DIPLOMARBEIT

DYNAMISCHES ROUTING UND VERSPÄTUNGSVORHERSAGE VON BEDARFSGESTEUERTEN LINIENBUSSEN AUFGRUND VON GPS MESSDATEN

Ausgeführt zum Zweck der Erlangung des akademischen Grades eines
Dipl.-Ing. (FH) für Computersimulation
am Fachhochschul-Diplomstudiengang Computersimulation St. Pölten

unter der Leitung von

Mag. Otto Reichel

Dipl.-Ing. Dr. Christian Harlander

ausgeführt von

Christian Alois Sterzl
si011040

St. Pölten, am 12. September, 2005 Unterschrift:

EHRENWÖRTLICHE ERKLÄRUNG

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der vom Begutachter beurteilten Arbeit überein.

....

Ort, Datum Unterschrift

Für meine Mutter,

Margarete Sterzl

(† 04. August 2005)

Kurzfassung

Diese Arbeit behandelt alternative Bedienungsformen im öffentlichen Verkehr (kurz: ÖPNV) und deren Besonderheiten im Hinblick auf Fahrzeugortung, Informationsmanagement und Verspätungsvorhersage im Allgemeinen und das Projekt „BEHA - Bedarfshaltestellen“ im Besonderen. Die vorliegende Arbeit beschäftigt sich mit den Besonderheiten der alternativen Bedienungsformen und den darauf angepassten technischen Implementationen.

Zunächst wird das Projekt BEHA und dessen Entwicklungsgeschichte vorgestellt, bevor in der Einleitung Forschungsmethoden und -strategien vorgestellt werden. Danach wird BEHA im Detail betrachtet und es werden Vergleiche zu anderen Systemen des ÖPNV gezogen. Die einzelnen technischen Komponenten des BEHA-Systems werden ebenfalls genauer betrachtet. Hier sind vor allem zwei physikalisch voneinander getrennte Applikationen von Interesse:

- BusApplikation
- RoutingModul

Die BusApplikation ist eine J2ME-Applikation für Mobiltelefone, welche als Schnittstelle zum BEHA-System für die Busfahrer dient. Für die Fahrzeugortung wird die BusApplikation zum Sammeln von Positionsdaten via GPS verwendet. Die besondere Schwierigkeit liegt hier in der intelligenten Datenreduktion und der Programmierung für *Embedded Systems*. Neben der Software selbst wird auch die Wahl der Hardware beschrieben.

Das RoutingModul ist ein Framework für mehrere von BEHA genutzten Services. Zum einen dient es der Verarbeitung der von der BusApplikation kommenden Positionsdaten, zum anderen dem dynamischen Routing und der Verspätungsvorhersage. Auch weitere Services, wie etwa eine Anschlusssicherung, sind mit Hilfe des RoutingModuls denkbar.

Ein Ausblick auf zukünftige Entwicklungen und eine Zusammenfassung runden das Thema ab.

Abstract

This thesis deals with alternative services in public transport in general and especially with the project BEHA, a public on-demand bus service. Furthermore, it deals particularly with location of vehicles, management of information and delay-prediction. The technical implementation of such services is of special interest.

First, the project BEHA is presented in the introduction. Afterwards the strategy and the methods of research are given. The next part looks at the project BEHA in detail and all technical components. A comparison of BEHA with other systems in public transport is given too. The third and fourth part of this thesis describe two physically separated applications of BEHA:

- BusApplication
- RoutingModule

The BusApplication is a J2ME-based application for mobile phones. It serves as an interface BEHA and the drivers. Especially the location and collecting of GPS data is described. There will be also a discourse of the selection of the hardware.

The RoutingModule is a java-based framework for some of the services used by BEHA. It is used for processing of the incoming data from the BusApplication, for dynamic routing and delay-prediction. Other services like connection reliability are possible as well.

Finally, there are a summary and a discussion about the developments that can be expected in the future.

Inhaltsverzeichnis

Abkürzungsverzeichnis

Abb.	**Abb**ildung
API	**A**pplication **P**rogramming **I**nterface
ASF	The **A**pache **S**oftware **F**oundation
BEHA	**Be**darfs**ha**ltestellen
BMVIT	**B**undes**m**inisterium für **V**erkehr, **I**nnovation und **T**echnologie
bzw.	**bez**iehungs**w**eise
C/A-Code	**C**oarse **A**quisition
CAD	**C**omputer **A**ided **D**esign
CLDC	**C**onnected **L**imited **D**evice **C**onfiguration
CSD	**C**ircuit **S**witched **D**ata
d. h.	**d**ass **h**eißt
DGPS	**D**ifferential **GPS**
DRMS	**D**istance **R**oot **M**ean **S**quare
etc.	**et c**etera (lat.: und so weiter)
EU	**E**uropäische **U**nion
evtl.	**ev**en**t**u**e**ll
GFC	**G**eneric **C**onnection **F**ramework
ggf.	**g**e**g**ebenen**f**alls
GIS	**G**eographic **I**nformation **S**ystem
GLONASS	**Glo**bal'naya **na**vigatsionannaya **S**putnikovaya **S**istema
GPRS	**G**eneral **P**acket **R**adio **S**ervice
GPS	**G**lobal **P**ositioning **S**ystem
GSM	**G**lobal **S**ystem for **M**obile Communications
HSCSD	**H**igh **S**peed **C**ircuit **S**witched **D**ata
HTML	**H**yper**t**ext **M**arkup **L**anguage
HTTP	**H**yper**t**ext **T**ransfer **P**rotocol
IBM	**I**nternational **B**usiness **M**achines
ISDN	**I**ntegrated **S**ervices **D**igital **N**etwork
IV	**I**ndividual**v**erkehr
J2ME	**J**ava **2** Platform, **M**icro **E**dition
JCP	**J**ava **C**ommunity **P**rocess
JLS	**J**ava **L**anguage **S**pecification
JMS	**J**ava **M**essage **S**ervice
JSR	**J**ava **S**pecification **R**equests
JVM	**J**ava **V**irtual **M**achine
LIFO	**L**ast **I**n **F**irst **O**ut
LISB	**L**eit- und **I**nformations**s**ystem **B**erlin
LSB	**L**ast **s**ignificant **B**it
MGI	**M**ilitary **G**eographic **I**nstitute

MIDP	**M**obile **I**nformation **D**evice **P**rofile
move	**Mo**bilität und **Ve**rkehrstechnologie
NMEA	**N**ational **M**arine **E**lectronics **A**ssociation
OIR	**O**ptimierte **I**ntermodale **R**eisezeitprognose
PKW	**P**ersonen**k**raft**w**agen
PPS	**P**recision **P**ositioning **S**ervice
RBL	**r**echnergesteuertes **B**etriebs**l**eitsystem
RS-232	**R**ecommended **S**tandard 232
S.	**S**eite
SMS	**S**hort **M**essage **S**ervice
sog.	**sog**enannt
SPFV	**S**chienen**p**ersonen**f**ern**v**erkehr
SPS	**S**tandard **P**ositioning **S**ervice
TMS	**T**ransport **M**anagement **S**ystem
UKW	**U**ltra**k**urz**w**elle
UMTS	**U**niversal **M**obile **T**elecommunications **S**ystem
UTC	**U**niversal **T**ime **C**oordinated
VDV	**V**erband **D**eutscher **V**erkehrsunternehmen
WAP	**W**ireless **A**pplication **P**rotocol
XML	E**x**tensible **M**arkup **L**anguage
z. B.	**z**um **B**eispiel
ÖBB	**Ö**sterreichische **B**undes**b**ahnen
ÖPNV	**Ö**ffentlicher **P**ersonen**n**ah**v**erkehr
ÖV	**Ö**ffentlichen **V**erkehr

Formatierungshinweise

Abkürzungen werden bei ihrem ersten Auftreten im Text neben der Abkürzung in Klammer mit den Anfangsbuchstaben fettgedruckt ausgeschrieben.

Fremdsprachige Ausdrücke sind kursiv gedruckt.

URLs (**U**niform **R**esource **L**ocator) werden folgendermaßen geschrieben:
`www.fh-stpoelten.ac.at`, `12. September, 2005`

Beim ersten Auftreten einer Firma im Text folgt in Klammer die URL zum Webauftritt dieser Firma.

Zitiert wird nach dem *Harvard System* (`www.bournemouth.ac.uk/academic_services/documents/Library/Citing_References.pdf`, `12. September, 2005`).

Einleitung

0.1. Definition des Erkenntnisgegenstandes

Das Projekt „BEHA - **Be**darfs**ha**ltestellen“ wurde im Jahr 2000 im Rahmen einer Technologieoffensive der österreichischen Bundesregierung gestartet. Dabei hat das BMVIT (**B**undes**m**inisterium für **V**erkehr, **I**nnovation und **T**echnologie, `www.bmvit.gv.at`, `12. September, 2005`) das Impulsprogramm „move - **Mo**bilität und **Ve**rkehrstechnologie“ ins Leben gerufen.

> *Das Impulsprogramm „move - Mobilität und Verkehrstechnologie“ des BMVIT hat eine Reihe von Leitaktionen und Schwerpunkten zur Veränderung des österreichischen Verkehrssystems in Richtung Mobilität der Zukunft gesetzt. [Gorbach (November 2003), S. 3]*

0.1.1. Das Projekt „BEHA - Bedarfshaltestellen“

Die Firma CoCo MobileTelematics GmbH (`www.coco-mobile.at`, `12. September, 2005`) hat in Zusammenarbeit mit der ÖBB-Postbus GmbH (**Ö**sterreichische **B**undes**b**ahnen; `www.postbus.at`, `12. September, 2005`; `www.oebb.at`, `12. September, 2005`), VOR (**V**erkehrsverbund **O**st–**R**egion Gesellschaft m.b.H.; `www.vor.at`, `12. September, 2005`) und dem BMVIT das Projekt entwickelt und umgesetzt.
Das BMVIT beschreibt die Problemstellung dabei folgendermaßen:

> ***IST-Zustand - Problemstellung***
>
> *Der öffentliche Personenverkehr im Busbereich in Österreich ist insbesondere in ländlichen Bereichen von Haltestellen mit geringem Fahrgastaufkommen und unregelmäßiger, zeitlich schwer prognostizierbarer Nachfrageverteilung gekennzeichnet. Hohe Leerfahrtsanteile sind die Folge. Zudem ist die Anbindung an das öffentliche Liniennetz vielfach durch einen erheblichen Erschließungsaufwand (notwendige Abweich- und Stichfahrten von Hauptachsen) gekennzeichnet, wodurch die Attraktivität, bedingt durch lange Reisezeiten, für die Mehrzahl der Fahrgäste sinkt. Um einen hohen Erschließungsgrad unter geringstmöglichem Ressourceneinsatz und hoher Angebotsattraktivität zu gewährleisten, wurde das Projekt zur Einrichtung von Be-*

> *darfshaltestellen entwickelt. Durch das Bedarfshaltestellensystem (BEHA-System) können bestehende Bedienungsgebiete im regionalen Kraftfahrlinienverkehr effizienter erschlossen bzw. neue Erschließungsgebiete mit geringem zusätzlichen Aufwand und unter geringen zusätzlichen Reisezeiten in das Bedienungsangebot integriert werden. Das BEHA-System wurde so konzipiert, dass keine zusätzlichen Humanressourcen für die Abwicklung des laufenden Betriebes notwendig sind. [Stütz (November 2003), S. 12]*

Das BEHA-System ist derzeit (Stand: 12. September, 2005) auf 5 Linien in drei österreichischen Gebieten im Einsatz: Nördlich von Wien in der Kastralgemeinde Gaweinstal, sowie südlich von Wien in der Gemeinde Wienerwald. Beide Gemeinden weisen eine verteilte Siedlungsstruktur auf. Im dritten Gebiet, der Stadt Zwettl im nördlichen Waldviertel, beweist sich das BEHA-System erstmals in einem Stadtgebiet. [CoCo (2005), Screen 1]

0.1.1.1. Der Funktionsablauf

- An der Haltestelle ist eine Box mit Drucktasten und Display montiert. Die Anmeldung (5 bis 10 Minuten vor Fahrplanabfahrtszeit) kann direkt an der Bedarfshaltestelle (Abb. 0.1.1, S. 4) per Tastendruck oder auch per Mobiltelefon, Internet bzw. per Sprachtelefonie oder bei dem/der Busfahrer/in direkt durchgeführt werden. Die Anmeldung an der Box ist kostenlos. Bei den anderen Möglichkeiten zur Anmeldung fallen die normalen Gebühren für das Gespräch bzw. die Kurznachricht an.

- Von der Bedarfshaltestelle (bzw. vom Mobiltelefon) wird mittels SMS (**S**hort **M**essage **S**ervice) bzw. GPRS (**G**eneral **P**acket **R**adio **S**ervice) eine Nachricht an eine Rechnerzentrale gesendet, in der über ein Hintergrundsystem mit zugrunde liegenden Fahrplan- und Dienstplandaten automatisch eine Benachrichtigung an den/die entsprechende/n Buslenker/in auf ein Mobiltelefon ergeht.

- Der/Die Buslenker/in erhält so die Information, diese Bedarfshaltestelle anzufahren und bestätigt mittels Mobiltelefon die Ankunft an der gewünschten Bedarfshaltestelle bzw. gibt gegebenenfalls Verspätungen an.

- Der bei der Bedarfshaltestelle wartende Fahrgast wird dann über den Zeitpunkt der Busankunft (Fahrplanzeit und ggf. Verspätungen) informiert. Im Fall einer Bestellung per Mobiltelefon wird die Ankunftszeit auf das Telefon übermittelt. [Stütz (November 2003), S. 12f]

Abbildung 0.1.1.: Der Terminal an der Bedarfshaltestelle hat 3 Tasten für die Bedienung: 2 Tasten zur Auswahl einer Fahrt, 1 Taste zum Bestellen. [eigene Aufnahme, 2. August, 2005]

0.1.1.2. Die BusApplikation

Die BusApplikation ist die Schnittstelle von Busfahrer/in und dem BEHA-System. Die Java-Applikation auf dem Mobiltelefon ermöglicht es ihm mittels einfacher Bedienung mit der Rechnerzentrale zu kommunizieren. Für das Routing wurde diese BusApplikation erweitert, so dass GPS-Daten (**G**lobal **P**ositioning **S**ystem) eines angeschlossenen GPS-Empfängers ausgelesen, komprimiert und versendet werden können.

0.1.1.3. Das RoutingModul

Das RoutingModul, das dieser Arbeit zugrunde liegt, ist ein Zusatzmodul zu BEHA zur digitalen Umsetzung einer Busfahrt. Auf diesem Modul bauen Services wie etwa die Verspätungsvorhersage, Fahrzeugverfolgung oder Tools zur Auswertung von Busfahrten

auf. Des Weiteren wird dadurch eine Entlastung der Busfahrer/innen erreicht, da automatisch, sofern ein GPS-Empfänger im Bus an das Mobiltelefon des/der Buslenkers/in angeschlossen ist, die Position des Busses erfasst wird und der/die Buslenker/in somit nicht mehr jede angefahrene Bedarfshaltestelle bestätigen muss. Das RoutingModul entscheidet außerdem, ob eine Erweiterung der Fahrt zur Bedienung der gewünschten Haltestelle möglich ist. Dadurch soll auch eine Verkürzung der nötigen Voranmeldezeit des Fahrgastes erreicht werden. Musste sich der Fahrgast bisher noch mindestens 5 bis 10 Minuten vor einer Fahrt anmelden, kann so etwas kurzfristiger auf eine Bestellung reagiert werden.

Erzeugen der Routendaten

Das RoutingModul benötigt eine Vielzahl von Daten über eine Region in Form eines Graphen. Die Eingabe der Daten erfolgt jedoch manuell, da eine Anbindung an eine GIS- (**G**eographic **I**nformation **S**ystem) oder CAD- (**C**omputer **A**ided **D**esign) gestützte Datenbank der Straßenverwaltungen während der Entwicklung des Prototyps (November 2004 - Juni 2005) nicht möglich war. So entstanden im Laufe des Projektes einige Hilfstools zur Dateneingabe und -evaluierung, auf die später noch näher eingegangen wird.

0.1.2. Erkenntnisgegenstand

Diese Arbeit soll in erster Linie eine Machbarkeitsanalyse der oben beschriebenen Erweiterung von BEHA liefern. Zunächst wird das Projekt „BEHA“ vorgestellt. Danach wird gezeigt, wie sich das RoutingModul in das Gesamtbild einfügt. Anschließend wird die BusApplikation analysiert und gezeigt, wie mittels eines Mobiltelefons GPS-Daten eingelesen werden können und welche Anforderungen an die Hard- und Software dabei gestellt werden.

Das RoutingModul wird bei Abschluss dieser Arbeit aller Voraussicht nach noch keine Marktreife erlangt haben. Daher wird ein Prototyp einer Eigenentwicklung vorgestellt und analysiert. Dieser soll anhand von zwei Services, die das RoutingModul nutzen, auf seine Brauchbarkeit analysiert werden. Zum einen wird ein Tool zur Erstellung von Statistiken über die Fahrtleistung eines Busses, zum anderen eine Verspätungsvorhersage überprüft.

0.1.3. Relevanz des Themas

> ***Warum move*** *- Verkehr ist ein Kernthema der Wirtschafts- und Technologiepolitik. Ein leistungsfähiges Verkehrssystem bleibt auch im Informations-*

> *und Kommunikationszeitalter Voraussetzung für einen attraktiven Wirtschaftsstandort. Vom Verkehrssektor gehen zudem starke Beschäftigungs- und Investitionsimpulse für die gesamte Wirtschaft aus. Das steigende Verkehrsaufkommen beeinträchtigt alle Bereiche der Wirtschaft und Gesellschaft - Verkehrs- und Technologiepolitik soll das Gesamtsystem optimieren, sowohl ökologisch nachhaltig wie ökonomisch effizient und sozial verträglich. move trägt zur Verkehrsvermeidung und Reduzierung der negativen Folgeerscheinungen des Verkehrs bei. [Stütz (November 2003), S. 4]*

Neben der Verbesserung der allgemeinen Wirtschaftlichkeit des ÖPNV (**Ö**ffentlicher **P**ersonen**n**ah**v**erkehr) und der damit verbundenen Modernisierung wird auch die Qualität des Services für die Konsumenten, im Falle des ÖPNV für die Fahrgäste, verbessert. Verspätungsvorhersagen und automatisierte Fahrplanabfrage helfen das Vertrauen des Fahrgastes in den ÖPNV zu heben und den Individualverkehr zu verringern.

0.2. Problembenennung

0.2.1. Anforderungsprofil Hard- und Software

Eine Vielzahl an Mobiltelefonen wird heutzutage angeboten. Eine große Zahl dieser wird den gestellten Anforderungen (serielle Schnittstelle, Java-fähig, etc.) jedoch nicht gerecht. Da bei der Anschaffung eines Mobiltelefons stark auf den Preis geachtet wird, muss die Software mit minimalen Anforderungen laufen. Bei der Erweiterung um GPS, darf die derzeit laufende BusApplikation allerdings nicht beeinträchtigt werden.

Ähnlich verhält es sich bei der Architektur und Konzeption des RoutingModuls. Die derzeitige auf den Servern laufende Software darf durch das RoutingModul nicht beeinträchtigt werden und muss dennoch eine große Datenmenge in Echtzeit verarbeiten können. Verzögerungen führen im Regelfall zu Ärgernissen bei den Fahrgästen.

0.2.2. Analyse und Effizienz

Es müssen also optimale Algorithmen und eine gute Datenstruktur gefunden werden, damit das Routing effizient und verlässlich vor sich geht. Darüber hinaus sollen die Entscheidungen für die Wahl der Datenstrukturen und Algorithmen sowie deren Funktionsweisen dokumentiert und begründet werden.

Nebenbei sollen Benchmarktests die Effizienz des RoutingModuls beweisen und Entscheidungsgrundlagen für spätere Optimierungen bilden.

0.3. Forschungsleitende Fragestellung

0.3.1. Anforderungen an die Hardware

Es ist ein Anforderungsprofil zu erstellen und mittels Recherche wird eine kleine Anzahl von Mobiltelefonen ausgewählt und getestet.

0.3.2. Anforderungen an die Software

Die Software ist aufgeteilt in die BusApplikation und das RoutingModul selbst. Für beide Teile ist ein Anforderungsprofil zu erstellen. In weiterer Folge werden die Einschränkungen aufgezeigt, welche sich durch die Einbettung in das vorhandene System ergeben.

0.3.3. Funktionsweise

Es ist die grundsätzliche Funktionsweise des RoutingModuls zu dokumentieren und auf ihre Richtigkeit zu überprüfen.

0.3.4. Automatisierbarkeit

Da das RoutingModul selbst nach einmaliger Dateneingabe ohne Benutzerinteraktion läuft, müssen verschiedene Analysetools die korrekte Funktionsweise überprüfen. Diese soll anhand einiger Tests bewiesen werden. Darüber hinaus sind Überlegungen anzustellen, wie das System gewartet werden kann.

0.3.5. Datenmanagement

Das Routing basiert auf einem Graphen, welcher eine Region, in der ein bedarfsgesteuerter Linienbus fährt, darstellt. Dieser wird kategorisiert und sein Aufbau wird erklärt. Der Graph wird in einer Datenbank gespeichert, daher müssen effiziente Methoden zum Laden, Speichern und Löschen eines Graphen gefunden werden. Neben den rein graphrelevanten Daten müssen auch noch Fahrplandaten verarbeitet werden.

Trotz großer zu verarbeitender Datenmengen müssen die darauf angewandten Routingalgorithmen schnell und effizient ablaufen. Darum werden verschiedene Routingalgorithmen implementiert, getestet und verglichen.

Jene von der BusApplikation kommenden GPS-Daten müssen den Kanten im Graphen zugeordnet werden. Dieser Vorgang wird als *Matching* bezeichnet. Die daraus bezogenen Daten dienen später unter anderem der Verspätungsvorhersage. Sie müssen daher auch aufgezeichnet werden. [Wiltschko (Juni 2005), Screen 1]

In dieser Arbeit wird jedoch nur die grundsätzliche Funktionsweise des *Matchings* geklärt.

0.3.6. Wie sieht eine ideale Verspätungsvorhersage aus?

In dieser Arbeit werden auch die Grundlagen für eine Verspätungsvorhersage bei BEHA erarbeitet. Darum wird theoretisch die ideale Verspätungsvorhersage behandelt, parametrisiert und beschrieben.

Nebenbei soll ein Vergleich zu schon bestehenden Verspätungsvorhersagen im ÖPNV gezogen werden.

0.3.7. Ausblick auf zukünftige Optimierungen und alternative Anwendungen für das RoutingModul

Das RoutingModul ist seit 1. Juni 2005 im Einsatz mit einem Service zur Erstellung von Statistiken über die Fahrtleistung eines bedarfsgesteuerten Linienbusses. Darum kann schon ein erstes Resumee über die Schwachstellen des RoutingModuls gezogen werden. Es werden daher Überlegungen zur Optimierung angestellt und wie die Anwendung des Systems erweitert und auf andere Bereiche ausgeweitet werden kann.

0.4. Forschungsstrategie

Diese Arbeit umfasst drei grundlegende Teile. Nach der Einleitung im ersten Teil, folgen die konzeptionellen Erkenntnisse im zweiten Teil. Dieser zweite Teil wurde analog zu den physischen Komponenten des Systems in zwei Teile, die BusApplikation und das RoutingModul, getrennt. Ausblick und Zusammenfassung bilden den dritten Teil dieser Arbeit.

Durch eine Literaturanalyse größtenteils deutschsprachiger Publikationen zum Thema „BEHA“ werden Teile der Einleitung und die konzeptionellen Grundlagen erarbeitet und hergeleitet. Anhand dieser gewonnenen Erkenntnisse wird ein Aufbau erstellt, der die Formulierung von Problemdefinitionen, die Zerlegung umfassender Probleme in einfachere Teilprobleme und die Entwicklung entsprechender Lösungsansätze erleichtern soll.

Der zweite Teil beschreibt neben den Konzepten die bisher durchgeführte Implementation und zeigt qualitativ und quantitativ die Gründe für die gewählten Lösungen auf. In methodisch aufgebauten, funktionsorientierten Experimenten werden die Teillösungen der Implementation sowohl der BusApplikation, als auch des RoutingModuls verifiziert. Bei den noch nicht implementierten Systemteilen beschränkt sich die Arbeit auf eine Konzeptbeschreibung und - soweit vorhanden - Analyse der bereits erarbeiteten Lösungsvorschläge.

Der dritte Teil soll klären, inwieweit verschiedene Probleme gelöst werden konnten.

Teil I.

Das Projekt BEHA im Detail

1.1. Alternative Bedienungsformen im ÖPNV

Im ländlichen und suburbanen Raum Österreichs werden seit Beginn des Privatisierungsprozesses zunehmend verstärkte Anstrengungen zur Senkung der Kosten des öffentlichen Verkehrs unternommen. Hierbei spielen alternative Bedienungsformen eine wesentliche Rolle, um durch höhere Kosteneffizienz ein flächendeckendes und nutzerfreundliches ÖPNV-Angebot aufrecht erhalten zu können. Bei der Konzeption alternativer Bedienungsformen sind das Fahrgastaufkommen und die Kosten-/Einnahmensituation ebenso wie im traditionellen Linienverkehr wichtige Grundlagen zur Entscheidungsfindung für die Betreiber. [Walther (Juni 2002), Screen 1]

1.1.1. Begriffsbestimmungen

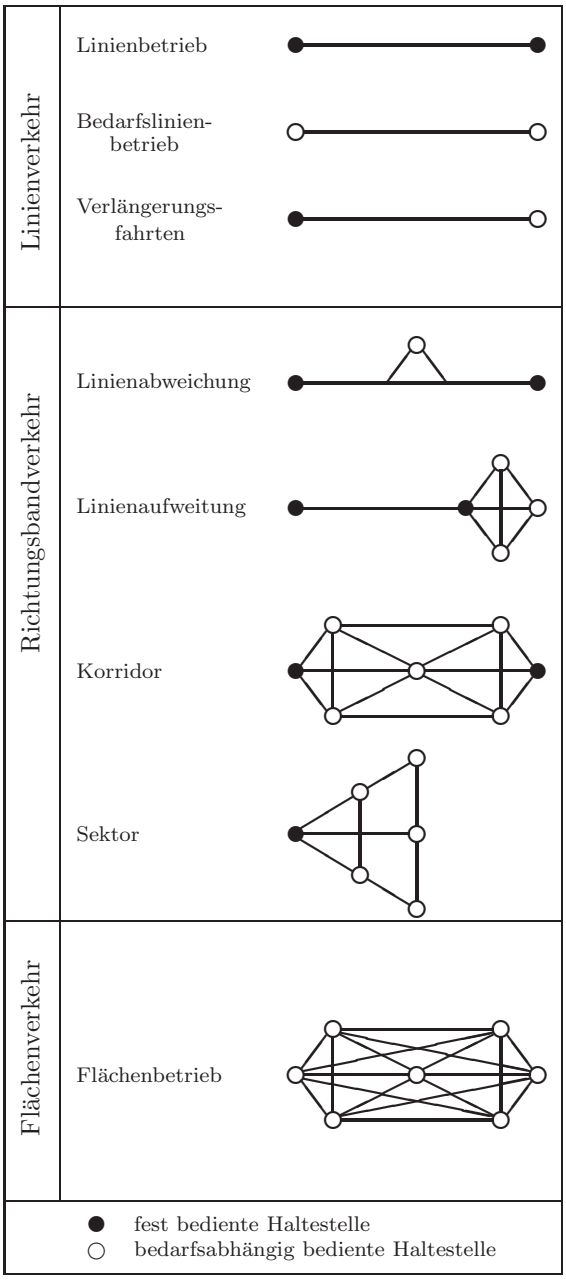


Abbildung 1.1.1.: Betriebsformen nach räumlicher Ausprägung [Eder (2001), S. 55]

Die verschiedenen Bedienungsformen können sowohl nach ihrer räumlichen als auch nach ihrer zeitlichen Ausprägung unterteilt werden. Tabelle 1.1.1 zeigt die verschiedenen Arten der alternativen Betriebsformen kategorisiert nach ihrer räumlichen Ausprägung.

Der Linienverkehr zeichnet sich durch eine fixe Linienführung aus. Beim Richtungsbandverkehr wird von einem fixen Startpunkt aus eine Fläche bzw. eine Region in eine bestimmte Richtung bedient und führt entweder wieder zum Startpunkt, wie beim Sektor oder der Linienaufweitung, zurück oder zu einem Zielpunkt, wie bei der Linienabweichung oder dem Korridor. Der Flächenverkehr ermöglicht Quell-Ziel-Kombinationen unabhängig von Richtung und Startpunkt.

Trotz der räumlichen Flexibilisierung sind noch immer Leerfahrten, welche eigentlich verhindert werden sollten, möglich. Eine zeitliche Flexibilisierung wird durch eine bedarfsabhängige Betriebsweise erreicht, womit die unerwünschten Leerfahrten auf ein Minimum reduziert werden können.

Die zeitliche Ausprägung unterscheidet nach den beiden markanten Merkmalen:

- fahrplangebunden/fahrplanungebunden
- bedarfsabhängig/bedarfsunabhängig

Aus der Kombination sowohl der räumlichen, als auch der zeitlichen Unterscheidungsmerkmale ergeben sich nun die verschiedenen alternativen Betriebsformen, aus denen sich unterschiedliche Bezeichnungen ergeben. Die Nomenklatur ist in Tabelle 1.1.1 angeführt.

		zeitliche Ausprägung		
räumliche Ausprägung		fahrplangebunden bedarfsunabhängig	fahrplangebunden bedarfsabhängig	fahrplanungebunden bedarfsabhängig
	Linienverkehr	Linienverkehr Bürgerbus	Bedarfs-Linien-Bus Bedarfs-Linien-Taxi	-
	Richtungsbandverkehr	Festzeit-Sammel-Taxi	Anruf-Sammel-Bus Anruf-Sammel-Taxi	Festpunkt-Sammel-Taxi
	Flächenverkehr	-	-	Anruf-Bus Anruf-Taxi

Tabelle 1.1.1.: Alternative Betriebsformen nach räumlicher und zeitlicher Ausprägung [Eder (2001), S. 56]

Eine weitere Effizienzsteigerung kann durch eine kapazitive Flexibilisierung erreicht werden, d. h. eine Anpassung der Fahrzeuggrößen an die jeweilige Bedienungsform. [Hoffmann (1993), S. 37]

Da das BEHA-System derzeit nur von der österreichischen Postbus AG in Österreich betrieben wird und deren Fuhrpark homogen aus Standardbussen besteht, welche 60 bis 90 Fahrgästen Platz bieten, zudem BEHA-Busse immer nur einen Teil ihrer Fahrten bedarfsabhängig bestreiten, kann an dieser Stelle auf eine Einteilung der verschiedenen Bedienungsformen nach der optimalen Fahrzeuggröße verzichtet werden. Die Zusammenhänge zwischen den Betriebsformen und Fahrzeuggrößen können an folgenden Stellen nachgelesen werden:

- [Eder (2001), S. 56ff]
- [Mehlert (Juni 1998), S. 56ff]
- [Hoffmann (1993), S. 38ff]
- [Hoopmann (April 1997), Screen 8f]

1.1.2. BEHA als Bedarfslinienverkehr

Das BEHA-System fällt in die Kategorie des Bedarfslinienverkehrs, genauer gesagt ist BEHA ein Linienverkehr mit teilweiser Bedarfssteuerung. Das BEHA-System ermöglicht aber grundsätzlich alle Möglichkeiten des Linienverkehrs und des Richtungsbandverkehrs. Der Flächenbetrieb ist aufgrund der Bindung an einen Fahrplan nicht möglich. Das BEHA-System ist fahrplangebunden und bedarfsabhängig und ermöglicht dadurch sowohl eine räumliche als auch eine zeitliche Flexibilisierung, weshalb Fahrgastströme zu den Stoßzeiten, etwa Schülerfahrten oder Pendlerfahrten, bewältigt werden können, genauso wie geringerer Freizeit- und Einkaufsverkehr ökonomischer geführt werden können.

Das BEHA-System weist alle Merkmale eines gewöhnlichen Linienverkehrs auf:

- Fahrplan

- Haltestellen
- Linientarif

Dadurch bleibt die gewohnte Bedienungsqualität erhalten. Bei einer kompletten Umstellung auf z. B. einen Flächenbetrieb wäre diese nicht mehr garantierbar gewesen.

1.2. BEHA im Einsatz

In der Kastralgemeinde Gaweinstal im Bezirk Mistelbach ist das BEHA-System seit 2002 auf der Linie 1018 installiert. Am Beispiel Gaweinstal lassen sich einige mögliche Anwendungsgebiete für BEHA ablesen. Es wurden Verlängerungsfahrten, Linienabweichungen, -aufweitungen und Sektoren realisiert. Abbildung 1.2.1 stellt das Gebiet schematisch dar.

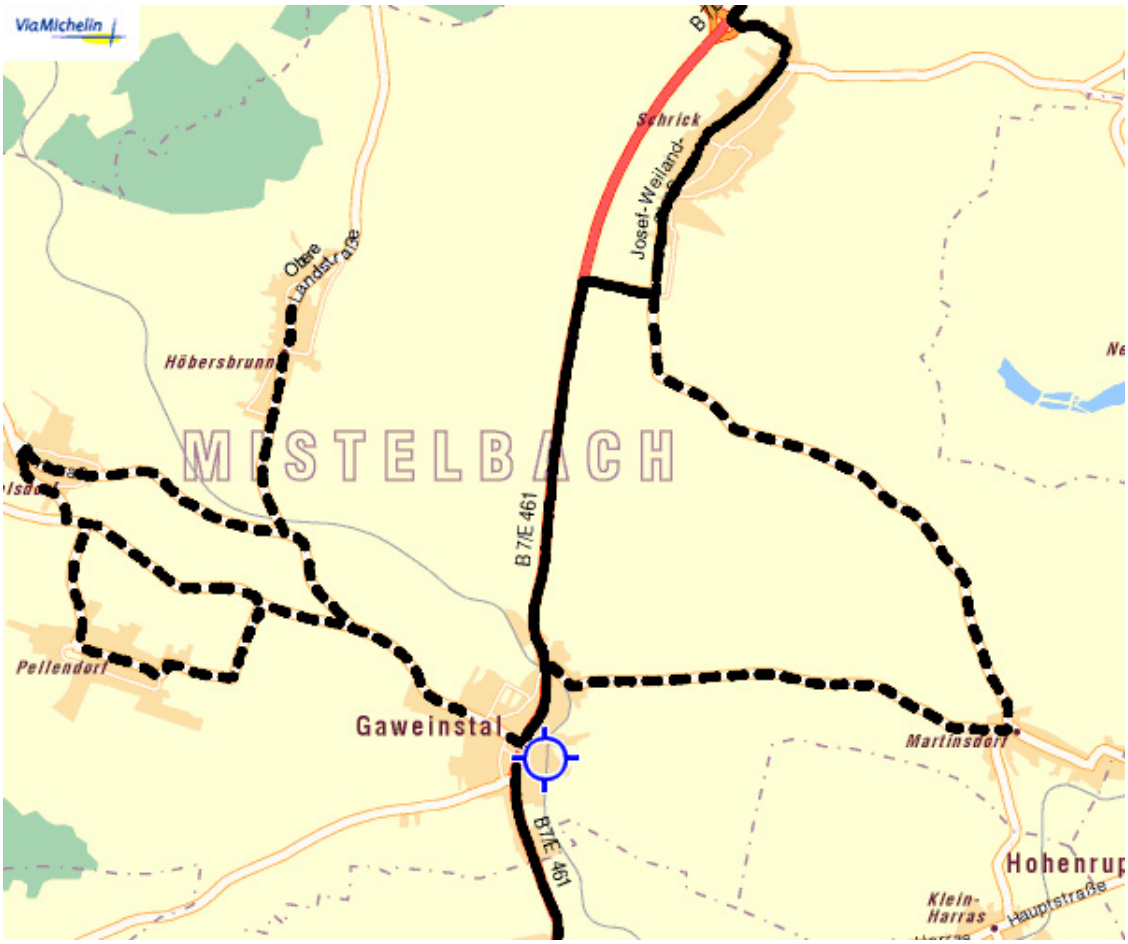


Abbildung 1.2.1.: Karte der Gemeinde Gaweinstal - Die Linienführung der Buslinie 1018 ist schwarz (fix) und schwarz gestrichelt (bedarfsabhängig) eingezeichnet. [ViaMichelin (2001–2005)]

Acht BEHA-Terminals (Abb. 0.1.1, S. 4) wurden installiert, an denen Fahrgäste einen Bus per Knopfdruck bestellen können.

Durch die weiteren Bestellmöglichkeiten via Internet, per SMS oder per Telefon bei einer Hotline der Verkehrsregion Ost, ist es auch möglich, sogenannte virtuelle BEHA-Points zu errichten. Diese sind speziell als solche gekennzeichnet, es fehlt nur der Terminal. Eine weitere Bestellmöglichkeit ist bei dem/der Busfahrer/in. Dieser kann über sein Mobiltelefon die nächsten Fahrten abrufen und durch einen einfachen Klick auf eine bestimmte Station ebenfalls eine Fahrt bestellen. Dies ist zum Beispiel sehr hilfreich, wenn ein Fahrgast von einem bestimmten Punkt wieder abgeholt werden möchte.

1.2.1. Vor der Umsetzung

Strukturelle Maßnahmen und die Umstellung der Linie 1018 auf einen bedarfsorientierten Betrieb würden nur erfolgreich sein, wenn auch die Bevölkerung diese Änderungen akzeptiert. Das war allen Beteiligten - Politik, Betreiber und der Bevölkerung selbst - von Anfang an klar. Die Linie 1018 ist in den Dörfern Martinsdorf, Pellendorf, Höbersbrunn und Atzelsdorf die einzige Anbindung an das Netz des ÖPNV.

Folgende Faktoren machen in ländlichen Regionen im Allgemeinen den ÖPNV unverzichtbar:

- Angewiesenheit von mehr als einem Viertel der Bevölkerung trotz hohen Motorisierungsgrades
- Anstieg der Lebenserwartung mit abnehmender Fahrtauglichkeit älterer Mitmenschen
- Verändertes Freizeitverhalten junger Menschen und damit verbundene Erhöhung des Unfallrisikos durch fehlendes öffentliches Verkehrsangebot
- Umweltaspekte (z. B. Verbesserung der Umweltqualität, Lärmemissionen, Abgasemissionen, Energieverbrauch, Kyoto-Protokoll, ...)
- Chancengleichheit aller Bürgerinnen und Bürger in Beruf, Ausbildung und Freizeit
- Verringerung des Unfallrisikos
- Erhöhung der Lebensqualität
- Verringerung des Verkehrsflächenbedarfs
- Verhinderung von Landflucht
  [Verkehrsplanung (1990), S. 5f]

Aus diesen Gründen musste bei der Umstellung sehr vorsichtig vorgegangen werden, damit sich die Bevölkerung nicht von vornherein BEHA verschließt. Aus folgenden Gründen hätte die Linie 1018 nicht mehr wirtschaftlich geführt werden können:

- Demografische Entwicklungen
  - Alterstruktur: Der Anteil der 10 - 24-jährigen an der Gesamtbevölkerung nimmt stetig ab
  - Rückgang der Schülerzahlen in ländlichen Regionen
    [Seidl (22. Juni, 2005), S. 10]
- höhere Verfügbarkeit von PKWs
- Arbeitszeitveränderungen - Rückgang der Arbeitstage je Jahr
- Abwanderung - Urbanisierung

Da ein Auflassen bzw. eine Verringerung der Frequenz trotz rückläufiger Fahrgastzahlen auf der betroffenen Linie der Region und ihrer Infrastruktur schaden könnte, musste das Angebot des ÖPNV in Gaweinstal ohne Verringerung geändert werden. Um die Akzeptanz des ÖPNV in der Bevölkerung trotz einer Umstellung zu erhöhen, durften die Technik-Skepsis und die Vorbehalte vor allem auch bei der älteren Bevölkerungsgruppe nicht ignoriert werden. Darum wurden Angebotsstrategien von allen Beteiligten gemeinsam erarbeitet. So wurde etwa die ursprüngliche Linienführung beibehalten. Eine weitere Maßnahme um das Image und die Akzeptanz des ÖPNV im Allgemeinen zu verbessern, war, die Bedienungsfrequenz zu erhöhen. Dadurch sollte auch das Attraktivitätsverhältnis zugunsten des öffentlichen Verkehrs gegenüber des Individualverkehrs verändert werden, wodurch wiederum neue Kunden gewonnen werden sollten.

1.2.2. Auswirkungen auf den Fahrgast und sein Verhalten

Für den Fahrgast hat die Einführung kaum Auswirkungen auf sein Verkehrsverhalten. Er muss lediglich seinen Fahrtwunsch ungefähr fünf Minuten vor der fahrplanmäßigen Abfahrt bekannt geben. Via Internet oder auch via Telefon kann er außerdem ein sogenanntes Abonnement bestellen, mit dem er eine tägliche (oder an bestimmten Tagen der Woche gebundene) Abfahrt bei einer bestimmten Station bestellen kann. Möchte der Fahrgast bei einer BEHA-Station aussteigen, so muss er seinen Aussteigewunsch dem/der Busfahrer/in beim Einsteigen bekannt geben. Der Busfahrer macht daraufhin eine Eingabe auf seinem Mobiltelefon. Dadurch wird die Anfahrt in der Zentrale bestätigt, welche diese Information wiederum an alle notwendigen Stellen weiterleitet, was vor allem die BEHA-Station betrifft.

1.2.2.1. Fahrgastinformationen

Ein weiterer Vorteil für den Fahrgast ist auch die vermehrte Information durch die Terminals an den Bushaltestellen, an denen BEHA im Einsatz ist. Nach Erhebungen des VDV (**V**erband **D**eutscher **V**erkehrsunternehmen) werden ca. 17 % aller Wege nicht mit dem ÖPNV durchgeführt, weil Informationen fehlen. Damit entspricht dieser Teil in etwa der Gesamtnutzung des ÖPNV. Eine theoretische Steigerungsrate von 100 % ist also allein durch Steigerung der Informationen, ohne eine Ausweitung des Angebots, an potentielle Fahrgäste möglich.

Ziel der Information ist es, Kunden sowohl unabhängig von einer konkreten Fahrt als auch bei der Wegeplanung und vom Anfang bis zum Ende des Weges zuverlässig und bequem zu unterstützen und zu leiten. Der Informationsbedarf wird in die folgenden fünf Gruppen kategorisiert:

- Allgemeine Grundinformation
- Persönliche Grundinformation
- Informationen vor Antritt einer Fahrt
- Informationen während der Fahrt
- Informationen nach Ende einer Fahrt
  [Müller-Hellmann und Nickel (Oktober 2000), S. 213ff]

Allgemeine Grundinformation

Allgemeine Informationen sollen die Bevölkerung mit den Vorteilen der Benutzung des ÖPNV bekannt machen. Durch diese Art der Information sollen insbesondere Menschen erreicht werden, die noch nicht regelmäßig das Angebot des ÖPNV nutzen. Ziel ist es, das Angebot allgemein vorzustellen. Der Kunde verlangt hier vor allem folgende Informationen:

- Landkarte mit Liniennetz
- Verbindungsmöglichkeiten, Anschlussstellen
- Fahrzeiten
- Preise
- weitere Service-Angebote

Dadurch soll ein emotional positives Bild des ÖPNV in der Öffentlichkeit verankert werden und Vorurteile bezüglich Zeitaufwand, Preis und (Un-)Pünktlichkeit abgebaut werden.

Der Verkehrsbetreiber nutzte die Einführungsveranstaltungen von BEHA als Gelegenheit, für sich und sein Produkt zu werben und erreichte dadurch eine Diskussion über

den ÖPNV im Allgemeinen und als Alternative zum Individualverkehr. So feierten die angeschlossenen Gemeinden im September 2002 das neue BEHA-System im Haus der Freiwilligen Feuerwehr Pellendorf. Durch diese Veranstaltung wollte der Verkehrsbetreiber eine Erhöhung der allgemeinen Grundinformation und eine damit verbundene Erhöhung der Nutzungshäufigkeiten, welche an der Anzahl der Fahrten pro Jahr und Einwohner gemessen wird, erreichen. [Girnau u. a. (Mai, 1997), S. 152]

Persönliche Grundinformation

Die persönliche Grundinformation soll über das konkrete ÖPNV-Angebot an der Haltestelle informieren. Dies umfasst Fahrpläne, Tarifinformationen, Verbindungsübersichten und zielgruppen- oder reisezweckspezifische Informationen, wie z. B. Informationen über Sehenswürdigkeiten, Gasthäuser oder über nahe gelegene Einkaufsmöglichkeiten. Sie sollen die allgemeinen Grundinformationen gebietsbezogen ergänzen.

Informationen vor Antritt einer Fahrt

Informationen vor Antritt einer Fahrt sollen über zweckmäßig zu benutzende Linien und Strecken, Fahrzeiten und Tarife Auskunft geben. Neben einem Liniennetzplan können dies auch Routenplaner im Internet sein, wie sie z. B. VOR auf ihrer Homepage anbietet.

Informationen während der Fahrt

Während der Reise muss der Fahrgast von verständlichen und einheitlichen Leitinformationen begleitet werden. Zu den Informationsmitteln während der Reise gehören

- die Wegweisung zu den Haltestellen,
- Fahrplan-, Tarif- und Informationsaushänge an den Haltestellen,
- gegebenenfalls aktuelle Hinweise zu besonderen Vorkommnissen an den Haltestellen,
- Informationseinrichtungen in den Fahrzeugen, wie
  - Haltestellenansage und -anzeige,
  - Übersichten über den Linienverlauf mit Umsteigemöglichkeiten,
  - Liniennetzpläne,
  - weitere Informationsansagen,
- zusätzliche Informationsmöglichkeiten an den Umsteigehaltestellen.

Informationen nach Ende der Fahrt

Zu den Informationen nach Ende einer Fahrt gehören Wegweiser zu wichtigen Zielen und Aushänge über die Fahrtmöglichkeiten für die Rückfahrt.

1.2.2.2. BEHA als Informationsquelle

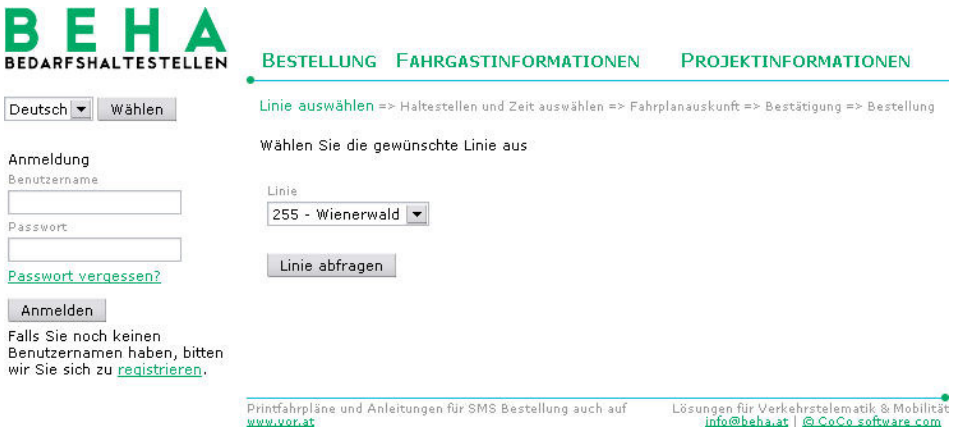
BEHA wird vom Verkehrsbetreiber als Möglichkeit zur Verbreitung von allgemeiner Grundinformation genutzt, wie in 1.2.2.1 schon beschrieben. Auch über die Homepage von BEHA (siehe Abb. 1.2.2 auf S. 22f) können allgemeine sowie persönliche Grundinformationen abgerufen werden. Die BEHA-Terminals werden zudem als dynamische Informationsquellen für persönliche Grundinformationen sowie für Informationen vor einer Fahrt genutzt. Auf den BEHA-Terminals kann der Fahrgast linien- und haltestellenbezogene Informationen beziehen, wie etwa Abfahrts- und Ankunftszeiten, eventuelle Verspätungen oder ob bestimmte Fahrten bereits bestellt sind.

Auch über das Mobiltelefon kann der Kunde Bestellungen oder Fahrplanabfragen tätigen. Hierzu muss er eine Kurznachricht via SMS mit einem Buchstaben- und Nummerncode an die BEHA-Zentrale senden und bekommt eine Antwortnachricht. Die Eingabe des Codes ist jedoch sehr kompliziert. Als einfacherer Prozess käme auch die Abfrage via WAP-Technologie (**W**ireless **A**pplication **P**rotocol) in Frage. Eine Umsetzung von Fahrgastinformationen via WAP-fähigem Mobiltelefon am Beispiel des Stadtbusses Schweinfurt, Deutschland, ist auf S. 219ff in [Müller-Hellmann und Nickel (Oktober 2000)] beschrieben.

Weitere Informationen wie unvorhergesehene Störungen oder Informationen über Anschlussfahrten sind derzeit zwar noch nicht realisiert, aber machbar. Ebenso könnten über Angebote wie der von A1 (`www.a1.net`, `12. September, 2005`) angebotenen *paybox* die Fahrtkosten bezahlt werden und das herkömmliche Fahrtticket ersetzen. Diese Maßnahme wäre jedoch nur betriebsweit umsetzbar.

1.2.3. Auswirkungen auf den Fahrplan

Auf Seite 23 sieht man den Fahrplan der Linie 1018. Die Bedarfshaltestellen sind gelb unterlegt. Sehr deutlich ist zu sehen, dass die Bedarfshaltestellen zu den Stoßzeiten fix angefahren werden. Die Bedarfshaltestellen haben auch bei bedarfsorientierter Bedienung im Fahrplan eine Zeit eingetragen. Dies geschah aus zwei Gründen. So musste zum einen der Betrieb auch bei einem Ausfall von BEHA oder eines BEHA-Terminals aufrecht erhalten bleiben, zum anderen sind dies die Zeiten zu denen auch die Bedarfshaltestellen bedient werden. Wenn ein Bus eine Haltestelle nicht anfahren muss, so wartet er an einem bestimmten Punkt jene Zeit ab, die er für die Umfahrung gebraucht hätte. Der Fahrplan kann dadurch weiterhin eingehalten werden. Für die Busfahrer bedeutet dies außerdem längere Ruhephasen.



(a) Bestellung einer Fahrt als Information vor Fahrtantritt

(b) Allg. Fahrgastinformationen als allg. und pers. Grundinformationen

(c) Allgemeine Projektinformationen als allgemeine Grundinformation

Abbildung 1.2.2.: Die Homepage von BEHA deckt drei Bereiche der Fahrgastinformation ab.

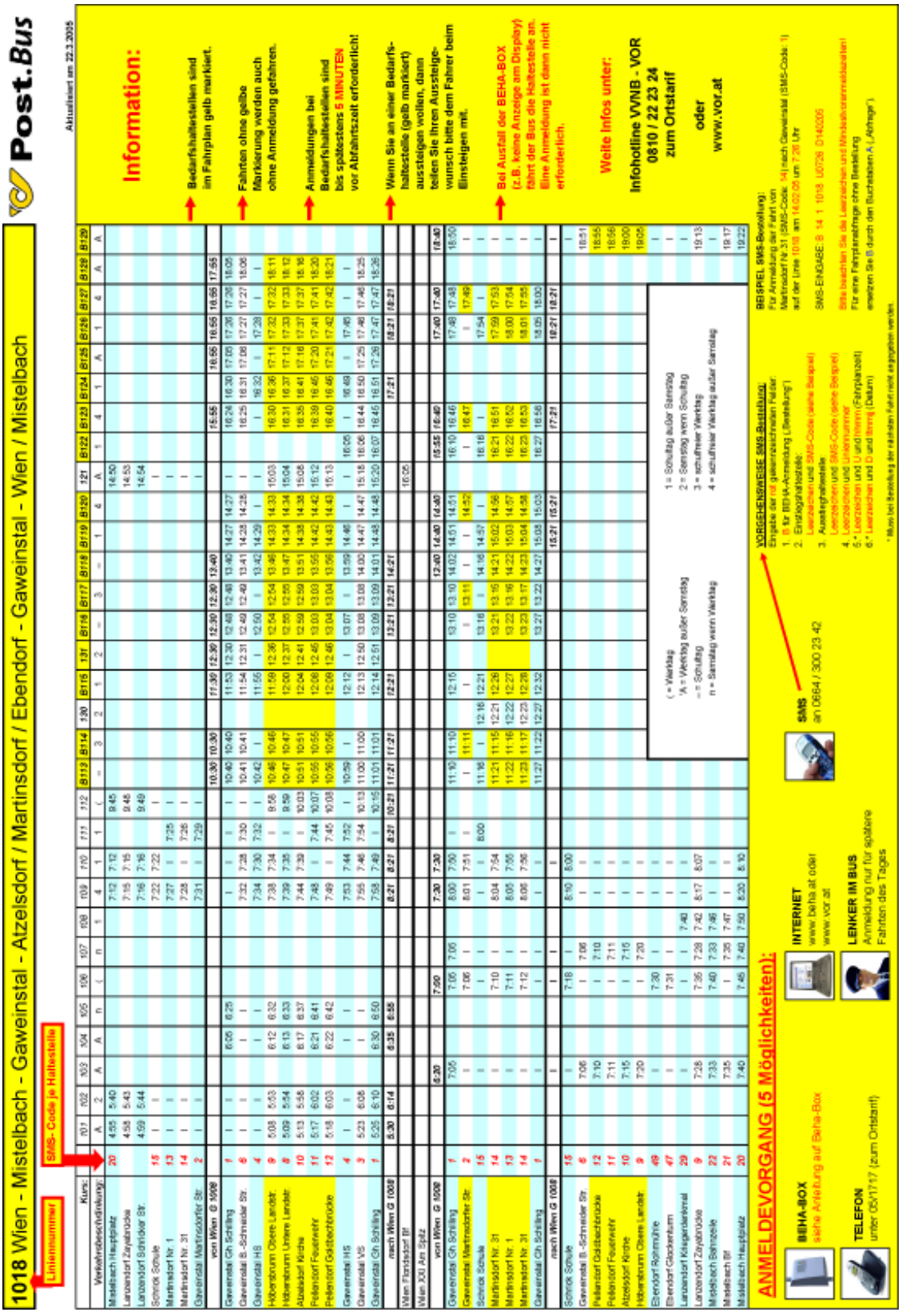


Abbildung 1.2.3.: Der Fahrplanaushang in Gaweinstal, wie er auf den BEHA-Terminals zu sehen ist [CoCo (2005)]

Zudem kann dadurch größere Pünktlichkeit garantiert werden, da eventuelle verkehrsbedingte Verspätungen leichter aufgeholt werden können und Vorplanabfahrten können vermieden werden, da der Fahrer über sein Mobiltelefon ständig über die aktuelle Fahrplanlage informiert ist. [Hoepke (1995), S. 117]

BEHA hat den Fahrplan nicht beschnitten, sondern konnte sogar die Frequenzen an den Haltestellen erhöhen, da sich das System selbst optimiert und durch einen geringen Grad der Mitwirkung des Kunden auf zeitliche Veränderungen der Fahrgastströme reagiert.

1.2.4. Auswirkungen auf das Fahrpersonal

Auf der einen Seite wird das Fahrpersonal entlastet. Zum einen wird dem/der Busfahrer/in durch den ständig durchgeführten Soll-Ist-Vergleich der Fahrplanlage die Einhaltung der Abfahrtszeiten erleichtert, zum anderen wird ihm/ihr durch die oben schon erwähnten längeren Ruhephasen die Arbeit im Allgemeinen erleichtert.

Auf der anderen Seite wurde der/die Busfahrer/in mit mehr Aufgaben betraut. Vor allem bei der Einführung des Systems entpuppte sich der/die Busfahrer/in als erster/erste Ansprechpartner/in für allgemeine und persönliche Grundinformationen über das BEHA-System. Zudem muss er Verspätungen und Bestellungen über sein Mobiltelefon eingeben. Auch bei der Routendisposition ist der/die Fahrer/in auf sich allein gestellt, da er selbst entscheiden muss, welche BEHA-Stationen er/sie anfahren kann. Damit das alles funktioniert, müssen die Busfahrer/innen das System auch annehmen und akzeptieren. Sie müssen motiviert werden und dürfen sich nicht überwacht fühlen.

Die Verspätungseingabe und die Routendisposition werden dem Fahrer zukünftig vom RoutingModul abgenommen.

1.3. Die Zentrale von BEHA

Ein RBL (**r**echnergesteuertes **B**etriebs**l**eitsystem) wie BEHA besteht aus folgenden drei wesentlichen Komponenten:

- Betriebsleitzentrale
- Fahrzeugausrüstung
- Streckenausrüstung

Die Streckenausrüstung wird durch die BEHA-Terminals und die Fahrzeugausrüstung durch die Mobiltelefone der Busfahrer repräsentiert. Die Betriebsleitzentrale ist im Fall von BEHA ein Server, der von VOR betrieben wird.

1.3.1. Die Betriebsleitzentrale

1.3.1.1. Aufgaben der Betriebsleitzentrale

Für das Verkehrsunternehmen ist es wichtig, einen Überblick über das Geschehen des Betriebes zu haben. Die Analyse der Daten hilft diesem bei planerischen Entscheidungen. Somit ist die Betriebsleitzentrale von BEHA nicht nur für den Ablauf und die Steuerung des Betriebes notwendig, sondern erfüllt auch noch andere Aufgaben. [Blum u. a. (Jänner 2002), S. 33f]

> ***Der Server*** *- Im zentralen BEHA Server wird das von CoCo Mobile Telematics entwickelte TMS (**T**ransport **M**anagement **S**ystem) eingesetzt. Dieses verfügt über alle Fahr- und Dienstpläne für die vollautomatische Disposition. Dieser Server vermittelt die Bestellungen zwischen BEHA Point, Fahrgast und Busfahrer vollautomatisch, und koordiniert die Busse. [CoCo (2003), S. 3]*

Der Server verarbeitet alle eingehenden Daten und weiß stets über die mit ihm verbundenen Fahrzeuge und Terminals Bescheid. Seine weiteren Aufgaben bestehen in der Verwaltung der Bestellungen und Anfragen, sowie dem Loggen aller Ereignisse. Des Weiteren stellt er verschiedene Services für Kunden und Mitarbeiter, wie Fahrplanabfragen oder Auswertung und Analyse bestimmter Daten, wie etwa Fahrzeiten, zur Verfügung

oder gibt Auskunft über aktuelle Betriebszustände und Ausfälle. Die aufgezeichneten Daten dienen dem Verkehrsbetreiber auch zur Planung eines verbesserten Betriebsablaufs und stellen die Grundlage für Kosten-Nutzen-Rechnungen dar. [Girnau u. a. (Mai, 1997), S. 164]

1.3.1.2. Technische Beschreibung der Betriebsleitzentrale

Der BEHA Server (siehe Abb. 1.3.1 auf S. 27) ist eine J2EE basierte asynchrone Applikation, mit dem Zweck, die zentrale Intelligenz für den Betrieb von BEHA zur Verfügung zu stellen. Externe *Clients* (BEHA-Terminals, BusApplikation, SMS *User*) kommunizieren mit dem System entweder über HTTP (**H**yper**t**ext **T**ransfer **P**rotocol, Schnittstelle: Webapplikationen) oder durch SMS, GPRS oder ähnliche Träger (Schnittstelle: *Communication Module*). [Lichtl (2002), S. 1]

Wie zuvor schon erwähnt, wird im BEHA Server das von CoCo Mobile Telematics entwickelte TMS eingesetzt. Das TMS versteht sich als *Middleware*, um *Agents* von Anbietern von Verkehrsdienstleistungen in einem *Electronic Marketplace* zu hosten.

Das *Communication Module* ist eine weitere wichtige Softwarekomponente, das einen *Unified Messaging Service* über JMS (**J**ava **M**essage **S**ervice) anbietet. Dadurch können weitere proprietäre *Clients* einfach in J2EE Systeme eingebunden werden. Das *Communication Module* liefert einkommende *Messages* zu einem *JMS Topic*, und sendet ausgehende Messages, die an einem anderen *JMS Topic* empfangen wurden, an die Adressaten weiter. Die *Messages* werden hierbei in einem einheitlichen XML-Format (E**x**tensible **M**arkup **L**anguage) abgefasst.

Der BEHA-Server ist also nachrichtenbasierend, da Objekte ausschließlich nachrichtengesteuert über ein spezifiziertes *Interface* kommunizieren können. Seine lose gekoppelte Architektur erlaubt eine hohe Autonomie.

Bei der betriebenen Datenbank handelt es sich um ein relationales Datenbankmodell, das alle Objekte des betriebsspezifischen Bereiches von BEHA abbildet. Die Datenbankzugriffe werden von dem Applicationserver JBoss verwaltet. [Girnau u. a. (Mai, 1997), S. 156]

1.3.1.3. Das RoutingModul als wichtige Erweiterung

Das RoutingModul versteht sich als serverseitige Erweiterung des BEHA-Systems. Es ist also nicht für die Aufrechterhaltung des Betriebes notwendig, bietet jedoch Vorteile und Erleichterungen im Einsatz für Kunden, Fahrer/innen und den Betreiber.

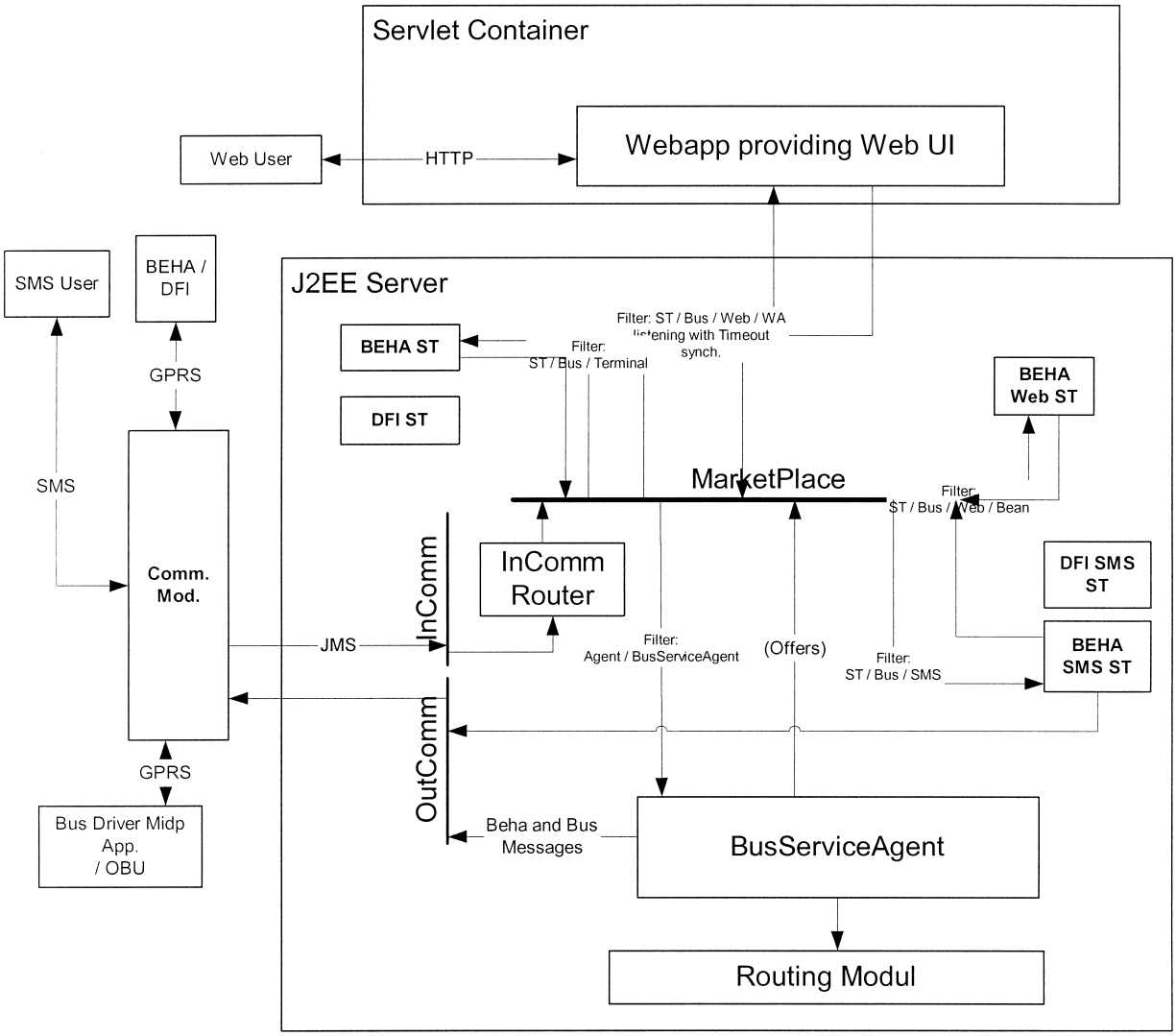


Abbildung 1.3.1.: Die schematische Darstellung des BEHA-Servers und all seiner Einzelkomponenten, auch das RoutingModul ist deutlich zu sehen. [Lichtl (2002), S. 3]

Aufgaben des RoutingModuls

Die meisten Aufgaben des RoutingModuls wurden schon in 0.1.1.3 beschrieben. Sie seien hier noch einmal kurz genannt:

- Fahrzeugortung
- Verspätungsvorhersage
- Routendisposition
- Datenanalyse und -auswertung

Die Bedeutung der Fahrzeugortung

Erst eine effiziente Fahrzeugortung ermöglicht alle oben genannten Dienste. Ist die Datenanalyse und -auswertung auch ohne ständige Positionsbestimmung möglich, sind die anderen drei Dienste ohne diese nicht denkbar. Eine Auswertung der Fahrtleistung ist z. B. mittels der Daten über Bestellungen und die Bestätigungen der Fahrer ebenso möglich, indem ein repräsentativer Graph der betreffenden Region mit statischen Entfernungsangaben zwischen den Stationen und wichtigen Kreuzungspunkten zu Grunde gelegt wird.

Konventionelle Systeme der Fahrzeugortung

Folgende Systeme werden im ÖPNV in Österreich zur Fahrzeugortung eingesetzt:

- Elektromechanische Systeme (Fahrdrahtkontakte, Weichenkontakte, Schienenkontakte, Schlüsselschalter oder -taster)
- Induktive Systeme (Schleifendetektoren in Fahrbahnen, Induktionsschleifen mit spezieller Auswertung, Gleiskoppelspulen)
- „Baken“ -Systeme (Datenübertragung zwischen Fahrzeuggeräten und Ortsbaken durch Infrarotstrahlen)

Um eine kontinuierliche Positionsbestimmung zu erreichen, müssten die Detektoren in dichtem Abstand angeordnet werden. Aus ökonomischen Gründen ist dies jedoch nicht immer durchführbar.

So werden zumeist Kombinationen aus Wegrandbaken und Radumdrehungszähler verwendet, da unter der Annahme, dass der öffentliche Verkehr sich auf bekannten, vordefinierten Strecken bewegt, zur automatischen Positionsbestimmung ein Fixpunkt auf der Strecke reicht, von dem aus dann nur mehr die zurückgelegte Weglänge gemessen wird. Ein anderer Zugang ist die Anbringung von Sensoren bei jeder Haltestelle. Anhand von Tagesganglinien (Erfahrungswerte), die die Fahrzeiten für die einzelnen Abschnitte abhängig von der Tageszeit speichern, kann dann die genaue Position bestimmt werden.

Alle genannten Systeme eignen sich jedoch nicht für die Nutzung im ländlichen Bereich, wo zum Teil die Infrastruktur für die Anbringung aller Ortsbaken fehlt. Für die Benutzung bei BEHA sind jedoch alle diese Systeme ungeeignet, da jeweils von Sollwerten für Zeit und Weglänge ausgegangen wird, beides jedoch von Fahrt zu Fahrt variieren kann. [Girnau u. a. (2001), S. 46ff]

Alternative Systeme der Fahrzeugortung

Zu den alternativen Systemen der Fahrzeugortung zählen neben den Satellitengestützten auch die GSM-basierten (**G**lobal **S**ystem for **M**obile Communications) Systeme. Sie gewinnen zunehmend an Bedeutung.

Zu den satellitengestützten Ortungssystemen gehören neben dem am weitesten verbreiteten GPS noch GLONASS (**Glo**bal'naya **na**vigatsionannaya **S**putnikovaya **S**istema) und GALILEO.

GSM-basierte Ortungsverfahren werden in Österreich derzeit nur vom Mobilfunkbetreiber T-Mobile angeboten. Der Vorteil dieser Verfahrens liegt zum einen in der höheren Verfügbarkeit von Positionsinformationen, zum anderen in den geringeren Kosten, da keine externen Geräte zur Positionsbestimmung angeschafft werden müssen, sondern das ohnehin zur Verfügung stehende Mobiltelefon genutzt werden kann. [T-Mobile (Juni, 2005), S. 5]

Vorteile der Farzeugortung mittels GPS

Während sich für Großstädte mit dicht befahrenen Streckennetzen die Ortung der Fahrzeuge mit einem Mix aus Wegrand-Baken und Radumdrehungszählung bewährt hat, hat GPS neue Möglichkeiten der Ortung auch in schwach befahrenen Netzen in ländlichen und suburbanen Räumen eröffnet.

Um einen ständigen Soll/Ist-Vergleich mit dem Fahrplan durchzuführen, der erst durch GPS ermöglicht wird, muss das RoutingModul auf eine große Menge Bestandsdaten zurückgreifen, z. B. den gesamten Fahrplan und das Streckennetz. Aus dem ständigen Soll/Ist-Vergleich in der RBL lässt sich eine neue Qualität der Fahrgastinformation ableiten, nämlich die dynamischen Fahrgastinformationen.

Bisher sind Verspätungsvorhersagen nur Schätzungen des/der Busfahrers/in, der seine Eingaben zu Verspätungen über sein Mobiltelefon eingibt. Da sich viele Busfahrer/innen dadurch aber überwacht fühlen oder diese Funktion aus anderen Gründen kaum nutzen, werden dem Fahrgast Verspätungen nicht gemeldet.

Dynamische Fahrgastinformationen erleichtern dem Kunden das Warten. Er muss nicht mehr ohne Informationen an der Haltestelle warten, bis sein Bus kommt. Am Display kann dem Fahrgast in Form eines Countdowns angezeigt werden, wie lange er noch warten muss. Der Countdown zählt die Minuten bis zur Ankunft des Busses herunter. Das besser informierte Warten wird auch als „Positives Warten“ bezeichnet. [Müller-Hellmann und Nickel (Oktober 2000), S. 214]

Der ständige Soll/Ist-Vergleich und das Lernen aus getätigten Fahrten und dem daraus resultierenden Schließen auf zukünftige Fahrtzeiten zu bestellten Haltestellen erleichtert auch dem/der Busfahrer/in den Betrieb von BEHA. Bestellt nämlich ein Kun-

de seine Fahrt sehr kurzfristig, so muss der/die Busfahrer/in durch einen Knopfdruck auf seinem Mobiltelefon bestätigen, ob er die bestellte Haltestelle unter Einhaltung des Fahrplans noch bedienen kann oder nicht. Macht er dies nicht, oder bestätigt er eine Fahrt durch Drücken des falschen Knopfes oder aus anderen Gründen falsch, führt dies oft zu großem Ärger und Unzufriedenheit bei den Kunden. Das RoutingModul wird bei Verwirklichung den Busfahrern diese Entscheidung abnehmen.

Doch abgesehen von den Vorteilen, die eine permanente Fahrzeugortung an sich bietet, hat GPS auch Nachteile. Zum einen gibt es eine natürliche statistische Streuung bei der Berechnung der Position, zum anderen sind die Informationen nicht immer und überall abrufbar, da zur Standortbestimmung via GPS freie Sicht auf die Satelliten gegeben sein muss, die durch Abschattungen gestört sein kann. GSM-basierte Ortungsverfahren haben hier den Vorteil, dass man schon im Vorhinein über die Verfügbarkeit der Information Bescheid weiß.

1.3.1.4. GALILEO

> *Das System GALILEO soll es jedem Bürger ermöglichen, mittels eines kleinen, erschwinglichen Empfängers seine Position auf einige Meter genau zu bestimmen. Die garantierte Kontinuität der Signalaussendung gewährleistet eine hundertprozentige Zuverlässigkeit des Systems. [Sutter (September 2000), S. 12]*

Obwohl inzwischen sowohl das amerikanische GPS als auch das russische GLONASS für zivile Nutzung zugänglich gemacht wurden, ist für Europa ein eigenes System zur satellitengestützten Positionsbestimmung geplant.

Das System GALILEO soll dem amerikanischen GPS überlegen sein. Dazu sollen bis 2008 30 Satelliten in Umlaufbahnen mit 23.000 km Abstand zur Erde sein. Diese Entwicklung wird vor allem aus politischen Gründen vorangetrieben, um von den Vereinigten Statten von Amerika unabhängig zu sein.

Auch aus wirtschaftlicher Sicht ist die Entwicklung von GALILEO interessant, da dadurch viele Arbeitsplätze geschaffen werden. Aus terminlichen Gründen konnte BEHA nicht GALILEO einsetzen, obwohl die EU (**E**uropäische **U**nion) Projekte, welche auf GALILEO setzen, stark fördert und finanzielle Anreize bietet.

Das beste System zur Fahrzeugortung

Alle Systeme haben ihre Vorteile und Nachteile. Allgemein kann jedoch gesagt werden, dass das beste System jenes ist, dass der realen Situation am besten angepasst ist. So hat zwar die Ortung über das GSM-Netz Vorteile gegenüber GPS im urbanen und im

alpinen Bereich aufgrund von Abschattungen durch Tunnel oder hohe Gebäude, sofern das GSM-Netz gut ausgebaut ist. Im ländlichen Bereich wird jedoch der Ausbau des GSM-Netzes aus wirtschaftlichen Gründen oftmals weniger stark vorangetrieben als in stark besiedelten Gebieten.

In jenen Gebieten, in denen BEHA zum Einsatz kommt, ist jedoch GPS oder eine andere satellitengestützte Ortung die bessere Wahl, da das GSM-Netz weniger stark ausgebaut ist und so weniger zuverlässig für die Ortung ist.

Entwicklungsstand des RoutingModuls

Bisher wurde neben den grundlegenden Funktionen des RoutingModuls vor allem die Entwicklung eines Auswertungstools zur Analyse der Fahrtwege und Einsparungen durch BEHA forciert. Sowohl das RoutingModul selbst als auch das Analysetool sind ausführlich ab S. 65 beschrieben.

1.3.2. Die Fahrzeugausrüstung

Die Fahrzeugausrüstung, die für den Betrieb von BEHA als RBL notwendig ist, besteht im Grunde nur aus der BusApplikation, welche in 2.3 ab Seite 54 ausführlich beschrieben ist.

1.3.3. Die Streckenausrüstung

Die für BEHA installierte Streckenausrüstung ist nicht relevant für den Betrieb von BEHA, sondern stellt lediglich ein *User Interface* für die Fahrgäste dar und dient, wie schon oben erwähnt, vordergründig dem Bestellvorgang und als Informationsquelle. Daher ist in dieser Arbeit keine detaillierte technische Beschreibung der Terminals zu finden. Eine genauere Beschreibung der Hardware der BEHA-Terminals ist in [CoCo (2002)] zu finden.

Teil II.

Die BusApplikation

Das BEHA-System ist dem/der Busfahrer/in mittels einer J2ME-Anwendung (im folgenden BusApplikation genannt) über ein GPRS-fähiges Mobiltelefon zugänglich. Um die für das RoutingModul benötigten GPS-Daten zu erhalten, wird ein GPS-Empfänger über die serielle Schnittstelle des Mobiltelefons angeschlossen. Ein Hintergrund-*Thread* der bereits entwickelten J2ME-Anwendung verarbeitet die vom GPS-Empfänger im NMEA-Format (**N**ational **M**arine **E**lectronics **A**ssociation) kommenden Daten und versendet sie via GPRS an den Server, der diese Daten dann weiterverarbeitet.

Im Folgenden sollen nun

- die BusApplikation selbst,
- die Anforderungen an die BusApplikation aus der Sicht des Fahrpersonals,
- die Anforderungen an das Mobiltelefon,
- die gewählte Hardware,
- die Einbindung eines GPS-Empfängers in die BusApplikation,
- der Komprimierungsalgorithums zur intelligenten Datenreduktion der einkommenden GPS-Daten und
- alle dafür notwendigen technischen Standards

beschrieben werden.

2.1. Begriffsbestimmungen

2.1.1. GPS

Das Global Positioning System ist Teil des satellitengestützten Navigationssystems, das vom Verteidigungsministerium (Departement of Defense) der U.S.A. unter dem Namen NAVSTAR (eine Reihe künstlicher Sterne zur Navigation auf der Erde, in Anlehnung an Seefahrer, die mit Hilfe von Sextanten ihre Position bestimmen konnten) entwickelt wurde. Das Ziel des Verteidigungsministeriums war, mit GPS einen PPS-Dienst (**P**recision **P**ositioning **S**ervice) zu schaffen, der den eigenen und verbündeten Streitkräften bei der Navigation von Flugzeugen, Schiffen und Fahrzeugen den Rückgriff auf eigene und fremde Funknavigationsdienste ersparen sollte. Ein GPS-SPS-Dienst (**S**tandard **P**ositioning **S**ervice) für zivile Nutzung war ursprünglich nicht geplant.

Heute, 32 Jahre nach Beginn der Konzeptphase, hat sich die anfangs ins Auge gefasste Rolle des GPS gründlich gewandelt. So sind heute mehr als 95 % der Benutzer zivil. [Dodel und Häupler (2004), S. 160f; Thaller (1999), S. 17f; Grewal u. a. (2001), S. 2ff]

2.1.1.1. Die Systemparameter

Die wesentlichen Systemparameter von GPS in Kurzform:

- 24 Satelliten in Betrieb
- sechs Bahnebenen belegt mit je vier Satelliten, Bahninklination (Neigung der Bahn gegenüber dem Äquator) ist 55° Grad
- Orbithöhe 20.200 km
- Umlaufzeit eines Satelliten ca. 12 Stunden
- Satellitensolllebensdauer 10 Jahre, durchschnittliche Satellitenlebensdauer 12-15 Jahre
- Fläche der Solargeneratoren 13,4 m^2, Flügelweite der Solargeneratoren 19,3 m
- Jeder Satellit verfügt über eine Cäsium- oder Rubidium-Atomuhr als Zeitgeber [Dodel und Häupler (2004), S. 162]

2.1.1.2. Die Dienste

Derzeit stehen die folgenden beiden GPS-Dienste zur Verfügung:

- SPS
- PPS

SPS

Der SPS-Dienst oder auch SPS-Einfrequenzdienst wird auf der L1-Frequenz im C/A-Code (**C**oarse **A**quisition) übertragen. Es ist ein offener und von jedem nutzbarer Funknavigationsdienst, der wie alle amerikanischen Funknavigationsdienste unter dem Vorbehalt steht, dass die *National Command Authority* im Falle einer schweren Bedrohung der nationalen Sicherheit der U.S.A. Nutzungsrestriktionen anordnen kann.

Die *National Command Authority* schuf sich die Möglichkeit mittels GPS-Jammers (Störsender) das Signal künstlich zu verschlechtern jedoch erst später, nachdem man erkannte, das die Positionsbestimmung mittels GPS für zivile Empfangseinrichtungen um den Faktor 10 besser war, als erwartet. Erst eine Vereinbarung zwischen dem Pentagon und dem US-Handelsministerium verhinderte 1990 eine zukünftige künstliche Verschlechterung des Signals. [Dodel und Häupler (2004), S. 162]

Epoche	1 ms
Wiederholungsrate (f_0)	1,023 MHz
X1-Epoche	1,5 s
Übertragungsrate	50 baud
Frequenz L1	$154 \cdot 10,23 = 1.575,42$ MHz

Tabelle 2.1.1.: Eigenschaften des SPS-Dienstes

Eigenschaften des SPS-Dienstes Epochen sind interne Zeitintervalle. Bei dem Zähler X1 handelt es sich um eine Größe, die in der Elektronik eines Satelliten erzeugt wird.

Die Genauigkeit ist mit 100 Metern (2 drms, 95 %) horizontal und 156 Metern (95 %) vertikal angegeben. DRMS steht dabei für **D**istance **R**oot **M**ean **S**quare. Der zweifache Wert von drms bezeichnet den Radius jenes Kreises, in dem 95 % aller Messungen eines Empfängers fallen würden. Die Zeit darf maximal um 340 ns von der wahren Zeit, wie sie durch den UTC-Standard (**U**niversal **T**ime **C**oordinated, der international koordinierte Atomzeitstandard) festgelegt wird, abweichen. [Thaller (1999), S. 20f]

PPS

Der PPS-Dienst oder auch PPS-Zweifrequenzdienst wird auf der L1- und der L2-Frequenz übertragen und ist hoch verschlüsselt und somit nur von den amerikanischen

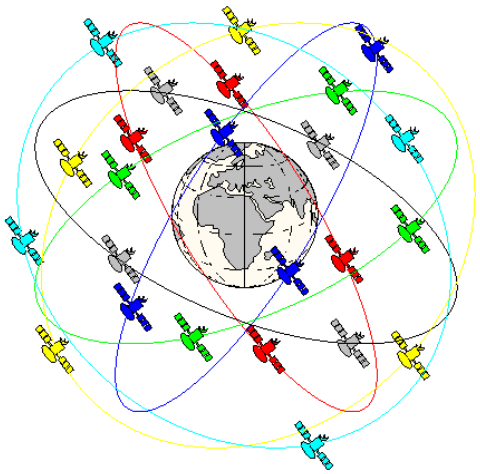


Abbildung 2.1.1.: Konstellation der 24 GPS-Satelliten [Schumann (1996), Screen 1]

Streitkräften und ihren Verbündeten nutzbar. Durch den gleichzeitigen Empfang beider Frequenzen kann der Ionosphärenfehler, ein durch die Erdatmosphäre verursachter Fehler, teilweise korrigiert werden.

Eigenschaften des PPS-Dienstes Die Trägerfrequenz des L2-Signals hat die Frequenz $120 \cdot 10,23 = 1.227,6$ MHz. Alle sieben Tage wird der Code zur Verschlüsselung neu aufgesetzt.

Die Genauigkeit ist mit 22 Metern (2 drms, 95 %) horizontal und maximal 27,7 Metern (95 %) vertikal angegeben. Die Zeit darf maximal um 200 ns vom UTC-Standard abweichen. Die Positionsbestimmung für Benutzer im militärischen Bereich ist also weit besser als für zivile Nutzer. [Thaller (1999), S. 25f]

2.1.1.3. Das Grundprinzip des GPS

Das bei GPS angewandte Ortungsverfahren beruht auf dem Prinzip der Entfernungsbestimmung durch Messung der Laufzeit von Signalen zwischen dem Nutzer und je einem von mehreren Bezugspunkten (hier Satelliten), deren Positionen genau bekannt sind.

Positionsbestimmung mit GPS

Der geometrische Ort aller Punkte im Raum, die von einem Bezugspunkt die gleiche Entfernung haben ist eine Kugel. Auf einer Ebene ist der geometrische Ort ein Kreis. Der gesuchte Standpunkt ist der Schnittpunkt dreier Kugeln. Die Genauigkeit der Positionsbestimmung hängt dabei von der Genauigkeit der Entfernungsmessung ab.

Die Laufzeit eines Signals, anhand derer die Entfernung bestimmt wird, hängt von der Genauigkeit der Uhrzeiten ab, bei Ausstrahlung des Signals vom Satelliten und bei Empfang beim Nutzer. Eine Zeitabweichung von 1 μs würde eine Entfernungsabweichung von 300 m verursachen. Darum ist für eine Positionsbestimmung mittels GPS ein vierter Satellit für die Zeitkorrektur notwendig. Die Satellitenanordnung des GPS gewährleistet, dass fast auf der ganzen Erdoberfläche immer vier Satelliten empfangen werden können.

Durch Messung der Dopplerfrequenzverschiebungen, die durch die relative Bewegung des Nutzers gegenüber den drei Satelliten entstehen, kann der Nutzer auch seine Geschwindigkeit nach Betrag und Richtung ermitteln.

Wie bereits erwähnt sind die beiden Träger L1 und L2 mit satellitenspezifischen Codesequenzen, die der Laufzeitmessung dienen, moduliert. Diese Codesequenzen sind selbst mit einem Bitstrom moduliert, der Navigationsmitteilungen enthält. Diese Informationen werden zur Berechnung der Koordinaten eines Ortes benötigt.

Dazu gehören:

- Koordinaten der Position der Satelliten im Moment der Ausstrahlung
- Bahnparameter der Satelliten
- Korrekturwerte für die Uhren der Satelliten
- Korrekturwerte für die Wellenausbreitung in der hohen Atmosphäre
- Almanach (Sammlung von Daten sämtlicher Satelliten, nach denen der Nutzer die Wahl der jeweils günstigsten Satelliten treffen kann)
  [Schumann (1996), GPS - das Grundprinzip!, ca. Screen 7]

2.1.1.4. Differential GPS

DGPS (**D**ifferential **GPS**) ist eine Technik zur Korrektur von Fehlern, die bei der Positionsbestimmung mittels GPS auftreten. Dabei wird ein anderer GPS-Empfänger, dessen Position bekannt ist, als Referenz herangezogen. Der Fehler wird für jeden empfangenen GPS-Satelliten einzeln bestimmt und durch die Referenzstation (siehe Abb. 2.1.2 auf S. 38) ausgestrahlt - üblicherweise mittels Funk im UKW-Bereich (**U**ltra**k**urz**w**elle) oder für spezielle Anwendungen mittels (Funk-)Telefonverbindung. Da sich die Fehler der einzelnen Satelliten nur langsam ändern, ist diese Übertragung nicht zeitkritisch. Für einfache DGPS-Korrektur reicht eine Korrektur alle drei Sekunden aus.

Ein Empfänger für DGPS empfängt nun neben den Signalen der GPS-Satelliten auch das Korrektursignal der stationären Referenzanlage. Die dazu nötige Empfangsantenne für die Korrektursignale ist oft schon in die GPS-Antennen integriert. Damit kann der Empfänger - für jeden Satelliten separat - die GPS-Signale korrigieren und auf

Abbildung 2.1.2.: Bild einer DGPS-Referenzstation auf dem Dach des Landesvermessungsamts Koblenz (SAPOS - Satellitenpositionierungsdienst) [Wikipedia, DGPS, Screen 1]

diese Weise seine Positionsbestimmung verbessern. Fällt die Funkverbindung zur DGPS-Sendeanlage aus, schaltet der DGPS-Empfänger in den normalen GPS-Modus ohne Korrektur um und verliert damit an Genauigkeit. Die erreichbare Genauigkeit liegt je nach Qualität des Empfängers und der Korrekturdaten bei ca. 0,3 bis 2,5 Meter in der Ebene und bei ca. 0,6 bis 5 Meter bei der Höhenmessung. [Grewal u. a. (2001), S. 5f]

2.1.1.5. Der NMEA-Standard

Die NMEA ist eine US-amerikanische Vereinigung von Elektronikherstellern und -händlern der Schifffahrtsindustrie. Die Vereinigung wurde 1957 gegründet und 1969 als Gesellschaft eingetragen. Ihre Hauptziele sind die Förderung von Standards und technischen Entwicklungen in der Marineelektronik sowie die technische Weiterbildung ihrer Mitglieder.

Unter dem NMEA-Standard versteht man einen im maritimen Bereich sehr verbreiteten Übertragungsstandard. NMEA 0183 ist der Standard für die Kommunikation zwischen Navigationsgeräten auf Schiffen. Er existiert seit März 1983 und liegt derzeit in der Version 0183-2.3 vor, die seit Februar 1998 etabliert ist. Der neueste Standard NMEA 2000 hat sich noch nicht durchgesetzt, weshalb weiterhin an einer inzwischen

veralteten Technik festgehalten wird. Nachteile des NMEA 0183 Standards sind die fehlende Erkennung von Kollisionen und Übertragungsfehlern.

Der NMEA Standard definiert die elektrischen Signalanforderungen und das Protokoll für die Übertragung von Positionsdaten von GPS-Empfängern zu anderen Geräten über eine serielle Schnittstelle. Die Schnittstelle wird dabei mit folgenden Parametern betrieben:

- Übertragungsrate: 4.800 Baud
- *Data* Bits: 8
- Stop Bits: 1
- kein *Parity* Bit
- kein *Handshake*

Die Datenübertragung läuft in kleinen Einheiten, den sogenannten *sentences*. Jeder Datensatz ist auf maximal 80 Zeichen beschränkt, beginnt mit einem $-Zeichen und endet mit <CR><LF>. Nach dem $-Zeichen folgt die Geräte-ID und eine Datensatz-ID. Anschließend werden durch Kommata getrennt die Datensätze angehängt. Der letzte Datensatz ist eine Prüfziffer, die durch ein ∗-Zeichen gekennzeichnet ist.

Ein Beispiel für NMEA-Daten, geliefert von einem in den Bussen verwendeten GPS-Empfängern (siehe 2.2.4 ab S. 53):

```
$GPGGA,085303.356,4813.1348,N,01622.2230,E,1,06,1.3,217.8,M,43.4,M,0.0,0000*70
$GPRMC,085303.356,A,4813.1348,N,01622.2230,E,0.000000,325.17,230704,,*0A
$GPVTG,2.92,T,,M,0.089861,N,0.166420,K*50
$GPGGA,085304.356,4813.1348,N,01622.2225,E,1,06,1.3,217.9,M,43.4,M,0.0,0000*72
$GPRMC,085304.356,A,4813.1348,N,01622.2225,E,0.000000,325.17,230704,,*09
$GPVTG,23.35,T,,M,0.089699,N,0.166120,K*62
$GPGGA,085305.355,4813.1348,N,01622.2222,E,1,06,1.3,217.7,M,43.4,M,0.0,0000*79
$GPRMC,085305.355,A,4813.1348,N,01622.2222,E,0.000000,325.17,230704,,*0C
$GPVTG,27.25,T,,M,0.057418,N,0.106336,K*6C
$GPGGA,085306.355,4813.1347,N,01622.2220,E,1,08,1.0,217.4,M,43.4,M,0.0,0000*79
$GPGSA,A,3,01,25,07,20,04,11,13,31,,,,,2.0,1.0,1.7*30
$GPGSV,2,1,08,01,80,078,42,20,71,072,42,13,36,227,45,04,33,308,41*7E
$GPGSV,2,2,08,25,27,051,44,11,26,162,38,07,22,265,44,31,14,230,40*79
$GPRMC,085306.355,A,4813.1347,N,01622.2220,E,0.000000,325.17,230704,,*02
$GPVTG,125.86,T,,M,0.085814,N,0.158924,K*5B
```

Abbildung 2.1.3.: Ausschnitt aus den gelesenen GPS-Daten

Der verwendete GPS-Empfänger verwendet mehrere Datensätze. Da für die spätere Verwendung nur die Daten des RMC-Datensatzes relevant sind, soll an dieser Stelle nur dieser Datensatz im Detail erläutert werden. Genaue Informationen zu den anderen Datensätzen sind in [Betke (August 2001)] zu finden.

[Betke (August 2001), S. 4ff; Köhne und Wößner (Mai, 2005), Screen 1ff]

GGA	*Global Positioning System Fix Data: Time, Position and fix related data for a GPS receiver*
RMC	*Recommended Minimum Navigation Information*
VTG	*Track Made Good and Ground Speed*
GSA	*GPS DOP (dilution of precision; Verschlechterung der Genauigkeit) and active satellites*
GSV	*Satellites in view*

Tabelle 2.1.2.: Verwendete Datensätze des GPS-Empfängers Holux GM-210 [Holux, Technology Inc., S. 2]

GPRMC

Es gibt 3 Arten von RM-Datensätzen: A, B und C. Die neuesten GPS-Empfänger verwenden jedoch nur mehr den RMC-Datensatz. Sein Format ist in Abb. 2.1.4 dargestellt.

```
$--RMC,hhmmss.sss,A,llll.llll,a,yyyyy.yyyy,a,xx.xxxxxx,xxx.xx,xxxxxx,x.x,a*hh
$GPRMC,085324.354,A,4813.1635,N,01622.2463,E,17.986132,359.59,230704,   , *32
| | |      |      |    |    |      |    |      |       |      |    | |  |
1 2 3      4      5    6    7      8    9      10      11     12   13 14 15
```

Abbildung 2.1.4.: Der RMC-Datensatz

Die Zeichen (siehe Abb. 2.1.4):

1 Einleitendes $-Zeichen

2 Geräte-ID (hier: GP)

3 Datensatz-ID (hier: RMC)

4 Zeit im UTC-Format (hier: 08:53:03.356)

5 Status (A-ok, V-invalid)

6 aktueller Breitengrad (hier: 48° 13,1635m bzw. 48° 13m 13.48s)

7 Hemisphäre (N-Nord, S-Süd)

8 aktueller Längengrad (hier: 16° 22,2463m bzw. 16° 22m 22.3s)

9 E-Ost, W-West

10 Geschwindigkeit in Knoten (hier: 17,99 Kn oder 33,32 km/h; 1 Kn = 1,85 km/h)

11 *True course* (hier: 359,59°; Nord = 0°; siehe Abb. 2.1.5 auf S. 41)

12 Datum (hier: 23. Juli, 2004)

13 Betrag der Variation (hier nicht angezeigt)

14 Richtung der Variation (E-Ost, W-West)

15 *Checksum* (hexadezimal)

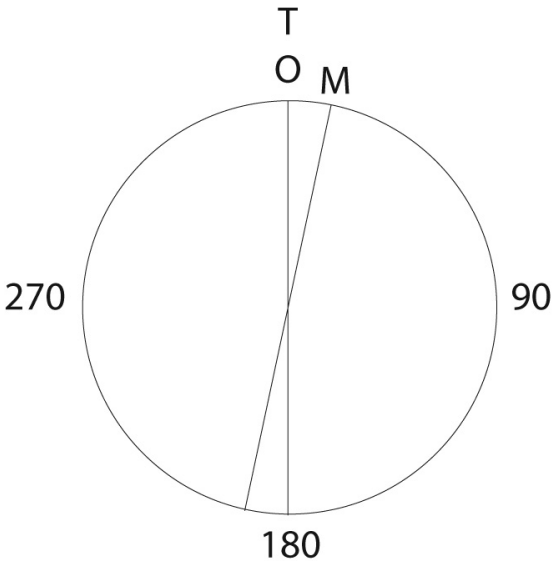


Abbildung 2.1.5.: Der *True Course* (T) weicht vom *Magnetic Course* (M) und der Anzeige eines Kompass erheblich ab. Den Unterschied bezeichnet die Variation.

WGS84

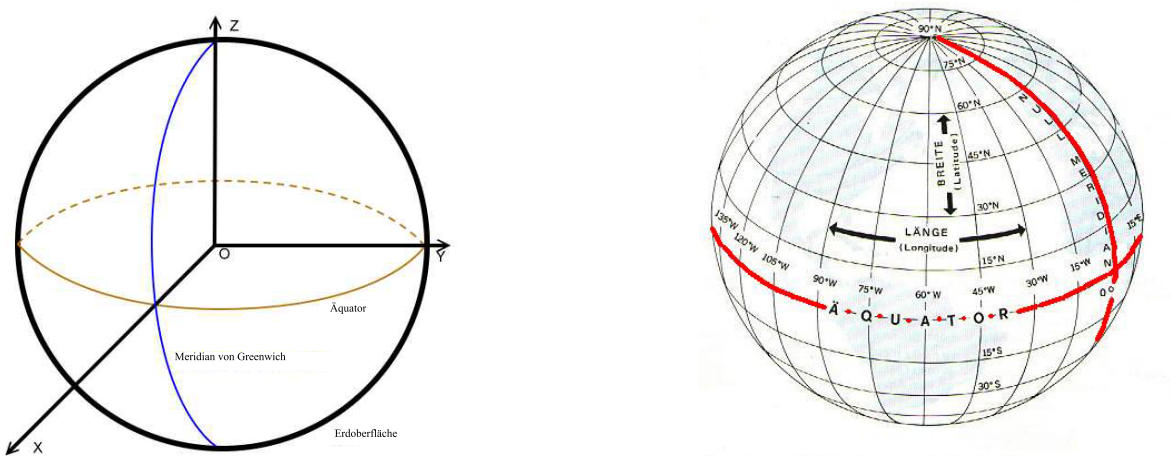


Abbildung 2.1.6.: Das weltweite geodätische Koordinatensystem von 1984 - WGS84 [eigene Darstellung, Chouquet-Stringer (2004), Screen 4]

Die in NMEA-Format angegebenen Längen- und Breitengrade beziehen sich auf das geodätische Koordinatensystem WGS84. Ein Koordinatensystem wird durch das Datum charakterisiert. Das Datum besteht aus dem Referenzellipsoid und einem oder mehreren Fundamentalpunkten.

Im Falle von WGS84 ist das Referenzellipsoid das Erdellipsoid und Fundamentalpunkt Greenwich. Der Erdellipsoid hat folgende Eigenschaften [Bilajbegovic (Chmielorz, 2001), S. 143]:

- $a = 6.378.137,000$ m
- $b = 6.356.752,315$ m
- $f = 1 : 298,257223563$

a und b sind die Achsen des Ellipsoids und f die Abplattung:

$$f = \frac{a - b}{a}$$

2.1.2. Mobile Datenübertragungsverfahren

Bei der Datenübertragung im Mobilfunknetz unterscheidet man grundsätzlich zwei Arten von Verfahren. Zu den leitungsvermittelten Verfahren, ähnlich der Sprachtelefonie, gehören z. B. CSD (**C**ircuit **S**witched **D**ata) bzw. HSCSD (**H**igh **S**peed **C**ircuit **S**witched **D**ata). Zu den paketvermittelten Übertragungsverfahren zählen SMS, GPRS und UMTS (**U**niversal **M**obile **T**elecommunications **S**ystem).

2.1.2.1. Leitungsvermittelte Übertragungsverfahren

Leitungsvermittelte Datenübertragungsverfahren funktionieren ähnlich wie Sprachtelefonie. Als Adressierung dient eine Telefonnummer, die Datenübertragung selbst ist dann jener von ISDN (**I**ntegrated **S**ervices **D**igital **N**etwork) sehr ähnlich. Durch eine Bündelung mehrerer benachbarter Zeitschlitze auf eine logische Verbindung können höhere Datenübertragungsraten erreicht werden. Dieses Verfahren verwendet HSCSD. Die Abrechnung bei leitungsvermittelten Übertragungsverfahren erfolgt auf Basis der Verbindungszeit. [Schnabel (1997–2005), Screen 1f; Wikipedia, HSCSD, Screen 1f]

2.1.2.2. Paketvermittelte Übertragungsverfahren

Die paketorientierten Übertragungsverfahren verfolgen einen alternativen Ansatz. Hierbei werden die Daten beim Sender in einzelne Pakete umgewandelt, als solche übertragen und beim Empfänger wieder zusammengesetzt. Die Übertragungsrate ist dabei von der Fähigkeit des Mobiltelefons abhängig, wie viele Zeitschlitze es parallel nutzen kann. Wenn GPRS aktiviert ist, besteht nur virtuell eine dauerhafte Verbindung zur Gegenstelle (sog. *Always-on*-Betrieb). Erst wenn wirklich Daten übertragen werden sollen, werden auch Daten gesendet. Der Vorteil besteht zum einen in der höheren

Datenübertragungsrate und zum anderen in der Abrechnung. Da während der eigentlichen Datenübertragung nicht dauerhaft ein Funkkanal für einen Benutzer reserviert sein muss, ist die Abrechnung bei GPRS hauptsächlich von der übertragenen Datenmenge und nicht primär von der Verbindungsdauer abhängig. [Schnabel (1997–2005), Screen 1f; Wikipedia, GPRS, Screen 1f]

2.1.3. Java am Mobiltelefon

Inzwischen verfügen fast alle neu auf den Markt gebrachten Mobiltelefone über Java-Unterstützung, was sicherlich als eine der populärsten Anwendungen von *Embedded Java* gelten darf. Die Umsetzung der Programmiersprache für *embedded consumer products*, wie etwa Mobiltelefone oder PDAs, heißt ***J**ava 2 Platform, **M**icro **E**dition*, abgekürzt J2ME.

Die genauen Spezifikationen eines jeden Java-APIs (**A**pplication **P**rogramming **I**nterface) wurden als JSRs (**J**ava **S**pecification **R**equests) veröffentlicht. Eine Sammlung aller JSRs finden sich auf `http://www.jcp.org/en/jsr/overview`, `12. September, 2005`.

Die Grundlage von J2ME bilden die Konfigurationen und die Profile.

2.1.3.1. CLDC

Die CLDC (**C**onnected **L**imited **D**evice **C**onfiguration) wurde im JSR 30 und im JSR 139 definiert. Die Konfigurationen stellen verschiedene Bibliotheken und eine Virtuelle Maschine bereit. Für Mobiltelefone und andere mobile Geräte steht die CLDC zur Verfügung. Die CLDC wird noch ständig weiterentwickelt und steht derzeit (Stand: 30. Juli, 2005) als Version 1.1 zum freien *Download* auf `http://java.sun.com/products/cldc/index.jsp`, `12. September, 2005` bereit. Die Unterschiede zur ersten Version liegen vor allem in den besseren Möglichkeiten mit Grafik umzugehen und zum anderen in der fehlenden Unterstützung von Fließkommazahlen, ein Umstand der die Verwendung von CLDC 1.0 bei der Implementierung der BusApplikation sehr erschwerte.

2.1.3.2. MIDP

Profile sind die APIs, die es zu einer Konfiguration gibt. So wurde vor allem für Mobiltelefone das MIDP (**M**obile **I**nformation **D**evice **P**rofile) entwickelt. MIDP 1.0 ist im JSR 37 definiert. Java-Applikationen, die auf Grundlage des MIDP entwickelt wurden, nennt man auch kurz MIDlet. Dabei kann es sich zum Beispiel um die häufig beworbenen „Java-Spiele für Handys“ handeln.

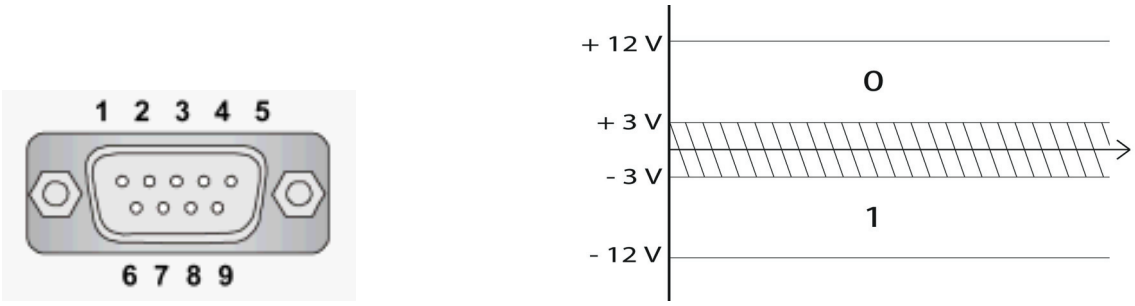
Mittlerweile gibt es auch MIDP 2.0, welches im JSR 118 definiert wurde. Diese Version ist vor allem um die Bedürfnisse der Spieleentwickler erweitert worden. So gibt es erweiterte Unterstützung für Sounds, aber auch ein einfacher XML-*Parser* wurde verwirklicht. [Wikipedia, J2ME, Screen 1]

2.1.4. RS-232 - Die serielle Schnittstelle

Die RS-232 (**R**ecommended **S**tandard 232) ist eine Spannungsschnittstelle im Gegensatz zu einer Stromschnittstelle. D.h. die verschiedenen Spannungspegel stellen die Information dar. Sie entspricht einer V.24-Schnittstelle hinsichtlich Signalsemantik, Elektrik und Steckerbelegung. Dabei geht der Spannungsbereich für die logische Eins von -3 Volt bis -12 Volt und die logische Null wird durch Spannungen zwischen +3 Volt und +12 Volt abgebildet. Die Abbildung der logischen Null als positive Spannung und der logischen Eins als negative Spannung nennt man negative Logik.

Eine RS-232 Verbindung stellt eine serielle Datenübertragung dar, d.h. die Bits werden nacheinander auf einer Leitung übertragen, im Gegensatz zur parallelen Datenübertragung, bei der mehrere Bits gleichzeitig auf mehreren verschiedenen Leitungen übertragen werden.

Da die Spannung mit der Länge einer Leitung (wegen des größer werdenden elektrischen Widerstandes) immer kleiner wird, ist die Leitungslänge begrenzt (auf ca. 25 bis 100 Meter, je nach Kabel- und Steckverbinderqualität). Ein weiterer begrenzender Faktor ist die Laufzeit des Signals. Da eine RS-232 Schnittstelle am Ende der Leitung nicht mit einem Wellenwiderstand abgeschlossen ist, gibt es unweigerlich eine Leitungsreflektion. Mit zunehmender Übertragungsrate und Kabellänge stören die Reflexionen immer mehr die Datenübertragung.



(a) (b)

Abbildung 2.1.7.: Schematische Darstellung einer RS-232-Schnittstelle mit neun Pins (a) Die Spannungspegel einer V.24-Schnittstelle (b)

Als Steckverbindung wurden nach der ursprünglichen Norm 25-polige Stecker und Buchsen (Sub-D) benutzt. Da viele der 25 Leitungen reine Drucker- bzw. Terminal-Steuerleitungen aus der elektromechanischen Ära sind, die für die meisten Verbindungen mit moderneren Peripheriegeräten nicht benötigt werden, haben sich heute 9-polige Sub-D-Stecker und Buchsen (siehe Abb. 2.1.7(a) auf S. 44) etabliert, die beim ersten IBM PC (**I**nternational **B**usiness **M**achines) ursprünglich als reine Notlösung zum Platzsparen eingeführt worden waren.

Da es keine Taktleitung gibt, die die Übertragung synchronisiert, muss auf beiden Seiten der Übertragungsstrecke dieselbe Übertragungsgeschwindigkeit (in Baud) eingestellt sein, damit die Übertragung klappt.

Pro übertragenem Datensatz (Byte) wird die Übertragung mittels des Startbits eingeleitet. Serielle Übertragungen funktionieren grundsätzlich aber auch ohne Startbit. Darauf folgen 5-8 Datenbits (Nutzdaten) mit dem niederwertigsten Bit (LSB - **L**ast **s**ignificant **B**it) zuerst, wahlweise ein Parity-Bit und schließlich ein oder zwei Stoppbits.

Da alle diese Variationen in den Standards nicht festgelegt sind, müssen bei beiden Geräten, die an der Kommunikation beteiligt sind, die Parameter gleich eingestellt sein, bevor eine erfolgreiche Kommunikation zustande kommen kann.

Heutzutage hat sich für diese Einstellungen ein „üblicher Standard“ herauskristallisiert, der von vielen Geräten benutzt wird. Diese „üblichen“ Einstellungen sind: 8 Datenbits, kein Parity, 1 Stoppbit, was oft als 8N1 abgekürzt wird. Die Baudrate variiert stark, je nach Qualität der Leitung und des Geräts. Für die verwendeten Parameter bei einer Übertragung von einem GPS-Empfänger siehe 2.1.1.5 auf S. 39. [Wikipedia, RS_232, Screen 1f, Janovsky (November 2002). S. 1ff]

2.2. Die Fahrzeugausrüstung als Komponente einer RBL

In 1.3 wurde schon gezeigt, wie sich die Fahrzeugausrüstung, in diesem Falle die BusApplikation in das Gesamtbild einer RBL fügt. In anderen RBLs gehören zudem Funkgerät, der Bordrechner, die Wegstreckenzähler, das Fahrgasterfassungsgerät sowie die Sender/-Empfänger als Schnittstelle zum Datenaustausch zwischen Fahrzeug und Zentrale zur Standardausrüstung. [Hoepke (1995), S.115f; Girnau u. a. (2001), S.142]

Mit einigen Erweiterungen bietet die BusApplikation zusammen mit dem Mobiltelefon nun die Möglichkeit, eine Vielzahl der Funktionen all dieser Geräte in einem Gerät zu vereinen.

2.2.1. Die Aufgaben der BusApplikation

Die BusApplikation hat folgende Aufgaben zu erfüllen:

- Schnittstelle und *User Interface* des BEHA-Systems für den/die Buslenker/in (siehe dazu auch 0.1.1.2 auf S. 4 und Abb. 1.3.1 auf S. 27)
- Informationsquelle für den/die Buslenker/in mit folgenden Informationen:
  - Fahrplan
  - bestellte/nicht bestellte Bedarfshaltestellen
- Weitere Bestellmöglichkeit für Fahrgäste (siehe dazu auch 1.2 auf S. 17)

Über die Schnittstelle kann der/die Busfahrer/in folgende Aufgaben erledigen:

- Sich zu einem Dienst anmelden
- Abfahrten an Bedarfshaltestellen für Fahrgäste bestellen
- Aussteigewünsche eingeben
- Bestellungen annehmen/ablehnen
- Verspätungen melden (siehe dazu auch 1.2.4 auf S. 24)
  [Lichtl u. a. (2005), S.10ff]

2.2.2. Die Anforderungen an die BusApplikation

An die BusApplikation werden Anforderungen von verschiedener Seite gestellt. Zum einen muss sie den technischen Anforderungen des Mobiltelefons gerecht werden und muss mit diesem kompatibel sein. Dies hängt vor allem von der Version der JVM (**J**ava **V**irtual **M**achine) ab (siehe dazu auch 2.1.3 auf S. 43). Zum anderen muss sie den Gegebenheiten eines Linienbusbetriebs entsprechen und darf den/die Busfahrer/in vor, während und nach einer Fahrt nicht beeinträchtigen. So wurden beim Entwurf der BusApplikation besonders hohe Anforderungen in Bezug auf Bedienungsfreundlichkeit und benutzergerechte Informationsdarstellung gestellt. Die BusApplikation muss *Plug&Play*-tauglich sein und darf keine Konfigurationsarbeit von dem/der Busfahrer/in verlangen. Sie darf den/die Busfahrer/in nicht mit Informationen überfluten und der/die Busfahrer/in muss trotz eines extrem kleinen Bildschirms mit einem Blick alle relevanten Informationen erhalten, die er/sie benötigt. Wenn von ihm/ihr eine Interaktion benötigt wird, so sollte diese rasch, ohne Komplikationen und möglichst mit einem einzigen Tastendruck erfolgen können. [Pausits (2001), S.49; Pfleger und Linauer (2001), S.118]

Die Kenntnisnahme besonders wichtiger Informationen, wie etwa eine eingegangene Bestellung und eine damit verbundene Änderung der Fahrtroute, auf deren besondere Wichtigkeit mit einer blinkenden Anzeige und einem lauten Signalton aufmerksam gemacht wird, muss vom Fahrer mit einem Tastendruck quittiert werden. Darum ist das Mobiltelefon in einer Halterung in einem günstigen Griffbereich des/der Fahrers/in positioniert.

2.2.3. Das Mobiltelefon

Die heutige Generation von Mobilgeräten, im speziellen der Mobiltelefone oder „Handys“, wie sie umgangssprachlich genannt werden, lässt kaum mehr Wünsche offen. Nebst dem eigentlichen Telefonieren ist es möglich, Kurznachrichten bzw. SMS zu versenden, zu fotografieren, Spiele zu spielen und vieles mehr. Die Telefonie spielt im Marketingbereich nur noch eine nebensächliche Rolle. Längst sind die verschiedensten *Features* in den Vordergrund getreten und ständig wird nach dem Mehrwert eines Mobiltelefons gesucht.

So war die Suche nach einem geeigneten Mobiltelefon keine leichte Aufgabe, auch wenn man bei der Vielfalt des Angebots anderes vermuten möchte.

2.2.3.1. Das Anforderungsprofil des Mobiltelefons

Das Anforderungsprofil des Mobiltelefons sieht sehr einfach aus. Es muss

1. einfach zu bedienen sein (Bedienbarkeit).
2. das Display gut lesbar sein.
3. der Standardakku, auch bei starker Benutzung, lange genug halten.
4. GPRS-fähig sein.
5. J2ME-fähig sein.
6. die serielle Schnittstelle über die JVM erreichbar sein.

Die ersten vier Punkte scheinen auf den ersten Blick von jedem modernen Mobiltelefon erfüllt zu sein. Doch die Auswahl wurde schnell schon beim ersten Punkt eingeschränkt. So kamen „Klapphandys“ schon bald nicht in Frage, da das Scharnier anfällig für Schäden ist und es teilweise Probleme mit den dafür vorgesehenen Halterungen im Bus gibt.

Ein leicht zu bedienendes Mobiltelefon hat klassisch angeordnete Tasten mit einem Abstand, der so groß ist, dass unabsichtliches Drücken einer daneben liegenden Taste ausgeschlossen werden kann. Das Menü, über das die verschiedenen Applikationen aufgerufen und gestartet werden, soll übersichtlich gestaltet und leicht bedienbar sein. Die Bedienbarkeit soll zudem auch dann noch gegeben sein, wenn es in der Halterung im Bus befestigt wird.

Das Display soll kontrastreich und groß genug sein, um Schriften leicht lesen zu können. Es muss außerdem mehrzeilig sein und soll problemlos fünf Zeilen Platz bieten. Die Anzahl der Farben spielt keine Rolle. Ein 8-Farben Graustufendisplay ist kein Problem, ein mehrfarbiges Display ist jedoch vorzuziehen und heutzutage ohnehin Standard.

Auch der Akku stellt zumeist kein Problem mehr da, da durch Lithium-Ionen-Akkus eine ausreichende Betriebsdauer gewährleistet ist. Zudem ist es meist möglich den Akku während des Betriebes über ein mit dem Zigarettenanzünder verbundenes Kabel aufzuladen. Ein Problem hierbei ist, das durch das Ladekabel fast immer die Schnittstelle für Daten oder Freisprechanlagen, welche zumeist die selbe ist, unzugänglich wird. Siemens hat dieses Problem durch ein Y-Kabel nach dem Prinzip einer Mehrfachsteckdose gelöst. Dieses Y-Kabel lässt das Aufladen des Akkus und gleichzeitige Datenübertragung zu.

Das Mobiltelefon soll aufgrund der Vorteile, die GPRS bietet, wie die höhere Datenübertragungsrate und die günstigere Abrechnung, GPRS-fähig sein (siehe auch 2.1.2.2 ab S. 42).

Die meisten Mobiltelefone unterstützen inzwischen eine Anbindung an einen PC, entweder über eine serielle Schnittstelle oder via USB, Infrarot oder Bluetooth. Die J2ME-Fähigkeit ergibt sich aus der Notwendigkeit, dass die BusApplikation in Java geschrieben wurde. Einzige Bedingung an die JVM ist dabei die Möglichkeit die serielle

Schnittstelle des Mobiltelefons via Java ansprechen zu können, eine nicht selbstverständliche Eigenschaft.

2.2.3.2. Die Wahl des passenden Mobiltelefons

Bei der Wahl des Mobiltelefons galt es, ein Mobiltelefon zu finden, dass die oben genannten Anforderungen erfüllte unter Berücksichtigung des Preises und der Verfügbarkeit. Da Postbus eine Kooperation mit A1 eingegangen war, musste das Mobiltelefon aus dem Angebot von A1 kommen.

Das Angebot und die Informationen über die angebotenen Mobiltelefone sind von A1 zwar sehr umfangreich gestaltet, aber Auskünfte über die genauen technischen Spezifikationen sind trotzdem nur schwer zu beschaffen. So spielte z. B. die Java-Version bei der Wahl des Mobiltelefons zwar nur eine untergeordnete Rolle, aber trotzdem war es wünschenswert schon vor der Anschaffung die Version zu kennen, um Kompatibilitätsprobleme zu vermeiden. Welche Version ein Mobiltelefon unterstützt bzw. ob J2ME überhaupt unterstützt wird, ist oftmals nur schwer herauszufinden. Da technische Details in der Werbung oft nur schwer vermittelbar sind, werden sie oftmals verhüllt bzw. bleiben unerwähnt. Für den Endverbraucher haben diese zumeist auch keine Bedeutung. So wird die J2ME-Fähigkeit oft durch das Wort „Spiele“ umschrieben, da viele Hersteller und Provider darin den Hauptnutzen von J2ME sehen. Manchmal wird noch in Klammer Java gesetzt, oder auch als eigener Punkt angeführt, um die Liste der verfügbaren *Features* zu verlängern. Die Version wird geflissentlich verschwiegen, da das Gros der Mobiltelefone erst CLDC 1.0 und MIDP 1.0 unterstützen. Auch bei neueren Mobiltelefonen ist dies aufgrund älterer Prozessoren oft nicht anders.

Die genaue Spezifikation des J2ME, welche von Hersteller zu Hersteller und von Mobiltelefon zu Mobiltelefon variieren kann, ist auf den *Homepages* zumeist sehr versteckt oder gar nicht zu finden. Erst in Foren, *Communities* oder Tests in Zeitschriften wird die Version aufgedeckt. Eine gute *Community* im Netz mit vielen Beiträgen, einem Forum und guten, umfassenden Testberichten ist z. B. `http://www.inside-handy.de`, `22. Juli, 2005`. Manchmal aber ist der Gang zu einem Verkäufer die einfachste Methode.

Da die BusApplikation jedoch mit Bedacht auf Abwärtskompatibilität geschrieben wurde, spielte die Version, wie eben erwähnt, nur eine untergeordnete Rolle, da die Versionen CLDC 1.0 und MIDP 1.0 genügen, auch wenn diese einige Einschränkungen, vor allem im Bereich der graphischen Darstellung, mit sich brachten. Darum genügte oftmals schon der Hinweis auf Java. Die Frage nach der Fähigkeit der JVM, die serielle Schnittstelle anzusprechen, bedurfte hingegen aufwändiger Recherchen. Letztendlich

(a) Sony Ericsoon T610

(b) Sony Ericsson T630

(c) Motorola V525

(d) Motorola A835

(e) Motorola A920

(f) Motorola A925

(g) Siemens S55

(h) Siemens MC60

(i) Siemens M55

Abbildung 2.2.1.: Die überprüften Mobiltelefone im Überblick

konnten hier nur eigene Tests Aufschluss geben, da eine genaue Definition der J2ME selten vom Hersteller veröffentlicht wird.

Folgende Geräte wurden auf die oben genannten Kriterien hin überprüft (siehe Abb. 2.2.1 auf S. 50):

- Sony Ericsson T610
- Sony Ericsson T630
- Motorola A-Serie (A925, A920, A835)
- Motorola V525
- Siemens S55
- Siemens MC60
- Siemens M55

Geräte des Weltmarktführers Nokia wurden nicht überprüft, da der Projektleiter Dipl.-Ing. Balázs Lichtl Bedenken bei J2ME und der seriellen Schnittstelle meldete. Es konnten nicht alle Mobiltelefone getestet werden, da nicht für alle Modelle Testgeräte beschafft werden konnten. Im Folgenden sollen nun die eben genannten Mobiltelefone untersucht werden. Die Preisangaben beziehen sich dabei auf die von A1 angegebenen Preise ohne Erstanmeldung im August 2004.

Sony Ericsson T610 und Sony Ericsson T630

Das T630 ist der Nachfolger des T610. Die Unterschiede sind nur marginal und liegen vor allem in den erweiterten Business-Funktionen (Kalender, E-Mail-Empfang, ...) des T630. GPRS und eine Anbindung an einen PC via USB oder RS-232 gehört bei den Mobiltelefonen von Sony Ericsson zur Standardausführung. Beide Mobiltelefone bestechen vor allem durch ihre gute Bedienbarkeit, das gut lesbare, farbige, hochauflösende Display mit einer Auflösung von 128 x 160 Pixeln. Auch Java wird unterstützt. Die Version wird vom Hersteller nicht angegeben. Der Preis des T610 liegt bei 179 €. Das T630 kostet 379 € und kommt aufgrund des höheren Preises nicht in Frage.

Motorola A-Serie

Die drei Mobiltelefone der A-Serie A835, A920 und A925 sind schon für die nächste Generation des mobilen Internets gerüstet und unterstützen UMTS. Auch ein DGPS-Empfänger ist integriert. Das A835 liegt besonders gut in der Hand und scheint durch seine Tastatur gut für den Einsatz bei BEHA geeignet zu sein. Die anderen beiden Geräte sind über einen für BEHA ungeeigneten Touchscreen zu bedienen. Java und Bluetooth sind weitere *Features*. Ein Akku mit einer Standbyzeit von bis zu 140 Stunden komplettiert das Angebot. Daher käme das Mobiltelefon ohne Verkabelung aus. A1 bot

dieses Mobiltelefon im August 2004 noch nicht an, aber die Konkurrenz (Hutchinson 3G) gibt die Geräte der A-Serie bei Erstanmeldung umsonst dazu.

Motorola V525

Das V525 ist dem T610 von den *Features* her ähnlich. Es unterstützt CLDC 1.1. Als „Klapphandy " kam es zwar von Anfang an nicht für den Einsatz in Frage, trotzdem wurde ein Testgerät zur Verfügung gestellt. Wie sich herausstellte, konnte zwar die serielle Schnittstelle via Java angesprochen werden, aber sie konnte nicht via Java aktiviert werden, da eine Stromsparfunktion die Schnittstelle erst aktivierte, nachdem ein externes Gerät angeschlossen worden ist. Das bedeutet, wenn ein/e Busfahrer/in das Mobiltelefon aus irgendeinem Grund neu starten muss, müsste er auch den GPS-Empfänger erst aus- und einstecken, bevor er funktioniert.

Siemens S55

Das Display des S55 ist zwar mit 101 x 80 Pixel sehr klein, aber noch immer gut lesbar. Die Tasten sind sehr leichtgängig und es ist einfach in der Bedienung. Das Gerät konnte getestet werden und es unterstützt alle verlangten *Features*. Trotzdem ist es den oben genannten Mobiltelefonen unterlegen und rechtfertigt nicht den Preis von 159 €.

Siemens MC60

Das Display ist das gleiche wie beim S55. Die Tastatur und Menüführung sind aber unzureichend. Die Verarbeitung wirkt billig und das Gerät schnitt auch in anderen Tests in verschiedenen Fachzeitschriften sehr schlecht ab. Aufgrund des geringen Preises von gerade einmal 65 € konnte ein Testgerät angeschafft werden. Das Testgerät bestätigte jedoch nur die Vermutung, dass die Java-Version sehr eingeschränkt und die serielle Schnittstelle nicht ansprechbar ist.

Siemens M55

Auch in diesem Gerät wird das gleiche Display wie in den anderen beiden Mobiltelefonen von Siemens verwendet. Es wurde bei der Einführung von BEHA in Zwettl und im Wienerwald angeschafft. Ein Test deckte jedoch auf, dass die serielle Schnittstelle nicht via Java angesprochen werden kann.

Resumee

Das Motorola A835 wird zwar nicht von A1 angeboten, war aber das am besten geeignetste, da GPS bereits integriert ist. Unter den von A1 angebotenen Geräten bot das Sony

Modell	Bedien-barkeit	Display (Pixel)	max. Akku-laufzeit	GPRS	CLDC/ MIDP-Version	serielle Schnitt-stelle
Sony Ericsson T610	1	128 x 160	315h	✓	1.0/1.0	✓
Sony Ericsson T630	1	128 x 126	300h	✓	1.1/2.0	✓
A925, A920, A835	2	208 x 320	70h	✓	k.A.	✓
Motorola V525	3	176 x 220	200h	✓	1.0/1.0	✓
Siemens S55	2	101 x 80	300h	✓	1.0/1.0	✓
Siemens MC60	4	101 x 80	250h	✓	1.0/1.0	
Siemens M55	2	101 x 80	250h	✓	1.0/1.0	✓

Tabelle 2.2.1.: Die überprüften Mobiltelefone im Überblick, Bewertungen im Schulnotensystem von der Community „inside-handy.de“ übernommen

Ericsson T610 das beste Preis/Leistungs-Verhältnis. Schlussendlich setzte sich aber das Siemens MC60 wegen seines geringen Preises durch.

2.2.4. Der GPS-Empfänger

Abbildung 2.2.2.: Die GPS-Maus Holux GM-210

Als GPS-Empfänger wird die GPS-Maus Holux GM-210 verwendet. Ihre großen Stärken liegen zum einen im geringen Preis, zum anderen in ihrem geringen Gewicht und ihrer geringen Größe. Mithilfe eines Magneten an der Unterseite kann die Maus leicht im Inneren eines Busses angebracht werden.

Die GM-210 unterstützt das NMEA-Protokoll 0183 v.2.2, wiegt knapp 100 g und verbraucht wenig Strom ($<$ 170 mA bei 5 V *Input*). Sie ist DGPS-fähig. Die genauen Spezifikationen sind in der Tabelle 5.0.1 ab Seite 100 nachzulesen.

Von einer Bluetooth-fähigen Maus wurde aufgrund der Kosten Abstand genommen.

2.3. Die BusApplikation

(a) Anmeldevorgang (b) Kursliste

(c) Annahme einer Bestellung (d) Menüfunktionen

Abbildung 2.3.1.: Einige *Screenshots* der BusApplikation

Die Aufgaben und Anforderungen der BusApplikation wurden schon in 2.2.1 ab Seite 46 erläutert. An dieser Stelle soll nur mehr auf die Besonderheiten der BusApplikation in Bezug auf die Fahrzeugortung mittels GPS eingegangen werden.

Die BusApplikation ist nachrichtengesteuert und kann sowohl Nachrichten an die Zentrale senden, als auch von dieser empfangen. Der *Thread* für die Verarbeitung der GPS-Daten läuft im Hintergrund ohne dass der *User* (hier: der Lenker) etwas davon mitbekommt. Neben dem Einlesen und Sammeln der GPS-Daten, liegt die Hauptaufgabe des GPS-*Threads* in der Komprimierung der Daten, da ein GPS-Empfänger schon in kürzester Zeit eine große Menge von Daten erzeugt. Pro Sekunde sind es zwischen

200 und 400 Bytes. Das entspricht in etwa einem MegaByte pro Stunde. Da der Datentransfer jedoch auf sieben MegaByte pro Monat vom Betreiber limitiert ist, müssen ausgehend von einer Betriebsdauer eines Busses von 21 Stunden pro Tag bei mindestens 24 Tagen pro Monat die Daten um mehr als 98 % reduziert werden. Eine zweite Bedingung war, die Komprimierungsrate steuern zu können sowie eine Wiederherstellung der Daten.

2.3.1. Java am Mobiltelefon

Die folgenden Kapitel sollen den Aufbau einer J2ME-Applikation im Allgemeinen und die Funktionsweise des GPS-*Threads* im Besonderen erörtern. Jene Passagen im Quellcode, die besonders wichtig sind, sind hier abgedruckt.

2.3.1.1. Das MIDlet

```
1  public class MIDletTest extends MIDlet {
2      public MIDletTest() {
3           ...
4      }
5      protected void startApp() throws MIDletStateChangeException {
6          ...
7      }
8      protected void pauseApp() {
9          ...
10     }
11     protected void destroyApp(boolean unconditional) throws
           MIDletStateChangeException {
12         ...
13     }
14 }
```

Listing 2.3.1: Ein minimales MIDlet

Die Methode `startApp()` in der Klasse `javax.microedition.midlet.MIDlet` ist der Einstiegspunkt für jede J2ME-Anwendung. Daher muss jede J2ME-Anwendung eine Klasse besitzen die von MIDlet abgeleitet ist und muss diese in ihrem Manifest als Einstiegspunkt ausdrücklich deklarieren.

Ein MIDlet besitzt einen wohldefinierten Lebenszyklus. Dieser spiegelt sich in den grundlegenden Methoden eines MIDlets wider. Es ähnelt in seinem Aufbau einem Applet.

Genau wie ein Applet kennt das MIDlet drei Zustände, welche sich in den Methoden widerspiegeln:

- Paused
- Destroyed
- Active
  [Tödter (Februar 2004), S. 4f]

2.3.1.2. CommReader

Die Klasse `CommReader` liest via serielle Schnittstelle empfangene Daten ein. Zur Verbindung zweier Kommunikationspartner im CLDC wurde das GFC (**G**eneric **C**onnection **F**ramework) im Paket `javax.microedition.io` definiert. Eine Verbindung wird durch das *Interface* `javax.microedition.io.Connection` repräsentiert. Für jedes Protokoll existiert ein direkt oder indirekt von `Connection` abgeleitetes *Interface*. Das Öffnen einer `Connection` geschieht in der Klasse `javax.microedition.io.Connector` mittels einer der zur Verfügung gestellten statischen Methoden. Hier wird die Methode `open(String name)` verwendet. Der Parameter *name* hat die allgemeine Struktur:

scheme:target[;parameter]

Die einzelnen Bestandteile haben folgende Bedeutung:

scheme definiert das Protokoll.

target ist die Netzwerkadresse im protokollabhängigen Format.

parameters sind die zusätzlichen Verbindungsparameter, welche als *name=wert*-Paare mit Semikolons getrennt geschrieben werden.

Für eine Verbindung mit der GPS-Maus via serielle Schnittstelle ergibt sich daher folgender Verbindungsname:

comm:com0;baudrate=4800;bitsperchar=8

Für alle anderen Parameter wie *Parity* Bit, *Handshake* und Stop Bits gelten die *Default*-Werte.

Das MIDlet der BusApplikation startet einen eigenen *Thread*, der mittels der Klasse `CommReader` die von der GPS-Maus kommenden Daten einliest. Die Daten können nur Zeichen für Zeichen aus dem Buffer vom `Connector` geöffneten `InputStream` eingelesen werden. Sobald ein vollständiger Satz ausgelesen wurde, wird eine Instanz von `NMEAReader` mit dem gelesenen Satz als Parameter erzeugt. Der `NMEAReader` funktioniert wie ein *Parser* und erzeugt einen `BusPoint`. Dieser wird dann in einem Container gespeichert. Sobald der Bus eine einstellbare Mindestgeschwindigkeit unterschreitet, wird

ein *Event* ausgelöst (siehe Listing 2.3.3). Dieser *Event* startet die Komprimierung und versendet anschließend die reduzierten GPS-Daten an die Zentrale via GPRS in einem vorgegebenen Nachrichtenformat. Je nach Höhe der Mindestgeschwindigkeit kann die Fahrzeugortung in der Zentrale somit in Echtzeit ablaufen, auf alle Fälle jedoch wird bei jedem Stop eine Nachricht versendet.

```
1  package com.cocosoftware.beha.bus.gps;
2
3  import java.io.*;
4  import javax.microedition.io.*;
5
6  public class CommReader {
7      private CommConnection conn;
8      private InputStream in;
9      private String string_comm = "comm:";
10     private String string_baud = ";baudrate=4800";
11     private String string_bitsperchar = ";bitsperchar=8";
12     private String string_port = "com0";
13
14     public CommReader() throws IOException, SecurityException,
           IllegalArgumentException {
15         String connection = string_comm + string_port + string_baud +
               string_bitsperchar;
16         InputConnection inputConn = (InputConnection) Connector.open(
               connection);
17         in = inputConn.openInputStream();
18     }
19
20     /**
21      * This constructor is only for testing purpose on a PC,
22      * to read a NMEA-output from a textfile.
23      * @param a_testfile The textfile to read from
24      */
25     public CommReader(String a_input_filename) {
26         File testfile = new File(a_input_filename);
27         try {
28             in = new FileInputStream(testfile);
29         } catch (FileNotFoundException e) {
30             e.printStackTrace();
31         }
32     }
33
34     public BusPoint read() throws IOException {
35         char x[];
36         while ((x = getLine()) != null) {
37             NMEAReader reader = new NMEAReader(x);
38             BusPoint bp = reader.getBusPoint();
39             if (bp != null)
```

```
40                 return bp;
41         }
42         return null;
43     }
44
45     private char[] getLine() throws IOException {
46         char[] cbuf = new char[256];
47         int c;
48         int i = 0;
49
50         while ((c = in.read()) != '\n' && c != -1)
51             cbuf[i++] = (char) c;
52         if (c == -1)
53             return null;
54         char nmea[] = new char[i];
55         for (int k = 0; k < i; k++)
56             nmea[k] = cbuf[k];
57         return nmea;
58     }
59
60     public void close() throws IOException {
61         if (in != null) {
62             in.close();
63         }
64         if (conn != null) {
65             conn.close();
66         }
67     }
68
69     public InputStream getIn() {
70         return in;
71     }
72 }
```

Listing 2.3.2: Die Klasse CommReader

```
1     [...]
2     BusPoint bp;
3     while ((bp = cr.read()) != null) {
4         synchronized (container) {
5             container.add(bp);
6             container.notifyAll();
7             if (bp.getCurrentSpeed() < MIN_SPEED) {
8                 doEventAction();
9             }
10        }
11    }
12    [...]
```

Listing 2.3.3: Ein BusPoint wird gelesen, gespeichert und löst einen Event aus

Ein `BusPoint` repräsentiert einen NMEA-Satz. In einer Instanz von `BusPoint` werden Datum, Uhrzeit, Längen- und Breitengrad, Geschwindigkeit und der Kurs mit möglichst wenig Genauigkeitsverlust gespeichert. In Listing 2.3.4 wird in Zeile 1 gezeigt wie ein GPRMC-Satz vor dem Einlesen aussieht, als BusPoint in Zeile 2 und zuletzt wieder als GPRMC-Satz in Zeile 3.

```
1 $GPRMC,085327.354,A,4813.1785,N,01622.2361,E,21.585619,320.37,230704,,*3C
2 85327 482196 163706 39975 32037 230704
3 $GPRMC,085327.000,A,4813.1760,N,01622.2360,E,21.584000,320.37,230704,,*3B
```

Listing 2.3.4: Ein GPRMC vor und nach der Konvertierung in einen BusPoint

Alleine schon durch die Filterung der NMEA-Sätze und nur das Lesen der GPRMC-Sätze kann eine Komprimierungsrate bis zu 70 % erreicht werden. Durch das Speichern nur der notwendigen Daten in Form des `BusPoints` kann diese Rate auf 85 % gesteigert werden. Durch die Speicherung der Rohdaten in Form des `BusPoints` leidet jedoch die Genauigkeit.

Bei der Normalisierung der Längen- und Breitengrade werden die Werte im Format `dd°mm.mmmmm'` in einen Dezimalwert mit vier Nachkommastellen `dd.dddd°` umgerechnet. Im Beispiel im Listing 2.3.4 entstand dabei beim Längengrad ein Fehler von 0.0025 Minuten oder 4,6 Meter und beim Breitengrad ein Fehler von 0.0001 Minuten oder 0,2 Meter – ein durchaus vertretbarer Fehler.

Die Genauigkeit der Geschwindigkeit leidet ebenfalls kaum unter der Umwandlung. Die Abweichung beträgt hier 0.00162 kts oder 3 m/h. Kurs, Datum und Uhrzeit sind von der Umwandlung nicht betroffen, lediglich auf die Zehntelsekunden wurde beim Zeitstempel verzichtet.

```
1     /**
2      * Normalises a value in degree and minutes into a decimal value with
3      * four decimals after the imaginary comma point.
4      *
5      * @param a_val a_val could be even a latitude or a longitude in the
6      * format ddmm.mmmmm for a latitude or dddmm.mmmm for a longitude
7      *
8      * @return the normalised value of a latitude or a longitude
9      */
10    private int normalise(String a_val) {
11        int hour = 0;
12        int minute = 0;
13        int k = (a_val.length() > 8) ? 1 : 0;
14        hour = 10000 * Integer.parseInt(a_val.substring(0, 2 + k));
15        minute = Integer.parseInt(a_val.substring(2 + k)) / 60;
16
17        return hour + minute;
```

```
18      }
19
20      private String denormalize(int a_val) {
21          int decimal = 4;
22          int hour = a_val/10000;
23          int minute = (a_val-hour*10000)*60;
24          String degree = "" + hour + minute;
25          while(degree.length()<(decimal+1))
26              degree = '0' + degree;
27          return (degree.substring(0,degree.length()-decimal)).concat('.'+degree
                .substring(degree.length()-decimal));
28      }
```

Listing 2.3.5: Normalisierung und Denormalisierung eines Längen- oder Breitengrades

Da eine möglichst gute Rekonstruktion eines gefahrenen Kurses angestrebt wird, wurde auf eine statische Komprimierung (z. B. nur jeden fünften GPRMC zu übermitteln) verzichtet. Stattdessen wurde eine dynamische Komprimierung entwickelt, bei der signifikante Streckenpunkte erkannt und übermittelt werden.

Wenn ein Bus in Fahrt ist und z. B. bei einer Haltestelle stehenbleibt so fällt seine Geschwindigkeit unter die eingestellte Mindestgeschwindigkeit (*Default*: 10 km/h). Daraufhin wird ein Event ausgelöst, auf den hinauf alle bis dahin gesammelten Punkte folgendermaßen komprimiert und anschließend versendet werden:

```
1       /**
2        * This method inherits the compressing algorithm for a route. A route
3        * consists of many measuring points, but not all are needed to describe
4        * a route.
5        *
6        * Every measuring point is compressed to a BusPoint inheriting just the
7        * most important information and just the significant points of a route
8        * are chosen. Cause of failure redundance in GPS-data the algorithm just
9        * takes points to compare, which are at least ten meters away of each
10       * other. Thus, also just one point is taken if i.e. the bus stands still
11       * or in a station. That saves most of the data.
12       *
13       * The second step is to take just Beginning- and Endpoint of a direct
14       * route. So direction changes are just registered above a specified
15       * Sensitivity-constant. At least points are also taken where a big
16       * change of speed is registered.
17       *
18       * @return a list inheriting all significant points. This list is
19       *         naturally a lot smaller than the list, given to the
20       *         constructor.
21       */
22      public Enumeration getSignificantPoints() {
23          // compenum are all collected BusPoints
24          if(!compenum.hasMoreElements())
```

```
25              return null;
26
27          //The new Vector to store all significant points.
28          Vector significantPoints = new Vector();
29
30          //The lastSignificant is the point compared to the next BusPoint.
31          BusPoint lastSignificant = (BusPoint) compenum.nextElement();
32
33          //The lastSignificant is stored
34          significantPoints.addElement(lastSignificant);
35
36          //The next BusPoint to compare.
37          BusPoint next = null;
38
39          //The distance between two points.
40          int distance = 0;
41
42          //The course difference between two points.
43          int course = 0;
44
45          //The current speed at the lastSignificant point.
46          int speed_1 = lastSignificant.getCurrentSpeed();
47
48          //The current speed at the next point.
49          int speed_2 = 0;
50
51          //Tells if a big acceleration or deceleration was registered.
52          boolean speed_exceeding = false;
53
54          while (compenum.hasMoreElements()) {
55              while ((course < SENSITIVITY) && compenum.hasMoreElements()) {
56                  while ((distance < MIN_LENGTH) && compenum.hasMoreElements())
                        {
57                      next = (BusPoint) compenum.nextElement();
58                      distance = lastSignificant.getDistance(next);
59                      speed_2 = next.getCurrentSpeed();
60
61                      if ((speed_exceeding = (Math.abs(speed_1 - speed_2) >
                            DELTA_SPEED))) {
62                          //If a big change of speed was registered, take this
63                          //point as an significant Point and store it. Go out
64                          //of this loop.
65                          break;
66                      }
67                  }
68
69                  if (speed_exceeding) {
70                      //Go also out of this loop.
71                      break;
72                  }
```

```
73
74              distance = 0; //Set the distance to zero to get to the next
75                            //point and into the next loop.
76
77              course += Math.abs(lastSignificant.getCourse()
78                  - next.getCourse());
79          }
80
81          course = 0;
82          distance = 0;
83          lastSignificant = next; //Do the same again with the last
84                                  //calculated significant point.
85
86          speed_1 = next.getCurrentSpeed();
87          significantPoints.addElement(next); //store the lastSignificant
88          speed_exceeding = false;
89      }
90
91      return significantPoints.elements();
92  }
```

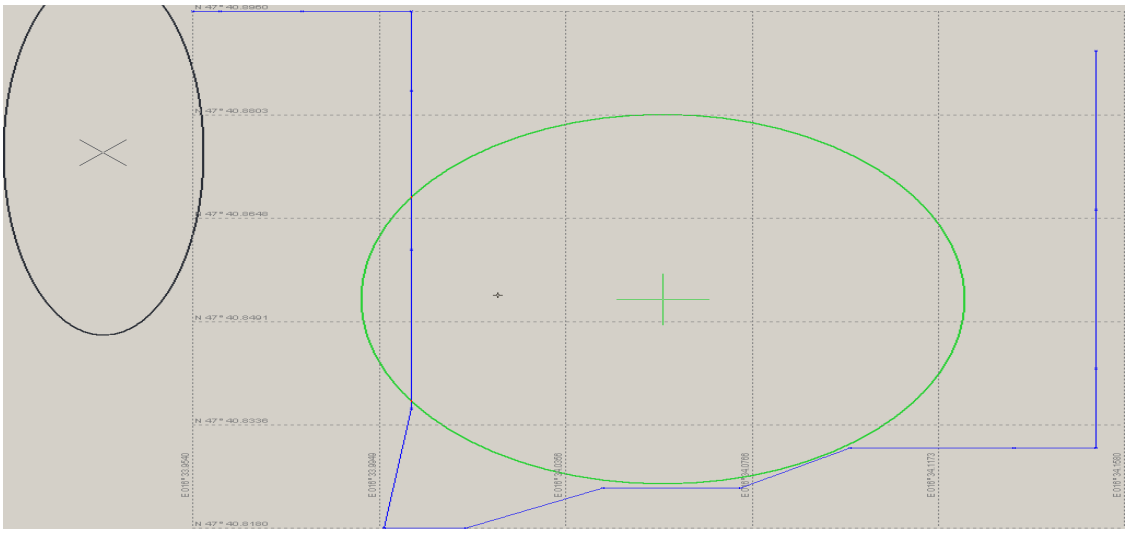
Listing 2.3.6: Suchen der signifikanten Punkte

Bei jedem Punkt wird die Richtungsänderung registriert. Beträgt diese mehr als eine einstellbare Sensitivitätsgrenze und ist sie mehr als eine minimale Länge vom letzten gemessenen Punkt entfernt so wird dieser Punkt in die Liste der signifikanten Punkte aufgenommen. Desweiteren werden Punkte aufgenommen, bei denen große Geschwindigkeitsunterschiede registriert wurden, dies kann z. B. sein wenn der Bus abbremst um in eine Kurve zu fahren oder aufgrund eines Staus oder wenn er stark beschleunigen kann.

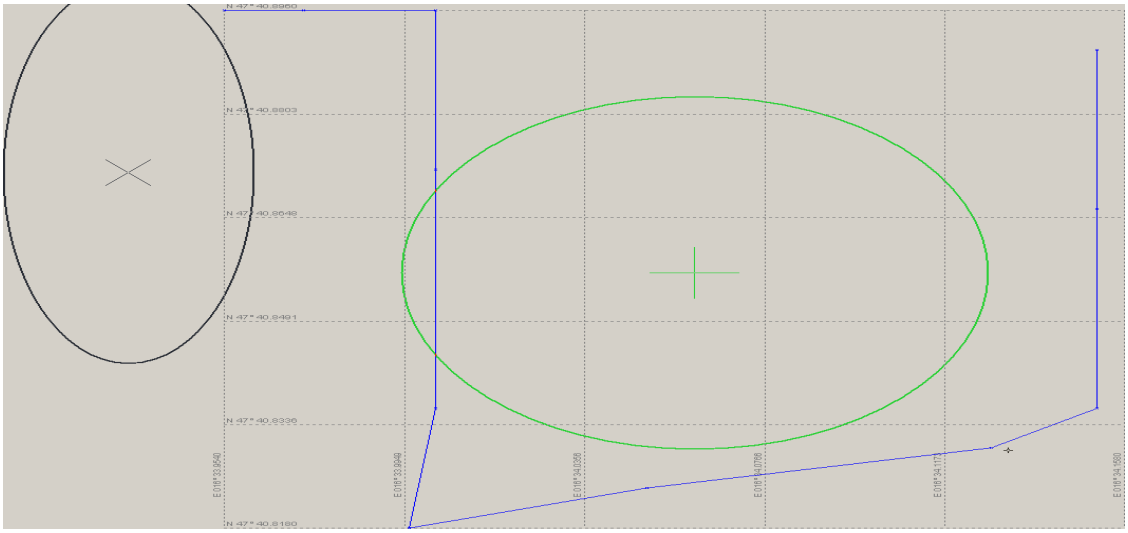
Das Ergebnis ist in Abbildung 2.3.2 zu sehen. Eine Strecke zwischen zwei Bushaltestellen wurde hier aufgezeichnet. Bei einer Sensitivität von 50 und einem minimalen Abstand von 10 m ist die Strecke noch sehr genau erkennbar. Ebenso wie bei einer Sensitivität von 200 und einem minimalen Abstand von 30 m. Bei Abbildung 2.3.2(b) konnte eine Komprimierungsrate von 96,2 % erreicht werden, bei Abbildung 2.3.2(c) konnte bereits die gewünschte Komprimierungsrate von 98,4 % erreicht werden.

Die Konglomerate an Messpunkten, die beim Halten und Wegfahren entstehen, können durch die Komprimierung vollständig eliminiert werden, da lediglich dann Messpunkte gespeichert und übertragen werden, wenn sich der Bus in Bewegung befindet.

(a) Die Messung einer Strecke mittels GPS im Original ohne Komprimierung



(b) Komprimierte Strecke (Sensitivität: 50, min. Abstand: 10m)



(c) Komprimierte Strecke (Sensitivität: 200, min. Abstand: 30m)

Abbildung 2.3.2.: Eine Demonstration der GPS-Komprimierung mit zwei unterschiedlichen Komprimierungsraten

Teil III.

Das RoutingModul

Nachdem nun die Beschaffung der Daten im letzten Kapitel beschrieben wurde, wird in diesem Teil der Arbeit nun die Verarbeitung und der Nutzen beschrieben. In 1.3.1.3 ab Seite 26 wurden das RoutingModul und seine Aufgaben schon vorgestellt. Die Aufgaben seien hier noch einmal kurz genannt:

- Fahrzeugortung
- Verspätungsvorhersage
- Routendisposition
- Datenanalyse und -auswertung

An dieser Stelle werden das grundlegende Graph-API und seine Implementation vorgestellt. Danach wird in einem Experiment ein darauf aufbauender Service für die Erstellung von Statistiken getestet, anhand dessen der für BEHA entwickelte Routing-Algorithmus erläutert wird.

Zuletzt werden geplante, angewandte und ideale Arten der Verspätungsvorhersage erläutert, ein Service, den einmal das RoutingModul übernehmen soll.

3.1. Graphen

3.1.1. Definitionen

Ein Graph ist ein Quadrupel (V, E, α, ω), bestehend aus einer nichtleeren Menge V (Knotenmenge, *Vertex*), einer nichtleeren Menge E (Kantenmenge, *Edge*), einer Abbildung $\alpha : E \to V$, die jeder Kante e ihren Anfangsknoten $\alpha(e)$ zuordnet, und einer Abbildung $\omega : E \to V$, die jeder Kante e ihren Endknoten $\omega(e)$ zuordnet. Mit den Abbildungen α und ω wird der Graph zu einem gerichteten Graphen.

Eine Kante e mit $\alpha(e) = \omega(e)$ heißt Schlinge. Zwei Kanten e, e' heißen parallel, wenn $\alpha(e) = \alpha(e')$ und $\omega(e) = \omega(e')$. Gibt es zu einer Kante e eine oder mehrere parallele Kanten, so nennt man e auch eine Multikante.

Falls es in einem Graphen keine parallelen Kanten gibt, sprechen wir von einem parallelenfreien Graphen. Hier kann die Kantenmenge E in $V \times V$ eingebettet werden. Die Einbettung ι wird definiert als $\iota(e) := (\alpha(e), \omega(e))$.

Ein Graph (V, E, α, ω) zusammen mit einer Abbildung $\beta : E \to \Re$ heißt gewichteter Graph. Falls $\beta(e)\ \epsilon\ \Re_0^+$, dann heißt $\beta(e)$ die Länge der Kante e. Ist $\beta(e) \geq 0$ für alle e, dann heißt der Graph positiv.

Sei $G = (V, E, \alpha, \omega)$ ein Graph. Eine Kante e heißt bidirektional, wenn zu e ein $e'\ \epsilon\ E$ existiert mit $\alpha(e) = \omega(e')$ und $\omega(e) = \alpha(e')$. Ist der Graph gewichtet, so soll außerdem $\beta(e) = \beta(e')$ gelten. Eine Kante, für die dies nicht gilt, heißt unidirektional. Sind alle Kanten bidirektional, so heißt der Graph ungerichtet.

Es seien $G = (V, E, \alpha, \omega)$ ein Graph und $e_1, \ldots, e_n\ \epsilon\ E$. Das n-Tupel $(e_1, e_2, \ldots, e_n)$ heißt Weg (von $\alpha(e_1)$ nach $\omega(e_n)$), wenn $\omega(e_i) = \alpha(e_{i+1})$ für alle $i = 1, \ldots, n-1$ gilt. Ist $w = (e_1, e_2, \ldots, e_n)$ ein Weg, dann heißt e_1 die Anfangskante, e_n die Endkante, $\alpha(e_1)$ der Anfangsknoten und $\omega(e_n)$ der Endknoten des Weges. Die Menge aller Wege in einem Graphen heißt W. Ein Weg $w' = (e'_1, e'_2, \ldots, e'_m)$ heißt Teilweg des Weges w, wenn ein Index i mit $1 \leq i \leq n - m + 1$ existiert mit $e_{i+j-1} = e'_j$ für $j = 1, \ldots, m$. Ein Weg $(e_1, \ldots, e_n)$ heißt geschlossen, wenn sein Anfangsknoten gleichzeitig sein Endknoten ist, das heißt $\alpha(e_1) = \omega(e_n)$.

Der Graph G heißt zusammenhängend, wenn zu je zwei Knoten $u, v\ \epsilon\ V$ immer mindestens ein Weg von u nach v existiert.

Ein Weg (e_1, e_2) heißt ein Abbiegeverbot bei $\omega(e_1)$, wenn die Sequenz $\ldots, e_1, e_2, \ldots$ in einem Weg ausdrücklich nicht vorkommen darf. Ein Umkehrverbot heißt ein Weg wenn die Sequenz $\ldots, e_1, e_1, \ldots$ in einem Weg ausdrücklich nicht vorkommen darf. T ist die Menge aller Abbiegeverbote und Umkehrverbote. [Schmid (2000), S. 14ff]

3.1.2. Die Darstellung eines Graphen im RoutingModul

Der verwendete Graph bei BEHA zur Darstellung einer Region mit allen regulären Haltestellen und Bedarfshaltestellen kann mit folgenden Eigenschaften beschrieben werden:

- gerichtet
- gewichtet
- parallelfrei
- schlingenfrei
- unidirektional
- zusammenhängend
- planar

Im Graph-API des RoutingModuls wird ein Graph in Form einer Adjazenz- oder Nachbarschaftsliste gespeichert, bei der zu jedem Knoten die Menge aller von ihm wegweisenden Kanten inklusive deren Endpunkte, Gewichte und andere Eigenschaften, wie z. B. Abbiegeverbote, der Kante aufgeführt sind.

Folgender gegebener Graph soll als Beispiel für weitere Berechnungen dienen:

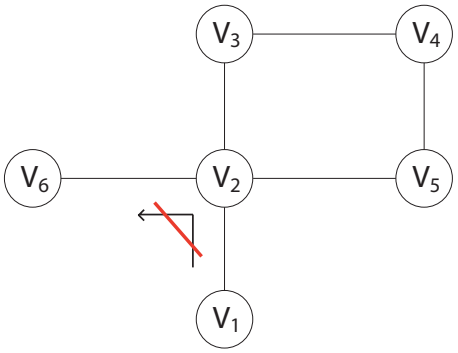


Abbildung 3.1.1.: Ein Graph für das Routing mit einem Abbiegeverbot, sechs Kanten und sechs Knoten

$$
\begin{aligned}
G &= (V, E, \alpha, \omega, \beta, T) \\
V &= \{v_1, v_2, v_3, v_4, v_5, v_6\} \\
E &= \{e_{12}, e_{23}, e_{34}, e_{45}, e_{52}, e_{26}\} \\
T &= \{t(e_{12}, e_{26})\}
\end{aligned}
$$

Für alle $e \,\epsilon\, E$ gilt $\beta(e) = 10$. Alle Kanten sind bidirektional. Im folgenden gilt auch folgende Notation:

Eine Kante heißt e_v wenn $v = \alpha(e_{vu})$ und e'_v wenn $v = \omega(e_{uv})$.

Anfangsknoten	**Endknoten**	**Kante**	**Eigenschaften der Kante**
$v_1 = \alpha(e_1)$	$v_2 = \omega(e_1)$	e_1	$\beta(e_1) = 10$, $t(e_1, e'_6)$
$v_2 = \begin{cases} \alpha(e'_1) \\ \alpha(e_2) \\ \alpha(e'_5) \\ \alpha(e'_6) \end{cases}$	$v_1 = \omega(e'_1)$	e'_1	$\beta(e'_1) = 10$
	$v_3 = \omega(e_2)$	e_2	$\beta(e_2) = 10$
	$v_5 = \omega(e'_5)$	e'_5	$\beta(e'_5) = 10$
	$v_6 = \omega(e'_6)$	e'_6	$\beta(e'_6) = 10$
$v_3 = \begin{cases} \alpha(e_3) \\ \alpha(e'_2) \end{cases}$	$v_4 = \omega(e_3)$	e_3	$\beta(e_3) = 10$
	$v_2 = \omega(e_2)$	e'_2	$\beta(e'_2) = 10$
$v_4 = \begin{cases} \alpha(e_4) \\ \alpha(e'_3) \end{cases}$	$v_5 = \omega(e_4)$	e_4	$\beta(e_4) = 10$
	$v_3 = \omega(e'_3)$	e'_3	$\beta(e'_3) = 10$
$v_5 = \begin{cases} \alpha(e_5) \\ \alpha(e'_4) \end{cases}$	$v_2 = \omega(e_5)$	e_5	$\beta(e_5) = 10$
	$v_4 = \omega(e'_4)$	e'_4	$\beta(e'_4) = 10$
$v_6 = \alpha(e_6)$	$v_2 = \omega(e_6)$	e_6	$\beta(e_6) = 10$

Tabelle 3.1.1.: Speicherung des Beispielgraphen in einer Adjazenzliste

Diese Art der Speicherung übernimmt die Klasse `com.cocosoftware.util12.graph.AdjacencyList`. Sie speichert Kanten (`com.cocosoftware.util12.graph.Edge`) und Knoten (`com.cocosoftware.util12.graph.Vertex`) in einer *Key-Value*-Struktur.

Die Klasse `AdjacencyList` ist dabei von `java.util.HashMap` abgeleitet. Key ist der Anfangsknoten einer Kante und Value wiederum eine `HashMap`, deren Key der Endknoten der Kante und Value alle Attribute einer Kante sind.

Die Abbildung einer Region mit allen Haltestellen und Kreuzungen, wobei jeder Punkt ein *Vertex* ist, in einer Datenbank übernimmt die Klasse `at.cocomobile.bus.server.routing.region.RegionDAO`. Sie übernimmt das Lesen und Schreiben aus und in eine Datenbank. Die Daten für das erstmalige Schrei-

ben in die Datenbank kommen aus einer nach festen Vorgaben geschriebenen Excel-Tabelle. Diese wird unter Zuhilfenahme des Open-Source-Projektes von Apache Jakarta (`http://jakarta.apache.org/`, `12. September, 2005`) in eine XML-Datei konvertiert und hernach können die Daten sehr einfach eingelesen werden.

3.1.3. Routing

Bei der Suche nach dem kürzesten bzw. optimalsten Weg (im Folgenden als Routing bezeichnet) gibt es 3 verschiedene Ansätze:

STSP ***S**ource **T**arget **S**hortest **P**ath*: Von nur einem Startknoten wird zu nur einem Zielknoten der kürzeste Weg gesucht.

SSSP ***S**ingle **S**ource **S**hortest **P**ath*: Von nur einem Startknoten wird der kürzeste Weg zu allen Knoten gesucht.

APSP ***A**ll **P**air **S**hortest **P**ath*: Zwischen allen Paaren von Start- und Zielknoten werden die kürzesten Wege gesucht.

Verallgemeinernd kann gesagt werden, dass im schlechtesten Fall, so dass ein vordefiniertes Abbruchkriterium nicht greift, das STSP-Problem ebenso aufwändig ist, wie das SSSP-Problem. Das APSP-Problem kann durch $|V|$-fache Anwendung eines SSSP-Algorithmus gelöst werden. [Schmid (2000), S. 21]

Das größte Problem bei der Suche nach einem geeigneten Routing-Algorithmus passend für BEHA, war die Berücksichtigung der Abbiege- und Umkehrverbote. Während der Entwicklung des letztendlich verwendeten Algorithmus wurden noch zwei andere Algorithmen getestet. Im Folgenden werden nun zwei bekannte Algorithmen, der Dijkstra- und der Bellman-Ford-Algorithmus, vorgestellt. Danach wird der bei BEHA verwendete Algorithmus vorgestellt.

3.1.3.1. Der Dijkstra-Algorithmus

Edsger Wybe Dijkstra wurde 1930 in Rotterdam geboren und gilt heute als einer der einflussreichsten niederländischen Informatiker. Er arbeitete von 1952 bis 1962 am *Mathematisch Centrum* in Amsterdam als Professor für Mathematik. Später lehrte er noch an der TU Eindhoven und an der Universität von Texas in Austin. 2002 starb er in den Niederlanden an Krebs.

Unter seinen Beiträgen zur Informatik finden sich Dijkstras Algorithmus zur Berechnung des kürzesten Weges in einem Graphen, die erstmalige Einführung von Semaphoren sowie eine Abhandlung über den Goto-Befehl und warum er nicht benutzt werden sollte. Er führte den Begriff der strukturierten Programmierung in die Informatik ein.

Im Jahre 1972 erhielt Dijkstra den Turing-Preis für seine Abhandlung über „Technik und Kunst der Programmiersprachen“.

Der 1959 entwickelte Dijkstra-Algorithmus basiert auf dem *Greedy*-Prinzip (*engl.*: gierig). Das Prinzip geht davon aus, dass man das annähernd beste Ergebnis erzielt, indem man immer das nächstbeste Stück (hier: Kante) nimmt und ist eine Weiterentwicklung der Breitensuche. Grundsätzlich ist der Algorithmus von Dijkstra nur zur Ermittlung der Entfernung zweier Knoten in einem Graphen gedacht. Durch eine Erweiterung jedoch, bei der jeder Knoten von einem Startknoten ausgehend seinen Vorgänger speichert, kann die Kanten- bzw. Knotenfolge ermittelt werden.

Um den kürzesten Weg w_{min} von einem Startknoten v_s zu einem Zielknoten v_d zu finden, wird zunächst eine Liste aller Knoten V initialisiert. Jedem Knoten wird dabei eine Distanz d und ein Vorgänger p zugeordnet. d hat zunächst einmal den Wert unendlich (∞) und p, der zuletzt besuchte Knoten, *null*. n ist die Anzahl aller Knoten und m die Anzahl aller Kanten. [Fellner (2004), S. 47; Wikipedia, Dijkstra, Screen 1]

$d(V) = \infty$
$p(V) = \text{null}$
$d(v_s) = 0$
$p(v_s) = v_s$
für $i = 1, 2, \ldots, n$:
    wähle v aus V mit $d(v)_{min}$
    OPTIONAL: falls $v = v_d$: STOP
    $V = V - v$
    für alle e aus E mit $\alpha(e) = v$
        wenn $d(v) + \beta(e) < d(\omega(e))$
            $d(\omega(e)) = d(v) + \beta(e)$
            $p(\omega(e)) = v$

In der Klasse `AdjacencyList` ist der Dijkstra-Algorithmus folgendermaßen implementiert:

```
1  private MarkableVertex shortesPathDijkstra(MarkableVertex a_vertex1,
2      MarkableVertex a_vertex2) {
3          Map markableVerteces = getMarkableVertexMap();
4          markableVerteces.put(a_vertex1.getVert(), a_vertex1);
5          markableVerteces.put(a_vertex2.getVert(), a_vertex2);
6          Stack queue = new Stack();
7          MarkableVertex startVertex = a_vertex1;
8          startVertex.setMark(true);
9          startVertex.setDist(0.0);
10         startVertex.setPrev(a_vertex1.getPrev());
11         queue.push(startVertex);
```

```
12          while (!queue.isEmpty()) {
13              MarkableVertex current = (MarkableVertex) queue.pop();
14              current.setMark(true);
15              Set adjacents = null;
16              try {
17                  adjacents = getReachableAdjacentVerteces(current);
18                  Iterator adjIter = adjacents.iterator();
19                  while (adjIter.hasNext()) {
20                      MarkableVertex next = (MarkableVertex) markableVerteces
21                              .get(adjIter.next());
22
23                      double edgeDist = 0.0;
24                      Edge edge = null;
25                      try {
26                          edge = getEdge(current.getVert(), next.getVert());
27                          edgeDist = ((Double) edge.getDistance().getValue())
28                                  .doubleValue();
29                          if(edgeDist < 0.0) {
30                              edgeDist = 0.0;
31                              throw new EdgeNotFoundException("Edge has invalid
                                    distance of less 0.");
32                          }
33                      } catch (EdgeNotFoundException ex) {
34                          ex.printStackTrace();
35
36                      } catch (NullPointerException ex) {
37                          System.out.println("NullPointerException: " + edge);
38                      }
39                      double weight = current.getDist() + edgeDist;
40                      if (weight < next.getDist()) {
41                          next.setDist(weight);
42                          next.setPrev(current.getVert());
43                          queue.push(next);
44                      }
45                      if (!next.isMark()) {
46                          queue.push(next);
47                      }
48                  }
49              } catch (EdgeNotFoundException ignore) {
50                  ignore.printStackTrace();
51              }
52          }
53          return (MarkableVertex) markableVerteces.get(a_vertex2.getVert());
54      }
```

Listing 3.1.1: Die Implementation des Dijkstra-Algorithmus

`MarkableVertex` ist hierbei eine *Wrapper*-Klasse für Vertex, damit ein Vertex markiert werden, ob er schon besucht wurde, sowie die beiden Eigenschaften d und p.

```
1  package com.cocosoftware.util12.graph;
2
3  import java.util.*;
4
5  public class MarkableVertex {
6      private Vertex vert; // the vertex to be represented
7      private boolean mark = false; // if visited
8      private double dist = Double.MAX_VALUE; // distance of infinity
9      private Vertex prev = null; // predecessor
10     // a List of all predecessors
11     private List visitedList = Collections.synchronizedList(new Vector());
12
13     // Constructor
14     public MarkableVertex(Vertex v) {
15         vert = v;
16     }
17
18     public boolean equals(Object obj) {
19         if (obj instanceof MarkableVertex) {
20             MarkableVertex mvertex = (MarkableVertex) obj;
21             return mvertex.vert.equals(vert);
22         }
23         return super.equals(obj);
24     }
25
26     [...]
```

Listing 3.1.2: Die Wrapper-Klasse MarkableVertex

Zunächst wird V initialisiert. Die Menge aller Knoten wird im Code durch eine `java.util.Map` repräsentiert, in der jeder `Vertex` zu seinem Pendant, dem `MarkableVertex`, gemappt wird. Danach wird ein `java.util.Stack` zur Abarbeitung aller Knoten initialisert. Dann werden dem Starknoten die Distanz 0 und der Vorgänger *null* zugeordnet, das boolsche *Flag*, welches darauf hinweist, dass dieser Knoten schon besucht wurde, wird auf *true* gesetzt. Der Startknoten wird dann dem `Stack` hinzugefügt. Diesen Vorgang bezeichnet man allgemein als *push*. Etwas wegzunehmen bezeichnet man hingegen als *pop*.

Die Methode `getReachableAdjacentVerteces(MarkableVertex)` liefert alle direkt erreichbaren Knoten unter Berücksichtigung aller Abbiegeverbote zurück. Danach wird die Liste aller erreichbaren Knoten abgearbeitet und jeweils, wie schon beim ersten Knoten, werden immer dann, wenn ein Knoten eine kürzere Distanz hat, die neue Distanz und der neue Vorgänger eingetragen. Sind alle Knoten abgearbeitet, terminiert der Algorithmus. Die optionale Abfrage, den Algorithmus terminieren zu lassen, sobald der Zielknoten erreicht wurde, wurde nicht implementiert, da so nicht gewährleistet ist,

dass der kürzeste Weg gefunden wird, da wenn eine direkte Verbindung länger ist als eine indirekte, nur die direkte gefunden wird und diese als kürzer ausgegeben wird, auch wenn sie es nicht ist.

Stellt man die einzelnen Iterationsschritte im Graphen in Abb. 3.1.1 dar, ergibt sich Abb. 3.1.2.

Ein Knoten kann nur einmal besucht werden, da der Algorithmus sonst nicht terminieren würde. Somit findet der Dijkstra-Algorithmus den Weg von v_1 nicht zu v_6. v_2 müsste daher zwei Mal abgearbeitet werden. Der Dijkstra-Algorithmus in seiner ursprünglichen Form kann hier daher nicht angewandt werden.

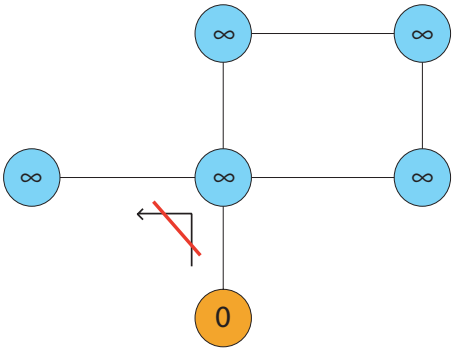
3.1.3.2. Der Bellman-Ford-Algorithmus

Da eine Mehrfachaufnahme von Knoten in einen Weg nicht möglich war um Abbiegeverbote in Graphen zu berücksichtigen, wurden Überlegungen angestellt einen kantenorientierten Algorithmus zu implementieren

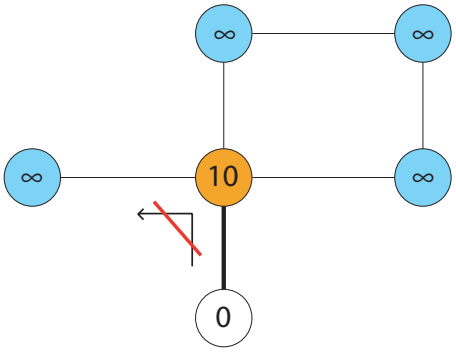
Der Bellman-Ford-Algorithmus ist nach seinen Erfindern Richard Bellman und Lester Ford, beides Mathematiker an der Universität von Wisconsin, benannt. Ein großer Unterschied zum Algorithmus von Dijkstra besteht in der Möglichkeit auch negative Kantengewichte im Graphen zuzulassen. Das Konzept findet sich in der Netzwerktechnik in allen *Distance-Vector*-Protokollen wieder. Ansonsten ist das Prinzip dieses Algorithmus jenem von Dijkstra sehr ähnlich.

$d(V) = \infty$
$p(V) = \text{null}$
$d(v_s) = 0$
$p(v_s) = v_s$
für $i = 1, 2, \ldots, m$:
    für alle e aus E:
        wenn $d(\alpha(e)) + \beta(e) < d(\omega(e))$
            $d(\omega(e)) = d(\alpha(e)) + \beta(e)$
            $p(\omega(e)) = v$
für alle e aus E:
    wenn $d(\alpha(e)) + \beta(e) < d(\omega(e))$
        Zyklus negativer Länge gefunden: STOP

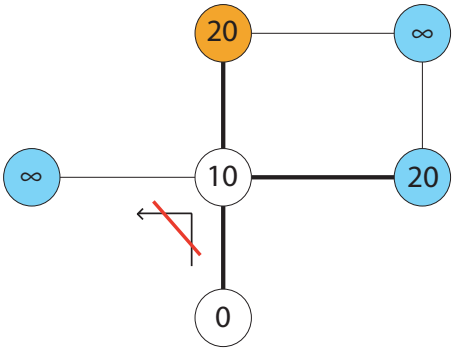
Im Gegensatz zu Dijkstra werden bei Bellman-Ford nicht Knoten, sondern Kanten abgearbeitet. Die Knoten müssen jedoch ebenso ihre Distanz zum Startknoten und ihren Vorgänger speichern. Hier liegt das Problem jedoch nicht in der fehlenden Möglichkeit einen Knoten mehrfach aufnehmen zu können, sondern in der simplen Tatsache, dass



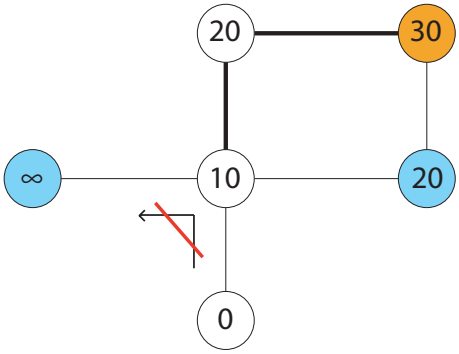
(a) Graph nach Initialisierung



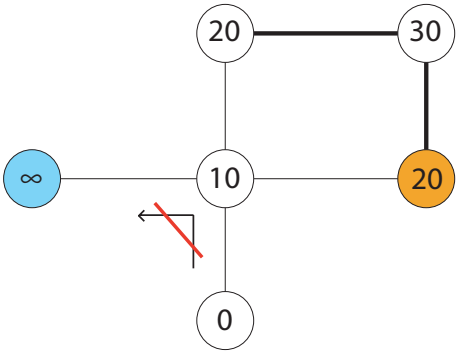
(b) Iteration 2



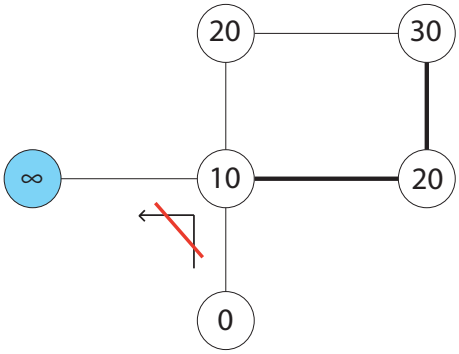
(c) Iteration 3



(d) Iteration 4



(e) Iteration 5



(f) Iteration 6

Abbildung 3.1.2.: Die einzelnen Iterationsschritte des Dijkstra-Algorithmus. In den Knoten stehen die Distanzen zum Startknoten. Der aktuelle Knoten ist orange markiert, die noch zu besuchenden sind blau und die bereits abgearbeiteten Knoten weiß gefüllt.

ein Knoten eine größere Distanz nicht speichert und einen Weg, der ein zweites Mal am selben Knoten vorbeiführt, verwirft. Der richtige Weg von v_1 nach v_6 führt über v_5, wo man umkehren müsste. So wäre $d(v_2)$ bei der 1. Iteration 10 und bei der 3. Iteration 30. Dieser Weg ist länger und daher nicht der optimale Weg zu v_2 von v_5. [Fellner (2004), S. 58ff, Wikipedia, Algorithmus_von_Bellman_und_Ford, Screen 1f]

3.1.3.3. Der BEHA-Algorithmus

Schlussendlich musste ein Algorithmus entwickelt werden, der sowohl früher terminiert, als auch die Mehrfachaufnahme von Knoten erlaubt. So wurde der BEHA-Algorithmus entwickelt, welcher im Grunde ein speziell an das Problem angepasster *Branch-and-Bound*-Algorithmus ist.

Die *Branch-and-Bound*-Algorithmen gehören zu der Gruppe der Entscheidungsbaum-Verfahren. Sie bestehen aus zwei Teilen, dem *Branching* (engl.: Verzweigen) und *Bounding* (engl.: Beschränken). Mit diesen beiden Methoden wird versucht den Lösungsraum möglichst klein zu halten, indem alle nicht optimalen bzw. nicht erlaubten Zweige beschränkt oder abgeschnitten werden. Der nächste zu bearbeitende Knoten kann entweder mittels Tiefensuche, Bestensuche oder wie im Falle des BEHA-Algorithmus mittels Breitensuche ermittelt werden. [Wikipedia, Branch_and_Bound, Screen 1f]

Ausgehend von einer Wurzel (hier: v_s) werden alle direkt verbundenen Knoten mit der Wurzel verzweigt. In allen weiteren Iterationen werden die Knoten in der Reihenfolge bearbeitet, in der sie in den Baum eingefügt wurden. Bevor also tiefer liegende Knoten betrachtet werden können, werden die Knoten im Baum pro Ebene bearbeitet. Im ersten Schritt werden nun alle nicht erlaubten Knoten wieder gestrichen. Die nächste Iteration verzweigt nun jeden Knoten mit allen direkt verbundenen Knoten. Danach werden unter Berücksichtigung aller Abbiege-, Umkehrverbote, der Überprüfung nach Wegen, die im Kreis führen, und des kürzesten bisher gefunden Weges wieder die nicht erlaubten Knoten gestrichen, bevor neue Knoten in der nächsten Iteration verzweigt werden. Ist ein Weg gefunden zu v_d, so wird die Distanz des Weges gespeichert und sobald alle Zweige beschränkt wurden oder die Wege, die sie darstellen länger sind als der bisher gefunden kürzeste Weg, terminiert der Algorithmus.

Die Breitensuche wurde als beste Methode gewählt, da sie als einzige Methode garantiert terminiert. Tiefen- und Bestensuche haben zwar die Chance wesentlich schneller zu terminieren, doch fehlt hier die Sicherheit. Dies wird vor allem in großen Graphen, in denen ein kurzer Weg mit nur wenigen Knoten gesucht wird, deutlich. Da diese Bedingung bei der Wegesuche zwischen zwei Haltestellen gegeben ist, war die Breitensuche die Methode der ersten Wahl.

Um Zyklen zu vermeiden, wurde die Überlegung angestellt, dass ein Knoten nur dann mehrmals in einen Weg aufgenommen werden muss, wenn ein Abbiege- bzw. ein Umkehrverbot besteht, das diesen Knoten umgibt. Im Falle von 3.1.1 dürfte der Knoten v_2 zwei Mal in einem Weg aufgenommen werden, da das Abbiegeverbot $t(e_1, e'_6)$ besteht, was gleichzusetzen ist mit $t(e_{12}, e_{26})$. Bestünde auch noch ein Abbiegeverbot $t(e'_2, e'_6)$, so dürfte v_2 auch zwei Mal in einen Weg aufgenommen werden. Der Graph wird dabei wie in Abb. 3.1.3 auf S. 77 dargestellt abgearbeitet. Für das Beispiel wird angenommen, dass alle Kanten bidirektional sind und mit einem Umkehrverbot belegt sind.

Ein Weg wird dabei durch einen durchgängigen Ast im Baum dargestellt. Ein Ast kann durch folgende Umstände terminieren:

- Wegeverbot
  - Abbiegeverbot
  - Umkehrverbot
- Wenn der Zielknoten schon erreicht ist, eine längere Gesamtdistanz eines Weges.

Der Algorithmus kann folgendermaßen beschrieben werden, wobei zu beachten ist, dass $V_{adj}(v)$ sind alle $\omega(e)$ aus E wo $\alpha(e) = v$ gilt. W ist der Wegebaum. Ein Zweig von W wird mit b für *branch* (engl.: Zweig) bezeichnet. $W.next()$ liefert den nächsten zu bearbeitenden Knoten. Dabei wird immer Ebene für Ebene abgearbeitet. $b.last()$ gibt den letzten zu diesem Zweig hinzugefügten Knoten v zurück, $b.dist()$ die Länge des Weges, den der Zweig darstellt.

$W.branch(v_s)$
$w_{opt} = \text{null}$
$d(w_{opt}) = \infty$
$v_{act} = v_s$
für alle v aus $V_{adj}(v_{act})$
    $W.branch(v)$
    für alle b aus W
        wenn $b \in T$ oder $b.dist() > w_{opt}$
            $W.bound(b)$
        wenn $b.last() = v_d$ und $b.dist() < w_{opt}$
            $w_{opt} = b$
            $d(w_{opt}) = b.dist()$
    $v_{act} = W.next()$

In der Implementation in `AdjacencyList` sieht der Algorithmus dann folgendermaßen aus:

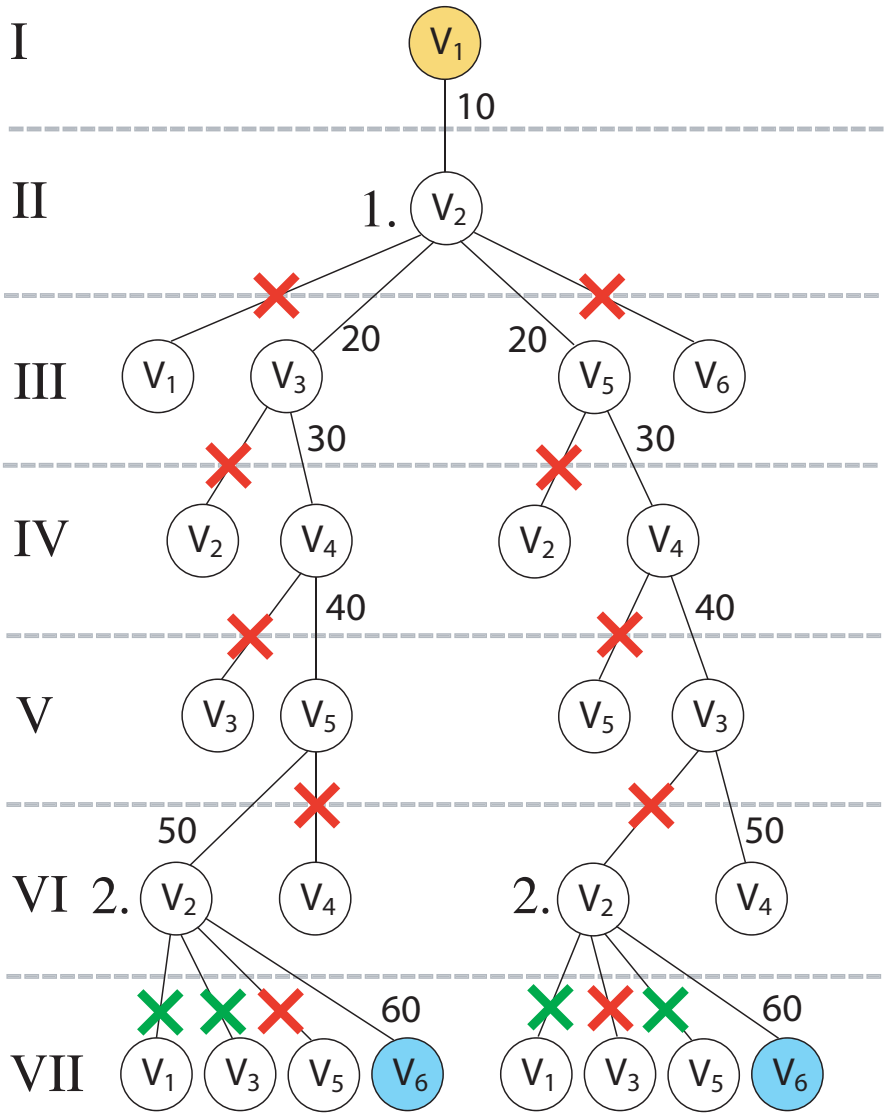


Abbildung 3.1.3.: Die Abarbeitung des BEHA-Algorithmus amBeispiel von 3.1.1

```
1  public TreeBranch shortestPathBEHA(List a_verteces) {
2          // Initializing
3          double shortestWay = Double.MAX_VALUE;
4          double totalWay = 0.0;
5          List visitedVerteces = Collections.synchronizedList(new Vector());
6          TreeBranch actualBestBranch = null;
7
8          List branches = Collections.synchronizedList(new Vector());
9
10         Vertex av = null,   // actual Vertex
11                dv = null,   // destination Vertex
12                lvv = null,  // last visited Vertex
13                nv = null,   // next Vertex
14                plvv = null; // the Vertex visited before the last Vertex
15
16         Set adjacents = Collections.synchronizedSet(new HashSet());
17
18         TreeBranch actualBranch;
19
20         Iterator param0Iter = a_verteces.listIterator();
21         if(param0Iter.hasNext()) {
22             av = (Vertex) param0Iter.next();
23         }
24         while(param0Iter.hasNext()) {
25             dv = (Vertex) param0Iter.next();
26             // erzeuge einen Branch
27             // füge diesen einer Liste hinzu
28             if(av.equals(dv)) {
29                 continue;
30             }
31             actualBranch = new TreeBranch(lvv, av, dv, this);
32             branches.add(actualBranch);
33             // iteriere jeden dieser Branches
34             while(branches.size() > 0) {
35                 Iterator branchesIter = branches.iterator();
36                 List newBranches = new ArrayList();
37                 while(branchesIter.hasNext()) {
38                     actualBranch = (TreeBranch) branchesIter.next();
39                     if(actualBranch.isTerminated()) {
40                         branchesIter.remove();
41                     } else {
42                         lvv = actualBranch.getLastVertex();
43                         plvv = actualBranch.getPreLastVertex();
44                         adjacents = getReachableAdjacentVerteces(lvv, plvv);
45                         Iterator adjaIter = adjacents.iterator();
46                         while(adjaIter.hasNext()) {
47                             nv = (Vertex) adjaIter.next();
48                             TreeBranch tb = new TreeBranch(actualBranch, nv,
                                   shortestWay);
```

```
49                          if(tb.isTerminated() && tb.isSuccessful() && tb.
                                getWay() < shortestWay) {
50                              actualBestBranch = tb;
51                              shortestWay = actualBestBranch.getWay();
52                          } else if(!tb.isTerminated()) {
53                              newBranches.add(tb);
54                          }
55                      }
56                  }
57              }
58              branches = newBranches;
59          }
60          if(actualBestBranch != null) {
61              if(visitedVerteces.isEmpty()) {
62                  visitedVerteces.add(actualBestBranch.getStartVertex());
63              }
64              visitedVerteces.addAll(actualBestBranch.getVisitedVerteces());
65              av = dv;
66              lvv = actualBestBranch.getPreLastVertex();
67              totalWay += actualBestBranch.getWay();
68              shortestWay = Double.MAX_VALUE;
69              actualBestBranch = null;
70          }
71      }
72
73      return new TreeBranch(visitedVerteces, totalWay);
74  }
```

Listing 3.1.3: Der BEHA-Algorithmus

Ein Ast wird durch eine Instanz der Klasse `com.cocosoftware.util12.graph.TreeBranch` repräsentiert.

```
1  package com.cocosoftware.util12.graph;
2
3  import java.util.*;
4
5  public class TreeBranch {
6
7      private AdjacencyList graph;
8
9      // The List of visited Verteces
10     private List visitedVerteces;
11
12     // The Vertex before the StartVertex
13     private Vertex bsv;
14
15     // The StartVertex
16     private Vertex sv;
```

```
17
18      // The DestinationVertex
19      private Vertex dv;
20
21      // The way of the List
22      private double way;
23
24      private boolean terminated = false;
25
26      private boolean successful = false;
27
28      public TreeBranch(Vertex a_bsv, Vertex a_sv, Vertex a_dv, AdjacencyList
            a_graph) {
29          bsv = a_bsv;
30          sv = a_sv;
31          dv = a_dv;
32          visitedVerteces = new ArrayList();
33          way = 0.0;
34          graph = a_graph;
35      }
36
37      public TreeBranch(List a_visitedVerteces, Vertex a_bsv, Vertex a_sv,
            Vertex a_dv, double a_way, AdjacencyList a_graph) {
38          bsv = a_bsv;
39          sv = a_sv;
40          dv = a_dv;
41          visitedVerteces = new ArrayList();
42          visitedVerteces.addAll(a_visitedVerteces);
43          way = a_way;
44          graph = a_graph;
45      }
46
47      public TreeBranch(TreeBranch a_branch, Vertex a_nextVertex, double
            actualShortestPath) {
48          this(a_branch.visitedVerteces, a_branch.bsv, a_branch.sv, a_branch.dv,
                 a_branch.way, a_branch.graph);
49          add(a_nextVertex, actualShortestPath);
50      }
51
52      public TreeBranch(List a_visitedVertexList, double a_shortestPath) {
53          visitedVerteces = a_visitedVertexList;
54          way = a_shortestPath;
55      }
56
57      public void add(Vertex a_vertex, double actualShortestPath) {
58          if(a_vertex.equals(dv)) {
59              // Ist dieser Vertex der Destination_vertex
60              // Terminiere diesen Branch
61              actualizeWay(a_vertex);
62              visitedVerteces.add(a_vertex);
```

```
63              terminated = true;
64              successful = true;
65              return;
66          }
67
68          int vertexCount = countVertex(a_vertex);
69          int vertexCountAllowed = graph.bansOnVertex(a_vertex);
70          if(vertexCount < vertexCountAllowed) {
71              // Darf noch aufgenommen werden
72              actualizeWay(a_vertex);
73              visitedVerteces.add(a_vertex);
74              if(way > actualShortestPath) {
75                  // Terminiere
76                  terminated = true;
77              }
78          } else {
79              // Darf nicht mehr aufgenommen werden
80              terminated = true;
81          }
82      }
83
84      private int countVertex(Vertex a_vertex) {
85          int retVal = 0;
86          Iterator iter = visitedVerteces.listIterator();
87          while(iter.hasNext()) {
88              Vertex v = (Vertex) iter.next();
89              if(v.equals(a_vertex)) {
90                  retVal++;
91              }
92          }
93          if(sv.equals(a_vertex)) {
94              retVal++;
95          }
96          /*if(bsv != null && bsv.equals(a_vertex)) {
97              retVal++;
98          }*/
99          return retVal;
100     }
101
102     public Vertex getLastVertex() {
103         int size = visitedVerteces.size();
104         if(size > 0) {
105             return (Vertex) visitedVerteces.get(size-1);
106         } else {
107             return sv;
108         }
109     }
110
111     public Vertex getPreLastVertex() {
112         int size = visitedVerteces.size();
```

```
113         if(size > 1) {
114             return (Vertex) visitedVerteces.get(size-2);
115         } else if (size == 1){
116             return sv;
117         } else if (size == 0) {
118             return bsv;
119         } else {
120             return null;
121         }
122     }
123 
124     private void actualizeWay(Vertex a_vertex) {
125         try {
126             Edge edge = graph.getEdge(getLastVertex(), a_vertex);
127             double edgeDist = ((Double) edge.getDistance().getValue()).
                    doubleValue();
128             way += edgeDist;
129         } catch (EdgeNotFoundException ex) {
130             ex.printStackTrace();
131         }
132     }
133 
134     public boolean isTerminated() {
135         return terminated;
136     }
137 
138     public double getWay() {
139         return way;
140     }
141 
142     public List getVisitedVerteces() {
143         return visitedVerteces;
144     }
145 
146     public boolean isSuccessful() {
147         return successful;
148     }
149 
150     public Vertex getStartVertex() {
151         return sv;
152     }
153 }
```

Listing 3.1.4: Die Klasse TreeBranch

Sobald ein Ast den Knoten v_d erreicht hat, wird er als `actualBestBranch` gespeichert. Der zuletzt gespeicherte `actualBestBranch` ist dann der optimalste Weg von v_s nach v_d. Für den Baum in Abb. 3.1.3 wäre dies der Weg $w_1(v_1, v_2, v_3, v_4, v_5, v_2, v_6)$.

Der zweite Weg $w_2(v_1, v_2, v_5, v_4, v_3, v_2, v_6)$ würde nicht gewählt werden, da er nicht als `actualBestBranch` gespeichert wird, da $d(w_1) \leq d(w_2)$ gilt.

Neben der Möglichkeit Abbiege- und Umkehrverbote zu berücksichtigen, liegt die größte Stärke des Algorithmus darin, dass er schnell terminiert. Angenommen ein großes Wegenetz wäre noch über v_5 erreichbar, so würde der Algorithmus trotzdem nach der siebten Iteration terminieren, da alle Wege, die folgen würden, länger als der Weg von v_1 nach v_6 wären. Die Algorithmen von Bellman-Ford und von Dijkstra lösen immer nur ein vollständiges SSSP-Problem. Der BEHA-Algorithmus terminiert jedoch schon früher und löst daher das SSSP-Problem nur teilweise und ist vor allem in größeren Graphen wesentlich schneller als die beiden anderen vorgestellten *Shortest-Path-Algorithmen*.

3.1.4. Statistikerstellung für BEHA

Der BEHA-Algorithmus wurde in einer Implementation zur Erstellung von Statistiken verwendet. Dabei wurden die mittels eines proprietären Tools erstellten Graphen ausgewertet. Dieses Tool liest ein Microsoft Excel Dokument ein, wandelt es in eine XML-Datei um, validiert dieses und schreibt die Daten anschließend in eine Datenbank. In dem Excel Dokument sind alle für die Erstellung eins Graphen notwendigen Daten geschrieben:

- Regionen-ID
- Buslinie
- Gültigkeitsdauer
- Koordinatensystem
- Kanten-ID
- Anfangsknoten
- Endknoten
- Wegeverbote
- Bidirektional
- Länge in
  - Meter
  - Sekunden
- Koordinaten von Anfangs- und Endknoten
- Anmerkungen

Anfangs- und Endknoten müssen dabei nicht zwangsweise eine Haltestelle sein. Auch wichtige Punkte, wie etwa Kreuzungen und Umkehrmöglichkeiten werden in dem Gra-

phen festgehalten. Der Parameter Koordinatensystem wurde eingeführt, um Internationalität zu garantieren. Für die Graphen in Ostösterreich wurde das Koordinatensystem MGI Lambert Austria (neu) (**M**ilitary **G**eographic **I**nstitute) gewählt. Die Koordinaten haben jedoch keine Bedeutung für die Erstellung der Statistiken.

Schnittstellen zu einer GIS-gestützten Datenbank wurden bisher noch nicht realisiert, sind aber für das Projekt geplant. Aufgrund der hohen Lizenzkosten und des Aufwandes für die Integration einer solchen Datenbank wurde die manuelle Eingabe der Koordinaten vorgezogen. Die Open-Source-Bewegung arbeitet jedoch schon an einigen Projekten zum Sammeln von Daten und *Frameworks* zum Auslesen dieser. Die meisten relevanten Projekte sind auf `http://freegis.org/`, `12. September, 2005` aufgelistet.

Zur Statistikerstellung werden die sich ebenfalls in der selben Datenbank befindlichen Fahr- und Dienstpläne ausgelesen. In diesen wird auch vermerkt, ob eine Haltestelle angefahren wurde oder nicht. Da dies aber manuell vom Fahrpersonal via BusApplikation geschieht sind die Daten nicht ganz zuverlässig, aber die Busfahrer/innen werden angehalten hier sehr genau vorzugehen. Anhand dieser Daten kann dann die Reihenfolge der Haltestellen einer Fahrt ermittelt werden. Die Länge zwischen zwei Stationen wird nur in Meter gemessen. Die Längenangabe in Sekunden wird erst mit der vollständigen Integration von GPS kommen, da diese zum einen sehr stark variieren kann und zum anderen eine genaue Angabe erst für die Verspätungsvorhersage gebraucht wird.

Da die Busfahrer/innen sehr ortskundig sind, kann davon ausgegangen werden, dass sie zwischen zwei bestellten Haltestellen immer den kürzesten Weg wählen. Eine zweite Annahme, die bis zur vollständigen Integration von GPS getroffen werden muss, ist, dass der zeitlich kürzeste Weg auch der räumlich kürzeste Weg ist. Unter diesen Annahmen, muss für die Erstellung einfach nur mehr die Länge aller Wege zwischen den angefahrenen Haltestellen, allen Haltestellen und allen nicht bedarfsgesteuerten Haltestellen zusammengerechnet werden.

So ergeben sich folgende Werte für eine Fahrt, auf der die weit entfernten Bedarfshaltestellen nicht angefahren wurden. Die Auswertung kann aber auch für mehrere Fahrten gemacht werden, auf denen zum Teil unterschiedliche Haltestellen angefahren werden.

Durchschnittliche Anzahl an Stationen	19
Größte Anzahl an Stationen	19
Kleinste Anzahl an Stationen	19
Durchschnittliche Anzahl an BEHAs	9
Größte Anzahl an BEHAs	9
Kleinste Anzahl an BEHAs	9
Durchschnittliche Anzahl der bedienten BEHAs	0
Größte Anzahl an bedienten BEHAs	0
Kleinste Anzahl an bedienten BEHAs	0
Einsparung (Prozent der Stationen)	47
Distanz, wenn alle bedient (Meter)	23942
Distanz, ohne BEHAs angefahren (Meter))	3143
Real gefahrene Distanz (Meter)	3143
Einsparung (Prozent der Distanz)	86

Je nach Bedarf kann das Statistiktool noch weitere Parameter auswerten. Ein für die Firmenstrategie und das Marketing wichtiger Faktor ist die Verspätung oder die Pünktlichkeit, die sich durch die bedarfsgesteuerte Bedienung ergibt.

Auch die Ausgabe der Auswertungen kann je nach Wunsch des Benutzers und Art der Weiterverwendung als Microsoft Excel-Dokument, XML-, HTML- (**H**yper**t**ext **M**arkup **L**anguage) oder einfache Textdatei geschehen. Das Tool selbst wurde als Webanwendung für die Mitarbeiter des öffentlichen Verkehrsbetreibers konzipiert und kommt ohne Konfiguration und nur wenigen Eingabeparametern aus. Es wurde mithilfe des Struts-Frameworks, welches einen standardisierten Prozess zur Verarbeitung von Requests vorgibt, verwirklicht.

3.1.4.1. Jakarta Struts

Das Jakarta-Projekt ist ein Open-Source-Projekt, welches zahlreiche Softwarelösungen in Java anbietet. Das Projekt ist Teil der ASF (The **A**pache **S**oftware **F**oundation) und ist zu finden unter `http://jakarta.apache.org/`, `12. September, 2005`.

Das Struts-Projekt ist eines der bekanntesten Jakarta-Projekte. Es bietet Basisfunktionalitäten und Infrastruktur für Webanwendungen, die dadurch vom Programmierer nicht mehr implementiert werden müssen. Zahlreiche Entwicklungstools (z. B.: Struts Console (`http://www.jamesholmes.com/struts/console/`, `12. September, 2005`), Easy Struts (`http://easystruts.sourceforge.net/`, `12. September, 2005`), Struts Builder (`http://sourceforge.net/project/showfiles.php?group_id=55282`, `12. September, 2005`), Karapan Sapi (`http://sourceforge.net/projects/`

strutsgenerator/, 12. September, 2005), ...) bieten inzwischen einen professionellen Rahmen zur Entwicklung von Webanwendungen mit Struts.

Als Server wird bei BEHA JBoss (http://www.jboss.com/index, 12. September, 2005) verwendet. Er arbeitet als Container sehr gut mit den Jakarta Struts zusammen, da diese nach den J2EE-Patterns entworfen wurden. Eine Auflistung aller J2EE-Patterns und eine Einführung in die J2EE-Architektur ist auf http://java.sun.com/blueprints/corej2eepatterns/, 12. September, 2005 zu finden. Eine Einführung für versierte Java-Programmierer bietet auch [Eickstädt und Reuhl (2004)].

3.2. Verarbeitung der GPS-Messpunkte

Das Statistiktool ist ein Service der das API des RoutingModuls zur Berechnung von Weglängen verwendet. Ein weiterer vor allem für die Fahrgäste wichtiger Service ist die Verspätungsvorhersage und die Verfolgung der Busse in der RBL. Gerade für die Verspätungsvorhersage müssen die GPS-Messpunkte den Kanten im Graphen zugeordnet werden, um daraus Ganglinien abzuleiten.

3.2.1. Verspätungsvorhersage

Als Basis für die Verspätungsvorhersage dienen Standardganglinien S von Fahrtzeiten in den einzelnen Streckenabschnitten i. Jeder Streckenabschnitt wird dabei von einer oder mehreren Kanten im Graph repräsentiert. Standardganglinien werden fortgeschrieben aus den Werten von den Tagesganglinien T. In diesen werden die Fahrtzeitwerte t gesammelt, die im Laufe eines Tages anfallen. Die Werte aus den Standardganglinien dienen dann als Prognosewerte. Dazu werden die Standardganglinien mit einem Glättungsfaktor α_f modifiziert, der das Verhältnis zwischen den Werten der Tageskennlinien und Standardganglinien über einen bestimmten Zeitraum beschreibt. Der Faktor soll auch verhindern, dass sich seltene Ereignisse, wie etwa Störungen im Verkehrsgeschehen durch einen Unfall, nur wenig auf die Standardganglinien auswirken. [Janko (1994), S. 6f]

Das Ganglinienintervall wurde in der ersten Konzeptphase mit einer Stunde gewählt. Die Erfahrungen werden zeigen, ob dieses Intervall ausreichend ist. Der Faktor für die exponentielle Glättung α_f wurde nach Janko (1994) mit 0,8 gewählt.

Die Prognoseganglinie P wird dynamisch berechnet. Die Fortschreibung der Standardganglinien funktioniert dabei folgendermaßen:

$$t_{S,i,d,n} = \alpha_f \cdot t_{T,i,n} + (1 - \alpha_f) \cdot t_{S,i,d-1,n}$$

$t_{S,i,d,n}$ ist der Wert der Standardganglinie S des aktuellen Tages d für die Teilstrecke i im Zeitintervall n. Der Vortag wird mit $d-1$ bezeichnet. Zur Berechnung der Prognose wird ein Prognosefaktor p berechnet, der sich aus dem Verhältnis der Tagesganglinienen und den Standardganglinien für eine gesamte Region zusammensetzt.

$$p_{i,d,n} = \frac{t_{T,i,d,n}}{t_{S,i,d,n}}$$

Der Prognosefaktor $p_{i,d,n}$ wird ebenfalls exponentiell mit dem Faktor α_p geglättet, nachdem zuvor über alle $p_{i,d,n}$ der Mittelwert gebildet wurde. Der Faktor wurde nach [Janko (1994)] mit 0,2 gewählt.

$$\begin{aligned} p_d &= \frac{\sum_{i=0,n=0}^{k,m} p_{i,d,n}}{k * m} \\ p &= \alpha_p \cdot p_d + (1 - \alpha_p) \cdot p_{d-1} \\ t_{P,i,d,n} &= p \cdot t_{S,i,d,n} \end{aligned}$$

Aufgrund der exponentiellen Glättung in Fortschreibung und Prognose wird nur schwach auf singuläre Störungen eingegangen und bei jahreszeitlichen Tendenzen oder Schulferien reguliert sich die Prognose von selbst.

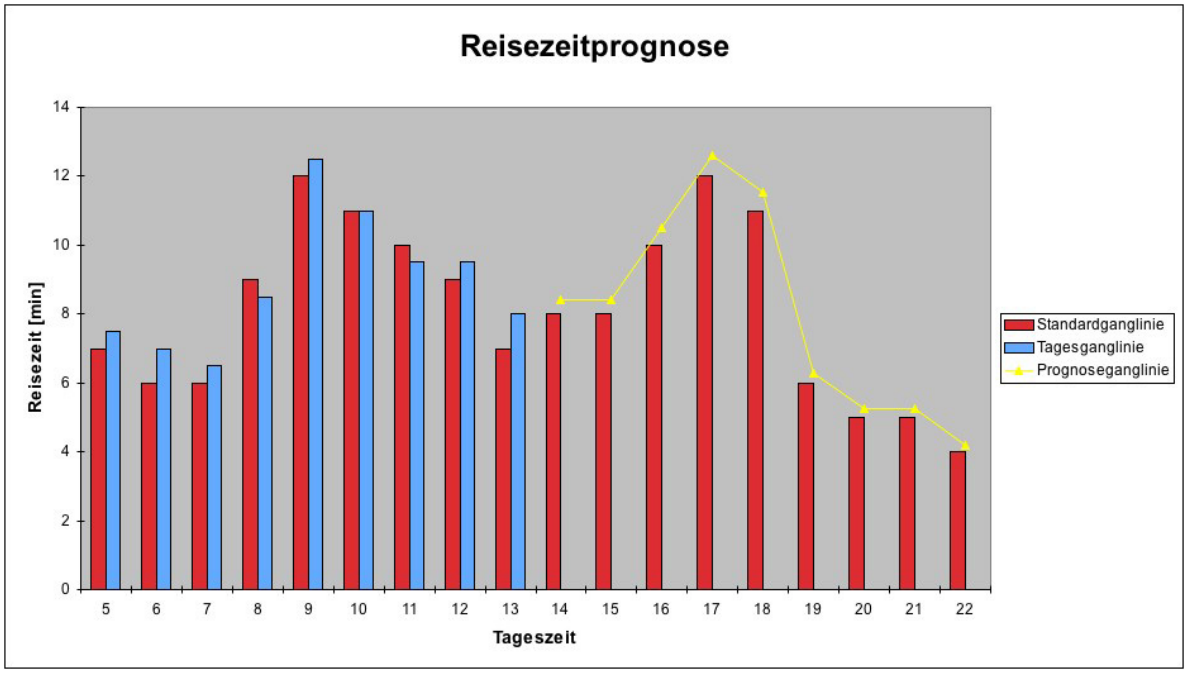


Abbildung 3.2.1.: Ganglinienfortschreibung und Prognose

Um starke Beeinflussungen durch kurzfristige Änderungen zu vermeiden, werden mehrere Standardganglinien pro Teilstrecke gespeichert. Diese werden in folgende Gruppen eingeteilt:

- Montag bis Freitag
- Samstag

- Sonn- und Feiertag
- Montag bis Freitag (Ferien)

So müssen nur wenige Daten für jede Tagesgruppe für die Fahrtzeitprognose gespeichert werden. Diese wären:

- Kantenbezogene Daten:
  - Standardganglinien
  - aktuelle Tageskennlinie
  - Tageskennlinien des letzten Tages
- Regionenweite Daten:
  - Prognosefaktor

Mit der Routendisposition, also der Planung der Reihenfolge, in der die Haltestellen angefahren werden, aufgrund der Vorbestellungen, können so die Anfahrtszeiten und somit die Verspätungen berechnet werden.

Diese Vorgehensweise der Fahrtzeitprognose ist wesentlich genauer als Systeme vieler Routenplaner, in der statische Fahrtzeiten verwendet werden, oder von Verspätungsvorhersagen von öffentlichen Verkehrsmitteln in urbanen Gebieten, wo vor allem Wegrandbaken zum Einsatz kommen.

3.2.2. Matching

Um die Ganglinien erstellen zu können, ist es wichtig, dass die einkommenden GPS-Daten den richtigen Kanten im Graphen zugeordnet werden. Dieser Vorgang wird als *Matching* bezeichnet.

Da während der Erstellung dieser Arbeit sich das *Matching* erst in einer frühen Entwicklungsphase befand, wird hier nur auf das Konzept des *Matchings* eingegangen.

Mittels Zeitstempel jedes GPS-Messpunktes und Fahrplan kann eine Vorauswahl getroffen werden, welcher Kante eine Gruppe von GPS-Messpunkten zugeordnet werden soll. Die BusApplikation sendet die GPS-Daten immer paketweise, sobald der Bus stehenbleibt oder dessen Geschwindigkeit eine bestimmte Toleranzgrenze unterschreitet. Durch Fortschreibung kann das *Matching* sehr konsistent betrieben werden. Wie auf Umfahrungen reagiert wird, ist im Moment (Stand: Juli, 2005) noch nicht geklärt. Da aber für die Verspätungsvorhersage grundsätzlich nur die Ankunfts- und Abfahrtszeiten des Busses benötigt werden, muss nicht unbedingt auf Streckenänderungen, z. B. durch Baustellen, reagiert werden, sofern die Reihenfolge, in der die Haltestellen angefahren werden, eingehalten wird.

Da immer nur der erste und letzte GPS-Punkt eines Datenpakets eine Haltestelle oder eine Kreuzung sein kann, können diese Punkte dem nächstgelegnenen Knoten zugeordnet werden, der in einem bestimmten Entfernungsintervall liegt. Aufgrund der Größe der Haltestellenbereiche und potentieller Messungenauigkeiten kann dieser Intervall großzügig mit 0 bis 100 Meter gewählt werden.

3.2.3. Koordinatensysteme

Für die Berechnung der Entfernung ist es zuerst notwendig die GPS-Koordinaten vom geodätischen Koordinatensystem WGS84 in ein kartesisches Koordinatensystem zu transformieren.

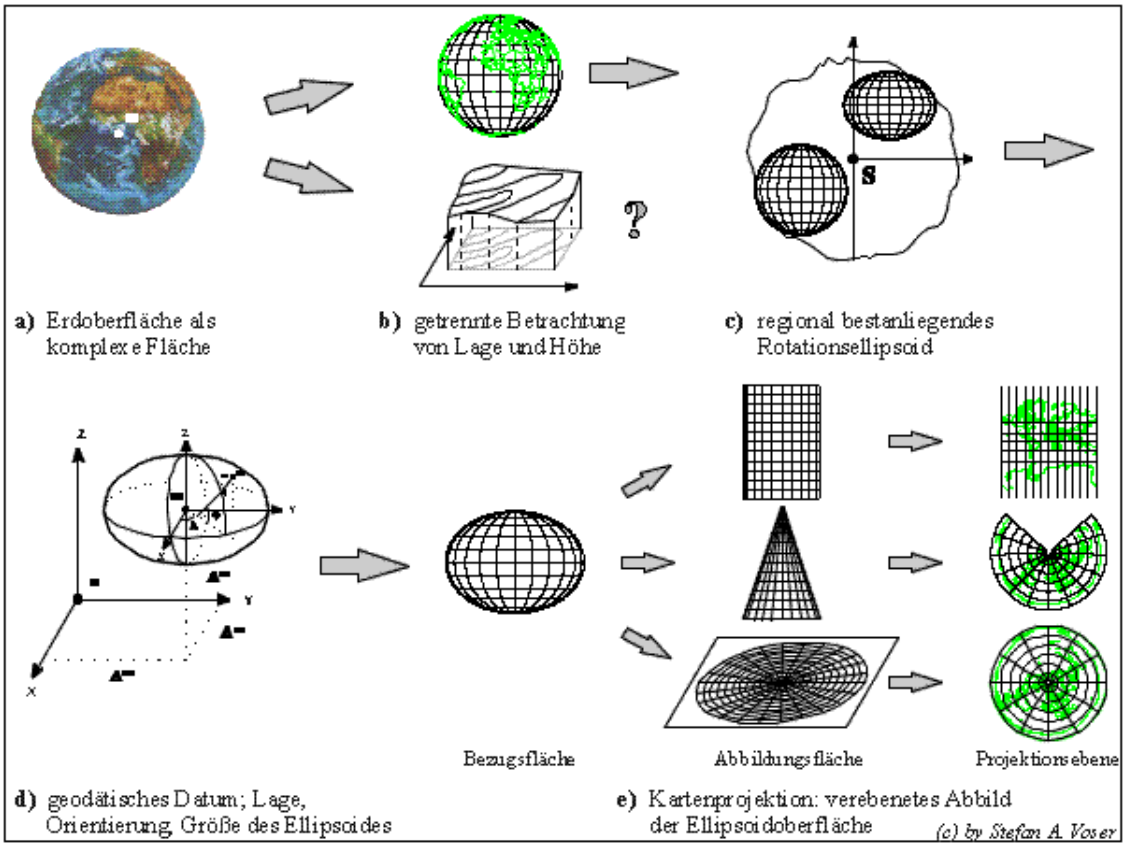


Abbildung 3.2.2.: Koordinatentransformation von der Erdoberfläche zu Lagekoordinaten [Voser (Mai, 1998), Screen 3f]

In der BusApplikation basiert die Entfernungsberechnung mehr auf einer Schätzung als auf einer Berechnung, da keine Floatzahlen unterstützt werden und somit trigonometrische Gleichungen oder Wurzel ziehen nicht möglich sind. Dabei wird angenommen, dass ein Breitengrad 111.136 km und ein Längengrad 111.324 km entsprechen. Diese Werte gelten zwar nur am Äquator exakt, stimmen aber noch einigermaßen, solange man

sich nicht in Polnähe befindet. Der Abstand zwischen zwei Punkten wird dann mittels des Manhattan-Algorithmus berechnet, der annimmt, das die Hypotenuse ungefähr die Länge der Summe von Ankathete und Gegenkathete entspricht. Die Manhattan-Distanz ist aber nur für sehr kleine Werte gültig und dient in der BusApplikation hauptsächlich zur Vermeidung von Konglomeraten von GPS-Messpunkten auf sehr kleinen Flächen, die durch Messungenauigkeiten entstehen.

$$c = \sqrt{a^2 + b^2} \approx a + b$$

Um genauere Werte zu erzielen, müssen die Werte in ein anderes Koordinatensystem transformiert werden. In Österreich gelten folgende amtliche Koordinatensysteme:

- Koordinatensysteme
  - Österreichisches Bundesmeldenetz Zone M28
  - Österreichisches Bundesmeldenetz Zone M31
  - Österreichisches Bundesmeldenetz Zone M34
  - Österreichische Gauß-Krüger Zone M28
  - Österreichische Gauß-Krüger Zone M31
  - Österreichische Gauß-Krüger Zone M34
  - Österreichische Lambert Koordinaten (alt)
  - Österreichische Lambert Koordinaten (neu)
  - Geographische Koord. (Greenwich) [Grad, Min, Sek]
  - Geographische Koordinaten (Greenwich) [Grad]
  - UTM Koordinaten (nördliche Hemisphäre)
- Bezugssysteme
  - ETRS89 (Europa), geozentrisch, GRS80
  - MGI (AT/CZ), Hermannskogel, Bessel
  - WGS84 (Weltweit GPS), geozentrisch, WGS84
  - WGS72 (Weltweit), geozentrisch, WGS72
  - ED50 (Europa), Potsdam, Hayford/Int.

Der Verkehrsbetreiber verwendet ein GIS-System zur geographischen Erfassung der Haltestellen, weshalb auch in BEHA das System MGI Austria Lambert (neu) verwendet wurde. Die Lambert Koordinaten gehören zur Gruppe der Kegelprojektionen, wo die Projektion der abzubildenden Erdoberfläche auf einen sie umhüllenden Kegelmantel erfolgt. Die Umrechnung ist mathematisch sehr aufwändig und wird an dieser Stelle nicht weiter erklärt. Eine gute Einführung zu diesem Thema findet sich in [Bilajbegovic (Chmielorz, 2001)].

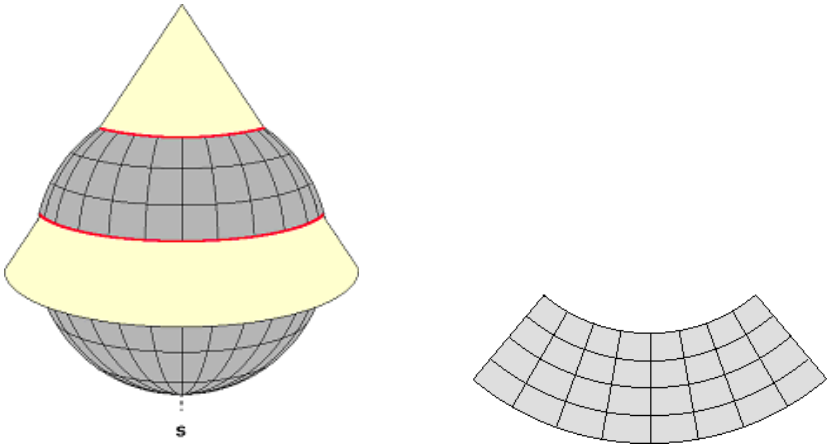


Abbildung 3.2.3.: Projektion der abzubildenden Erdoberfläche auf einen Kegelmantel [Panitzki (2004), Kegelprojektion, Screen 1]

Die Umrechnung bei BEHA wird mit dem Framework des Open-Source-Projekts deegree, zu finden unter `http://deegree.sourceforge.net/`, `12. September, 2005`, durchgeführt. Dieses Paket bietet auch Methoden zur Distanzberechnung.

Die Lambertsche Kegelprojektion bietet eine weitgehend flächentreue Abbildung der Oberfläche einer Kugel. Als Bezugssystem wird der Bessel-Ellipsoid verwendet. Sein Fundamentalpunkt liegt am Hermannskogel.

Abbildung 3.2.4.: Der Hermannskogel bei Wien als Fundamentalpunkt der österreichischen Landesvermessung [Wikipedia, Hermannskogel, Screen 1]

- $a = 6.377.397,155\,\mathrm{m}$
- $f = 1/299,152815$
- $b = 6.356.078,963\,\mathrm{m}$

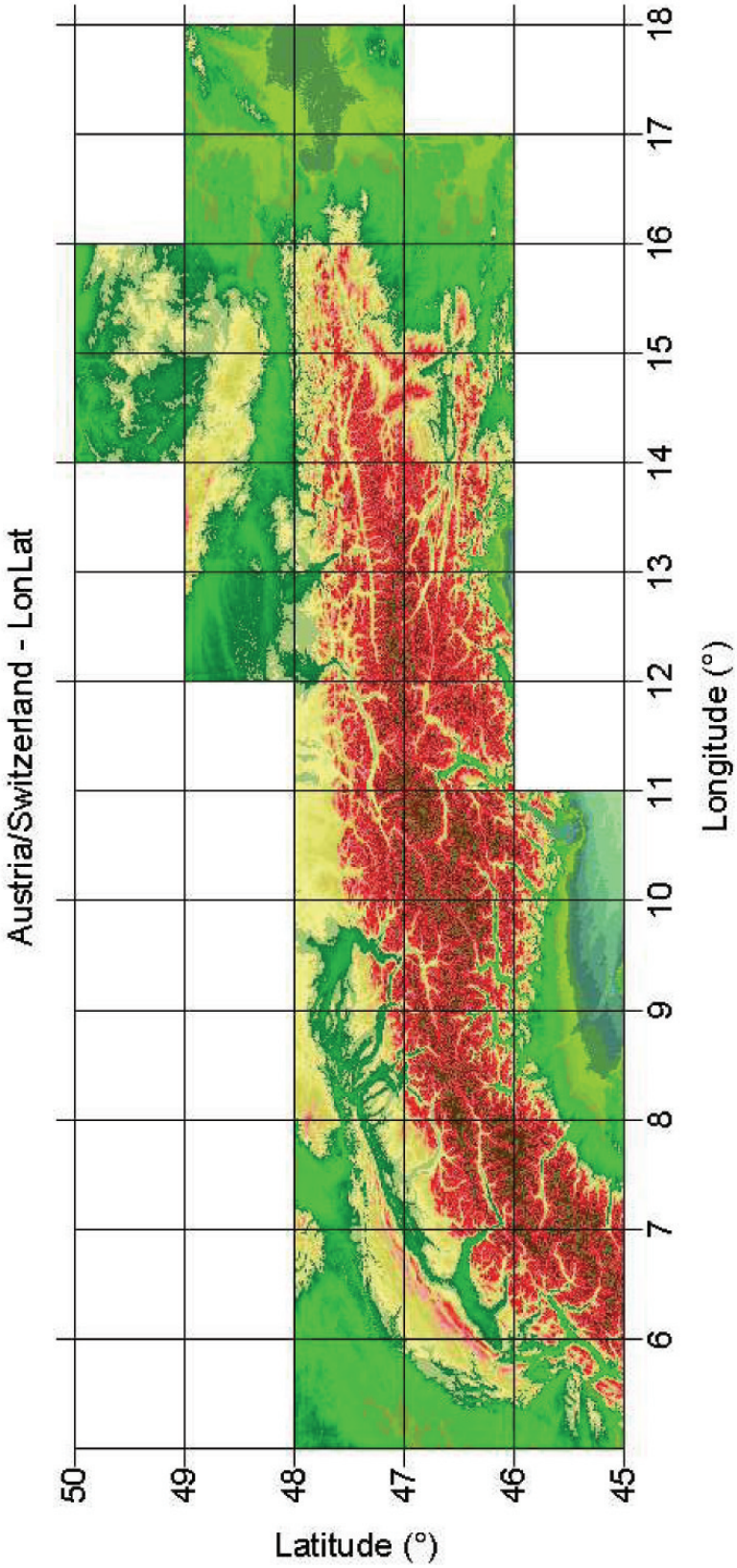


Abbildung 3.2.5.: Satellitenbild Österreichs in Lambert Koordinaten

3.3. Ausblick

Das Framework des RoutingModuls bietet derzeit Klassen zum Finden kürzester Wege in Graphen mit Abbiegeverboten und zur Koordinatentransformation an. *Matching* und Fahrtzeitprognose sind derzeit erst in der Konzeptionsphase bzw. einer frühen Testphase. Zusammen aber mit den Daten der schon bestehenden BusApplikation bietet das RoutingModul zahlreiche Einsatzmöglichkeiten und Services sowohl für den Betreiber als auch für die Kunden von BEHA.

Fahrzeugverfolgung in Echtzeit und ein Service zur Anschlusssicherung sind des Weiteren noch für den öffentlichen Verkehr denkbar. Auch ist das RoutingModul mit weiteren Schnittstellen, z. B. zu einer GIS-gestützten Datenbank, und somit ohne die manuelle Dateneingabe, für den ÖPNV abseits von BEHA denkbar. Auch Logistikunternehmen oder Routenplaner könnten Dienste des RoutingModuls beanspruchen.

Das Problem in späteren Ausbaustufen des RoutingModuls und für andere Bereiche als BEHA wird dann vor allem die Datenbeschaffung sein. In Berlin wurde bereits 1988 bis 1990 der Feldversuch LISB (**L**eit- und **I**nformations**s**ystem **B**erlin) mit Taxis zur Datensammlung für die Reisezeitprognose gestartet, um so Routenempfehlungen geben zu können. Zur Datenerfassung wurden allerdings nur Baken verwendet, die an 240 Kreuzungen installiert wurden. Ein ähnliches Projekt wurde 2003 in Berlin mit GPS gestartet. Es läuft noch unter dem Namen OIR (**O**ptimierte **I**ntermodale **R**eisezeitprognose) bis voraussichtlich 2010 im Rahmen einer Technologieoffensive, ähnlich der des BMVIT. [Janko (1994), S. 6]

Teil IV.

Reflexion und Zusammenfassung

Wenn sich durch demografischen Wandel Zahl und Altersstruktur der Bevölkerung ändert, so ändern sich auch deren Bedürfnisse bei Mobilität und Verkehr. Der ÖPNV muss sich diesem Prozess anpassen. Um dies sowohl in ökonomischer als auch in ökologischer Hinsicht zu schaffen, müssen alternative Bedienungsformen im öffentlichen Personennahverkehr angedacht werden.

Das in dieser Arbeit vorgestellte BEHA-System ist nur eine von vielen Möglichkeiten. Ob sich dieses System jemals rentiert, muss durch ein geeignetes *Controlling*-Verfahren herausgefunden werden. Die in dieser Arbeit erwähnte Statistikerstellung für BEHA ist nur ein Werkzeug. Es kann jedoch gesagt werden, dass die Entwicklung mit BEHA nicht endet, sondern nur ein erster Ansatz ist, der mit voller Konsequenz, z. B. durch eine weitere kapazitive Flexibilisierung, weiter verfolgt werden muss. BEHA wird aber schon seit 2002 entwickelt und man darf die weiteren Entwicklungen mit Spannung erwarten. So war die Entwicklung bei Abschluss dieser Arbeit noch lange nicht am Ende und es wurden auch einige der noch nicht abgeschlossenen Entwicklungen innerhalb des Projektes BEHA vorgestellt.

Im Zuge dieser Arbeit konnten die meisten zu Anfang gestellten forschungsleitenden Fragestellungen beantwortet werden. Die Ergebnisse seien an dieser Stelle noch einmal kurz zusammengefasst.

Die BusApplikation war schon vor der Implementation eines GPS-Moduls voll funktionsfähig. Das GPS-Modul konnte ohne große Schwierigkeiten und ohne Einschränkungen in die BusApplikation eingebettet werden. Im Zuge dessen wurden auch einige Wünsche der Fahrer/innen in Hinsicht auf *Usability* adaptiert. Da die bisher benutzten Mobiltelefone jedoch nicht mit einer GPS-Maus kompatibel waren, musste hier ein passendes Mobiltelefon gefunden werden. Der Prozess war aus diversen Gründen schwieriger als erwartet. Zum einen verhinderten verschleierte und unkonkrete Herstellerangaben eine schnelle Entscheidung, zum anderen war die Beschaffung von Testgeräten nicht immer einfach. Im Endeffekt wurde mit dem Siemens MC60 das preislich günstigste Modell gewählt, welches jedoch nicht in vollem Umfang den gestellten Anforderungen genügt. Als GPS-Empfänger wurde die GPS-Maus Holux GM-210 gewählt.

Die Funktionsweise und Aufgaben des RoutingModuls konnten in dieser Arbeit ausführlich dargestellt werden. Durch einige wenige Tools können die für den Einsatz des RoutingModuls notwendigen Daten beinahe voll automatisch eingegeben und verarbeitet werden. Aber auch hier ist noch großes Entwicklungspotential vorhanden, z. B. durch die Einbindung von GIS-gestützten Datenbanken. Mit der Eingabe der Graphrelevanten Daten via Microsoft Excel konnte jedoch eine effektive Zwischenlösung gefunden werden. Die Auswertung funktioniert ebenfalls, ist derzeit aber noch sehr von der Eingabe der Fahrer/innen abhängig. Die vollständige Einbindung von GPS in das RoutingModul wird hier noch den Automatisierungsgrad steigern. Die Wartung der Da-

tenbank kann durch zeitgesteuerte Archivierungswerkzeuge übernommen werden, damit die Zugriffszeiten auf die Datenbank auf einem konstant niedrigen Niveau gehalten werden, womit der Wartungsaufwand sehr niedrig gehalten werden kann.

Die vom RoutingModul verwendete Datenstruktur wurde beschrieben und erläutert. Der zum Routing verwendete BEHA-Algorithmus konnte erarbeitet und erfolgreich implementiert werden. Seine Funktionalität und Eigenschaften waren den beiden anderen getesteten Algorithmen - Dijkstra und Bellman-Ford - überlegen. Der BEHA-Algorithmus eignet sich hervorragend für die Verwendung bei BEHA und kann leicht in seiner Funktionalität erweitert werden, falls dies durch eine Änderung der Restriktionen wie z. B. Maut notwendig sein sollte.

Das *Matching*, die Zuordnung der GPS-Daten auf einen Graphen, war bei Abschluss dieser Arbeit noch in der Konzeptphase. Das *Matching* bildet jedoch die Grundlage für eine Verspätungsvorhersage. Die Vorarbeiten wurden im Rahmen dieser Arbeit schon geleistet, so dass diese Funktionen in naher Zukunft verwirklicht werden können.

Weitere Funktionen bzw. Verwendungsmöglichkeiten für das RoutingModul wie z. B. die Fahrzeugverfolgung in Echtzeit, ein Service zur Anschlusssicherung oder der Einsatz bei Logistikunternehmen oder in Routenplanern wurden ebenfalls angedacht.

GPS und in weiterer Konsequenz das RoutingModul ist jedoch nur eine Möglichkeit zur Fahrzeugortung. Alternative Bedienungsformen kommen grundsätzlich auch ohne Fahrzeugortung aus, sind dann aber stark von der Motivation der Mitarbeiter abhängig, die das System anders und weniger komfortabel bedienen müssen. Die Fahrzeugortung macht erst ein effizientes *Controlling* und eine effiziente Steuerung der Busflotte möglich. Dies wirkt sich auch auf die Kundenzufriedenheit aus. GPS ist jedoch nicht das einzige System zur Fahrzeugortung. GSM-basierte Ortung oder andere Sensoren sind ebenfalls denkbar. Das beste und kostengünstigste System ist jenes, das der realen Situation am besten angepasst ist und sich bestehender Infrastruktur bedient.

Abschließend ist zu sagen, dass alternative Bedienungsformen und im Besonderen das Projekt BEHA den ÖPNV für die Zukunft rüsten. Solange die bestehende Angebotsqualität zumindest beibehalten wird, können weitere kontinuierliche, spannende Entwicklungen freudig erwartet werden und es ist abzuwarten ob sich eine bestimmte Alternative Bedienungsform durchsetzen kann.

Teil V.

Anhang

Spezifikationen der Holux GM-210

Item	Description
Specification	Weight: < 100 g Update Rate: 1 Hz Receiver: L1, C/A code Tracks up to 12 satellites Dimension: 66 x 51 x 22.5 mm Minimum Signal Tracked: -175 dBW Store Temperature: -45 °C - +100 °C Antenna Type: Built in Patch Antenna Operation Temperature: -40 °C - +80 °C Snap Start: < 3 sec (at < 25 minutes off period) Operation Humidity: 5 % to 95 %, No condensing Power Consumption: < 170 mA at 4.5 - 5.5 V input
Position Accuracy	**Non DGPS (Differential GPS):** Position: 5 - 25 m CEP without SA Velocity: 0.1 m/sec Time: 1 usec sync GPS time **DGPS (Differential GPS):** Position: 1 - 5 m, typical Velocity: 0.05 m/sec **EGNOS/WAAS/Beacon:** Position: < 2.2 m, horizontal 95 % of time < 5 m, vertical 95 % of time
Acquisition Time	Reacquisition: 0.1 sec. averaged Hot Start: 8 sec. averaged Warm Start: 38 sec. averaged Cold Start: 45 sec. averaged

Protocol and Interface	**NMEA Protocol Output:** Baud Rate: 4800 bps Data Bit: 8 Parity: N Stop Bit: 1 **Output Format:** Standard: GGA, GSA, GSV, RMC Optional: GGL, VTG, SiRF Binary **Interface:** RS-232 + CMOS TTL Level RS-232 + DGPS
Dynamic Conditions	Altitude: 18,000 m (60,000 feet) max Velocity: 515 m/sec (700 knots) max Acceleration: ±4 G, max Jerk: 20 m/sec, max
LED Function	Power on/off and Navigation Update Indication

Tabelle 5.0.1.: Spezifikationen der GPS-Maus Holux GM-210

Glossar

Adjazenzliste Repräsentation eines Graphen in der Informatik

Agent Als Software-Agent bezeichnet man ein Computerprogramm das weitgehend unabhängig von Benutzereingriffen arbeitet, es löst Aktionen aufgrund eigener Initiative aus, reagiert auf Änderung der Umgebung, es kommuniziert mit anderen Agenten und lernt aufgrund zuvor getätigter Entscheidungen bzw. Beobachtungen.

Almanach bei *GPS* Sammlung von Daten sämtlicher Satelliten, nach denen der Nutzer die Wahl der jeweils günstigsten Satelliten treffen kann.

API bezeichnet in der Informatik eine Programmschnittstelle

Bake ist im Verkehr oder in der Schifffahrt ein Orientierungszeichen zur Positionsbestimmung.

BEHA Projekt der Firma CoCoSoftware Engineering GmbH für bedarfsgesteuerte Linienbusse

C/A-Code Der *GPS*-Code, der auf das *GPS*-L1-Signal aufmoduliert ist. Dieser Code ist eine Folge von 1023 pseudozufälligen binären zweiphasigen Modulationen auf dem *GPS*-Träger mit einer Rate von 1023 Mhz und einer Wiederholungsrate von einer Millisekunde.

CAD computerunterstützte Konstruktion, finden sich vor allem in den Bereichen von Architektur, Vermessungswesen, Raumplanung, Anlagenbau, Produktdesign, Maschinenbau, Fahrzeugbau, mechanische Simulation, etc.

Standardisierte Datenformate ermöglichen den Austausch von Daten zwischen verschiedenen CAD-Programmen.

CLDC definiert die kleinstmöglichste Konfiguration von *Java*

CSD leitungsvermitteltes mobiles Datenübertragungsverfahren

DGPS Technik zur Korrektur von Fehlern, die bei der Positionsbestimmung mittels *GPS* auftreten

Disposition bezeichnet im ÖPNV die Aufteilung oder Zuordnung verwendeter Ressourcen (hier: v. a. Fahrzeuge)

DRMS Fehlerradius bei der Positionsbestimmung mittels *GPS*

Embedded Systems ist der englische Fachbegriff für eingebettete (Computer-) Systeme, die - weitestgehend unsichtbar - ihren Dienst in einer Vielzahl von Anwendungsbereichen und Geräten versehen, wie z. B. in Flugzeugen, Autos, Kühlschränken, Fernsehern, DVD-Playern, Mobiltelefonen oder allgemein Geräten der Unterhaltungselektronik.

Embedded Systems vereinigen daher durch ihre oftmals sehr hardwarenahe Konstruktion die große Flexibilität von Software mit der Leistungsfähigkeit der Hardware.

Flächenverkehr im ÖPNV ermöglicht Quell-Ziel-Kombinationen unabhängig von Richtung und Startpunkt (im Gegensatz zum *Richtungsbandverkehr*).

GALILEO europäisches satellitengestütztes Navigationssystem zur weltweiten Positionsbestimmung

GFC *API* zur Verbindung zweier Kommunikationspartner im *CLDC*

GIS Geoinformationssytem mit dem raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und graphisch präsentiert werden.

GLONASS russisches satellitengestütztes Navigationssystem zur weltweiten Positionsbestimmung

GPRMC Datensatz des *NMEA-Standards* bei *GPS*

GPRS paketorientiertes mobiles Datenübertragungsverfahren

GPS amerikanisches satellitengestütztes Navigationssystem zur weltweiten Positionsbestimmung

Graph mathematische Struktur aus Knoten, die durch Kanten verbunden sein können.

GSM Mobilfunk-Standard für leitungsvermittelte und paketvermittelte Datenübertragung

Handshake-Verfahren ist eine einfache Methode, wie sich zwei an einer Datenübertragung beteiligte Teilnehmer nach jedem Übertragungsvorgang durch unmittelbare Quittungssignale synchronisieren.

HSCSD leitungsvermitteltes mobiles Datenübertragunsverfahren

HTTP Protokoll zur Übertragung von Daten

Ionosphärenfehler ein durch die Erdatmosphäre verursachter Fehler bei der Positionsbestimmung mittels *GPS*

ISDN internationaler Standard für den Zugang in ein digitales Telekommunikationsnetz

Java objektorientierte, plattformunabhängige Programmiersprache entwickelt von der Firma Sun Microsystems

J2EE Spezifikation einer Standardarchitektur für die Ausführung von J2EE-Applikationen. Hierzu werden in der Spezifikation Softwarekomponenten und Dienste definiert, die primär in der Programmiersprache Java erstellt werden. Die Spezifikation dient dazu, einen allgemein akzeptierten Rahmen zur Verfügung zu haben, um mit modularen Komponenten verteilte, mehrschichtige Anwendungen zu entwickeln. Klar definierte Schnittstellen zwischen den Komponenten und Schichten sollen dafür sorgen, dass Softwarekomponenten unterschiedlicher Hersteller interoperabel sind, wenn sie sich an die Spezifikation halten, und dass die verteilte Anwendung gut skalierbar ist.

J2ME Umsetzung der Programmiersprache Java für so genannte *Embedded Consumer Products* wie etwa Mobiltelefone oder PDAs

JBoss ist eine Firma, deren Hauptprojekt die Implementierung eines kostenlosen J2EE-konformen Applikationsservers ist.

JMS API für asynchrone Nachrichtenverarbeitung

JSR bezeichnet eine Anforderung für eine Änderung der JLS (**J**ava **L**anguage **S**pecification). Dies wird im Rahmen des JCP (**J**ava **C**ommunity **P**rocess) getan.

Korridor im ÖPNV eine bestimmte Art des *Richtungsbandverkehrs*

Kyoto-Protokoll Das Kyoto-Protokoll (benannt nach dem Ort der Konferenz Kyoto in Japan) ist ein Zusatzprotokoll zur Ausgestaltung der Klima-Rahmenkonvention

(UNFCCC) der Vereinten Nationen für den Klimaschutz. Es schreibt verbindliche Ziele für die Verringerung des Ausstoßes von Treibhausgasen fest, welche als Auslöser der globalen Erwärmung gelten. Die Zunahme dieser Treibhausgase wird großteils auf menschliche Aktivitäten zurückgeführt, nämlich durch das Verbrennen fossiler Brennstoffe. Die reglementierten Gase sind: Kohlendioxid (CO_2, dient als Referenzwert), Methan (CH_4), Distickstoffoxid (Lachgas, N_2O), teilhalogenierte Fluorkohlenwasserstoffe (H-FKW/HFCs), perfluorierte Kohlenwasserstoffe (FKW/PFCs) und Schwefelhexafluorid (SF_6).

Linienabweichung im ÖPNV eine bestimmte Art des *Richtungsbandverkehrs*

Linienaufweitung im ÖPNV eine bestimmte Art des *Richtungsbandverkehrs*

Linienverkehr ist das fahrplanmäßige Verkehren von Fahrzeugen für die Personenbeförderung oder den Gütertransport. Er zeichnet sich durch eine fixe Linienführung aus.

Loggen bezeichnet den Vorgang der Aufzeichnung innerhalb eines abgegrenzten Systems bestimmter Aktionen in einem Protokoll, ohne dass der Benutzer dieses Systems oder das System selbst durch das Loggen beeinflusst wird.

Matching Mittels *Matching* wird eine gegebene Position eines bewegten Fahrzeuges oder seine Trajektorie (Bahn) auf die Knoten-Kanten-Struktur der digitalen Darstellung eines Gebietes eingepaßt. Diese Georeferenzierung transformiert die absoluten Koordinaten in relative Koordinaten im Graphen.

Middleware bezeichnet in der Informatik anwendungsunabhängige Technologien, die Dienstleistungen zur Vermittlung zwischen Anwendungen anbieten, so dass die Komplexität der zugrundeliegenden Applikationen und Infrastruktur verborgen wird.

MIDlet Softwareprogramm für ein Mobiltelefon in *Java* geschrieben

MIDP ist ein Profil *J2ME*, das speziell auf die Fähigkeiten kleiner mobiler Endgeräte wie Mobiltelefon ausgelegt ist.

NAVSTAR Name unter dem *GPS* vom amerikanischen Verteidigungsministerium entwickelt wurde

NMEA US-amerikanische Vereinigung von Elektronikherstellern und -händlern der Schifffahrtsindustrie. Die Vereinigung wurde 1957 gegründet und 1969 als Gesellschaft eingetragen. Ihre Hauptziele sind die Förderung von Standards und techni-

schen Entwicklungen in der Marineelektronik sowie die technische Weiterbildung ihrer Mitglieder.

NMEA-Standard Übertragungsstandard im maritimen Bereich für Positionsdaten

Parity Bit dient bei der Datenübertragung der Fehlererkennung

Plug & Play heißt soviel wie „einsetzen und los gehtt's". Industriestandard, der mit Microsofts Windows 95 eingeführt wurde und die Installations- und Konfigurationsaufgaben ohne Eingreifen des Anwenders selbsttätig löst/zu lösen versucht.

PPS System zur präzisen Positionsbestimmung, z. B. bei *GPS*

RBL Unter einem RBL versteht man ein im ÖPNV benutztes Rechnerverbund-System, das für vielfältige Aufgaben verwendet werden kann.

Mit dem RBL werden hauptsächlich folgende Bereiche gesteuert: Informations- und Kommunikationsmöglichkeit zwischen Fahrzeug und Leitstelle, Rechnergestützter Fahrbetrieb, Fahrgastinformation in Zügen und Bussen und an Haltestellen, über Mobilfunk und Internet, die so genannte dynamische Fahrgastinformation.

Richtungsbandverkehr bedeutet von einem fixen Startpunkt aus ein bestimmtes Gebiet in eine vorgegebene Richtung zu bedienen.

RMC Datensatz des *NMEA-Standards*

Routing ist ein Begriff aus der Netzwerktechnik und bezeichnet das Suchen kürzester bzw. günstigster Wege in einem Graphen.

RS-232 Die RS-232 oder serielle Schnittstelle ist eine Spannungsschnittstelle, d. h. die verschiedenen Spannungspegel stellen die Information dar.

Sektor im ÖPNV eine bestimmte des *Richtungsbandverkehrs*

Semaphor in der Informatik eine Datenstruktur zur Synchronisation von Prozessen

serielle Schnittstelle siehe *RS-232*

sentence bezeichnet eine Dateneinheit von maximal 80 Zeichen bei der Übertragung von *NMEA*-Datem

SMS paketorientiertes mobiles Datenübertragunsverfahren

SPS im Gegensatz zu *PPS*, ein Dienst von GPS, der weniger präzise arbeitet und daher für zivile Nutzung freigegeben wurde

Stack (engl.: Stapel) in der Informatik eine nach dem LIFO-Prinzip (**L**ast **I**n **F**irst **O**ut) organisierte Datenstruktur

Terminal (engl.: Endstation), bei BEHA Bezeichnung für das Datenendgerät bei einer Bushaltestelle, bei der der Kunde Fahrten bestellen kann und informiert wird.

Thread bezeichnet in der Software-Architektur einen Ausführungsstrang innerhalb eines Prozesses, der nebenläufig zu anderen Threads ausgeführt werden kann.

TMS Von der Firma CoCo Mobile Telematics entwickeltes System zur Überwachung und *Disposition* von Fahrzeugen im ÖPNV oder in Logistikunternehmen.

XML Standard zur Erstellung maschinen- und menschenlesbarer Dokumente in Form einer Baumstruktur

UKW elektromagnetische Wellen im Frequenzbereich von 30 MHz bis 300 MHz

UMTS paketorientiertes mobiles Datenübertragunsverfahren

UTC-Standard international koordinierter Atomzeitstandard

VDV Im VDV sind die Firmen des öffentlichen Personennahverkehrs und des Güterverkehrs mit Schwerpunkt Eisenbahngüterverkehr in Deutschland organisiert.

WGS84 geodätische Grundlage des *GPS*-Systems und der Vermessung der Erde

ÖPNV ist der Personenverkehr mit Verkehrsmitteln, die nicht zum SPFV (**S**chienen**p**ersonen**f**ern**v**erkehr) oder zum IV (**I**ndividual**v**erkehr) gerechnet werden. Der ÖPNV ist ein Teilsystem des ÖV (**Ö**ffentlichen **V**erkehr).

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[Betke August 2001] Betke, Klaus: The NMEA 0183 Protocol. (August 2001). – URL http://nmeatool.nmea2000.de/download/0183.pdf,15.Juni,2005

[Bilajbegovic Chmielorz, 2001] Bilajbegovic, Asim: Koordinatensysteme und Transformation. In: *VDV-Schriftenreihe - Der Vermessungsingenieur in der Praxis - GPS-Referenzstationsdienste - GPS-Antennen - Koordinatensystem und Transformationen* Band 19 (Chmielorz, 2001), S. 142–163

[Blum u. a. Jänner 2002] Blum, Martin; Hitter-Fertl, Susanne; Leihs, Dietrich; Nowak, Willi; Rasmussen, Ulla; Reiss-Enz, Viktoria; VCÖ - Verkehrsclub Österreich (Hrsg.): *Neue Technologien für sichere und barrierefreie Mobilität.* Jänner 2002

[Chouquet-Stringer 2004] Chouquet-Stringer, Mathieu: Le système GPS (Global Positionning System). In: *www.aviation-info.fr* (2004). – URL http://faq.bigip.mine.nu:8008/avion/GPS.php,22.August,2005

[CoCo 2002] CoCo, MobileTelematics GmbH: *BEHA - Hardware Beschreibung.* Jänner 2002

[CoCo 2003] CoCo, MobileTelematics GmbH: *BEHA - Bedarfshaltestellen - Innovative Lösungen für den öffentlichen Verkehr.* 2003

[CoCo 2005] CoCo, Software Engineering GmbH: *BEHA : Bedarfshaltestellen : Busbestellung.* 2005. – URL http://www.beha.at/businfo/app/SearchSchedule,29.Juni,2005

[Dodel und Häupler 2004] Dodel, Hans; Häupler, Dieter; Schulz, Sabine (Hrsg.); Jehle, Gunnar (Hrsg.): *Satellitennavigation - GALILEO, GPS, GLONASS, Integrierte Verfahren.* Hüthig Telekommunikation, Bonn, 2004

[Eder 2001] Eder, Martin: *Bedarfsgesteuerte Öffentliche Verkehrsarten für den Bregenzerwald - Bewertung mittels einer Nutzwertanalyse*, TU - Wien, Dissertation, 2001

[Eickstädt und Reuhl 2004] Eickstädt, Dieter; Reuhl, Thomas: *J2EE mit Struts & Co. - Java-Projekte mit Struts, Tomcat, JBoss und Eclipse.* Markt+Technik Verlag, München, 2004

[Fellner 2004] Fellner, Dieter: *Algorithmen & Datenstrukturen II.* Technische Universität Braunschweig, 2004. – URL http://www.cg.cs.tu-bs.de/bs_root/lv/LV/lvcg04/AuDII/Folien-Fellner-Kap08-sw.pdf,02.August,2005

[Girnau u. a. 2001] Girnau, Günter; Müller-Hellmann, Adolf; Blennemann, Friedhelm; Studiengesellschaft für unterirdische Verkehrsanlagen e.V., Köln (Hrsg.): *Telematik im ÖPNV in Deutschland*. Verband Deutscher Verkehrsunternehmen - VDV-Förderkreis e.V., 2001

[Girnau u. a. Mai, 1997] Girnau, Günter; Müller-Hellmann, Adolf; Blennemann, Friedhelm; Studiengesellschaft für unterirdische Verkehrsanlagen e.V., Köln (Hrsg.): *Zukunftsfähige Mobilität - Menschen bewegen - ÖPNV in Deutschland*. Verband Deutscher Verkehrsunternehmer - VDV-Förderkreis e.V., Mai, 1997

[Gorbach November 2003] Gorbach, Hubert (österreichischer Vizekanzler und Bundesminister): Vorwort. In: *TAKE ÖV - Entwicklungen der Verkehrstelematik 1999-2003 - Innovative Mobilitätsdienstleistungen - 3 Jahre nach dem Wettbewerb* (November 2003), S. 3. – URL `www.bmvit.gv.at/sixcms_upload/media/180/take_oev_broschuerekompr.pdf,02.Juni,2005`

[Grewal u. a. 2001] Grewal, Mohinder; Weill, Lawrence; Andrews, Angus; Grewal, Mohinder (Hrsg.): *Global Positioningsystems, Inertial Navigation, and Integration*. John Wiley & Sons, Inc., 2001

[Hoepke 1995] Hoepke, Erich; Technische Akademie Esslingen (Hrsg.): *Omnibusse im Verkehrssystem von Ballungsräumen - Planung, Betrieb und Leitsystem - Technik der Omnibusse, Obusse, Duo-Busse und Spurbusse*. expert verlag, 1995

[Hoffmann 1993] Hoffmann, Peter: *Flexible Bedienungsformen im ÖPNV - Elemente einer mehrstufig differenzierten Verkehrserschließung*. Schmidt, Bielefeld, 1993

[Holux, Technology Inc.] Holux, Technology Inc.: *GM-210 GPS Receiver*. – URL `http://www.holux.com.tw/Temp%20web/download/GM-210-Spec-E%20(930727).pdf,22.August,2005`

[Hoopmann April 1997] Hoopmann, Ralf: Rufbusse - Systemvergleich und aktuelle Entwicklungen. In: *Veröffentlichungen und Fachbeiträge der PGN und ihrer Mitarbeiter/innen* (April 1997). – URL `http://www.pgn-kassel.de/central/veroeff/art06/art06.htm,22.Juli,2005`

[Janko 1994] Janko, Josef: *Probleme der Reisezeitprognose in einem Leitsystem für den Straßenverkehr*, Technische Universität Berlin: Verkehrswesen und Angewandte Mathematik, Dissertation, 1994

[Janovsky November 2002] Janovsky, Harald: Serielle Übertragung von Daten / Höhere technische Bundeslehranstalt Linz. URL `http://manuel.mausz.at/5ahta/prt/skripten_jany/seriell.doc,24.August,2005`, November 2002. – Forschungsbericht

[Köhne und Wößner Mai, 2005] Köhne, Anja; Wössner, Michael: NMEA-0183 Daten. In: *www.kowoma.de* (Mai, 2005). – URL `http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm,22.August,2005`

[Lichtl u. a. 2005] Lichtl, Balázs; Wasner, Walter; Sterzl, Christian: *Bedarfshaltestellen (BEHA) - Funktionsweise des BEHA-Systems - Leitfaden für den Lenker*, 2005

[Lichtl 2002] Lichtl, DI B.: *BEHA - Der Server*. CoCo Software Engineering GmbH (Veranst.), 2002

[Mehlert Juni 1998] Mehlert, Christian: *Angebotsbezeichnung bei alternativen Bedienungsformen*. Alba, Düsseldorf, Juni 1998

[Müller-Hellmann und Nickel Oktober 2000] Müller-Hellmann, Adolf; Nickel, Bernhard; Verband Deutscher Verkehrsunternehmen, Köln (Hrsg.): *Stadtbus - mobil sein in Klein- und Mittelstädten*. Verband Deutscher Verkehrsunternehmen - VDV-Förderkreis e.V., Oktober 2000

[Panitzki 2004] Panitzki, Michael: *Wenn du weißt wo du bist, kannst du sein wo du willst*. URL `http://home.arcor.de/m.panitzki/html/navigation/inhaltsverzeichnis.htm,31.August,2005`, 2004

[Pausits 2001] Pausits, Attila: Mobile Traffic Environment/Anforderungen, Einsatzpotentiale und Funktionalitäten eines Car PC. In: *Schriftenreihe Telekommunikation, Information und Medien* Band 11 - Verkehrstelematik (2001), S. 21

[Pfleger und Linauer 2001] Pfleger, Ernst; Linauer, Martin: Untersuchung des Blickverhaltens in Abhängigkeit von Straßenlage und Verkehrsgeschehen. In: *Schriftenreihe Telekommunikation, Information und Medien* Band 11 - Verkehrstelematik (2001), S. 16

[Schmid 2000] Schmid, Wolfgang: *Berechnung kürzester Wege in Straßennetzen mit Wegeverboten*, Fakultät für Bauingenieur- und Vermessungswesen der Universität Stuttgart, Dissertation, 2000. – URL `http://www.ifp.uni-stuttgart.de/publications/dissertationen/Schmid_Diss.pdf,01.August,2005`

[Schnabel 1997–2005] Schnabel, Patrick: Datenübertragung im Mobilfunknetz. In: *das ELKO - das Elektronik Kompendium* (1997-2005). – URL `http://www.elektronik-kompendium.de/sites/kom/0910141.htm,23.August,2005`

[Schumann 1996] Schumann, Toralf: *GPS an der TU Ilmenau*. URL `http://ikmcip1.e-technik.tu-ilmenau.de/~traut/gps_www/gps_main.htm,22.August,2005`, 1996

[Seidl 22. Juni, 2005] Seidl, Conrad: Bevölkerung wächst in rasantem Tempo. In: *Der Standard* (22. Juni, 2005), S. 10

[Stütz November 2003] Stütz, Andrea: BEHA - Ihr Bus kommt auf Knopfdruck - Bedarfshaltestellen. In: *TAKE ÖV - Entwicklungen der Verkehrstelematik 1999-2003 - Innovative Mobilitätsdienstleistungen - 3 Jahre nach dem Wettbewerb* (November 2003), S. 12f. – URL `www.bmvit.gv.at/sixcms_upload/media/180/take_oev_broschuerekompr.pdf,02.Juni,2005`

[Sutter September 2000] Sutter, K.: GPS - Neue Chancen für den Einsatz im ÖPNV - Ortung über Satelliten/GPS und Funk-LAN im ÖPNV. In: *Vortrag beim beka Classic-Seminar in Königswinter*, September 2000

[T-Mobile Juni, 2005] T-Mobile: Transparenz auf allen Routen. In: *move - Magazin für Transport & Logistik* Nr.:6 (Juni, 2005), S. 5

[Tödter Februar 2004] Tödter, Kai: *J2ME MIDlet-Entwicklung mit Eclipse, Ant und Antenna*. URL `http://www.toedter.com/download/J2ME-Eclipse-Antenna.pdf,18.August,2005`, Februar 2004

[Thaller 1999] Thaller, Georg; Büge, Michael (Hrsg.): *Satellitennavigation - Das Global Positioning System (GPS)*. Verlag für Technik und Handwerk, Baden-Baden, 1999

[Verkehrsplanung 1990] Verkehrsplanung, Arbeitskreis „Änderung des Benutzerverhaltens“ des Arbeitsausschusses „Öffentlicher Verkehr“ der Arbeitsgruppe V.: *Öffentlicher Personennahverkehr - Empfehlungen zur Verbesserung der Akzeptanz des ÖPNV*. Forschungsgesellschaft für Straßen- und Verkehrswesen, 1990

[ViaMichelin 2001–2005] ViaMichelin, (Deutschland) GmbH: *ViaMichelin: Routenplaner, Straßenkarten online, Tourismusinformation, Restaurants & Hotels in Europa*. 2001-2005. – URL `http://www.viamichelin.de/,15.August,2005`

[Voser Mai, 1998] Voser, Stefan: Koordinatensalat - Wenn das Koordinatensystem nicht paßt. In: *GeoBIT* 5/98 (Mai, 1998), S. 26–27. – URL `http://www.geocities.com/CapeCanaveral/1224/savpub/savpub-104.htm,28.Oktober,2001`

[Walther Juni 2002] Walther, Christoph: Amabile - Ausschreibung und Modellierung von Alternativen Bedienungsformen in Form von Teilnetzen unter Integration traditioneller Linienverkehre. In: *EPOMM - European Platform On Mobility Management* (Juni 2002). – URL `http://www.epommweb.org/epomm_example.phtml?sprache=en&id=224,14.August,2005`

[Wikipedia] Wikipedia: Die freie Enzyklopädie. Wikimedia Foundation Inc.. – URL `http://de.wikipedia.org/,22.August,2005`

[Wiltschko Juni 2005] Wiltschko, Thomas: AG Verkehrsinformationstechnik. In: *Uni Stuttgart - Institut für Anwendungen der Geodaesie im Bauwesen - Forschung* (Juni 2005). – URL `http://www.uni-stuttgart.de/iagb/forschung/verkehr/agverkehr_index.htm,16.August,2005`

Schlagwortverzeichnis

V

W

Z

Ö