereich zusammen. Für den Prototypen sind für jeden Tag sechs Zeitfenster mit jeweils drei Zeitbereichen implementiert worden. Die Zeitbereiche haben jeweils eine Grösse von zwei Stunden. Sowohl die Anzahl der Zeitfenster und Zeitbereiche, als auch die Grösse der Zeitbereiche müssen für den Realeinsatz noch angepasst werden. Im Prototypen kann ein Zeitfenster vollständig mit Daten gebucht werden. Für jedes Zeitfenster wird eine eindeutige ID angelegt. Die Anmeldedatenkomponente bietet im Prototypen die Möglichkeit sich mit einem Zeitfenster mit Hilfe der Zeitfenster-ID anzumelden. Eine Anmeldung mit Zeitfenster-ID verkürzt den Fragenkatalog und somit die Anmeldedauer, da ein grosser Teil der Daten zu den Aktivitäten bereits bei der Zeitfensterbuchung erfasst worden sind. Ein Zeitfenster kann auch wieder gelöscht werden. Dadurch wird das belegte Zeitfenster wieder frei und buchbar. Im Prototypen wird noch nicht zwischen den einzelnen Benutzerrechten unterschieden. Im Realeinsatz dürfte bspw. ein normaler Benutzer keine Zeitfenster löschen dürfen.

Benutzungsoberfläche

Die Abbildung 5.7 zeigt die Benutzeroberfläche für die Zeitfensterkomponente. Im oberen Bereich befindet sich ein Kalender, mit dessen Hilfe der gewünschte Tag ausgewählt wird. Im unteren Bereich wird dann für den entsprechenden Tag eine Tabelle mit den Zeitfenstern dargestellt. Rote Zeitfenster sind belegt, grüne sind frei. Bei den belegten Zeitfenstern wird der Firmenname der Firma, die das Zeitfenster gebucht hat, angezeigt. Mittels eines rechten Mausklicks auf einem Zeitfenster können die Funktionen „buchen” oder „löschen” ausgeführt werden.

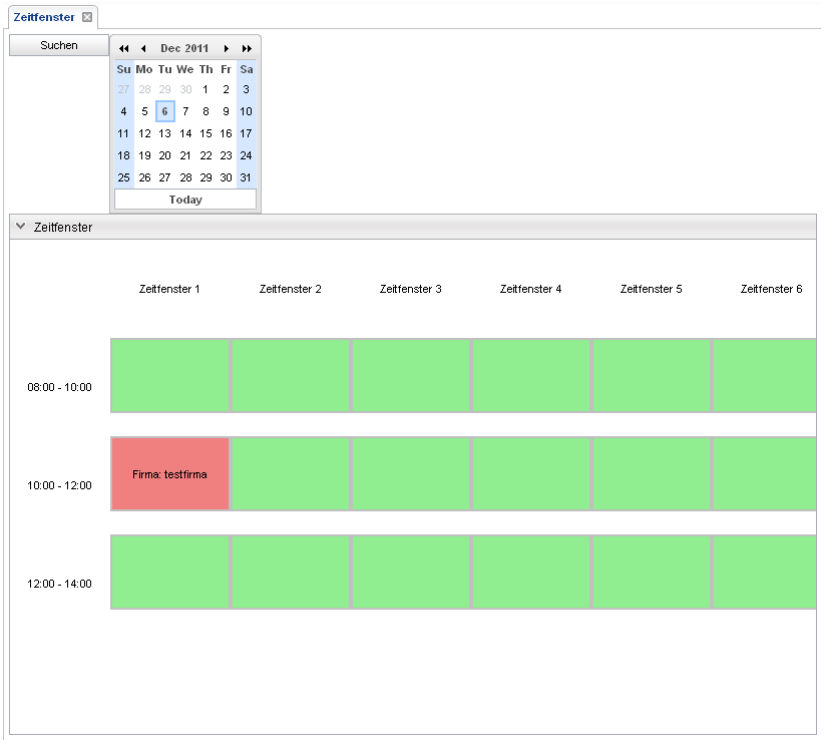

Abbildung 5.7: Zeitfensterbenutzungsoberfläche

5.5 Probleme

Bei der Entwicklung des Prototypen traten an vereinzelten Stellen Probleme verschiedener Kategorien auf, die im Folgenden zusammen mit den Lösungen etwas näher erläutert werden sollen.

GWT Softwarefehler

Die EntityBean-Klasse „Anmeldedaten” enthält Listen der einzelnen Aktivitäten. Jeder Anmeldedatensatz kann von jeder Aktivität beliebig viele Datensätze referenzieren. Diese Referenz wurde im Prototypen mittels der Listen und der Annotation „@OneToMany” umgesetzt. Die EntityBean-Klasse „Anmeldedaten” wie auch die EntityBean-Klassen, die die Aktivitäten repräsentieren implementieren das Interface Serializable, was Voraussetzung für die Übertragung zwischen GWT-Server und GWT-Client ist (siehe Abschnitt 5.3). Gem. GWT soll die Übertragung von Objekten auch funktionieren, wenn diese Objekte widerum Listen von Objekten beinhalten, die serialisierbar sind. Bei den Anmeldedaten funktionierte dies nur unidirektional, nämlich vom GWT-Client zum GWT-Server. Anmeldedaten-Objekte, die der GWT-Server zum GWT-Client schicken wollte, sind mit einer Exception quittiert worden.

Der erste Lösungsversuch bestand in der Erstellung eines benutzerdefinierten Feldserialisierers. Dies sind normale Klassen, denen am Klassennamen der Suffix „_CustomFieldSerializer" angehängt wird. In diese Klassen wird mittels der Klasse „SerializationStreamWriter" Daten und Objekte geschrieben, bzw. mittels der Klasse „SerializationStreamReader" gelesen. Bevor die Daten zwischen dem GWT-Client und dem GWT-Server transportiert werden, werden die Daten in die entsprechenden CustomFieldSerializer-Klassen verpackt. Dieser Lösungsansatz zeigte jedoch keine Verbesserung des Problems. Im zweiten Lösungsversuch ist die EntityBean-Klasse „AnmeldedatenSolo" implementiert worden. Diese Klasse ist mit der Klasse „Anmeldedaten" identisch bis auf das Fehlen der Referenzen auf die Aktivitäten. Im Prototypen funktioniert nun ein Abruf von Anmeldedaten mit den dazugehörigen Aktivitäten in zwei Schritten. Im ersten Schritt werden die Anmeldedaten mittels der Klasse „AnmeldatenSolo" abgerufen, und im zweiten Schritt werden dann sequentiell die Aktivitäten mit Hilfe der Fremdschlüsselreferenz der Aktivitäten auf die Anmeldedaten, nachgefordert. Dieser Lösungsversuch funktioniert. Im Prototypen lies sich das Problem auf diese Weise lösen, allerdings muss es dafür auch eine andere Lösung geben, denn in grossen Projekten mit EntityBean-Klassen, die viele Referenzen besitzen, ist diese Lösung nicht praktikabel.

GWT fehlende Funktionalität

Die asynchrone Kommunikation zwischen dem GWT-Client und dem GWT-Server ist zum Teil unangenehm. Denn muss auf die Antwort des Servers gewartet werden, ist es notwendig selber eine Blockierung des Kommunikations-Threads zu implementieren. Eleganter wäre es, würde das GWT zusätzlich eine synchrone Kommunikation anbieten.

GWT Deploymentschwierigkeiten

Zum Deployen des GWT-Projektes wurde mittels einer war-Datei versucht, diese in den JBoss-Anwendungsserver hochzuladen. Das Erstellen der war-Datei und auch das Hochladen in den JBoss verlief ohne Probleme. Jedoch lies sich die GWT-Applikation nicht starten. Als Lösungsversuch ist dann ein separater Tomcat-Webserver installiert worden. Um einen Parallelbetrieb eines Standalone Tomcat-Webservers mit dem integrierten Tomcat-Webserver von JBoss auf einem Hardwareserver zu ermöglichen, sind in der JBoss-Konfiguration die entsprechenden Ports angepasst worden.

Kapitel 6

Test

In diesem Kapitel soll nun der Prototyp mittels Softwaretests geprüft werden. Ziel ist es, mittels verschiedener Testverfahren Fehler im Programm zu finden, um den Qualitätsstand festzustellen. Das Testen ist ein destruktiver Vorgang, da hierbei mit der Prämisse gearbeitet wird, dass das System Fehler enthält, die verursacht werden können, indem man das System mit Daten rechnen lässt, die aus dem gültigen Randbereich stammen. Werden dabei keine Fehler verursacht, so bedeutet dies nicht, dass das Programm zu 100% fehlerfrei ist, da nicht alle Kombinationen von Zuständen, die ein System annehmen kann, getestet werden können. Allerdings ist ein hoher Qualitätsstandard der Software dadurch nachgewiesen.

6.1 Testkonzept

Das Testkonzept beschreibt zunächst mit der Teststrategie die allgemeine Vorgehensweise beim Testen des Prototypen. Danach werden exemplarisch einige Testfälle beschrieben.

6.1.1 Teststrategie

Der Prototyp soll in mehreren Teststufen getestet werden. In den ersten beiden Stufen wird das System verifiziert. Hierbei wird der Prototyp auf technische Programmfehler untersucht. In der ersten Stufe sollen die einzelnen Komponenten separat getestet werden. Eine Fehleranalyse und -korrektur ist hierbei noch relativ einfach, da die Komponenten als kleine Einheiten des Systems noch überschaubar sind. Die erste Stufe erhöht somit die Qualität der einzelnen Komponenten. Das System besteht aus mehreren fachlichen Komponenten, den A-Komponenten (siehe Abschnitt 4.1.1), und mehreren technischen Komponenten, den T-Komponenten (siehe Abschnitt 4.2.1). Im Prototypen werden die Komponenten durch jeweils eine Session Bean Klasse definiert, die hier getestet werden.

Damit das System überhaupt arbeiten kann, müssen die Komponenten miteinander kommunizieren und interagieren. Für sich alleine können Komponenten fehlerfrei arbeiten, da-

durch können aber keine Rückschlüsse auf die Zusammenarbeit mit anderen Komponenten bezüglich von Fehlern gezogen werden. Das Testen dieser Interaktionen und Kommunikation der Komponenten miteinander ist Aufgabe der zweiten Teststufe. In der dritten Stufe wird das System validiert. Bei der Validierung soll geprüft werden, ob der Prototyp die Anforderungsspezifikationen (siehe Abschnitt 3.2.2) umsetzt. Zum Validieren der funktionalen Anforderungen bedarf es der Mitarbeit des Kunden. Leider war dies aus zeitlichen Gründen nicht mehr durchführbar.

Das DYMS wird mit dem inkrementellen Entwicklungsprozess erstellt (siehe Abschnitt 3.2.1). Dabei sollte nach Fertigstellung eines Entwicklungsschrittes das System die drei Teststufen durchlaufen, jeweils dem neuen Entwicklungsstand angepasst. Der Prototyp stellt die Fertigstellung des ersten Entwicklungsschrittes dar.

Weiterhin sollen die Testfälle strategisch nach dem Black-Box-Testverfahren aufgebaut und zum Teil sowohl positiv als auch negativ getestet werden.

Black-Box-Testverfahren

Beim Black-Box-Testverfahren wird ohne Kenntnis des Quellcodes getestet. Hier wird geprüft, ob die Spezifikation nach aussen erfüllt wird. Das Gegenstück ist das White-Box-Testverfahren, bei dem der Quellcode bekannt ist. Dieser wird dann direkt getestet.

Positiv- / Negativtest

Beim Positivtest werden nur mit Eingabedaten gearbeitet, die auch erwartet werden. Dies zeigt, dass das System bei erwarteten Daten korrekt arbeitet.

Beim Negativtest hingegen werden unerwartete bzw. ungültige Eingabedaten verwendet. Erreicht das System trotz dieser Eingabedaten keinen Zustand, indem es keinen anderen Zustand mehr erreichen kann, so lässt sich hierdurch die Robustheit zeigen.

6.1.2 Testfallspezifikationen

In diesem Abschnitt werden jeweils exemplarische Spezifaktionen von Testfällen zum Komponenten- und Integrationstest beschrieben.

6.1.2.1 Komponententest

Exemplarisch sind im Folgenden zwei Testfälle zu den A-Komponenten „Hof” und „Stammdaten” und ein Testfall zur T-Komponente „Autorisierung” aufgeführt.

Testfall Nr. 1

Testobjekt:	A-Komponente Hof, Methode: getParkplatzWartezeitMinutes(). Diese Methode berechnet die Differenz aus zwei Zeitstempeln. Der erste Zeitstempel ist die Anmeldezeit eines Fahrzeuges und der zweite ist der jeweils aktuelle Zeitstempel. Der Rückgabewert ist in der Zeiteinheit Minuten zu verstehen.
Vorbedingungen:	keine, da ein gesonderter Testalgorithmus geschrieben worden ist, der den gleichen Berechnungsalgorithmus wie das Original verwendet, allerdings ohne Eingabewerte aus der Datenbank.
Beschreibung:	Die Methode wird mit zwei Zeitstempeln als Parameter aufgerufen. Der erste Parameter muss aus chronologischer Sicht später sein als der zweite Parameter. Das erwartete Ergebnis ist die Differenz der beiden Zeitstempel in Minuten.

Tabelle 6.1: Testfall Nr.1: testGetParkplatzWartezeitMinutes

Testfall Nr. 2

Testobjekt:	A-Komponente Stammdaten, Methode: addAdresse(Adressenobjekt). Die Methode wird mit einem Adressenobjekt als Parameter aufgerufen. Die Methode prüft, wenn die Variable Land mit „D" gefüllt ist, ob die Variable PLZ eine Länge von 5 besitzt. Ist dies der Fall, so wird das Adressenobjekt in die Datenbank geschrieben und als Rückgabewert erhält man das Adressenobjekt mit der von der Datenbank zugewiesenen ID. Beträgt die Länge nicht genau 5, so wird eine AdressenException mit entsprechender Fehlermeldung ausgelöst.
Positivtest	
Vorbedingungen:	keine
Beschreibung:	Die Methode wird aufgerufen, wobei darauf zu achten ist, dass das Adressenobjekt in den Variablen „Land" und „PLZ" die korrekten Daten enthält. Es wird dann geprüft, dass das zurückgelieferte Adressenobjekt nicht null sein darf und als ID einen Wert grösser 0 enthält.
Negativtest	
Vorbedingungen:	keine
Beschreibung:	Wie beim Positivtest, allerdings ist hier die Variable PLZ nur mit einem Wert der Länge 3 gefüllt. Nach dem Aufruf wird das Auslösen einer AdressenException erwartet.

Tabelle 6.2: Testfall Nr.2: testAddAdresse_Positiv und testAddAdresse_Negativ

Testfall Nr. 3

Testobjekt:	T-Komponente Autorisierung, Methode: login(String, String). Diese Methode wird mit zwei Parametern aufgerufen. Der erste ist ein Benutzername und der zweite ein Passwort. Die Methode prüft, ob ein Benutzereintrag in der Datenbank mit dem übergebenen Benutzernamen existiert. Ist dies der Fall, wird noch das Passwort auf Gleichheit geprüft. Bei Erfolg wird eine HashMap mit einem Datensatz zurückgeliefert. Im Key ist der String „ok" enthalten, und als Value das entsprechende Benutzerobjekt. Bei einem Misserfolg enthält der Key einen Fehlertext und als Value ein leeres Benutzerobjekt.
Positivtest	
Vorbedingung:	In der Datenbank muss ein Benutzer mit den folgenden Angaben existieren. Als Benutzernamen „testbname", als Vorname „testvorname", als Nachname „testnachname", als Firma „TestFirma" und dem Passwort „12345".
Beschreibung:	Die Methode wird mit einem Benutzernamen und einem Passwort als Parameter aufgerufen, die beide in der Datenbank existieren. Beim Rückgabewert wird überprüft, ob die HashMap einen Key mit dem Inhalt „ok" enthält. Das zurückgelieferte Benutzerobjekt muss Daten enthalten, die zu dem Benutzernamen passen.
Negativtest	
Vorbedingung:	In der Datenbank darf zwar ein Benutzer mit dem Benutzernamen „testbname" existieren, aber nicht mit dem Passwort „111".
Beschreibung:	Beim Testen der Autorisierung liegt der Hauptanteil auf dem Negativtest, denn es darf nicht durch Zufälligkeiten zu einem Einlogvorgang kommen, der nicht autorisiert ist. Es werden verschiedene Kombinationen von Benutzernamen und Passwort geprüft, wobei mindestens ein Element falsch sein muss. Auch soll geprüft werden, ob der Benutzername case sensitiv behandelt wird. Bei allen Kombinationen darf die zurückgelieferte HashMap keinen Key mit dem Wert „ok" enthalten und beim zurückgleiferten Benutzerobjekt müssen die Variablen Vorname, Nachname und Firma null sein.

Tabelle 6.3: Testfall Nr.3: testLogin_Positiv und testLogin_Negativ

6.1.2.2 Integrationstest

Beim Integrationstest wird die Kommunikation vom GWT-Client über den GWT-Server zum Anwendungsserver geprüft. Im Folgenden ist für den Integrationstest exemplarisch ein Testfall beschrieben.

Testfall Nr. 1

Testobjekte:	A-Komponente Anmeldedaten A-Komponente Aktivitäten A-Komponente Hof
Vorbedingungen:	In der Datenbanktabelle „anmeldedaten" darf kein Eintrag vorhanden sein, dessen Attributwert „firma" identisch ist mit dem Wert, der vom TestMaker-Skript eingetragen wird.
Beschreibung:	Im Browser wird die Startseite des Systems aufgerufen. Als erstes wird in die Anmeldung gewechselt. Hier werden nun Anmeldedaten und Aktivitätsdaten erfasst. Bei den Aktivitäten braucht nur eine Aktivität angelegt werden mit einem Datensatz. Der Anmeldedatensatz wird manuell angelegt ohne Nutzung der Zeitfenster-ID Angabe. Ist die Anmeldung erfolgt, wird geprüft, ob nach dem Ablauf einer Zeitspanne der Test „Sie sind angemeldet" im Browser erscheint. Mit diesem Text wird die erfolgreiche Persistierung der Anmeldedaten vom Anwendungsserver bestätigt. Als nächster Schritt erfolgt das Einloggen im DYMS. In der Benutzungsoberfläche wird in den Reiter „Leitstand" gewechselt, der unteranderem die Hof-Komponente darstellt. Ein Klick auf den Button „Aktualisieren" lädt die GWT-Tabelle „Parkplatz". In dieser Tabelle wird nun geprüft, ob der Firmenname, mit dem sich zuvor angemeldet worden ist, hier erscheint.

Tabelle 6.4: Integrationstest Testfall Nr. 1

6.2 Testrealisierung

In diesem Abschnitt werden die beschriebenen Testfälle aus dem Testkonzept (siehe Abschnitt 6.1) ausgeführt und anschliessend bewertet.

6.2.1 Testdurchführung

Einige Testfälle (siehe Abschnitt 6.1.2) müssen mit verschiedenen Eingabedaten wiederholt ausgeführt und bewertet werden, um Fehler auszuschliessen. Des Weiteren müssen nach Änderungen an einer Komponente der sog. Regressionstest durchgeführt werden, dies bedeutet, dass ein Teil der Testfälle wiederholt ausgeführt und bewertet werden muss, um den Qualitätsstand der Komponente zu prüfen. Das wiederholte Durchlaufen aller Testfälle ist sehr aufwendig, daher ist nach Möglichkeiten der Automatisierung gesucht worden. Testsoftware bietet diese Art der Testautomatisierung. Der Einsatz von Testsoftware erlaubt es, Testfälle mit unterschiedlichen Eingabedaten zeitsparend und kosteneffizient zu wiederholen und schnell auszuwerten.

Für die Durchführung der Testfälle zum Komponententest kam das Testsoftwareprodukt „JUnit" zum Einsatz, welches im Folgenden näher erläutert wird.

JUnit

JUnit ist ein Framework mit dessen Hilfe Java-Programme Funktionstests unterzogen werden können. Besonders geeignet ist es zum Testen von Units, also Klassen und Methoden ((Wik11c)). Ein JUnit-Test kennt als Ergebnis nur erfolgreich oder nicht erfolgreich. Zu einer Java-Klasse kann eine entsprechende Test-Klasse erstellt werden. Diese Test-Klasse enthält die zu testenden Methoden der Java-Klasse. Die Namen der Methoden in der Test-Klasse sind identisch mit den Methodennamen der Java-Klasse, mit dem Unterschied, dass in der Test-Klasse den Methodennamen der Präfix „test" vorangestellt wird. In den Test-Methoden kann mit Hilfe sog. assert-Methoden Ergebnisse ausgewertet werden. Diese assert-Methoden liefern dann das Ergebnis, welches eben erfolgreich oder nicht erfolgreich sein kann. Es wurde die Version 4.8.2 eingesetzt.

JMeter

Für den Integrationstest ist zunächst die Testsoftware „JMeter" auf Eignung geprüft worden. Mit JMeter können Last- und Funktionstests durchgeführt werden. Dazu wird die URI der zu testenden html-Seite benötigt. Bei GWT-Applikationen existiert idealerweise nur eine html-Seite. Diese beinhaltet neben den Angaben zum Seitentitel und der css-Datei ein Script, welches die kompilierten GWT-Module, also die GWT-Anwendung, startet. Die einzelnen GWT-Module lassen sich nicht per URI erreichen. Dies ist der erste Grund, weshalb JMeter ungeeignet ist für die Prüfung des Prototypen. Der zweite Grund besteht darin, dass JMeter JavaScript in den html-Seiten nicht ausführen kann (Fou11). GWT-Anwendungen bestehen fast auschliesslich aus JavaScript. Aus diesen zwei Gründen eignet sich JMeter nicht für die Unterstützung beim Integrationstest.

Mittels der JDBC-Schnittstelle wurde testweise ein Lasttest mit der verwendeten

Postgres-Datenbank durchgeführt. Hier zeigte sich, dass JMeter eine gute Testkonfiguration und anschliessende Auswertung anbietet. Die Performance der Postgres-Datenbank ist abhängig von der Hardware, auf der sie betrieben wird, somit stehen die Ergebnisse des Lasttestes auch in Korrelation mit der Hardware. Da die Postgres-Datenbank des Prototypen auf einer virtuellen Maschine läuft, bieten die Lasttestergebnisse keinen nennenswerten Informationsgewinn.

TestMaker

Da JMeter ungeeignet ist für den Integrationstest, ist nach einer anderen Testsoftware für den Integrationstest gesucht und das Produkt „TestMaker" gefunden worden. „TestMaker" wird von der Firma PushToTest entwickelt und vertrieben ((TM11)). Die Testsoftware TestMaker ist unteranderem speziell für das Testen von Ajax-Applikationen entwickelt worden. Das Besondere bei Ajax-Applikationen ist, dass auf die einzelnen html-, bzw. JavaScript-Dateien nicht per URI zugegriffen werden kann, um sie zu testen. Um Ajax-Appliaktionen zu testen, nutzt TestMaker das sog. „Capture & Replay"-Verfahren. Dabei wird mittels Skript, welches im Hintergrund des Browsers läuft, die Ereignisse, die der Benutzer im Browser ausführt, aufgenommen. Dieses Skript wird dann dazu genutzt, automatisch die Benutzereereignisse im Browser in der Reihenfolge wie sie aufgenommen worden sind, auszuführen. Im Skript können sog. assert-Methoden eingefügt werden, die dann bestimmte Ergebnisse überprüfen können. Dabei sind die assert-Methoden analog zu den assert-Methoden aus dem JUnit-Framework zu verstehen, da auch hier die assert-Methoden entweder Erfolg oder Misserfolg als Ergebnis ausgeben.

„TestMaker" eignet sich somit als Testsoftware für den Integrationstest. In dieser Arbeit ist die kostenfreie Version „TestMakerCommunity" verwendet worden, die nicht den vollen Funktionsumfang besitzt. Die zur Verfügung stehenden Funktionen waren aber ausreichend. Zum Einsatz kam die Version 6.

Vorbedingungen

Bei den Testfällen des Komponententests sind die Vorbedingungen zum grossen Teil im JUnit-Framework so hinterlegt worden, dass diese automatisch vor dem Starten des Testfalles erfüllt werden.

Die Vorbedingungen zum Integrationstest müssen manuell erfüllt werden.

Randbedingung

Die Testsoftware „TestMaker" ist nicht auf der virtuellen Maschine, auf der entwickelt worden ist, eingesetzt worden, da es zwischen TestMaker und dem JBoss Konflikte gab. Diese

sind aber nicht weiter untersucht worden. Daher ist TestMaker auf einer anderen virtuellen Maschine ausgeführt worden.

6.2.2 Testergebnisse

Die spezifizierten Testfälle aus Abschnitt 6.1.2 werden nun mit Hilfe der entsprechenden Testsoftware durchgeführt. Die Testdurchläufe werden bewertet und die Ergebnisse im Folgenden tabellarisch dargestellt.

Komponententest

Testfall:	Nr. 1			
Kurzbeschreibung:	Differenz zweier Zeitstempel			
Anmerkung	**Eingabe-paramter**	**Resultat Soll**	**Resultat Ist**	**Test-ergebnis**
Testdurchlauf 1: Differenz am selben Tag	Parameter 1: 10.10.2011 14:00:00 Parameter 2: 10.10.2011 14:01:00	1	1	**ok**
Testdurchlauf 2: Differenz mit Tageswechsel	Parameter 1: 10.10.2011 14:00:00 Parameter 2: 11.10.2011 14:00:00	1440	1440	**ok**
Testdurchlauf 3: Testen mit Monatswechsel	Parameter 1: 31.10.2011 23:00:00 Parameter 2: 01.11.2011 01:05:00	125	125	**ok**

Tabelle 6.5: Komponententest-Ergebnis Testfall Nr.1

Testfall:	Nr. 2
Kurzbeschreibung:	Hinzufügen einer Adresse. Bei Länderkennzeichen „D” muss die Postleitzahl genau 5 Zeichen enthalten.

Anmerkung	**Eingabe-paramter**	**Resultat Soll**	**Resultat Ist**	**Test-ergebnis**
Testdurchlauf 1: Positivtest	Adressenobjekt mit beliebigen Daten, allerdings Variablen Land = D PLZ = 12345	1. Adr.obj. nicht null 2. Adr.obj.-ID > 0	1. Adr.obj. nicht null 2. Adr.obj.-ID > 0	**ok**
Testdurchlauf 2: Negativtest	Adressenobjekt mit beliebigen Daten, allerdings Variablen Land = D PLZ = 123	Adressen- Exception	Adressen- Exception	**ok**

Tabelle 6.6: Komponententest-Ergebnis Testfall Nr.2

Testfall:	Nr. 3			
Kurzbeschreibung:	Prüfen der Autorisierung beim Einloggen.			
Anmerkung	**Eingabe-paramter**	**Resultat Soll**	**Resultat Ist**	**Test-ergebnis**
Testdurchlauf 1: Positivtest	Parameter 1: testbname Parameter 2: 12345	1. „ok” $\in$ HashMap 2. Benutzerobjekt-Variablen gefüllt	1. HashMap enthält Key=”ok” 2. Variablenwerte: Vorname= ”testvorname”, Nachname= ”testnachname” Firma= ”TestFirma”	**ok**
Testdurchlauf 2: Negativtest mit falschem Benutzernamen	Parameter 1: testbnamexxx Parameter 2: 12345	1. „ok”$\notin$ HashMap 2. Benutzerobjekt-Variablen sind null	1. HashMap enthält keinen Key mit Wert „ok” 2. Variablenwerte = null	**ok**
Testdurchlauf 3: Negativtest mit falschem Passwort	Parameter 1: testbname Parameter 2: 111	1. „ok”$\notin$ HasphMap 2. Benutzerobjekt-Variablen sind null	1. HashMap enthält keinen Key mit Wert „ok” 2. Variablenwerte = null	**ok**
Testdurchlauf 4: Negativtest mit Prüfung auf case sensivity beim Benutzernamen	Parameter 1: Testbname Parameter 2: 12345	1. „ok”$\notin$ HashMap 2. Benutzerobjekt-Variablen sind null	1. HashMap enthält keinen Key mit Wert „ok” 2. Variablenwerte = null	**ok**

Tabelle 6.7: Komponententest-Ergebnis Testfall Nr.3

Integrationstest

Testfall:	Nr. 1			
Kurzbeschreibung:	Fahrzeug anmelden			
Anmerkung	**Eingabe-paramter**	**Resultat Soll**	**Resultat Ist**	**Test-ergebnis**
Testdurchlauf 1:	Anmeldedaten-objekt. Wert der Variablen „Firma" = „testfirma"	Tabelle Parkplatz im Browser soll einen Eintrag mit dem Firmennamen „testfirma" enthalten	Tabelle Parkplatz enthält einen Eintrag mit dem Firmennamen „testfirma"	**ok**

Tabelle 6.8: Integrationstest-Ergebnis Testfall Nr.1

Die Ergebnisse der Software „TestMaker" dieses Tests können im Anhang E entnommen werden.

Die Testergebnisse haben gezeigt, dass die hier erstellten Testfälle keine Fehler im System hervorrufen. Zudem haben die Negativtests gezeigt, dass das System eine gewisse Robustheit aufweist. Somit ist das System in dem jetzigen Entwicklungsstand in einer guten Qualität.

6.3 Erfahrungen

Mit dem Framework JUnit liessen sich sehr schnell die ersten Testfälle umsetzen. Die Funktionsweise des Framework's ist intuitiv und die Bedienung in Eclipse erwies sich als unproblematisch.

Die Testsoftware „TestMaker" erwies sich auch als sehr hilfreich. Das ähnelnde Funktionsprinzip zum JUnit-Framework, gerade in Bezug auf die assert-Methoden, machte eine Einarbeitung sehr schnell möglich. Die Software ist übersichtlich aufgebaut. Eingabedaten können von der Software auch aus einer csv-Datei bezogen werden, somit ist es möglich, Testfälle zu realisieren, die es erfordern, grosse Datenmengen in ein System zu schreiben.

Allerdings gibt es bei der Software „TestMaker" auch einige negative Aspekte, die hier kurz genannt werden sollen.

Nach der Installation lies sich der Standardbrowser nicht aus der Software heraus starten, was für das Capture&Replay unbedingt erforderlich ist. Das Problem lag darin, dass in

der „windows.xml”-Datei der absolute Pfad zur exe-Datei des Browsers Leerezichen enthielt. Die Pfadangabe wurde in Hochkommata gesetzt, um das Problem zu lösen.

Weiterhin konnten zwar alle GWT-Widget's im Capture-Mode von TestMaker erkannt werden, beim Abspielen des erstellten Skriptes konnten allerdings nicht alle GWT-Widget's ausgeführt werden, so dass das Skript bei diesen Schritten mit einer Fehlermeldung abbrach. Um das Skript trotzdem vollständig ausführen zu können, musste bei den Schritten, die nicht ausführbare GWT-Widget's enthielten, mit workaround's gearbeitet werden.

Kapitel 7

Schlussbetrachtung

In der Schlussbetrachtung soll abschliessend ein Fazit zu dieser Arbeit gezogen werden. Anschliessend sollen noch Ausblicke gegeben werden, die das weitere Entwicklungspotential des Dock und Yard Management Systems zeigen sollen.

7.1 Fazit

Ziel dieser Arbeit war es, ein Konzept für ein Dock und Yard Management System zu entwerfen. Anschliessend ist die Umsetzbarkeit des entworfenen Konzeptes in einem explorativen Prototypen nachgewiesen worden. In dieser Arbeit wurde nach dem Modell der inkrementellen Softwareentwicklung gearbeitet. Der Prototyp bildet hierbei das erste Inkrement. Eine Analyse des Altsystems erzeugte bereits erste Anforderungen an das neue System. Zudem wurden Interviews durchgeführt, um weitere Anforderungen an das neue System zu ermitteln. Dabei wurde mit Hilfe von Mock-ups bereits in diesem frühen Analysestadium eine Visualisierung des späteren Systems geschaffen.

Beim Entwurf der Architektur für das neue System wurde sich an den Quasar-Richtlinien orientiert, um somit eine möglichst qualitativ hochwertige Software zu entwickeln. Diese Hochwertigkeit zeichnet sich unteranderem darin aus, dass das System leicht erweiterbar und wartbar ist. Zudem sind moderne Techniken verwendet worden, die die Vorteile von klassischen Desktop-Anwendungen mit den Vorteilen der Web-Entwicklung verbinden. Dadurch war es erst möglich, einige Anforderungen zu erfüllen. Der Prototyp hat anschliessend gezeigt, dass sowohl die Architektur als auch die Kombination der verwendeten Technologien funktionieren.

Der Prototyp ist während der Entwicklung und anschliessend in einem Testkonzept verschiedenen Testverfahren unterzogen worden, um seine Qualität zu messen. Bei den Testverfahren während der Entwicklung konnten Fehler beseitigt werden, so dass die Qualität des Systems, auch wenn es sich noch in einem Prototyp-Stadium befindet, bereits erhöht werden konnte.

Abschliessend betrachtet sind die Ziele der Arbeit, ein neues Dock und Yard Management System zu konzipieren und mit neuen Techniken zu realisieren, erfüllt worden.

7.2 Ausblick

Für den Realeinsatz muss der entwickelte Prototyp noch ingenieursmässig weiterentwickelt werden. Das Exceptionhandling sowie die Verifizierung von Benutzereingaben muss vervollständigt werden. Zudem muss versucht werden, für die dargestellten Probleme im Abschnitt 5.5 bessere Lösungen zu finden. Weiterhin müssen noch Funktionen hinzugefügt werden, um die gestellten Anforderungen vollständig zu erfüllen. Es bedarf dann einer Validierung des Prototypen zusammen mit dem Kunden, um zu prüfen, ob die Anforderungen gem. Kundenwunsch korrekt umgesetzt worden sind.

Ein DYMS bietet viel Raum für Weiterentwicklungsmöglichkeiten. So wäre es bspw. denkbar ein Telematiksystem des eigenen Fuhrparks anzubinden. Es könnten sog. Geofences um den Lagerstandort gelegt werden in einem bestimmten Radius. Fährt nun ein mit Telematik ausgestattetes Fahrzeug in den Geofence-Bereich ein, so kann über die Grösse des Geofencebereich eine ungefähre Ankunftszeit des Fahrzeuges am Lager hochgerechnet werden. Dieser voraussichtliche Eintreffzeitpunkt kann dann an das DYMS gemeldet werden, welches dann diese Information anzeigt. Oder weitergehend kann das DYMS widerum mit dem Lagerverwaltungssystem zusammenarbeiten, um mit der Information des voraussichtlichen Eintreffzeitpunktes einen Kommissionierauftrag auszulösen. Dadurch würden dann die Kommissionieraufträge just-in-time ablaufen können. Das Fahrzeug müsste nicht lange warten und in der Lagerhalle würde kein unnötiger Kommissionierplatz verschwendet werden.

Weiter wäre eine Kopplung mit einem Kamerasystem an und in der Lagerhalle denkbar. Die Kameras könnten dem DYMS Feedback liefern, wann genau ein Fahrzeug an eine Rampe fährt bzw. diese wieder verlässt. Darüber hinaus könnte die Kamera mittels Schrifterkennung das KFZ-Kennzeichen des betreffenden Fahrzeuges mitliefern. Zu prüfen wären hierbei natürlich noch Datenschutzgesetze.

Weiter könnten die fahrzeugspezifischen Daten dahingehend erweitert werden, indem Daten wie Abgasnorm des Zugfahrzeuges oder bspw. fahrzeugeffizienzsteigernde Mittel, wie bspw. Windabweiser, am Fahrzeug angebracht sind. Mit Hilfe dieser Daten könnte dann das Unternehmen, welches das DYMS einsetzt, eine Aussage über die Umweltverträglichkeit der eingesetzten Fahrzeuge treffen.

Das DYMS sollte in einem Lager der Cargo Service Nord GmbH aufgebaut und unter Realbedingungen getestet und weiterentwickelt werden. Dadurch bietet sich die Möglichkeit, das System anderen Firmen unter Realbedingen zu demonstrieren.

Anhang A

DVD's

Dieser Arbeit liegen insgesamt 5 DVD's bei. Der Prototyp dieser Arbeit, die Entwicklungsumgebung, die Entwicklungswerkzeuge, Server und weiteres Zubehör, sind in einer virtuellen Maschine installiert worden. Dies ermöglicht es einfach und ohne grossen Installationsaufwand in das Eclipse-Projekt des Prototypen zu schauen oder einen Testlauf durchzuführen. Diese virtuelle Maschine ist auf den beigefügten DVD's enthalten.

A.1 DVD#1

Inhalt:

- Bachelor_Buerger.pdf: dieses Dokument
- Verzeichnis „VM": enthält Dateien der virtuellen Maschine und Dokumentation
- Verzeichnis „src": enthält den Sourcecode des Prototypen als java-Dateien
- Verzeichnis „JBoss": enthält die jar-Datei mit SessionBeans und Datasource-Datei für Anbindung JBoss an Datenbank
- Verzeichnis „GWT": enthält das Kompilat der GWT-Anwendung
- Verzeichnis „DB": enthält ein Wiederherstellungsskript der Datenbank
- Verzeichnis „TestMaker" enthält Szenario- und Use Case-Datei zum Testen des Prototypen mit Software „TestMaker"

A.2 DVD#2 - DVD#5

Inhalt:

- weitere Dateien der virtuellen Maschine

Anhang B

Mock-ups

B.1 Ausgewählte Mock-ups aus den Vorgesprächen

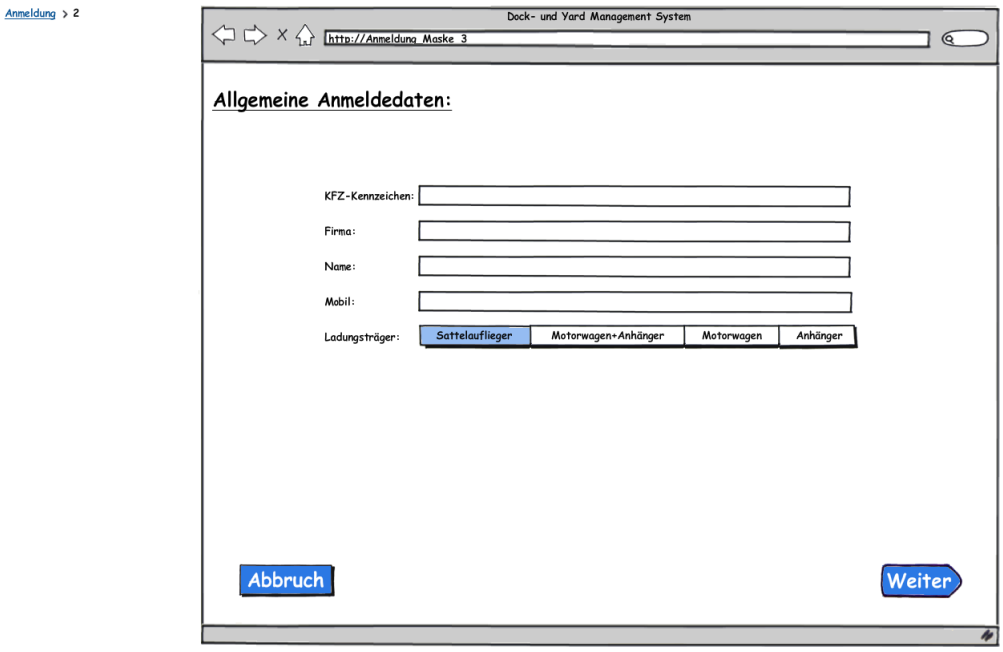

Abbildung B.1: Mock-upV1 - Anmeldung - Standarddatenerfassung

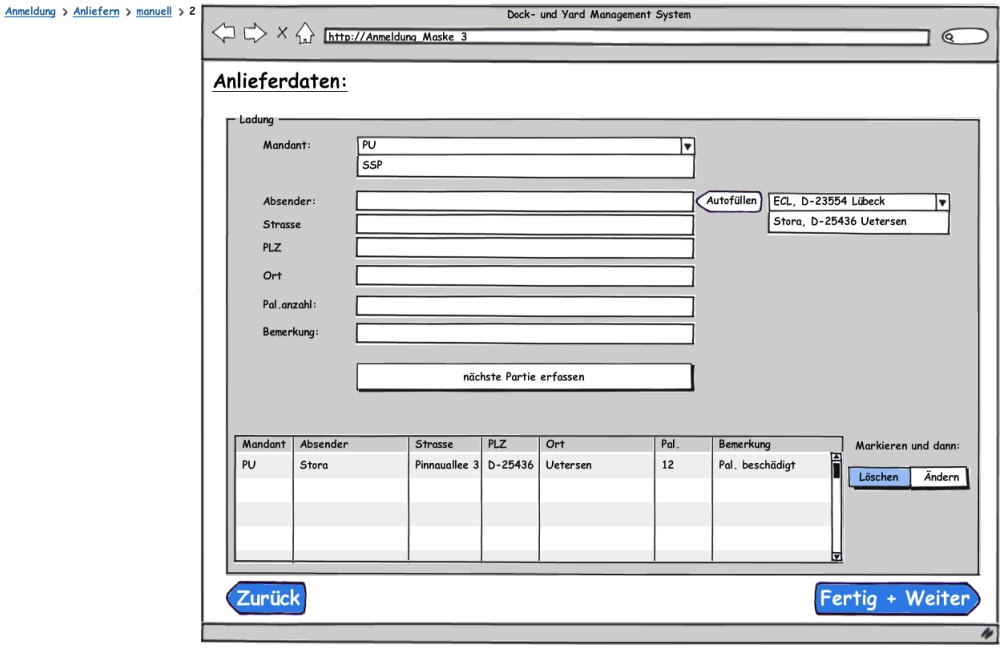

Abbildung B.2: Mock-upV1 - Anmeldung - Anlieferdatenerfassung

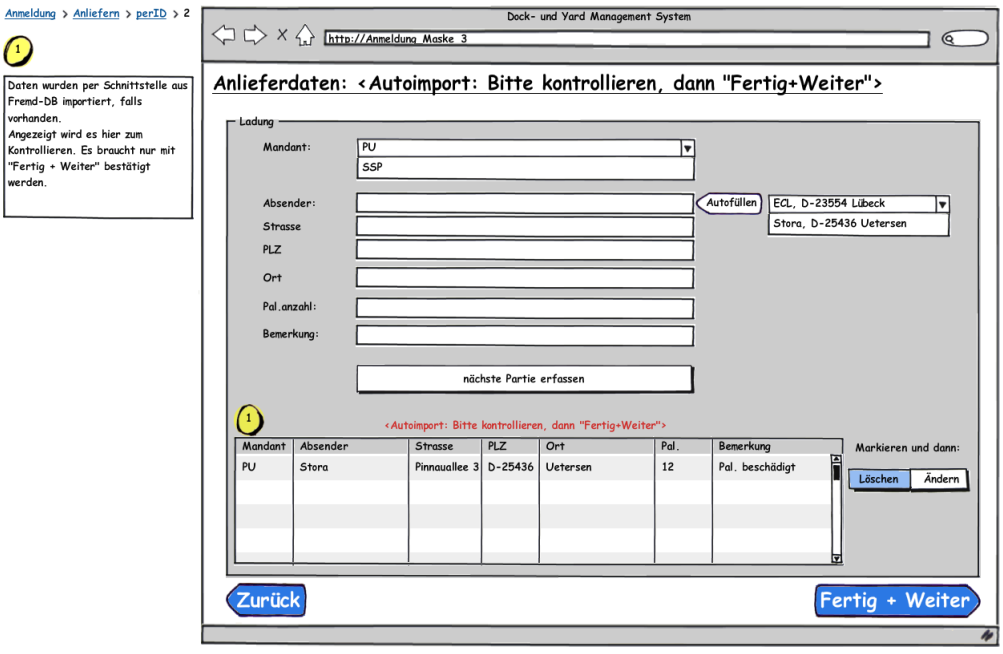

Abbildung B.3: Mock-upV1 - Anmeldung - Anlieferdaten automatischer Import

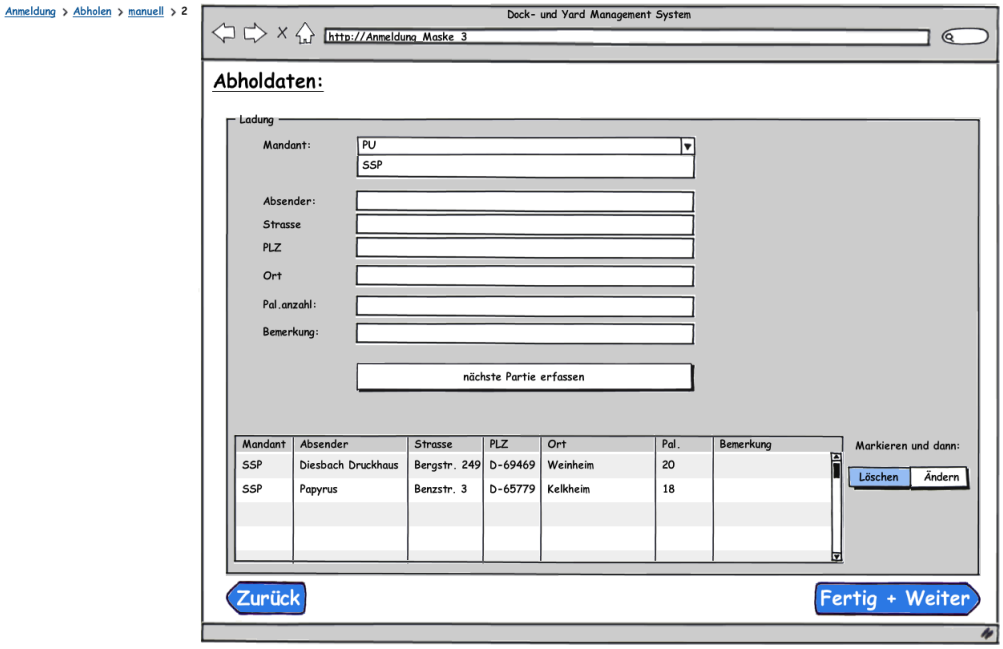

Abbildung B.4: Mock-upV1 - Anmeldung - Abholdatenerfassung

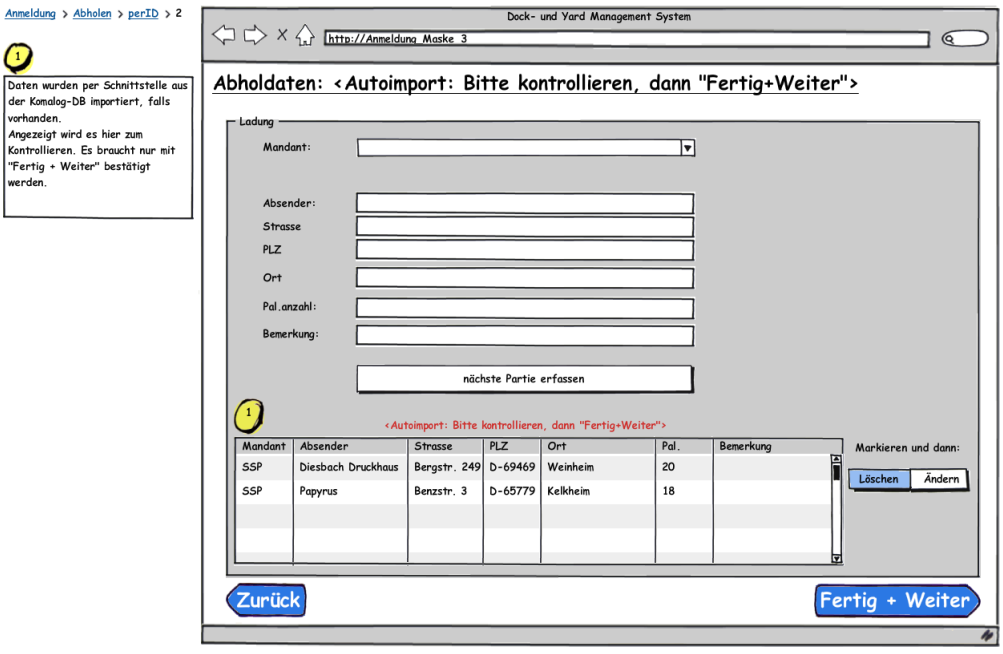

Abbildung B.5: Mock-upV1 - Anmeldung Abholdaten automatischer Import

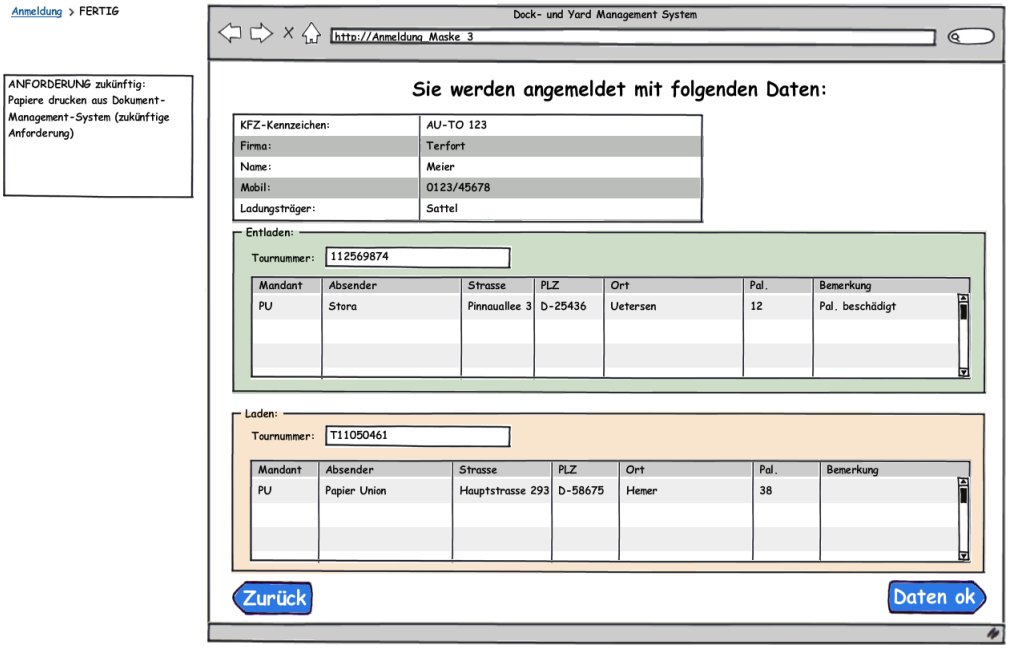

Abbildung B.6: Mock-upV1 - Anmeldung - Anmeldedatenübersicht

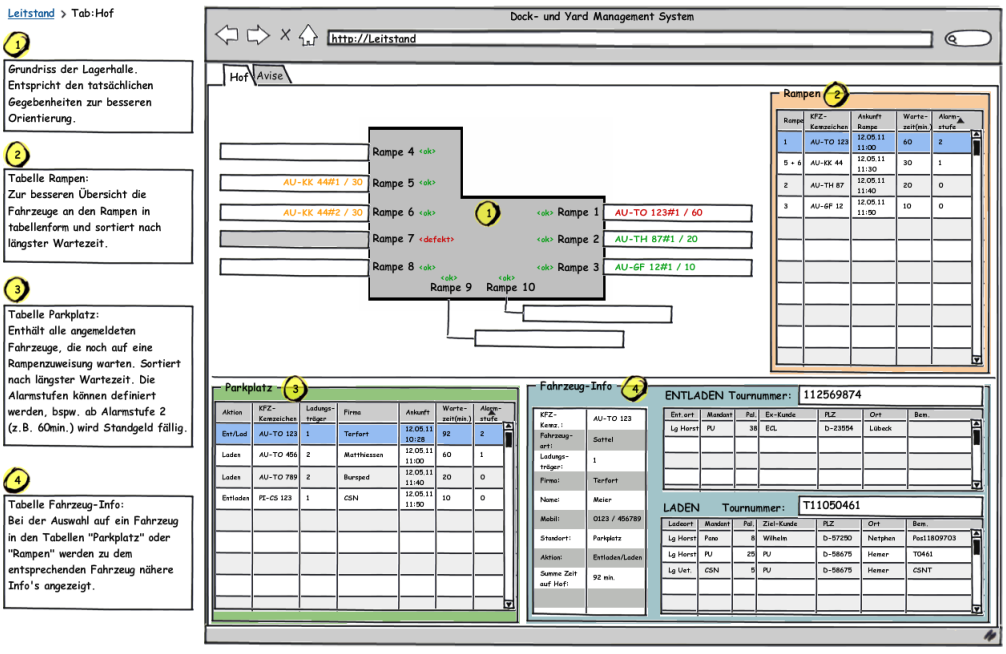

Abbildung B.7: Mock-upV1 - Leitstand

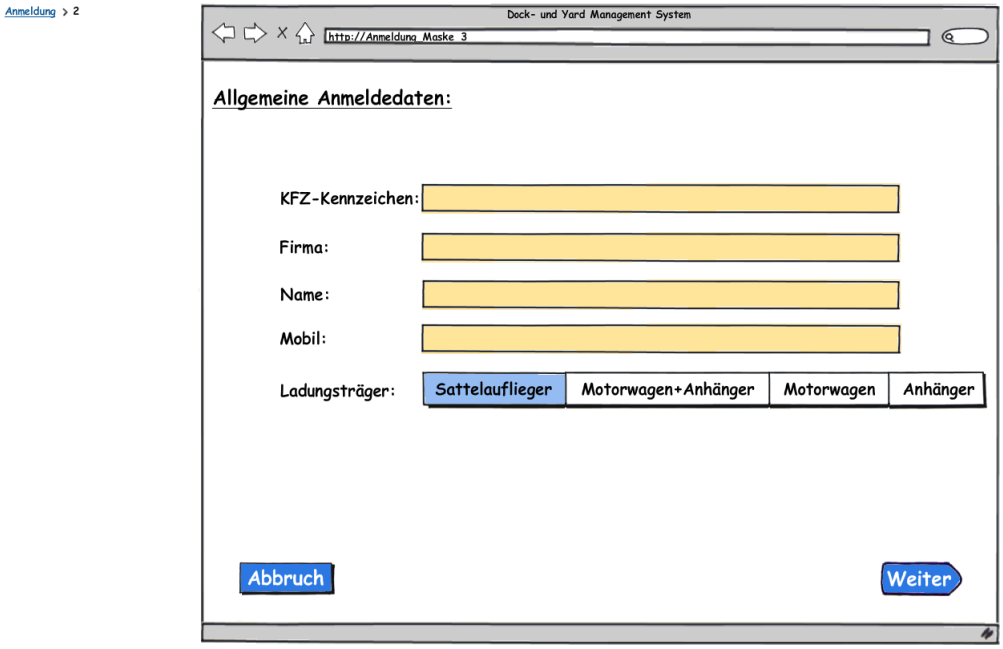

Abbildung B.8: Mock-upV2 - Anmeldung - Standarddatenerfassung

B.2 Ausgewählte Mock-ups nach den Interviews

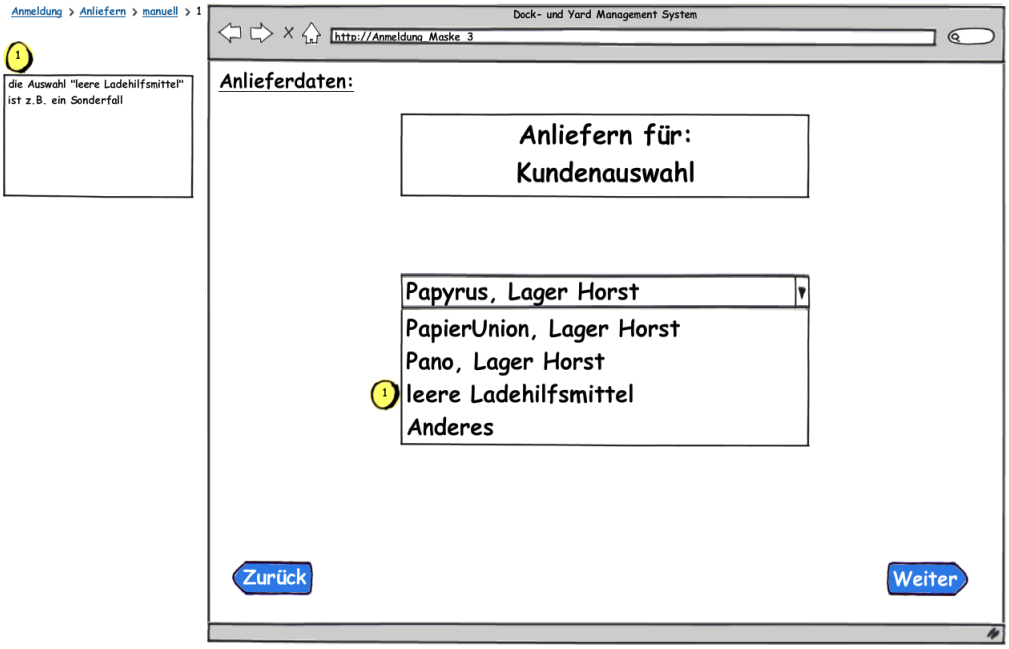

Abbildung B.9: Mock-upV2 - Anmeldung - Anliefern Kundenauswahl

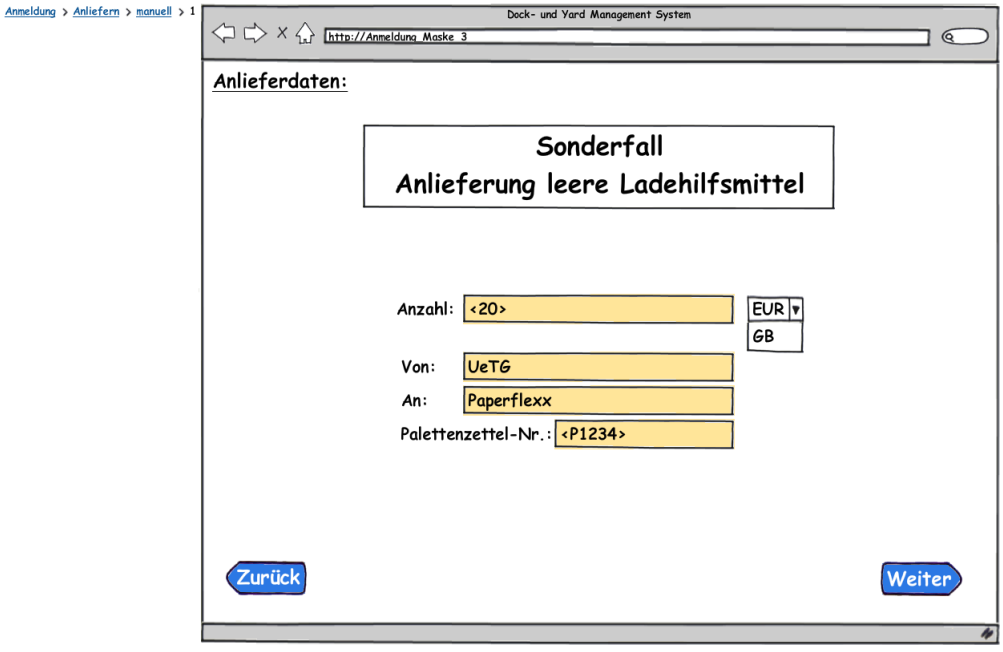

Abbildung B.10: Mock-upV2 - Anmeldung - Anlieferung Sonderfall

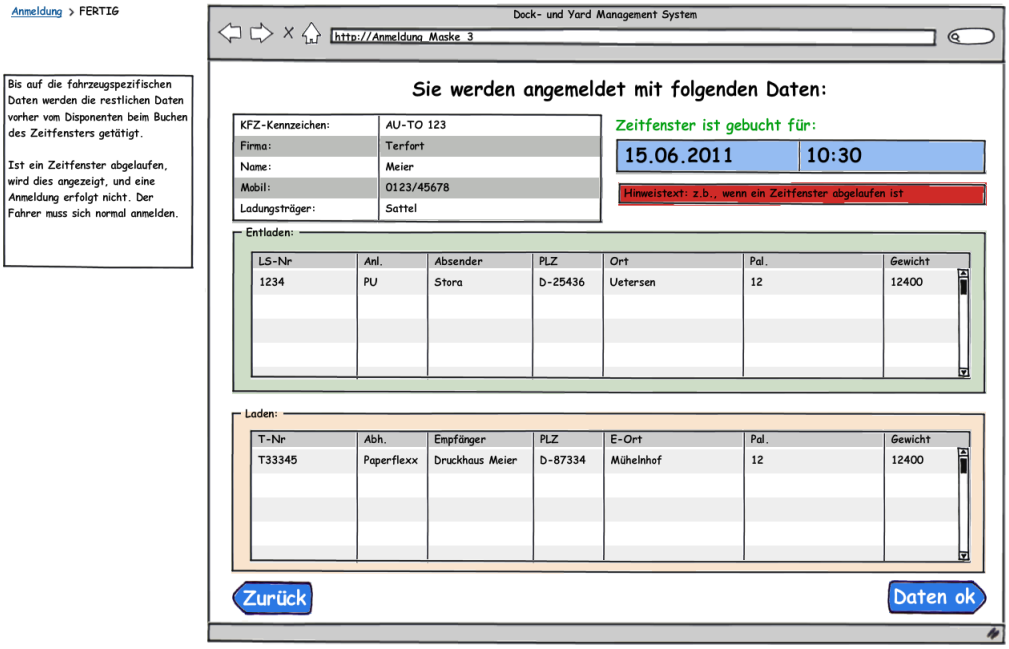

Abbildung B.11: Mock-upV2 - Anmeldung - Anmeldedatenübersicht mit Zeitfenster

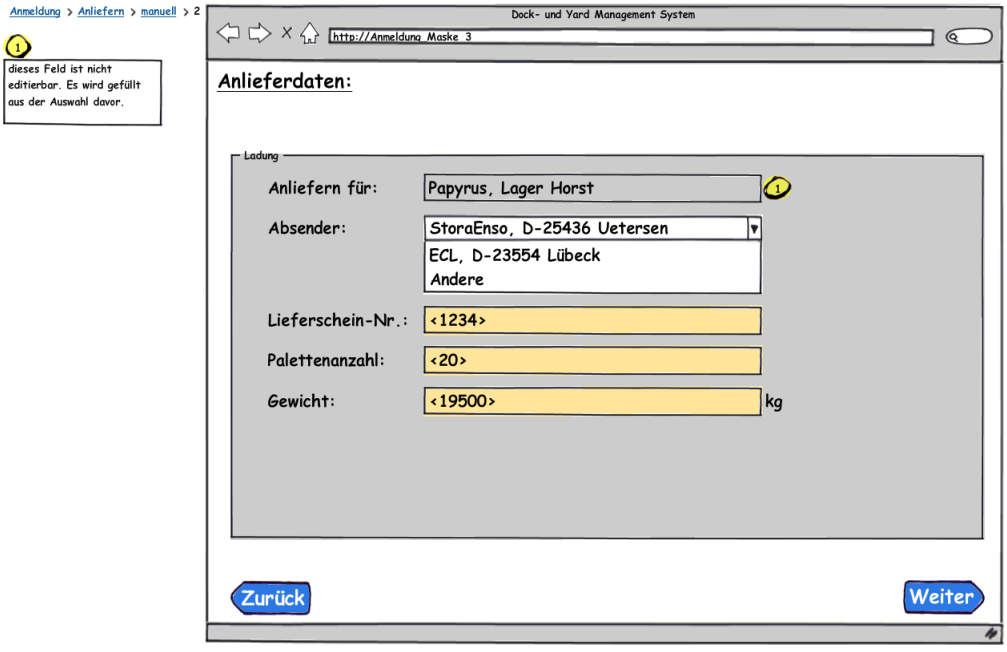

Abbildung B.12: Mock-upV2 - Anmeldung - Anlieferdatenerfassung verkürzt

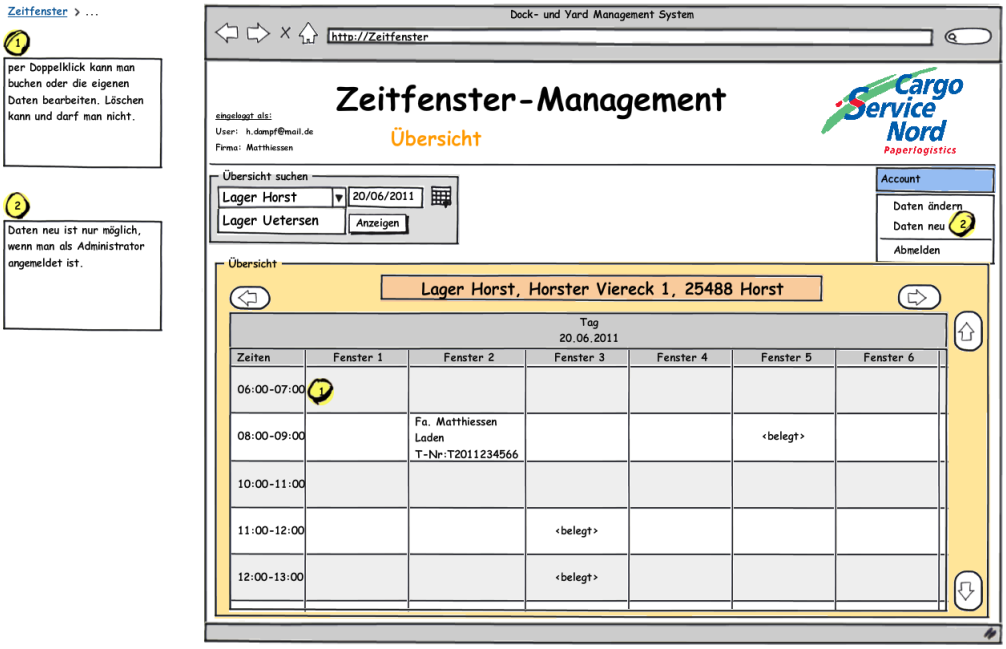

Abbildung B.13: Mock-upV2 - Zeitfenster - Zeitfenstertabelle

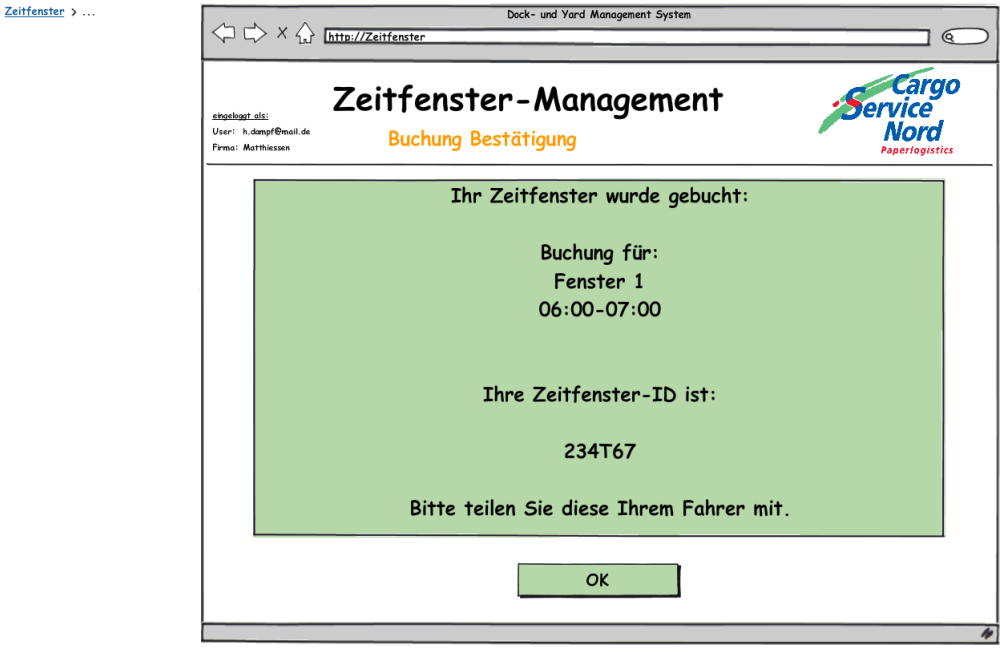

Abbildung B.14: Mock-upV2 - Anmeldung - Mitteilung Zeitfenster-ID für Anmeldung

Anhang C

Interviewprotokolle

C.1 Interviewprotokolle der Interviews

Papier:	**Gesprächsprotokoll**	
Thema:	**1. Version Mock-Ups Dock- und Yard-Management-System**	
Teilnehmer:	Klaus Ziesmann, Firma Cargo Service Nord GmbH Thorsten Kuschinski, Firma Cargo Service Nord GmbH Niels Böhlen, Firma Uetersener Transport&Logistik GmbH Peter Ulrich, Firma LKW-Pilot André Bürger, Firma Cargo Service Nord GmbH	
Datum:	18.05.2011	
Ort:	Große Twiete 92, 25436 Uetersen	erstellt am:
Autor:	André Bürger	07.06.2011

	Allgemein
	Time-Slot-Management: Time-Slot-Management ist ein wichtiger Punkt. Das Time-Slot-Management sollte bereits im Prototypen umgesetzt werden.
	Komponente Anmeldung
1	*Positive Funktionen*, die beibehalten werden sollten: - Möglichkeit der verschiedenen Sprachen. - Möglichkeit die Anliefer- bzw. Abholdaten per Schnittstelle einlesen zu lassen - Fensterführung. Der Benutzer wird durch den Fragenkatalog durch mehrere Fenster geleitet. Die Anzahl der Fenster sollte so gewählt sein, dass pro Fenster nur ein paar Informationen dem Benutzer abverlangt werden.
2	*Time-Slot-ID-Erfassung:* Nach der Eingabe der allgemeinen Anmeldedaten sollte die Möglichkeit bestehen, dass der Fahrer sich mit einer Time-Slot-ID anmeldet, die er vorher von seinem Disponenten erhalten hat. Der Disponent erhält diese Time-Slot-ID nach der Buchung eines Time-Slots über das Web. Der Fahrer sollte im Falle, dass er sich mit einer Time-Slot-ID anmeldet, Informationen über seinen Time-Slot erhalten, bspw. über Gültiglkeit und ähnliches.
3	*Kundenauswahl:* Es soll eine Kundenauswahl erfolgen. In Abhängigkeit der Auswahl werden bei den Anliefer- bzw. Abholdaten verschiedene Daten erfragt. Bspw. werden bei der Auswahl „leere Europaletten“ andere Daten abgefragt als bei der Auswahl eines Kunden. So kann bei der Auswahl „leere Europaletten“ das Fenster mit der Eingabe der Tournummer übersprungen werden.
4	*Anliefer- / Abholdaten:* Erfassen der Anlieferdaten und der Abholdaten ist zu umfangreich. Je nachdem, welche Auswahl in der Kundenauswahl getroffen (siehe 3) worden ist, sind die entsprechenden Datenfelder zu erfragen. Bei der Auswahl eines Kunden sind aber die bisherigen Felder zu viel. Bspw. braucht nicht die gesamte Adresse eingegeben zu werden, es reicht hier Ort und Gewicht oder Anzahl der Paletten. Auch bei den Anlieferdaten braucht nur der Ort, das Gewicht oder die Anzahl der Paletten gefüllt werden. Alternativ ist das Autofüllen mittels einer Auswahlliste gut und sollte beibehalten werden.
	Komponente Leitstand
	Aus zeitlichen Gründen konnte dies nicht mehr besprochen werden.

Abbildung C.1: Protokoll des Interviews mit der Firma Cargo Service Nord GmbH

Papier:	**Gesprächsprotokoll**	
Thema:	**1. Version Mock-Ups Dock- und Yard-Management-System**	
Teilnehmer:	Ulf Semmelhaack, Firma Semmelhaack-Logistik GmbH André Bürger, Firma Cargo Service Nord GmbH	
Datum:	18.05.2011	
Ort:	Werner-von-Siemens-Str. 6, 25335 Elmshorn	erstellt am:
Autor:	André Bürger	07.06.2011

	Allgemein
	Time-Slot-Management: Time-Slot-Management ist ein wichtiger Punkt. Das Time-Slot-Management ist im ersten Mock-Up nur als zu berücksichtigende Anforderung angedacht. Allerdings stellte sich im Gespräch heraus, dass das Time-Slot-Management ein Key-Feature sein würde, welches bereits im Prototypen umgesetzt werden sollte.
	Komponente Anmeldung
1	*Positive Funktionen*, die beibehalten werden sollten: - Möglichkeit der verschiedenen Sprachen. - Möglichkeit die Anliefer- bzw. Abholdaten per Schnittstelle einlesen zu lassen - Fensterführung. Der Benutzer wird durch den Fragenkatalog durch mehrere Fenster geleitet. Die Anzahl der Fenster sollte so gewählt sein, dass pro Fenster nur ein paar Informationen dem Benutzer abverlangt werden.
2	*Kundenauswahl:* Beim Anmelden sollte eine Auswahl des Kunden getroffen werden können. Im Workflow wäre die Position nach der Auswahl der Aktivität am besten geeignet, um für das Anliefern und das Abholen eine separate Auswahl treffen zu können.
3	*Anliefer- / Abholdaten:* Erfassen der Anlieferdaten und der Abholdaten ist zu umfangreich. Änderungen der Abholdaten: - Mandantenauswahl wird durch die Auswahl des Kunden ersetzt. - Angaben wie Empfänger, Strasse sind zuviel und unwichtig. - Es sollten nur Palettenanzahl, Gewicht und Ort erfasst werden. Änderungen der Anlieferdaten: - Auch hier wird die Mandantenauswahl durch die Auswahl des Kunden ersetzt. - Auswahl des Absenders entweder mittels Autofüllen oder falls der Absender nicht im Autofüllen enthalten ist, so braucht nur Ort, Palettenanzahl und/oder Gewicht erfasst werden. - Es brauchen keine mehrere Partien erfasst werden.
4	*Komplett- / Teilladung:* Die Anzahl der abzuholenden Paletten sollte eine Einteilung in Teilladung oder Komplettladung ermöglichen. Im Leitstand kann diese Information dazu genutzt werden, um dem Lagermitarbeiter einen Anhaltspunkt zum Schätzen der eventuellen Beladedauer zu geben. Eventuell sollte dies auch durch einen Algorithmus erfolgen.
5	*Time-Slot-Management:* Dem Fahrer sollte eine Info angezeigt werden, ob ein Time-Slot für ihn gebucht worden ist oder nicht. Falls einer gebucht worden ist, so soll eine Info darüber angezeigt werden.
	Komponente Leitstand
6	*Feld „Fahrzeug-Info“:* Dieses Feld sollte kleiner oder als Nebenfeld dargestellt werden. Dadurch ist mehr Platz vorhanden, um die schematische Darstellung der Halle mit den Rampen zu vergößern.

Abbildung C.2: Protokoll des Interviews mit der Firma Semmelhaack-Logistik GmbH

7	*Feld „Parkplatz“:* Im Feld „Parkplatz“ sollte die in der Anmeldung getätigte Kundenauswahl mit angezeigt werden. Anhand der Kundenauswahl, sowie Kriterien Palettenanzahl oder Gewicht sollte ein Hinweis auf Komplettladung oder Teilladung erfolgen.
8	*Feld „Rampen“:* Auch im Feld „Rampen“ sollte die getätigte Kundenauswahl angezeigt werden.
9	*Darstellung der Lagerhalle:* Die Rampen sollten anhand der Time-Slot-Buchungen eine Reservierung erhalten, so dass diese dann nicht mehr manuell vergeben werden können ohne sie vorher freizuschalten.
10	*Rampen-Sensor:* Als weitere Anforderung nach dem Prototypen sollte eine Schnittstelle zu Rampen-Sensoren geschaffen werden. Über die Rampen-Sensoren kann dann später die tatsächliche physische Belegung der Rampen angezeigt werden.

2 / 2

Abbildung C.3: Protokoll des Interviews mit der Firma Semmelhaack Logistik GmbH - Fortsetzung

Anhang D

Anwendungsfälle

D.1 Vollständige Sammlung der Anwendungsfälle

Nr.:	1
Anwendungsfall:	LKW anmelden
Ziel:	Ein Fahrer meldet sein Fahrzeug am System an
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	Im System existiert der LKW mit den benötigten Daten und die Anmeldung ist protokolliert
Nachbedingung Fehlschlag:	Der Fahrer muss sich an den Lagermeister wenden
Akteure:	Fahrer
Auslösendes Ereignis:	LKW trifft auf dem Betriebshof des Lagers ein.
Beschreibung:	1. Der Fahrer wählt eine Sprache 2. Die fahrzeug- und fahrerspezifischen Daten werden erfasst 3. Der Fahrer wählt eine der drei Eingabemöglichkeiten (per ID, manuell, per Zeitfenster-ID), um die ladungsspezifischen Daten zu erfassen 4. Wählt er die manuelle Möglichkeit, muss er einen Lagerkunden wählen 5. Anhand der Palettenmenge oder des Gewichtes setzt das System beim Fahrzeug das Attribut Teilladung oder Komplettladung 6. Der Fahrer kontrolliert die erfassten Daten und bestätigt diese, um seine Abmeldung damit erfolgreich abzuschliessen

Tabelle D.1: Anwendungsfall: LKW anmelden

Nr.:	2
Anwendungsfall:	Sprache wählen
Ziel:	Die Dialoge werden in der gewählten Sprache angezeigt
Kategorie:	should be
Vorbedingung:	-
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	Fahrer
Auslösendes Ereignis:	Ein Fahrer meldet sich an
Beschreibung:	Der Fahrer wählt eine Sprache, in der er die Dialoge der Anmeldung angezeigt haben möchte

Tabelle D.2: Anwendungsfall: Sprache wählen

Nr.:	3
Anwendungsfall:	Dateneingabe per ID
Ziel:	Bei der Anmeldung werden die ladungsspezifischen Daten mit Hilfe eines externen Systems erfasst
Kategorie:	must have
Vorbedingung:	Der Fahrer muss eine im externen System vorhandene Referenz-Nr. besitzen
Nachbedingung Erfolg:	Bis auf die fahrer- und fahrzeugspezifischen Daten muss der Fahrer keine weiteren Daten erfassen
Nachbedingung Fehlschlag:	Der Fahrer muss auch die ladungsspezifischen Daten manuell erfassen
Akteure:	Fahrer, Speditions-System „Komalog”
Auslösendes Ereignis:	Der Fahrer hat während der Anmeldung die Auswahl „Dateneingabe per ID” gewählt
Beschreibung:	1. Der Fahrer wählt eine Sprache 2. Die fahrzeug- und fahrerspezifischen Daten werden erfasst 3. Der Fahrer wählt die Eingabemöglichkeit „per ID” 4. Der Fahrer scannt oder gibt seine Referenz-Nr.(hier eine Tournummer) ein 5. Aus dem externen Speditions-System „Komalog” werden die ladungsspezifischen Daten geholt und angezeigt 6. Eventuelle Sonderfälle (wie z.B. das An- oder Abliefern von leeren Paletten) müssen zusätzlich manuell erfasst werden 7. Der Fahrer kontrolliert die erfassten Daten und bestätigt diese, um seine Abmeldung damit erfolgreich abzuschliessen

Tabelle D.3: Anwendungsfall: Dateneingabe per ID

Nr.:	4
Anwendungsfall:	Dateneingabe manuell
Ziel:	Die Daten für die Anmeldung werden komplett manuell eingegeben
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	Das Fahrzeug ist im System angemeldet
Nachbedingung Fehlschlag:	Der Fahrer muss sich an den Lagermeister wenden
Akteure:	Fahrer
Auslösendes Ereignis:	Der Fahrer hat während der Anmeldung die Auswahl „Dateneingabe manuell" gewählt
Beschreibung:	1. Der Fahrer wählt eine Sprache 2. Die fahrzeug- und fahrerspezifischen Daten werden erfasst 3. Der Fahrer wählt die Eingabemöglichkeit „manuell" 4. Der Fahrer gibt die ladungsspezifischen Daten ins System ein. 5. Falls Sonderfälle (wie z.B. das An- oder Abliefern von leeren Paletten) vorhanden sind, werden diese zusätzlich manuell erfasst 6. Der Fahrer kontrolliert die erfassten Daten und bestätigt diese, um seine Abmeldung damit erfolgreich abzuschliessen

Tabelle D.4: Anwendungsfall: Dateneingabe manuell

Nr.:	5
Anwendungsfall:	Dateneingabe per Zeitfenster-ID
Ziel:	Die ladungsspezifischen Daten sind im System bei der Anmeldung bereits vorhanden und zusätzlich erhält das Fahrzeug ein Zeitfenster zur Be-/Entladung
Kategorie:	must have
Vorbedingung:	Der Disponent hat für die Anlieferung, bzw. Abholung ein Zeitfenster im System gebucht
Nachbedingung Erfolg:	Das Fahrzeug ist mit einem Zeitfenster im System angemeldet
Nachbedingung Fehlschlag:	Fahrer muss sich manuell anmelden
Akteure:	Fahrer
Auslösendes Ereignis:	Der Fahrer wählt die Dateneingabe per Zeitfenster-ID
Beschreibung:	1. Der Fahrer wählt eine Sprache 2. Die fahrzeug- und fahrerspezifischen Daten werden erfasst 3. Der Fahrer wählt die Möglichkeit der Anmeldung per Zeitfenster-ID 4. Der Fahrer gibt die Zeitfenster-ID ins System ein 5. Der Fahrer kontrolliert die Daten und das für ihn gebuchte Zeitfenster und bestätigt diese, um die Anmeldung abzuschliessen

Tabelle D.5: Anwendungsfall: Dateneingabe per Zeitfenster-ID

Nr.:	6
Anwendungsfall:	Übersicht Hof und Rampen
Ziel:	Den aktuellen Zustand der Rampen und des Hofes sehen
Kategorie:	must have
Vorbedingung:	Die Übersicht muss von verschiedenen örtlich getrennten Standorten einsehbar sein
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, Lagermeister, Lagermitarbeiter
Auslösendes Ereignis:	-
Beschreibung:	Nach dem Einloggen am System ist eine Übersicht, bspw. in tabellenform, zu sehen. Die Übersicht zeigt aktuell, welche Fahrzeuge angemeldet auf dem Hof stehen und welche Rampen mit welchen Fahrzeugen belegt sind. Dazu gibt es noch weitere Informationen, wie bspw. wie lange sind die Fahrzeuge schon auf dem Hof oder an einer Rampe, was sollen sie laden und noch weitere Details. Zudem müssen sich wichtige Informationen, wie bspw. die Wartezeit, automatisch aktualisieren, damit der Lagermeister diese zeitkritischen Informationen immer aktuell zur Verfügung hat.

Tabelle D.6: Anwendungsfall: Übersicht Hof und Rampen

Nr.:	7
Anwendungsfall:	LKW Rampen zuordnen
Ziel:	einem LKW ist 1 bis n Rampen zugewiesen
Kategorie:	must have
Vorbedingung:	LKW ist korrekt angemeldet und die zuzuweisenden Rampen sind belegbar
Nachbedingung Erfolg:	Das Fahrzeug ist 1 oder n Rampen zugewiesen
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister, Lagermitarbeiter
Auslösendes Ereignis:	Ein Zeitfenster wird fällig oder ein Fahrzeug und die entsprechende Ladung sind bereit
Beschreibung:	Einer der Akteure prüft die angemeldeten Fahrzeuge auf dem Hof und die Verfügbarkeit der Rampen Nach Kriterien, wie bspw. Wartezeit, gebuchtes Zeitfenster oder Relation, wählt er ein Fahrzeug aus und weist es einer Rampe zu. Handelt es sich bei dem Fahrzeug z.B. um einen Gliederzug, so kann er diesem Fahrzeug auch 2 Rampen gleichzeitg zuweisen. Durch die Auswahl des Lagerkunden kann der Akteur erkennen, ob das Fahrzeug eventuell an mehreren Rampen laden muss. Der Akteur kann dann dem Fahrzeug die weiteren Rampen zuweisen. Durch das Attribut Teilladung oder Komplettladung kann der Akteur erkennen, ob ein Fahrzeug eventuell vorgezogen werden kann, da es nur eine Teilladung bekommt. Ist eine Rampe belegt, so kann trotzdem ein weiteres Fahrzeug dieser Rampe zugeordnet werden. Allerdings muss erkennbar sein, welches Fahrzeug aktiv an der Rampe tätig ist, und welches Fahrzeug auf das Freiwerden der Rampe wartet.

Tabelle D.7: Anwendungsfall: LKW Rampen zuordnen

Nr.:	8
Anwendungsfall:	LKW abmelden
Ziel:	Der LKW wird als abgemeldet gekennzeichnet und ist somit systemseitig nicht mehr auf dem Betriebsgelände
Kategorie:	must have
Vorbedingung:	LKW ist angemeldet.
Nachbedingung Erfolg:	Die Abmeldung ist protokolliert
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister, Lagermitarbeiter
Auslösendes Ereignis:	-
Beschreibung:	Einer der Akteure wählt einen LKW aus. Dabei ist unwichtig, ob der LKW einer Rampe zugewiesen ist oder nicht.

Tabelle D.8: Anwendungsfall: LKW abmelden

Nr.:	9
Anwendungsfall:	Stammdaten verwalten
Ziel:	Die Stammdaten werden angelegt, gelesen, geändert und gelöscht
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister
Auslösendes Ereignis:	-
Beschreibung:	Nach erfolgreichem Einloggen mit den Zugangsdaten des Lagermeisters können die Stammdaten manipuliert werden. Es können neue Daten angelegt werden, diese Daten können auch ausgelesen werden, die Daten können geändert oder gelöscht werden

Tabelle D.9: Anwendungsfall: Stammdaten verwalten

Nr.:	10
Anwendungsfall:	Stammdaten Zeitfenster verwalten
Ziel:	Die Stammdaten speziell für das Zeitfenster-Management werden verwaltet
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister, interner Disponent
Auslösendes Ereignis:	-
Beschreibung:	Die Stammdaten des Zeitfenster-Managements definieren die nötigen Informationen, die nötig sind, um ein Zeitfenster zu buchen. Zu den Stammdaten des Zeitfenster-Managements gehören • Accounts für die Firmen, die Zugang zum Zeitfenster-Management haben möchten • Zu jeder Firma können 1 bis n Benutzer angelegt werden, die jeweils einen Login erhalten • Die verschiedenen Lagerstandorte, die verwaltet werden sollen Die Akteure können diese Daten anlegen, ansehen, ändern oder löschen

Tabelle D.10: Anwendungsfall: Stammdaten Zeitfenster verwalten

Nr.:	11
Anwendungsfall:	Stammdaten Dock und Yard verwalten
Ziel:	Die Stammdaten speziell für das Dock und Yard Management-System werden verwaltet
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister
Auslösendes Ereignis:	-
Beschreibung:	Zu den nötigen Informationen, die für das Dock- und Yard Management System nötig sind, sind folgende: • Verschiedene Lagerstandorte müssen pflegbar sein • Anzahl der Rampen • Besonderheiten der Rampen, wie z.B. Rampenhöhe • Zeiten, an der die Rampe benutzt werden darf Der Lagermeister kann diese Daten anlegen, ansehen, ändern oder löschen

Tabelle D.11: Anwendungsfall: Stammdaten Dock und Yard verwalten

Nr.:	12
Anwendungsfall:	Zeitfenster buchen
Ziel:	Ein Zeitfenster für eine Aktivität (Laden oder Entladen) gebucht zu bekommen
Kategorie:	must have
Vorbedingung:	Der Akteur muss ein Account haben und das zu buchende Zeitfenster muss eine Mindestzeitspanne in die Zukunft haben
Nachbedingung Erfolg:	Der Akteur erhält eine Zeitfenster-ID. Diese kann später der Fahrer zur Anmeldung am System nutzen
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, externer Disponent
Auslösendes Ereignis:	-
Beschreibung:	1. Der Akteur loggt sich ein 2. Der Akteur gibt ladungsspezifische Daten in das System ein, dabei kann er zwischen der manuellen Eingabe oder der Eingabe per ID wählen 3. In Abhängigkeit der ladungsspezifischen Daten schaltet das System die möglichen Zeitfenster frei 4. Der Akteur wählt ein Zeitfenster 5. Der Akteur kontrolliert die Daten und bestätigt diese, um das Zeitfenster erfolgreich zu buchen

Tabelle D.12: Anwendungsfall: Zeitfenster buchen

Nr.:	13
Anwendungsfall:	Lagerstandort wählen
Ziel:	Eine Auswahl treffen, für welchen Lagerstandort die weitere Bearbeitung erfolgen soll
Kategorie:	must have
Vorbedingung:	Die Lagerstandorte sind in den Stammdaten gepflegt
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, externer Disponent
Auslösendes Ereignis:	Ein Akteur möchte ein Zeitfenster buchen
Beschreibung:	Nach dem Einloggen ins System kann der Akteur einen Lagerstandort wählen. Die Wahl eines Lagerstandortes muss möglich sein, da die Firma über mehrere örtlich getrennte Standorte verfügen kann. Somit muss eine getrennte Verwaltung für jeden der Standorte möglich sein.

Tabelle D.13: Anwendungsfall: Lagerstandort wählen

Nr.:	14
Anwendungsfall:	Zeitfenster ändern
Ziel:	Ein bereits gebuchtes Zeitfenster zu ändern
Kategorie:	must have
Vorbedingung:	Die Zeitspanne zwischen dem Zeitpunkt des Änderungswunsches und des Zeitfensters muss eine bestimmte Grösse aufweisen
Nachbedingung Erfolg:	Bestätigung über die Änderung
Nachbedingung Fehlschlag:	-
Akteure:	interner Disponent, externer Disponent
Auslösendes Ereignis:	-
Beschreibung:	Der Akteur loggt sich ein. In Form einer Tabelle hat er eine Übersicht, wie die Belegung der Zeitfenster ist. Detailinformationen erhält er nur bei den gebuchten Zeitfenstern, die von seiner Firma gebucht worden sind. Er kann auch nur diese ändern. Ändern kann er die Zeit des Zeitfensters, d.h. er kann bspw. ein gebuchtes Zeitfenster nach hinten verschieben auf einen freien Platz. Dansch erhält er eine Bestätigung über die Änderung.

Tabelle D.14: Anwendungsfall: Zeitfenster ändern

Nr.:	15
Anwendungsfall:	Zeitfenster löschen
Ziel:	Ein gebuchtes Zeitfenster aus dem System zu entfernen
Kategorie:	must have
Vorbedingung:	Mit der ID des zu löschenden Zeitfensters wurde sich bisher noch nicht angemeldet
Nachbedingung Erfolg:	Das Löschen des Zeitfensters ist protokolliert
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister
Auslösendes Ereignis:	-
Beschreibung:	Da der Lagermeister für die Verwaltung des Lagers zuständig ist, hat nur er die Möglichkeit ein Zeitfenster zu löschen. Das Löschen eines Zeitfensters ist an keine bestimmten Bedingungen geknüpft, es muss nur protokolliert werden. Der Lagermeister wählt im System das entsprechende Zeitfenster. Er wählt die Funktion löschen und muss dann eine Begründung für das Löschen in Freitext eingeben. Danach erhält er eine Bestätigung über den Löschvorgang.

Tabelle D.15: Anwendungsfall: Zeitfenster löschen

Nr.:	16
Anwendungsfall:	Löschvorgang protokollieren
Ziel:	Der Löschvorgang eines Zeitfensters wird protokolliert
Kategorie:	must have
Vorbedingung:	-
Nachbedingung Erfolg:	-
Nachbedingung Fehlschlag:	-
Akteure:	Lagermeister
Auslösendes Ereignis:	Ein Zeitfenster wurde gelöscht
Beschreibung:	Wählt der Lagermeister ein Zeitfenster und löscht dieses, dann muss er zunächst einen Freitext als Begründung eingeben. Dann wird der Datensatz des Zeitfensters als gelöscht markiert zusammen mit dem Freitext und einem Zeitstempel

Tabelle D.16: Anwendungsfall: Löschvorgang protokollieren

Anhang E

Screenshots Software TestMaker

E.1 Skript

Abbildung E.1: generiertes Skript mittels Capturing

E.2 Darstellung Testverlauf

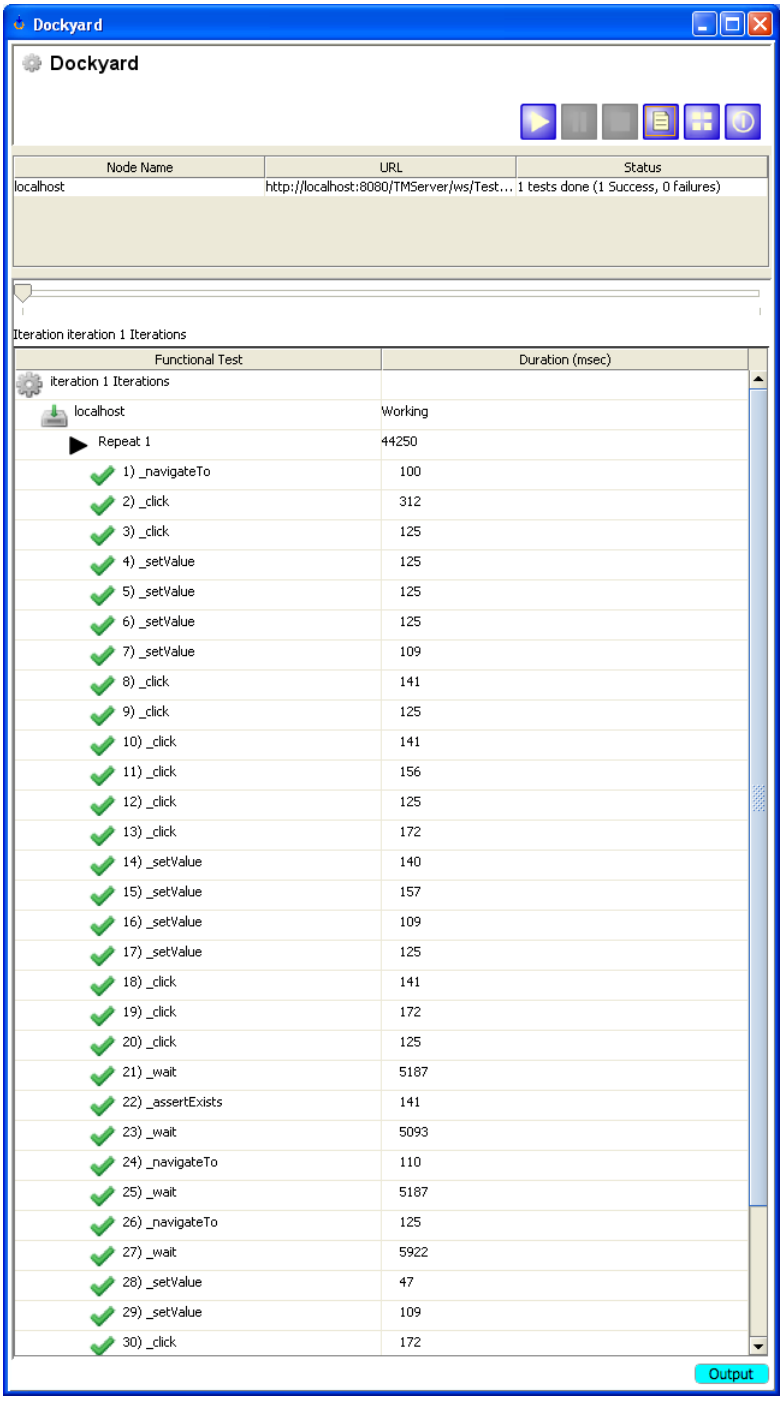

Abbildung E.2: Visualisierung des Testverlaufs

E.3 Logfile zum Testverlauf

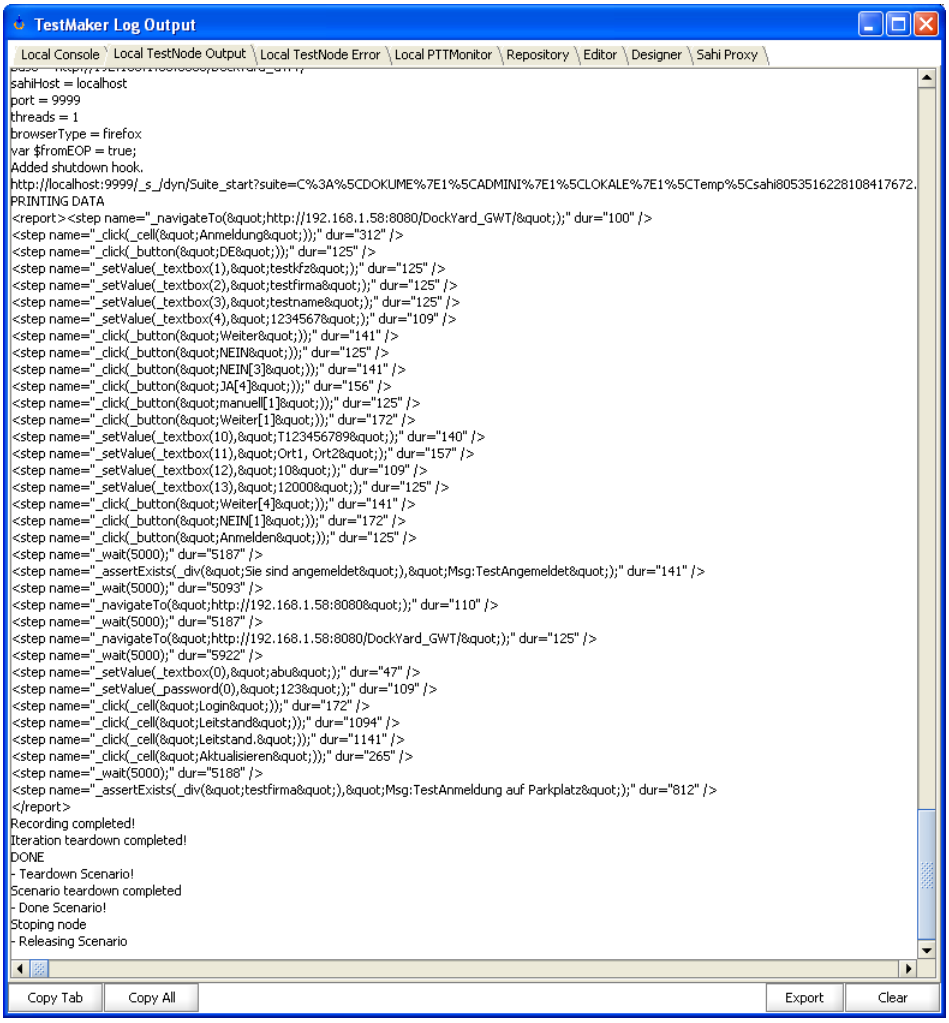

Abbildung E.3: Logfile zum Testverlauf

Literaturverzeichnis

[Bal00] BALZERT, Helmut: *Lehrbuch der Software-Technik - Software-Entwicklung*. Spektrum Akademischer Verlag, 2000. – ISBN 3–8274–0480–0

[BS11] BALSAMIQ STUDIOS, LLC: *Webseite Balsamiq*. http://balsamiq.com/. Version: 2011, Abruf: 20.05.2011

[BvJ09] BVJ/A.S.: Hofmanagement für Höchstleistungen. In: *LOGISTIK für Unternehmen* Sonderdruck (2009), Nr. 4/5

[Com11] COMMUNITY, Red Hat J.: *Webseite JBoss*. http://jboss.org. Version: 2011, Abruf: 20.05.2011

[DSD11] DHIMAN, Rohit ; SIGEL, Christian ; DÖRR, Jörg: *ISO/IEC 9126 Standard*. http://wwwagse.informatik.uni-kl.de/teaching/re/ws2010/ISO9126_Abstract.pdf. Version: 2011, Abruf: 17.07.2011

[Eur11] EUROPE, United Nations Economic C.: *CEFACT*. http://www.unece.org/cefact/. Version: 2011, Abruf: 16.04.2011

[fab11] FABFORCE.NET: *Webseite fabFORCE.net DBDesigner 4*. http://www.fabforce.net/dbdesigner4/. Version: 2011, Abruf: 11.07.2011

[Fis06] FISCHER, Oliver: Den Lkw am Gate per Mausklick steuern. In: *Deutsche Verkehrs Zeitung (DVZ)* BLOG (2006), Oktober

[Fou11] FOUNDATION, Apache S.: *Apache JMeter - JMeter is not a browser*. http://jmeter.apache.org/. Version: 2011, Abruf: 25.11.2011

[Fow04] FOWLER, Martin: *Inversion of Control Containers and the Dependency Injection pattern*. http://martinfowler.com/articles/injection.html#InversionOfControl. Version: 2004, Abruf: 14.05.2011

[GD10] GOEDICKE, Ina ; DEYMANN, Simon: Simulation von Strategien der Hoflogistik in Sortierzentren, KIT Scientific Publishing 2010, 2010, 253–260

[GH07] GILLERT, Frank ; HANSEN, Wolf-Rüdiger: *RFID - Für die optimierung von Geschüftsprozessen*. Carl Hanser Verlag, 2007. – ISBN 978–3–446–40507–3

[Gla11] *Webseite Glassfish*. http://glassfish.java.net. Version: 2011, Abruf: 20.05.2011

[Gmb02] GMBH, Gulp Information S.: *Unter der Lupe: Application Server*. http://www.gulp.de/kb/mk/chanpos/appserver.html. Version: 2002, Abruf: 10.05.2011

[Gmb11a] GMBH, IBM D.: *Webseite IBM Websphere*. http://www-01.ibm.com/software/de/websphere/. Version: 2011, Abruf: 20.05.2011

[Gmb11b] GMBH, SparxSystems S.: *Webseite Sparx Systems Enterprise Architect 9.0*. http://www.sparxsystems.de/uml/ea-function/. Version: 2011, Abruf: 07.07.2011

[Goo11a] GOOGLE: *Google code - Google Web Toolkit*. http://code.google.com/intl/de-DE/webtoolkit/. Version: 2011

[Goo11b] GOOGLE: *Google Web Toolkit - Showcase of Features*. http://gwt.google.com/samples/Showcase/Showcase.html#!CwCheckBox. Version: 2011, Abruf: 15.05.2011

[Han96] HANSEN, Hans R.: *Wirtschaftsinformatik 1*. Lucius und Lucius, 1996. – ISBN 3–8252–0802–8

[Hau06] HAUENSCHILD, Stefan: *Modellierung und Analyse der Geschäftsprozesse einer Hoflogistik mit UML2.0 und Petri-Netzen*, Universität Paderborn, Informatik, Bachelorarbeit, 2006. http://www2.cs.uni-paderborn.de/cs/ag-schaefer/Veroeffentlichungen/Quellen/Stud/2006/Studienarbeit_SHauenschild.pdf, Abruf: 21.03.2011

[Hob07] HOBKIRK, Ian: The Key to Getting the Most Out of Yard Management Systems. In: *Research Brief, Aberdeen Group* (2007), März. http://www.yardview.net/download/Aberdeen.pdf

[HT07] HANSON, Robert ; TACY, Adam: *GWT im Einsatz (dt. Fassung)*. Hanser, 2007. – ISBN 978–3–446–41241–5

[IHHK07] IHNS, Oliver ; HARBECK, Dierk ; HELDT, Stefan M. ; KOSCHEK, Holger: *EJB 3 professionell*. dpunkt.verlag, 2007. – ISBN 978–3–89864–431–0

[Inc11a] INC., Sencha: *Ext GWT Licensing Options*. http://www.sencha.com/products/extgwt/license/. Version: 2011, Abruf: 15.05.2011

[Inc11b] INC., Sencha: *Ext GWT Overview*. http://www.sencha.com/products/extgwt/. Version: 2011, Abruf: 15.05.2011

[Inc11c] INC., Sencha: *Ext GWT Samples and Demos*. http://www.sencha.com/products/extgwt/examples/. Version: 2011, Abruf: 15.05.2011

[JHA+08] JOHNSON, Rod ; HOELLER, Juergen ; ARENDSEN, Alef ; SAMPALEANU, Colin ; HARROP, Rob ; RISBERG, Thomas ; DAVISON, Darren ; KOPYLENKO, Dmitriy ; POLLACK, Mark ; TEMPLIER, Thierry ; VERVAET, Erwin ; TUNG, Portia ; HALE, Ben ; COLYER, Adrian ; LEWIS, John ; LEAU, Costin ; FISHER, Mark ; BRANNEN, Sam ; LADDAD, Ramnivas ; POUTSMA, Arjen: *The Spring Framework - Refernece Documentation (Chapter 3.3.1)*. http://static.springsource.org/spring/docs/2.5.x/reference/index.html. Version: 2008, Abruf: 14.05.2011

[Joc04] JOCHHEIM, Thorsten: *Dock & Yard Management Informationssysteme - Grundlagen, Geschäftsprozesse und Anwendungsszenarien*, Universität Paderborn, Wirtschaftsinformatik, Diplomarbeit, 2004. http://www.grin.com/e-book/31967/dock-yard-management-informationssysteme-grundlagen-geschaefts Abruf: 21.03.2011

[Joh11] JOHANNING, Ralf: Ohne Wartezeiten Güter abholen. In: *Deutsche Verkehrs Zeitung (DVZ)* (2011), Juni, Nr. 66/67

[Kai02] KAIB, Michael: *Enterprise Application Integration - Grundlagen, Integrationsprodukte, Anwendungsbeispiele*. DUV, 2002 http://books.google.de/books?id=9GVXCQjfJbMC&printsec=frontcover&dq=Enterprise+Application+Integration&source=bl&ots=klHY3F3NYr&sig=-9XB0pmUCcHZlYpz1TkY05MEiFM&hl=de&ei=0ZypTfneGYiBOqeUmM8J&sa=X&oi=book_result&ct=result&resnum=10&ved=0CG4Q6AEwCQ#v=onepage&q&f=false. – ISBN 3–8244–2163–1

[Kec09] KECHER, Christoph: *UML2 - Das umfassende Handbuch*. Galileo Press, 2009. – ISBN 978–3–8362–1419–3

[KPW10] KRUPP, Thomas ; PAFFRATH, Rainer ; WOLF, Johannes: *Praxishandbuch IT-Systeme in der Logistik*. Deutscher Verkehrs-Verlag, 2010. – ISBN 978–3–87154–425–5

[Küm11] KÜMMERLEN, Robert: Warenannahme muss schnell gehen. In: *Deutsche Verkehrs Zeitung (DVZ)* (2011), April

[Lix09] LIXENFELD, Christoph: *Standardsoftware muss draußen bleiben*. http://www.cio.de/2210899. Version: 2009, Abruf: 16.04.2011

[Möl07] MÖLLER, Prof.Dr.-Ing.D.P.F.: *Multimodaler Transport und Logistik*. Universität Hamburg - Seminarskript 18.168 WS 2007/08. http://www.informatik.uni-hamburg.de/TIS/files/MMTL_IP12.pdf. Version: 2007, Abruf: 10.04.2011

[NS08] NEUMANN, Larissa ; SZEWCZYK, Marcin ; RABE, Markus (Hrsg.): *Abbildung von Yard Management-Prozessen in Simulationsmodellen*. http://www.asim-fachtagung-spl.de/asim2008/papers/Proof_114-2b.pdf. Version: 2008, Abruf: 21.03.2011

[PBPMS04] PROCKL, Günter ; BAUER, Angela ; PFLAUM, Alexander ; MÜLLER-STEINFAHRT, Ulrich: *Entwicklungspfade und Meilensteine moderner Logistik*. Betriebswirtschaftlicher Verlag Dr. Th. Gabler/GWV Fachverlage GmbH, 2004. – 276–278 S. http://books.google.de/books?id=HwOTJL79FU0C&printsec=frontcover#v=onepage&q&f=false. – ISBN 3–409–12649–X

[Pfo10] PFOHL, Hans-Christian: *Logistiksysteme - Betriebswirtschaftliche Grundlagen*. Springer-Verlag, 2010. – ISBN 978–3–642–04161–7

[Rei09] REICHERT, Stefan: *Eclipse RCP im Unternehmenseinsatz*. dpunkt.verlag, 2009. – ISBN 978–3–89864–573–7

[Röm06] RÖMER, Kay: *Einführung in CORBA*. http://www.vs.inf.ethz.ch/edu/WS0607/VS/u2/corba.pdf. Version: 2006, Abruf: 10.05.2011

[Roz07] ROZANSKI, Uwe: *Enterprise JavaBeans 3.0 mit Eclipse und JBoss*. mitp, 2007. – ISBN 978–3–8266–1699–0

[Sav11] SAVELSBERG, Dr.Eva: Wie sich Hubs intelligent managen lassen. In: *Deutsche Verkehrs Zeitung (DVZ)* BKEP (2011), Januar

[Sch09a] SCHELTER, Sebastian: *Einführung in das Spring Framework*. http://www.inf.fu-berlin.de/inst/ag-se/teaching/S-BSE/134_schelter_spring_slides.pdf. Version: 2009, Abruf: 14.05.2011

[Sch09b] SCHULTE, Christof: *Logistik: Wege zur Optimierung der Supply Chain*. Vahlen, 2009. – ISBN 978–3800635160

[Sie04] SIEDERSLEBEN, Johannes: *Moderne Softwarearchitektur*. dpunkt.verlag GmbH, 2004. – ISBN 978–3–89864–292–7

[Sof11a] SOFTWARE, Isomorphic: *Smart GWT Showcase*. http://www.smartclient.com/smartgwt/showcase/#main. Version: 2011, Abruf: 15.05.2011

[Sof11b] SOFTWARE, Isomorphic: *SmartClient/SmartGWT Product Overview*. http://www.smartclient.com/product/. Version: 2011, Abruf: 15.05.2011

[Som07] SOMMERVILLE, Ian: *Software-Engineering*. Pearson Studium, 2007. – ISBN 978–3–8273–7257–4

[Sta07] STAHL, Jens: *Ajax mit dem Google Web Toolkit*. http://www.ordix.de/ORDIXNews/1_2007/Java_J2EE/google_web_toolkit_ajax.html. Version: 2007, Abruf: 15.05.2011

[Sta09] STARK, Thomas: *Java EE 5 - Einstieg für Anspruchsvolle*. Addison-Wesley, 2009. – ISBN 978–3827326485

[Sta11a] STANDARDIZATION, International O.: *ISO/IEC 25000:2005*. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35683. Version: 2011, Abruf: 12.07.2011

[Sta11b] STANDARDIZATION, The International O.: *EDIFACT*. http://www.gefeg.com/jswg/. Version: 2011, Abruf: 16.04.2011

[Ste07] STEYER, Ralph: *Google Web Toolkit schnell+kompakt*. entwickler.press, 2007. – ISBN 978–3–939084–37–2

[tbu11] TBU: Zeitfenster per Web. In: *Transport* (2011), Juni, Nr. 12

[TM11] TM, PushToTest: *Webseite PushToTest*. http://www.pushtotest.com/. Version: 2011, Abruf: 19.11.2011

[Tra11] TRAILERFORUM: Teure Wartezeiten an der Rampe. In: *Trailerforum Trends und Informationen von Krone* (2011), Nr. 1

[Tre09] TREBILCOCK, Bob: Beyond Yard Management. In: *Modern Material Handling, United States (MODEMATH), Thomson Gale* 64 (2009)

[Vog09] VOGLER, Anne: *Entwurfbeschreibung smartGWT*. http://pcai042.informatik.uni-leipzig.de/swp/SWP-09/swp09-5/Fremdprojekt.pdf. Version: 2009, Abruf: 26.04.2011

[VR01] VAWTER, Chad ; ROMAN, Ed: *J2EE vs. Microsoft.NET*. http://media.techtarget.com/tss/static/articles/pdf/J2EE-vs-DotNET.pdf. Version: 2001, Abruf: 10.05.2011

[Wik10] *Cross Docking*. http://de.wikipedia.org/wiki/Datei:Crossdocking.gif. Version: 2010, Abruf: 28.11.2011

[Wik11a] *ISO/IEC 9126*. http://de.wikipedia.org/wiki/ISO/IEC_9126. Version: 2011, Abruf: 17.07.2011

[Wik11b] *Java Platform, Enterprise Edition*. http://de.wikipedia.org/wiki/Java_EE. Version: 2011, Abruf: 14.05.2011

[Wik11c] *JUnit*. http://de.wikipedia.org/wiki/JUnit. Version: 2011, Abruf: 19.11.2011

[Wik11d] *Logistik*. http://de.wikipedia.org/wiki/Logistik. Version: 2011, Abruf: 20.05.2011

[Wik11e] *Martin-Notation*. http://de.wikipedia.org/wiki/Martin-Notation. Version: 2011, Abruf: 19.10.2011

[Wik11f] *Mock-up*. http://de.wikipedia.org/wiki/Mock-up. Version: 2011, Abruf: 20.05.2011

[Wik11g] *.NET*. http://de.wikipedia.org/wiki/.NET#Laufzeitumgebung. Version: 2011, Abruf: 10.05.2011

[Wik11h] *Spring (Framework)*. http://de.wikipedia.org/wiki/Spring_%28Framework%29. Version: 2011, Abruf: 14.05.2011

[Wüt10] WÜTHRICH, Jörg: *GWT 2.0*. http://www.infopoint-fhs.ch/_documents/11082010/Praesentation.pdf. Version: 2010, Abruf: 15.05.2011

[www07] WWW.GIGATON.DE: Warehouse-Management-Software Mit Dock- und Yard-Management. In: *FM Fracht + Materialfluss* (2007), Nr. 11, S. 47

[ZR06] ZUKUNFT, Olaf ; RAASCH, Jörg: *Architekturen von Informationssystemen*. HAW Hamburg - Vorlesungsskript WS 2006/07, 2006

[ZRK05] ZUKUNFT, Olaf ; RAASCH, Jörg ; KAHLBRANDT, Bernd: *Software Engineering*. HAW Hamburg - Vorlesungsskript Software-Engineering WS 2005/06, SS 2006, 2005

Glossar

Abk.:	Abkürzung
bspw.	beispielsweise
bzgl.	bezüglich
bzw.	beziehungsweise
ca.	circa
d.h.	das heisst
DYMS	Dock- und Yard Management System
EAI	Enterprise Application Integration
ECR	Efficient Consumer Response
ERD	Entity-Relationship-Modell
Gem.	Gemäß
ggf.	gegebenenfalls
GUI	Graphical User Interface
HTTP	Hyper Text Transfer Protocol
i.A.	im Allgemeinen
i.d.R.	in der Regel
ID	Identifikationsnummer
IT	Informationssystem
Java EE	Java Enterprise Edition

JIT Just-in-Time

LAN Local Area Network

LKW Lastkraftwagen

LVS Lagerverwaltungssystem

RDBMS Relational Database Management System

SDSL Symmetric Digital Subscriber Line

sog. sogenannte

URI Uniform Resource Identifier

usw. und so weiter

VPN Virtual Private Network

YMS Yard Management System

z.B. zum Beispiel

z.T. zum Teil

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 08.12.2011 André Bürger