CARL VON OSSIETZKY universität OLDENBURG

Fakultät II – Informatik, Wirtschafts- und Rechtswissenschaften
Department für Informatik

Bachelorarbeit

# Wegpunktnavigation für Fußgänger auf Basis von standardisierten Geodateninfrastrukturen

vorgelegt von

**Dennis Geesen**

Gutachter:

**Prof. Dr. Susanne Boll**
**Dr. Dietrich Boles**

Betreuer:

**Dipl. Inform. Martin Pielot**

19. November 2008

# Zusammenfassung

Ziel dieser Arbeit war die Implementierung einer Navigationssoftware für Fußgänger, die eine standardisierte Geodateninfrastruktur nutzt. Dazu sollten Probleme und Lösungsansätze vorgestellt werden, die im Vergleich zu einer klassischen Kraftfahrzeugnavigation bei einer Navigation für Fußgänger vorkommen können. Da für eine Navigation auch Straßendaten notwendig sind, war ein weiteres Ziel dieser Arbeit, dass die Navigationsanwendung über eine Geodateninfrastruktur die erforderlichen Daten abrufen kann. Dazu werden verschiedene vektorbasierte Austauschformate, Geoinformationssysteme und Standards betrachtet und verglichen. Des Weiteren wird das Problem, dass sich Fußgänger frei bewegen können, für die Routenführung betrachtet. Dieses wirkt sich auf die Positionsbestimmung und dem Erreichen eines Wegpunktes aus. Als Lösungsansätze werden Verfahren, wie Map Matching, Schwellwertverfahren oder die Projektion eines Lotpunktes vorgestellt. Abschließend wird basierend auf dem NICCIMON-Framework eine konkrete Navigationsanwendung entworfen, die eine Auswahl geeigneter Lösungsansätze verwendet.

Bei der Konzeption der Geodateninfrastruktur wurde als Datenquelle OpenStreetMap verwendet, da diese Daten frei nutzbar sind. Als Nachteil ist zu nennen, dass einige Daten nicht korrekt sind, da die Daten von unprofessionellen, freiwilligen Personen erfasst werden. Diese Daten stehen als Shapefile zur Verfügung, so dass sie auf Grund des Quasi-Standards, welches Shapefile ist, in jedes vorgestellte Geoinformationssystem geladen werden können. Die Wahl eines Geoinformationssystems fiel auf den GeoServer, da dieser zwar im Vergleich zum UMN MapServer vergleichbar ist, jedoch ist der GeoServer die Referenzimplementierung des Web Feature Service. Dieser Web Feature Service ist als offizieller OGC- und ISO-Standard für eine vektorbasierte Geodatenschnittstelle die konsequente Wahl, da nur dieser Service die Anforderungen nach einem Standard und hoher Interoperabilität erfüllt.

Beim Routing für Fußgänger wurde zum einen die Berechnung der Route betrachtet. Hier wurden der Dijkstra- und der A*-Algorithmus vorgestellt. Der A*-Algorithmus ist als Erweiterung des Dijkstra-Algorithmus effizienter und dient daher in der Software zur Berechnung des kürzesten Weges. Für die Berechnung ist es zusätzlich notwendig, dass die aktuelle Position als Startpunkt mit dem Straßennetz verknüpft wird, da sich die Person nicht zwangsweise auf einer Straße befindet. Hierzu wurde die Projektion eines Lotfußpunktes betrachtet, um die aktuelle Position mit der nächstgelegenen Straße zu verbinden. Auf Grund der Kugelform der Erde wurde eine sphärische Projektion hergeleitet. Nach der Routenberechnung wird der Benutzer bei der Routenführung von Wegpunkt zu Wegpunkt geleitet. Das Erreichen eines Wegpunktes kann durch ein Schwellwert festgestellt werden. Hier ist es jedoch schwer einen verallgemeinerten Schwellwert zu definieren, da die Vielfalt der Wegpunkte sehr groß ist. Als Maßstab für den Schwellwert kann z.B. der Straßentyp und damit die Straßenbreite genutzt werden.

# Inhalt

# 1 Einleitung

Unsere heutige Gesellschaft ist geprägt von Mobilität. Dadurch haben Navigationssysteme immer mehr an Bedeutung gewonnen. Im Gegensatz zu fest eingebauten Navigationssystemen in Kraftfahrzeugen, stieg in den vergangenen Jahren die Nachfrage nach mobilen Navigationssystemen, so genannte Personal Navigation Devices (PND). Laut des Marktforschungsunternehmen GfK wuchs im Jahr 2007 der Markt für PNDs um etwa 80 Prozent zum Vorjahr [GfK08]. Heute nutzen auch Fußgänger mobile Endgeräte um zu erfahren, wo sie sind oder wie sie ein Ziel erreichen können. Die für Kraftfahrzeuge konzipierten PNDs sind jedoch für Fußgänger ungeeignet, da z.B. die Routenführung Autobahnen verwendet, Fußgänger jedoch keine Autobahnen benutzen dürfen. Ebenso bleiben Rad- und Fußwege bei PNDs oft unberücksichtigt. Des Weiteren ist die Verwendung eines separaten Gerätes nötig, welches nicht immer bei Bedarf zur Hand ist. Daher greifen Fußgänger immer mehr auf GPS-basierter Navigationstechnik für Mobiltelefone zurück.

Die Navigationsführung für Fußgänger unterscheidet sich jedoch von der klassischen Kraftfahrzeugnavigation. Fußgänger sind nicht zwangsweise an eine Straße gebunden, sondern können sich frei bewegen. Hierdurch entstehen jedoch viele Probleme, die bei einer Kraftfahrzeugnavigation nicht zu finden sind. Durch die freie Bewegung, auch abseits bekannter Straßen, ist es schwer festzustellen, auf welcher Straße sich der Fußgänger befindet. Klassische Map-Matching-Verfahren können daher nicht angewandt werden. Auf Grund der langsameren Bewegung und des teilweise ungenauen GPS-Positionssignals, wird eine genaue Positionsbestimmung zusätzlich erschwert. Dies führt auch bei der Routenführung zu Problemen. Im Gegensatz zu einer klassischen Routenführung, können beim Routing für Fußgänger die Wegpunkte nicht immer nacheinander abgelaufen werden. Durch die Nutzung von Abkürzungen oder nicht verzeichneten Strecken, ist es dem Fußgänger möglich Wegpunkte zu überspringen. Dies stellt zum einen die Frage, wann ein Wegpunkt erreicht wurde, wann einer Übersprungen wurde und wann sich der Fußgänger so von der Route entfernt hat, dass sie neuberechnet werden muss. Wird eine Route berechnet, so ist dazu ein Startpunkt notwendig. Dieser muss mit dem bekannten Straßennetz verbunden sein. Die Festlegung und Verknüpfung der aktuellen Position als Startknoten stellt somit ein weiteres Problem dar. Als weitere Problemquelle sind die verwendeten Daten zu betrachten. Datenquellen für Navigationsanwendungen sind bei Kartenanbietern für Kraftfahrzeugnavigation optimiert, so dass Fuß- und Radwege oft nicht beachtet werden. Des Weiteren sind diese Daten oft in einem speziellen Format, so dass die Anwendung nur die Daten eines Anbieters nutzen kann.

Für die Lösung dieser Probleme existieren bereits verschiedene Lösungsansätze, die sowohl aus dem Bereich der Kraftfahrzeugnavigation, als auch aus dem Bereich der Geoinformatik stammen. Diese Lösungsansätze können auf eine Navigation für Fußgänger angepasst werden. Dazu werden die jeweiligen Lösungen auf ihre Eignung und im Kontext einer Fußgängernavigation untersucht. Die Eignung der untersuchten Lösungen finden in einer konkreten Navigationssoftware für Fußgänger eine Anwendung und werden somit auf ihre Eignung geprüft.

Diese Arbeit soll die verschiedenen Probleme bei der Fußgängernavigation und die Verwendung von verschiedenen Datenquellen untersuchen. Dabei werden die Probleme ausgehend von der Datenquelle bis zur Ausgabe der Routenführung fragmentweise betrachtet. Dabei gliedert sich die Arbeit in zwei theoretische und einen praktischen Bereich. Zum einen wird in Kapitel 2 die Bereitstellung und Abfrage der Geodaten betrachtet. Zum anderen beschäftigt sich Kapitel 3 mit der theoretischen

Darstellung der Probleme, die zum Routing für Fußgänger gehören. Diese beiden theoretischen Teile fließen in Kapitel 4 ein, in dem eine konkrete Navigationsanwendung für Fußgänger entworfen wird.

Für die Bereitstellung der Geodaten wird eine Geodateninfrastruktur (GDI), bestehend aus der Datenquelle, einem Geoinformationssystem, einem Client und deren Austauschformate, betrachtet. Für die GDI werden in Kapitel 2.1 zuerst die Anforderungen festgelegt, damit die geeigneten Komponenten bestimmt werden können. Die verschiedenen Austauschformate innerhalb der GDI sind ein wesentliches Auswahlkriterium für das Geoinformationssystem. Daher wird in Kapitel 2.2 eine Auswahl an möglichen Austauschformaten vorgestellt. Sowohl Anforderungen und Austauschformate fließen in Kapitel 2.3 ein, in dem geeignete Geoinformationssysteme zur Haltung der Geodaten betrachtet werden. Dabei wird unter anderem auf eine standardisierte Schnittstelle fokussiert, die der Client nutzen soll. Die Geodaten, die in dem Geoinformationssystem bereitgestellt werden, können von verschiedenen Anbietern stammen. Diese Anbieter werden in Kapitel 2.4 vorgestellt. Abschließend wird das Kapitel 2 durch einen Vergleich vervollständigt. Dort werden die vorgestellten Komponenten der GDI ausgehend von der Datenquelle verglichen, um so die konkreten Komponenten für die GDI zu bestimmen.

Kapitel 3 beschäftigt sich mit dem Routing für Fußgänger. Dazu werden verschiedene Probleme betrachtet und Lösungsansätze vorgestellt. Kapitel 3.1 betrachtet die Berechnung einer Route. Dort wird zuerst auf die möglichen Fehler der Geodaten eingegangen, da korrekte Geodaten für eine erfolgreiche Routenberechnung erforderlich sind. Für die eigentliche Berechnung der Route werden daraufhin konkrete Algorithmen vorgestellt, die den kürzesten Weg in einem Straßennetz berechnen können. Aufbauend auf der Routenberechnung beschäftigt sich Kapitel 3.2 mit der Routenführung. Dazu geht das Kapitel auf die Probleme beim Routing für Fußgänger ein. Zu Anfang wird das Festlegen eines Startpunktes beschrieben. Hierbei gibt es verschiedene Möglichkeiten, um den Startpunkt mit dem Straßennetz zu verknüpfen. Danach wird die eigentlich Routenführung betrachtet. Dazu wird zum einen gezeigt, welche Probleme bei Fußgängern auftreten können und wie diese mit verschiedenen Map Matching gelöst werden können. Dazu zählt auch das Erreichen eines Wegpunktes, sowie die Anzeige, wo sich der nächste Wegpunkt befindet. Im schlechtesten Fall muss eine Route neu berechnet werden, wenn der Fußgänger zu weit von der Strecke abgekommen ist. Auch dies wird hier kurz angesprochen. In Kapitel 3.3 wird aufbauend auf Kapitel 3.1 und 3.2 entschieden, welche Verfahren geeignet sind und welche in der konkreten Navigationsanwendung verwendet werden sollen. Diese Navigationsanwendung wird in Kapitel 4 entworfen. Dazu wird eine Anforderungsdefinition und -analyse durchgeführt, so dass abschließend eine Architektur für die eigentliche Implementierung vorliegt.

# 2 Auswahl der Standardgeodateninfrastrukturen

Um eine Navigationssoftware zu entwickeln, ist man auf entsprechende Daten angewiesen, welche alle nötigen Informationen liefern. Dies sind hier in erster Linie Straßen mit all ihren Ausprägungen, wie deren Darstellungsform, deren Straßenname, Beziehungen untereinander oder der Straßentyp. Auf Grund der geometrischen Darstellung werden diese als Geodaten bezeichnet. Um diese Geodaten austauschen zu können, wurden mehrere Austauschformate entwickelt. Im Folgenden sollen verschiedene Austauschformate betrachtet werden, die für diese Anwendung in Frage kommen. Dabei soll das Austauschformat Anforderungen erfüllen. Zum einen muss das Austauschformat Geodaten so liefern können, dass damit ein Routing möglich ist. Zum anderen sollen bestimmte Standards oder Quasi-Standards berücksichtigt werden, so dass eine hohe Interoperabilität und Unterstützung gewährleistet werden kann. Die Geodaten sollen zentral in einem Server gespeichert werden, der entsprechende Dienste zum Abruf dieser Daten bereitstellt. Dieser Server soll somit ein Geoinformationssystem (GIS) sein, welches zusammen mit der Navigationsanwendung als Dienstnutzer eine Geodateninfrastruktur (GDI) ergbit. Im Folgenden sollen die Anforderungen an diese GDI definiert werden, um nach einem Überblick über gängige Austauschformate und GIS die für diese Arbeit geeignetsten GDI-Komponenten zu bestimmen.

## 2.1 Anforderungen an die Geodateninfrastruktur

Die Anforderungen an die Komponenten der GDI, also das GIS, das Austauschformat und der Client, lehnen sich an die ISO-Norm 9126 an. Die ISO-Norm 9126 betrachtet Funktionalität, Zuverlässigkeit, Benutzbarkeit, Effizienz, Änderbarkeit und Übertragbarkeit um eine Softwarequalität zu beschreiben [Rez04]. Da diese Anforderungen jedoch für eine Software vorgesehen sind, sollen sich die nachfolgenden Punkte nur grob an die ISO-Norm 9126 richten. Diese Anforderungen sollen nachfolgenden im Kontext einer Navigationsanwendung, wie sie in dieser Arbeit entworfen werden soll, betrachtet werden. Dabei soll ebenso Rücksicht auf eine mögliche zukünftige Verwendung für eine andere Anwendung genommen werden.

### 2.1.1 Funktionalität

Das Austauschformat soll in der Lage sein Geodaten für verschiedene Aufgaben, wie Navigation oder Darstellung von POIs, zu liefern. Daher soll es das Austauschformat erlauben, Straßen mit Koordinaten, Verbindungen zwischen diesen, Straßennamen, Straßentypen und deren Beziehungen zu speichern. Dabei sollen alle nötigen Informationen korrekt und eindeutig gespeichert und extrahiert werden können, damit eine sichere und zuverlässige Navigation gewährleistet werden kann. Das GIS soll dabei in der Lage sein, das gewählte Austauschformat mit all seiner Funktionalität zu unterstützen. Eine weitere wichtige Eigenschaft ist die Konformität. Hierbei soll das Austauschformat und das GIS auf existierende Standards aufbauen. Zu diesen Standards gehört die ISO-Familie 191XX mit der im Vordergrund stehenden ISO-Norm 19107. Diese beschreibt ein Schema für Raumbezugsgrundformen, wie Punkt, Linie, Fläche und Körper. Aufgrund dieser Raumbezugsgrundformen muss es sich um ein vektorbasiertes Austauschformat handeln, da nur diese eine Speicherung von primitiven Grundformen erlaubt. Diese ISO-Norm soll ebenso als Grundlage für das GIS dienen. Des Weiteren soll eine hohe Interoperabilität gewährleistet werden. Dazu sollen auch Standards des Open

Geospatial Consortium (OGC)[1] betrachtet werden, die eine hohe Interoperabilität im Bereich der Geoinformationen anstreben.

### 2.1.2 Zuverlässigkeit

Das Austauschformat soll eine bestimmte Reife erreicht haben. Hier sollte das Austauschformat in einer finalen Version vorliegen und das GIS analog in einer stabilen Version. Ebenso sollte es möglich sein, dass das Austauschformat auf Fehler prüfbar ist, damit das Austauschformat bei Fehleingaben nicht die Stabilität der Software beeinträchtigt. Das GIS soll somit auch robust gegenüber unvorhergesehene Eingaben sein. Die Komponenten sollen dem Open-Source-Gedanken folgen, so dass dadurch eine bestimmte Langlebigkeit gewährleistet und die Benutzung nicht von lizenzrechtlichen Bestimmungen beeinflusst werden kann.

### 2.1.3 Benutzbarkeit

Das GIS soll über eine Benutzerschnittstelle zu administrieren sein. Dabei soll das Importieren und Exportieren von Daten während der Laufzeit möglich sein. Die Benutzbarkeit des Austauschformats bezieht sich hier auf die Attraktivität und Verständlichkeit. Eine hohe Attraktivität kann z.B. durch Nutzung von Standards gegeben sein, die bereits von möglichen Projektpartner genutzt werden. Eine hohe Verständlichkeit kann unter anderem durch eine Dokumentation erreicht werden. Dies soll eine hohe Akzeptanz bei möglichen Benutzern und Projektpartnern hervorrufen. Entsprechend sollte das Austauschformat gewissen Standards, wie in Abschnitt 2.1.1 beschrieben, folgen. Des Weiteren sollten alle Komponenten über eine ausführliche Dokumentation verfügen, welches die Verständlichkeit und Anwendbarkeit durch Dritte erleichtert. Diese Anforderung könnte auch durch eine existierende Community aufgewertet werden.

### 2.1.4 Effizienz

Da ein in Frage kommendes Austauschformat für eine Anwendung auf einem mobilen Endgerät gedacht ist, sollte der Ressourcenverbrauch im Vergleich zu den Funktionalitäten gering sein. Hierbei soll betrachtet werden, ob es sich um ein offenes oder bytebasiertes Austauschformat handelt, welches sich auf die Effizienz bei der De- und Enkodierung des Austauschformats ausdrückt. Somit soll erreicht werden, dass ein effizientes Routing durchgeführt werden kann und die Antwort- und Verarbeitungszeiten möglichst gering gehalten werden. Ebenso spiegelt sich die Größe der Datei in der Übertagungszeit wider. Das GIS sollte alle Anfragen schnell abarbeiten können und dabei wenig Betriebsmittel in Anspruch nehmen.

### 2.1.5 Änderbarkeit

Bei der Änderbarkeit soll in erster Linie die Analysierbarkeit und Modifizierbarkeit betrachtet werden. Bei der Analysierbarkeit soll geprüft werden, wie hoch der Aufwand zur Fehlererkennung ist, in dem z.B. Validatoren zur Verfügung stehen. Bei der Modifizierbarkeit soll betrachtet werden, inwie-

[1] `http://www.opengeospatial.org`

fern das Austauschformat und das GIS an eigene Bedürfnisse angepasst oder erweitert werden kann. Des Weiteren sollte die GDI in der Lage sein möglichst viele Datenlieferanten, wie z.B. OpenStreetMap[2] oder NavTeq[3], zu unterstützen. Aus diesem Grund sollte neben der Modifizierbarkeit auch eine gewisse Mächtigkeit vorhanden sein, so dass alle gewünschten Daten abgelegt werden können.

### 2.1.6 Übertragbarkeit

Bei der Übertragbarkeit soll betrachtet werden, wie das Austauschformat in andere Umgebungen eingebettet werden kann. Auch hier ist eine Verwendung von Standards als wesentliche Güte für eine Integration zu sehen. Zum einen soll dazu betrachtet werden, wie die Unterstützung eines Austauschformats bei den verschiedenen GIS ist und zum anderen soll betrachtet werden, wie aufwendig die Integrierbarkeit in eine neue Umgebung ist, in dem z.B. standardisierte Schemata für das Austauschformat vorliegen. Bei der Übertragbarkeit des GIS soll der Aufwand für eine Installation gering sein, in dem z.B. eine betriebsbereite Version zur Verfügung steht. Ein weiterer Vorteil wäre ebenso eine Plattformunabhängigkeit des GIS.

## 2.2 Formate zum Austausch von Geodaten

Die Wahl eines Austauschformats ist relevant für Art und Qualität der Daten. Sie bestimmt, welche Daten von einer Datenquelle übertragen werden können, da für eine Übertragung ein solches Austauschformat eingesetzt wird. Entsprechend ist die Wahl eines GIS abhängig von dem gewählten Austauschformat. Im Folgenden wird eine Auswahl von vektorbasierten Austauschformaten betrachtet. Diese Austauschformate finden in der GDI bei der Übertragung von Geodaten Anwendung. Da diese GDI aus einer Datenquelle, einem GIS und einem Client besteht, werden diese Austauschformate an den verschiedenen Schnittstellen zwischen diesen Komponenten genutzt. Entsprechend wird die folgende Vorstellung eine Zusammenfassung und einen Eindruck über deren Eignung geben.

### 2.2.1 ALK - Automatisierte Liegenschaftskarte

Die Automatisierte Liegenschaftskarte (ALK) ist ein Speicherformat, welches von der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland entwickelt worden ist. Die ALK bildet mit dem Automatisiertem Liegenschaftsbuch (ALB) das Liegenschaftskataster. Entsprechend wird es von den Katasterämtern in Deutschland verwendet um Liegenschaften zu kartographieren und zu beschreiben. Die Realisierung der ALK wurde zu Anfang allein durch die Vermessungsverwaltungen der einzelnen Länder durchgeführt. Erst zu einem späteren Zeitpunkt traten auch kommerzielle Anbieter auf. Bis 2010 soll die ALK mit dem ALB durch das Automatisierte Liegenschaftskatasterinformationssystem (ALKIS) abgelöst werden. Im Gegensatz zum ALK-Standard soll das ALKIS auch Standards der ISO und OGC berücksichtigen [Fri01].

[2] `http://www.openstreetmap.org`
[3] `http://www.navteq.com`

### 2.2.2 DXF - Drawing Interchange Format

Die Firma Autodesk entwickelte im Jahre 1982 das Drawing Interchange Format (DXF) als CAD-Austauschformat für das Programm AutoCAD. DXF wurde vorerst allein als Alternative für das native AutoCAD Format Drawing (DWG) entworfen. Mittlerweile wird DXF jedoch von vielen Programmen, wie Adboe Illustrator, ArcMap, Microsoft Word oder Corel Draw unterstützt. Die Daten werden in einem ASCII-Format gespeichert. Seit Version 10 ist es ebenso möglich die Daten in einem Binärformat zu speichern. Im Gegensatz zu DWG ist DXF frei verwendbar, besitzt jedoch einige Lücken in der Dokumentation [Bar00].

### 2.2.3 GDF - Geographic Data Files

Unter der ISO-Norm 14825:2004 werden die Geographic Data Files (GDF) beschrieben. Aus einer Initiative der Autonavigationsindustrie, zu denen auch Hersteller wie Bosch, Daimler Benz, Tele Atlas oder Volvo gehören, entstand in dem EU-Projekt European Digital Road Map die GDF. GDF ist für die Verwendung in Autonavigationssystemen ausgelegt worden, wird jedoch auch für Fuhrpark-, Logistik- oder Verkehrsmanagement verwendet. GDF-Dateien werden in einem ASCII-Format mit einer festen Zeilenbreite von 80 Zeichen beschrieben. GDF berücksichtigt hierbei drei verschiedene Levels. Im Level 0 werden einfache Geometrien, wie Punkte, Linien und Flächen beschrieben. Im Level 1 befinden sich einfache Features, bei denen die Geometrien mit Attributen verknüpft werden. Diese Features können im Level 2 zu Complex Features zusammengefasst werden. Da es GDF erlaubt über den Header neue Featuretypen hinzuzufügen, gibt es verschiedene GDF-Varianten. Diese Varianten werden vorwiegend von den Kartenlieferanten entwickelt, damit alle eigenen Attribute und Definitionen gespeichert werden können. Auf Grund dieser Variationen kann keine Interoperabilität sichergestellt werden. Dies soll jedoch in der nächsten Version berücksichtigt werden. Ebenso ist eine Version im XML-Format vorgesehen [Fri01].

### 2.2.4 GML - Geography Markup Language

In Zusammenarbeit mit dem Technical Committee 211 der ISO entwickelte das Open Geospatial Consortium (OGC) die Geography Markup Language (GML) als ein offenes Austauschformat für Geodaten. GML basierte ursprünglich auf dem Resource Description Framework (RDF) und ist somit eine XML-Anwendungssprache. Es ist ein frei verfügbares Format, welches für den Datenaustausch von Geoinformationen über Internetinfrastrukturen gedacht ist. Seit GML Version 3.2 besteht eine Normkonformität zu der der ISO 191XX-Familie. Hierbei ist die Spezifikation des OGC und die ISO 19136 identisch. Das GML-Modell stellt eine Reihe von einfachen Funktionen bereit. Zu diesen Primitiven gehören unter anderem die Darstellung eines Koordinatensystems, Geometrien, Features oder Gestaltungsregeln. Diese Primitiven erlauben jedoch eine sehr hohe Erweiterbarkeit und damit auch eine hohe Komplexität. Damit GML trotz der hohen Komplexität dennoch für einfache Probleme anwendbar ist, ist es möglich GML logisch einzuschränken, in dem über ein XML-Schema ein so genanntes GML-Profil vorgegeben wird. So gibt es z.B. ein GML-Profil für Really Simple Syndication (RSS) oder für den Abruf von Geodaten von einem Webdienst. GML gibt jedoch nur eine Richtlinie für ein konkretes anwendungsbezogenes Schema vor, den GML-Anwendungsschemata [Por07].

Ein bekanntes GML-Anwendungsschema ist CityGML. CityGML ist eine GML-Anwendungssprache, welche für die Darstellung von 3D Stadtmodellen entwickelt worden ist. CityGML ist ein OGC Best-Practices Dokument. CityGML berücksichtigt bei der Implementierung fünf Schichten als Level of Details (LoD). In der untersten Schicht, dem LoD0, werden 2,5D Modelle dargestellt, in dem auf einem geometrischen Geländemodell zusätzlich eine Karte oder ein Satellitenfoto gelegt wird. In LoD1 können einfache Blockmodelle modelliert werden, die dann mit dem LoD2 weiter verfeinert werden können. Mit dem LoD3 lassen sich detaillierte Dach- und Fassadenstrukturen sowie Gebäudeöffnungen darstellen. Als oberste Schicht dient der LoD4 um das Gebäudeinnere darzustellen [NH07]. Gemäß ISO 19109 werden auch bei CityGML alle Objekte als geographische Features modelliert. CityGML bietet dazu das Objekt *CityObject* als generalisiertes Feature, von denen sich dann weitere Feature-Klassen, wie *road*, *track*, *railway* oder *building* ableiten lassen. Zusätzlich ist es über eines der vielen Attribute möglich dem Feature eine Funktion zuzuordnen, wie z.B. *national highway*. Des Weiteren besteht die Möglichkeit verschiedene dieser Features zu einem Complex-Feature zusammenzufügen. Hierbei ist es z.B. möglich, dass Features wie z.B. Ampeln, Fußwege, Parkbuchten und Fahrbahnen als ein *TrafficArea* Feature definiert werden, um sie dann zu einem *TransportationComplex* zusammenzufassen. Man erhält so die Möglichkeit gesamte Umgebungen in Beziehungen zu setzen [GKC07].

### 2.2.5 GPX - GPS Exchange Format

Das GPS Exchange Format (GPX) wurde von der Firma TopoGrafix entwickelt um GPS Daten auszutauschen. GPX ist derzeit in der Version 1.1 verfügbar und als eine XML-Anwendung kann sie mit einem XML-Schema validiert werden. Es erlaubt neben Metadaten hauptsächlich Wegpunkte, Routen und Tracklogs zu speichern. Darüber hinaus ist es möglich GPX mit eigenen Elementen zu erweitern. GPX wird hauptsächlich zur Datenerfassung verwendet. Dabei wird mit einem Programm eine bestimmte Strecke zurückgelegt und aufgezeichnet. Die dabei entstehende Route kann zusätzlich mit POIs oder Wegpunkten angereichert werden. Diese Daten dienen dann z.B. als Ausgangsformat zu Generierung von vektorbasierten Karten [Fos04].

### 2.2.6 KML - Keyhole Markup Language

Keyhole Markup Language (KML) wurde ursprünglich als Austauschformat von der Firma Keyhole für den Keyhole Earth Browser entwickelt. Im Jahr 2004 übernahm die Firma Google den Earth Browser, welcher heute unter dem Namen Google Earth bekannt ist. Durch die rasche Verbreitung von Google Earth gewann auch KML an Bedeutung, so dass auch andere Programme eine KML Unterstützung anbieten. Google entwickelte KML daraufhin weiter und ist seit April 2008 mit der Version 2.2 ein OGC Standard [Wil08]. KML ist eine XML-Anwendung, die es erlaubt Features wie Punkte, Linien, Polygone, 3D-Modelle, Beschreibungen und Referenzen zu Bildern zu speichern. Jedem Feature wird dabei eine 3D-Koordinate zugewiesen und kann zusätzlich mit weiteren Daten beliebig angereichert werden. Neben der XML-Darstellung gibt es noch zusätzlich KMZ als eine ZIP-Variante, bei dem zusätzlich auch alle Referenzen mitgespeichert werden. Dies ermöglicht einen besseren Austausch über das Internet. Entgegen der weitläufigen Meinung ist KML keine GML-Anwendungssprache. KML wird vornehmlich für die Visualisierung verwendet und GML zum Speichern von Objekten und deren Eigenschaften. Jedoch ist es möglich KML für die Darstellung von GML-Objekten zu verwenden.

Im Gegensatz dazu ist es ebenso möglich GML in dem Maße zu erweitern, dass es wie KML arbeitet [Wil08].

### 2.2.7 MIF - MapInfo Interchange Format

Für die Software MapInfo wurde das MapInfo Interchange Format (MIF) entwickelt. MIF ist ein im ASCII-Format kodiertes Speicherformat und soll es ermöglichen die in MapInfo genutzten MapInfo-Tables zu speichern. Die Daten werden in zwei separate Dateien abgelegt. Dabei werden die graphischen Daten in der MIF-Datei abgelegt und die Textdaten in einer MID-Datei. Die MID-Datei enthält dabei jeweils ein Objekt pro Zeile. Die MID-Datei ist optional. Falls sie nicht vorhanden ist, wird der Eintrag als leere Zeichenkette interpretiert. Die MIF-Datei besteht aus einem Header und einem Datenteil. Der Header beschreibt, wie aus den Daten wieder eine MapInfo-Table hergestellt werden kann. Der Datenteil beinhaltet hingegen die geographischen Objekte. MIF unterstützt Punkte, Linien, Polygone, Regionen, Bögen, Texte, Rechtecke, gerundete Rechtecke und Ellipsen als Objekte. [Map99].

### 2.2.8 OSM - OpenStreetMap

OpenStreetMap ist ein Projekt mit dem Ziel geographische Daten unter einer freien Lizenz anzubieten. Die Daten werden dabei wie in einem Wiki von Freiwilligen eingefügt. Als Eingabeformat dient hierbei das GPX-Format. Als Ausgabeformat hingegen wurde ein eigenes OpenStreetMap-Format (OSM) entwickelt. OSM ist eine XML-Anwendung und entspricht der Darstellung der OSM-API. Jedes Objekt enthält eine ID, den Ersteller, einen Zeitstempel und eine Reihe von Tags. Tags sind Schlüssel-Wert-Paare, die von den Nutzern selbst erstellt werden. Ein Objekt kann wiederum ein *Node*, *Way* oder *Relation* sein. Ein *Node* ist ein Punkt mit einem Breiten- und einem Längengrad und dient z.B. zur Darstellung von POIs. Ein *Way* beinhaltet eine Liste von *Nodes*. Zusätzlich ist es möglich ein *Way* als einen *Closed Way* zu definieren, bei dem die erste und letzte *Node* die selbe ist. Ein *Closed Way* ist demnach ein Polygon, welches zusätzlich als eine Area definiert werden kann. In dem man z.B. ein implizites Tag wie natural=water setzt, wird des Polygon als eine Wasserfläche definiert. Als weiteren Basistyp steht mit einer *Relation* die Möglichkeit bereit eine oder mehrere Objekte in Beziehung zu stellen und mit zusätzlichen, gemeinsamen Tags anzureichern [Ope08].

### 2.2.9 SHP - Shapefile

Für das GIS ArcView entwickelte die Firma ESRI das Speicherformat Shapefile (SHP), welches sich auf Grund der langjährigen Existenz mittlerweile zu einem Quasi-Standard entwickelt hat. Demnach ist es das Geoformat, dessen Unterstützung von anderen Anbietern am höchsten ist. Ein Shapefile ist ein binärkodiertes Format, welches aus mindestens drei Dateien besteht. Folglich gibt es eine SHP-Datei um Geometrien zu speichern, eine DBF-Datei, welche die Sachdaten und Attribute enthält und eine SHX-Datei, welche die Verknüpfungen zwischen der SHP- und der DBF-Datei beinhaltet. Optional können noch weitere Dateien hinzugefügt werden. Mit diesem Dateien ist es möglich verschiedene Indizes zu beschreiben. Des Weiteren gibt es eine PRJ-Datei, welche die Kartenprojektion enthält. Mit einem Shapefile kann man jeweils nur Features eines Types beschreiben, so dass nur Punkte, Linien, Polygone oder Punktlisten enthalten sein können. Demnach ist es z.B. nicht möglich

Straßen und POIs in einem Shapefile zu speichern. Zusätzlich gibt es noch die Möglichkeit ein *MultiPatch* zu definieren, welche die Beschreibung bestimmter Oberflächeneigenschaften beschreibt. So kann man z.B. ein Polygon als inneren oder äußeren Ring definieren [Env98].

### 2.2.10 SVG - Scalable Vector Graphics

Das vom W3C initiierte Format Scalable Vector Graphics (SVG) ist eine XML-Anwendung, die es erlaubt zweidimensionale Grafiken zu beschreiben. SVG unterscheidet drei unterschiedliche Typen: Rastergrafiken, Text und Vektorgrafiken. Rastergrafiken werden über eine Referenz extern eingebunden. Um nach SVG konform zu sein, muss ein darstellendes Programm mindestens JPEG und PNG als Rasterdatenformate unterstützen. Ein Text kann mit einer bestimmten Schriftart direkt angegeben werden, jedoch setzt dies vorraus, dass die Schriftart bei der Zielplattform vorhanden ist. Eine Vektorgrafik wird aus grafischen Primitiven zusammengesetzt. Als oberstes Grundelement erlaubt SVG die Beschreibung von Pfaden. Aus diesen Pfaden können alle weiteren Primitiven, wie Kreise, Ellipsen, Rechtecke, Linien, Polygone und Polylines aufgebaut werden. Im Gegensatz zu anderen grafischen Vektorformaten erlaubt SVG Interaktion und Dynamik. Dazu bedient sich SVG an Sprachelementen aus dem SMIL-Standard um Animationen zu beschreiben. Neben der Speicherung als Unicode-konformes XML-Dokument mit der Dateiendung .svg erlaubt SVG auch eine Speicherung einer GZIP-komprimierten Version mit der Dateiendung .svgz [PZ04].

## 2.3 Das Geoinformationssystem

Um eine Navigationsanwendung zu entwerfen, ist man auf Daten, wie Koordinaten, Straßen und deren Attribute angewiesen, die einem eine Routenberechnung erlauben. Um solche Geodaten nutzen zu können, müssen diese der Navigationsanwendung in einer geeigneten Form bereitgestellt werden. In einer typischen GDI wird diese Aufgabe von einem GIS übernommen. Folgend wird daher auf die Auswahl eines geeigneten GIS, sowie deren Einbettung in die GDI eingegangen. Dabei werden die Schnittstellen zwischen dem GIS und der Navigationsanwendung betrachtet.

### 2.3.1 Schnittstellenarchitektur zwischen Client und GIS

Als eine wesentliche Eigenschaft einer GDI, ist die Betrachtung der Schnittstelle zwischen einem GIS als Server und der Navigationsanwendung als Client. Hierbei kann man im wesentlichen zwischen zwei Client-Server-Architekturen unterscheiden, eine Rich-Client-Architektur und eine Thin-Client-Architektur. Bei einer Rich-Client-Architektur wird die eigentliche Verarbeitung der Daten von dem Client durchgeführt. Im Gegensatz dazu steht ein Thin-Client, dessen Aufgabe meist lediglich die Präsentation der Daten ist und Verarbeitungen vom Server durchgeführt werden [MS04]. Im Kontext einer GIS-Architektur soll hier zwischen einem Thin- und einem Rich-Client wie folgt unterschieden werden. Die Architektur kann hierbei grob in drei Komponenten eingeteilt werden. Neben dem Client und der eigentlichen Datenhaltung in einer geographischen Datenbank, wie z.B. PostGIS[4] oder Oracle Spatial 11g[5], werden einige Basisfunktionalitäten zur Auswertung und Analyse von Geometrien bereitgestellt. Diese werden als GIS-Toolbox bezeichnet. Eine GIS-Toolbox ermöglicht z.B. den

[4] `http://postgis.refractions.net`
[5] `http://www.oracle.com/technology/products/spatial/index.html`

Vergleich, ob sich zwei geometrische Objekte schneiden [Fri01]. Wie in Abbildung 2.1 verdeutlicht, findet sich diese GIS-Toolbox bei einem Thin-Client als Serverkomponente wieder und entsprechend beim Rich-Client als Teil des Client.

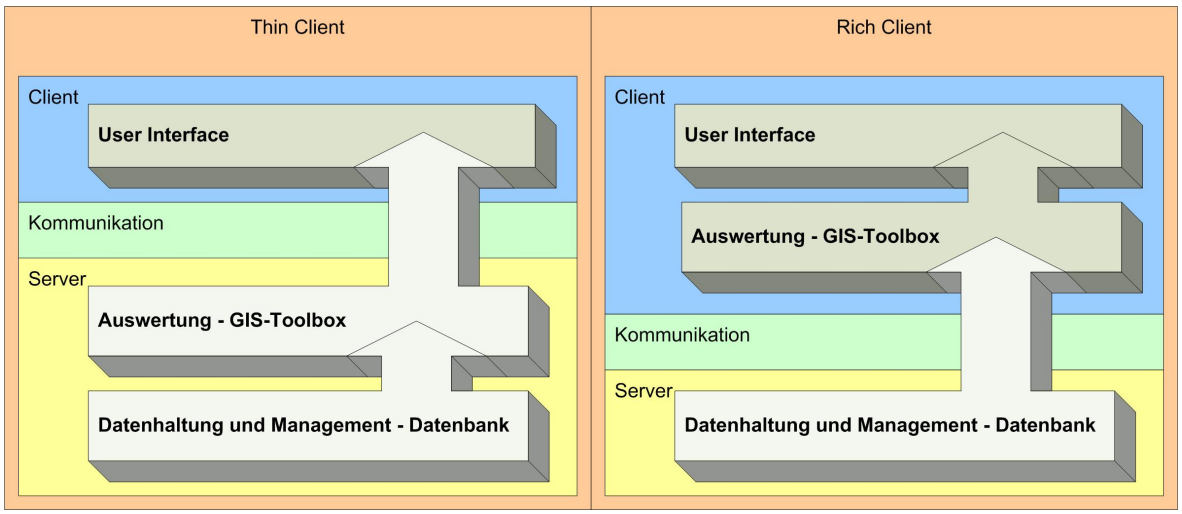


*Abbildung 2.1: Thin und Rich Client in Bezug auf GIS*

Um den Zugriff auf eine solche GIS-Toolbox so zu ermöglichen, dass die Geodaten in Zukunft neben der Navigationsanwendung auch für weitere Anwendungen zur Verfügung steht, soll diese Schnittstelle standardisiert bereitgestellt werden. Des Weiteren soll dadurch eine Interoperabilität gewährleistet werden. Aus diesem Grund ist hier die Verwendung einer standardisierten serviceorientierten Architektur (SOA) sinnvoll.

Im Bereich der GIS entwickelte das OGC die OpenGIS Web Services (OWS), die eine Rahmenarchitektur für eine solche SOA darstellt. Die OWS fassen den Cataloque Service (CAT), den Web Coverage Service (WCS), den Web Feature Service (WFS) und den Web Map Service (WMS) zusammen. Der CAT bietet den Zugriff auf Metadaten und entspricht somit einem Verzeichnis über alle zur Verfügung stehenden Dienste. Der WCS definiert den Zugriff auf Rasterdaten, wobei der WFS hingegen den Zugriff auf Vektordaten erlaubt, wie sie auch in der Datenbank abgelegt werden. Der WMS erlaubt die Generierung von Karten aus den Raster- und Vektordaten und liefert ein Grafikformat wie JPEG oder PNG zurück [GSL07]. Neben dem OWS gibt es noch die eXploration and Mining Markup Language (XMML). XMML ist eine GML-Anwendung, die speziell eine standardisierte Schnittstelle bereitstellen soll. Ebenso wie WFS erlaubt XMML die Abfrage von Features und liefert diese ISO-19107 konform als Punkte, Linien, Polygone oder zusammengesetzte Basistypen wieder. Ebenso soll XMML den OGC Standards folgen, damit eine hohe Interoperabilität gewährleistet werden kann. XMML befindet sich zur Zeit noch in einer Draft-Version und soll in Zukunft als eine offizielle GML-Anwendung von der OGC angenommen werden [Sol08]. Neben OWS und XMML gibt es noch ArcIMS der Firma ESRI als vergleichbare Schnittstelle. Diese Schnittstelle ist nicht frei nutzbar [Env08].

### 2.3.2 Das Geoinformationssystem

Für die Bereitstellung der Geodaten soll ein Server gefunden werden, der die in Abschnitt 2.1 beschriebenen Anforderungen erfüllt. Da der Server hier sowohl die Datenbank selbst, als auch die GIS-Toolbox mit entsprechenden Services zur Abfrage und Modifikation der Daten bereitstellen soll, bezeichnet man dieses ganzheitlich als Geoinformationssystem (GIS). Zur Zeit befinden sich mehrere GIS auf dem Markt. Neben proprietären und kommerziellen GIS, wie ArcGIS[6] oder MapInfo[7], gibt es auch diverse OpenSource-Anwendungen, welche auf Grund der Anforderung, dass hier OpenSource-Lösungen verwendet werden sollen, im Folgenden betrachtet werden.

#### 2.3.2.1 deegree

Das OpenSource Projekt deegree[8] der Firma lat/lon ist ein Java Framework, welches Implementationen verschiedener Webdienste anbietet. deegree bietet alle Services als einzelne Komponenten an, so dass sie je nach Gebrauch installiert werden können. Hierbei dient ein Tomcat-Server als ausführende Anwendung, wobei die Services als einzelne Tomcat-Anwendungen geladen werden können. In der aktuellen Version 2.1 werden WFS, WMS, WCS, CAT, Web Coordinate Transformation Service (WCTS), Web Map Print Service (WMPS) und Web Terrain Service (WTS) unterstützt. Alle diese Services sind OGC konform, wobei WMS und WCS zusätzlich die offiziellen Referenzimplementierungen der OGC sind. Durch die vollständige Implementation in Java ist deegree plattformunabhängig. Als Datenbanken werden Oracle Spatial, PostGIS und weitere JDBC-fähige Datenbanken unterstützt. Des Weiteren ist es möglich Shapefiles und Rasterformate als Datenquellen zu verwenden. Setzt man eine Datenbank ein, so muss diese, wie auch der Tomcat-Server, separat installiert und konfiguriert werden [Mit08].

#### 2.3.2.2 GeoServer

GeoServer wurde vom The Open Planning Project (TOPP)[9] 2001 initiiert und sollte vorerst zur Generierung von Karten eingesetzt werden. Als jedoch die erste Version des WFS veröffentlicht wurde, sollte GeoServer die erste OpenSource-Implementation dieser Spezifikation werden und ist bis heute die vom OGC genannte Referenzimplementierung des WFS. GeoServer ist in Java implementiert und ist auf der GeoTools-Bibliothek[10] aufgebaut. GeoServer verwendet einen Tomcat-Server als Umgebung und ist somit plattformunabhängig. Neben dem WFS unterstützt der GeoServer auch WMS, WCS, CAT, Filter Encoding Specification (FEC), WFS-Transactional (WFS-T) und Styled Layer Description (SLD). Durch WFS-T ist es somit auch möglich Daten über dieser Transaktionschnittstelle zu editieren. Als Ausgabeformat dieser Services werden unter anderem KML, GML, Shapefile, GeoRSS, PDF, GeoJSON, SVG, JPEG, GIF und PNG unterstützt. Als Datenbank können beim GeoServer PostGIS, Oracle Spatial, ArcSDE, DB2 und MySQL eingesetzt werden. Ebenso dienen auch Shapefiles, GeoTIFF und GTOPO30-Dateien als Datenquelle. Das Editieren von Daten wird jedoch nur bei einer Verwendung von PostGIS, Oracle, ArcSDE, DB2 oder Shapefiles unterstützt. GeoServer steht derzeit als Version 1.5 als eine sofort einsetzbares Paket zum Download zur Verfügung, bei

[6] http://www.esri.com/software/arcgis/index.html
[7] http://www.mapinfo.de
[8] http://www.deegree.org
[9] http://topp.openplans.org/
[10] http://geotools.codehaus.org

dem keine zusätzlichen Installationen oder Konfigurationen nötig sind. GeoServer verfügt über eine eigene grafische Benutzerschnittstelle, welche die Verwaltung des Servers ermöglicht [Mit08].

#### 2.3.2.3 GRASS

GRASS[11]steht für Geographic Resources Analysis Support System und befindet sich zur Zeit in der Version 6.2.3. Es wurde ursprünglich von 1982 bis 1995 von der US Army entwickelt, welches später von der Baylor University fortgeführt wurde. Heute wird es von einem internationalen GRASS Development Team entwickelt und ist eines der Projekte, die von der Open Source Geospatial Foundation (OSGeo)[12] unterstützt wird. Es ist ein modular aufgebautes GIS, welches verschiedene Funktionalitäten zur Verwaltung, Modifikation, Analyse, Produktion und Visualisierung von 2D und 3D Vektor- und Rasterdaten zur Verfügung stellt. Zur Zeit existieren über 350 dieser Module. GRASS erlaubt zwar das Importieren von Daten aus einem WFS, bietet jedoch selbst keine OWS an. GRASS benutzt mit dem GRASS-ASCII ein eigenes Dateiformat zur Speicherung der Daten. Alternativ werden auch SQL-basierende Datenbanken unterstützt. GRASS besitzt eine grafische Benutzerschnittstelle, als auch ein Kommandozeilenprogramm. Es ist somit ein Dektop-GIS, welches daher nicht ermöglicht GRASS als Server anzusprechen. Jedoch gibt es einige Projekte, die einen Zugriff über ein Web Processing Service (WPS) implementieren. Das in C geschriebene Projekt muss in Linux/Unix-Umgebungen selbst kompiliert werden und ist abhängig von weiteren Bibliotheken, die vorher installiert sein müssen. Für Windows existiert hingegen eine native Version [NM04][Mit08].

#### 2.3.2.4 Quantum GIS

Quantum GIS (QGIS)[13] ist eine OpenSource-Anwendung, die es erlaubt Geodaten zu betrachten und zu bearbeiten. Unter der Verwendung einer PostGIS-Datenbank ist es ebenso möglich geographische Analysen durchzuführen. Es werden von QGIS verschiedene Datenformate, wie GPX, Shapefiles, GML, MapInfo und die Einbindung von WMS und WFS unterstützt. QGIS selbst ist ein Desktop-GIS und stellt daher keinen Server für Schnittstellen bereit. Es ist jedoch unter anderem mit einem Plug-in möglich den UMN MapServer zu benutzen, um Daten über das Internet bereitzustellen. Ebenso existiert eine Möglichkeit GRASS einzubinden, um Geodaten zu editieren, zu erzeugen oder zu exportieren. Die in C++ entwickelte Anwendung ist unter Linux, Unix, MacOS X, und Windows lauffähig. Hierbei stehen für verschiedene Distributionen bereits native Downloads zur Verfügung [Mit08].

#### 2.3.2.5 UMN MapServer

Der UMN MapServer[14] ist ein von der University of Minnesota (UMN) mit der Unterstützung der NASA entwickeltes GIS. Der UMN MapServer ist nun, wie auch GRASS und Quantum GIS, ein Projekt der OSGeo. Der MapServer ist vergleichbar mit GeoServer. Da der UMN MapServer auf der GDAL/OGR-Bibliothek[15] aufbaut, werden viele Raster- und Vektorformate unterstützt. Hierzu zählen unter anderem als Vektorformate Shapefiles, GML, GeoJSON, KML, MapInfo, GPX und

[11] `http://grass.osgeo.org`
[12] `http://www.osgeo.org`
[13] `http://www.qgis.org`
[14] `http://mapserver.gis.umn.edu`
[15] `http://www.gdal.org`

GRASS-ASCII, sowie verschiedene Datenbanken, wie MySQL, Oracle Spatial, PostGIS oder SQ-Lite. Zusätzlich werden OGC-Standards, wie WMS, WFS, WCS, FEC, SLD und Sensor Observation Service (SOS) zur Steuerung und zum Datenempfang von Sensoren unterstützt. Der UMN MapServer ist unter Linux, Unix, MacOS X und Windows lauffähig. Der UMN MapServer wird über ein Common Gateway Interface (CGI) eingebunden und ist demnach auf einen HTTP-Server wie z.B. Apache angewiesen. Durch die Verwendung der MapScript API ist es möglich den MapServer durch verschiedene Browseroberflächen zu benutzten. So werden derzeit PHP, Python, Perl, Ruby, Java und C-Sharp unterstützt. Durch die Verwendung von GDAL/OGR und der damit vielfältigen Unterstützung verschiedener Datenformate, sowie der Plattformunabhängigkeit wird der UNM MapServer in vielen GDIs als Bindeglied zwischen sehr unterschiedlichen GIS benutzt [Mit08].

### 2.3.3 Der Client

Der Client ist in dieser GDI die Navigationsanwendung. Da die Gesamtarchitektur dem SOA-Gedanken folgt, ist die Navigationsanwendung als ein Dienstnutzer zu sehen, der die bereitgestellten Dienste des GIS in Anspruch nimmt. Entsprechend muss der Client in der Lage sein, den Dienst zu nutzen. Da der Client auf Vektordaten angewiesen ist, ist die Verwendung des WFS sinnvoll. Hierbei muss der Client dem SOA-Paradigma folgen können und somit den WFS finden, binden und ausführen können.

## 2.4 Die Datenquelle

Bei einer Navigationsanwendung ist man auf eine Fülle von Geodaten angewiesen. Diese Geodaten müssen alle Informationen liefern können, die zu einem Routing nötig sind. Hierzu zählen neben den eigentlichen Koordinaten auch Straßennamen, Straßentypen, Wegpunkte und gegebenenfalls Points of Interest (POI). Im Bereich der Geodaten gibt es ein Quasi-Duopol zwischen NAVTEQ und Tele Atlas. NAVTEQ und Tele Atlas sind jedoch kommerzielle Unternehmen, deren Daten nicht kostenlos zur Verfügung stehen. Seit 2004 existiert jedoch OpenStreetMap (OSM) als ein freies OpenSource-Projekt. Hierbei handelt es sich um ein Wiki, bei dem zur Zeit 55.800 Nutzer selbst Daten eingeben und pflegen. Die Daten werden dabei über GPX importiert. Durch eigene Anwendungen wie dem OSMTracker ist es mit einem PDA möglich während der Aufzeichnung die Geodaten mit OSM-spezifischen Annotationen, wie z.B. Geschwindigkeitsbegrenzungen, Straßentyp oder POIs zu ergänzen. Ebenso existiert mit Potlatch ein Flash-basierter Editor, der das direkte Editieren und Hinzufügen von Daten über einen Webbrowser erlaubt. Durch diese Unterstützungen stieg die Anzahl der erfassten Wege innerhalb eines Jahres von 2 Millionen auf über 20 Millionen[16]. Da die Behörden der USA ein frei zugängliches GIS anbieten, konnte bereits die ganze USA in OSM importiert werden. Die aktuellen Aktivitäten beziehen sich daher derzeit auf Mittel- und Westeuropa. In Deutschland sind demzufolge nicht alle Regionen vollständig erfasst. Neben einer API bietet OSM wöchentlich eine Ausgabe der gesamten Daten als Download an. Diese Daten liegen in einem eigenen OSM-Format, wie in Abschnitt 2.2.8 beschrieben, vor. Da dieses Format jedoch nicht von gängigen GIS als Importformat unterstützt wird, ist es nötig, dass die Daten vorher in ein anderes Format konvertiert werden. Da OSM-Dateien eine XML-Anwendung sind, bietet es sich an die Daten in ein GML-Format zu transformieren. Des Weiteren stellt die Firma Geofabrik bereits in Shapefiles konvertierte Dateien von Europa kostenlos zum Download zur Verfügung[17].

[16] `http://wiki.openstreetmap.org/index.php/Stats`
[17] `http://download.geofabrik.de/osm/`

## 2.5 Vergleich und Fazit

Im Folgenden werden die zuvor vorgestellten Komponenten der Geodateninfrastruktur verglichen. Dabei werden die in Abschnitt 2.1 genannten Anforderungen als Bewertungskriterien dienen. Aus diesem Vergleich werden daraufhin die für diese Anwendung geeigneten Komponenten gewählt. Da die einzelnen Komponenten voneinander abhängig sind, wird hier die Auswahl der Komponenten von der Datenquelle ausgehend betrachtet. Dies ist erforderlich, da momentan nur OpenStreetMap als Datenquelle in Frage kommt und somit jene Komponente ist, die keine weiteren Optionen zulässt.

### 2.5.1 Datenquelle

Als Datenquelle ist OpenStreetMap derzeit die einzige Lösung, die eine kostenlose Verwendung erlaubt. Des Weiteren hat OpenStreetMap im Gegensatz zu den Daten von Tele Atlas oder NAVTEQ den Vorteil, dass sie die Anforderung nach einer OpenSource-Lösung erfüllen. Die Verwendung der Daten ist somit auch in Zukunft frei von lizenzrechtlichen Problemen. OpenStreetMap liefert alle nötigen Informationen, die für ein Routing gebraucht werden. Neben den eigentlichen Koordinaten sind unter anderem auch Straßennamen und Straßentyp verfügbar. Der Zugriff auf Daten ist jederzeit über das Archiv oder der API von OpenStreetMap möglich. Zusätzlich existiert zu OpenStreetMap ein Wiki, welches neben der Dokumentation auch Informationen über Eigenschaften und Verwendung der Straßentypen angibt.

### 2.5.2 Austauschformat zwischen Datenquelle und GIS

OpenStreetMap erlaubt als Datenquelle nur den Export von OSM-Dateien als einziges Vektorformat, welches noch alle notwendigen Attribute beinhaltet. Aus diesem Grund muss dieses Format noch zusätzlich in ein anderes Austauschformat konvertiert werden, da OSM-Dateien von keinem der genannten GIS unterstützt wird. Da OSM-Dateien eine XML-Anwendung ist, bietet es sich an die Daten in eine andere XML-Anwendung zu transformieren. Hierbei würden GML, KML und SVG in Frage kommen. Jedoch werden nicht alle Formate von allen GIS als Eingabeformat bedient. So wird KML von keinem der genannten GIS als Eingabeformat unterstützt. SVG wird zwar teilweise unterstützt, jedoch gehen bei der Konvertierung von OSM zu SVG wichtige Eigenschaften, wie z.B. die GPS-Koordinaten, verloren. GML beinhaltet zwar alle Daten, jedoch unterstützt zum einen nicht jedes GIS den Import von GML Dateien und zum anderen ist eine Konvertierung von OSM zu GML mit sehr viel Aufwand verbunden, da es keine passende GML-Anwendung gibt. Eine weitere Alternative zu einer XML-Transformation wäre die Nutzung von Shapefiles. Dieses Format wird als Quasi-Standard von allen gängigen GIS unterstützt. Neben dem Tool osm2shp[18] zum konvertieren der Daten existieren bereits frei verfügbare Shapefiles, welche aus OSM-Daten generiert wurden. Mit Shapefiles wäre somit der Aufwand für die Konvertierung gering und ein Import in jedes GIS möglich.

### 2.5.3 Geoinformationssystem

Die Verwendung von Shapefiles als Eingabeformat verursacht keine Abhängigkeiten gegenüber eines GIS, da alle genannten GIS als Eingabeformat Shapefiles erlauben. Entsprechend wird hierdurch die

[18] `http://code.google.com/p/osm2shp/`

Auswahl nicht beschränkt. Mit der Anforderung, dass das GIS interoperabel integriert werden soll und dabei Standards eingehalten werden sollen, soll das GIS mindestens eine Schnittstelle bereitstellen. Diese sollte über ein Netzwerk zugreifbar sein. Demnach muss das GIS einen Server bereitstellen. Diese Möglichkeit werden nur von deegree, GeoServer und dem UMN MapServer gegeben. GRASS und Quantum GIS sind beides Desktop-Anwendungen und haben keinen Server zur Bereitstellung von Geodaten. deegree ist in der Benutzbarkeit und Übertragbarkeit gegenüber GeoServer und dem UMN MapServer im Nachteil, da man jeden Service separat installieren muss. Außerdem müssen der Tomcat-Server und alle nötigen Bibliotheken per Hand installiert werden. Sowohl GeoServer als auch der UMN MapServer sind in der Lage über eine Schnittstelle alle nötigen Daten anzubieten, deren Benutzung und Konfiguration beiderseits auch ausführlich über ein Wiki dokumentiert werden. Der UMN MapServer unterstützt auf Grund der Verwendung der GDLA/OGR-Bibliothek zwar mehr Formate als der GeoServer, jedoch ist der GeoServer in der Lage über WFS-T auch Transaktionen durchzuführen. Dies erlaubt dem Nutzer das Editieren von Daten über eine standardisierte Schnittstelle. Ebenso ist eine Installation des GeoServers wesentlich einfacher, da er sofort einsatzbereit ist. Der UMN MapServer ist zusätzlich auf einen HTTP-Server wie Apache angewiesen. Einen direkten Vorteil für diese GDI hält weder der GeoServer noch der UMN MapServer vor. Auf Grund der einfacheren Installation, Bedienbarkeit und da der GeoServer von der OGC als Referenzimplementierung für den WFS gilt, wird hier der GeoServer verwendet.

### 2.5.4 Schnittstelle zwischen GIS und Client

Als Ausgangspunkt, dass der GeoServer als GIS verwendet wird, ist die Frage nach einem geeigneten Austauschformat für den Client. Dieses Austauschformat ist abhängig von der Schnittstelle, die das GIS zur Verfügung stellt. Diese Schnittstelle muss in der Lage sein vektorbasierte Daten zu liefern. Obwohl es mit XMML, ArcIMS und den OWS mehrere Architekturen gibt, die eine solche Schnittstelle spezifizieren, ist OWS die einzige Option, da sie auch vom GeoServer angeboten wird. OWS ist nicht nur durch die Abhängigkeit vom GeoServer die bessere Wahl, sondern auch, weil es gegenüber ArcIMS kostenlos ist und gegenüber XMML ein zertifizierter und verbreiteter Standard ist. Obwohl für diese Anwendung Vektordaten entscheidend sind und somit der Schwerpunkt auf die Nutzung des WFS liegt, ist es beim GeoServer ebenso von Vorteil, das mit dem CAT, WMS und WCS die gesamten OWS zur Verfügung stehen. Dies fördert eine höhere Akzeptanz bei Projektpartner, die unter Umständen auch auf Rasterdatenformate angewiesen sind und dann den WCS nutzen können. Ebenso wird damit eine zukünftige Nutzung durch andere Anwendungen gewährleistet und somit die Anforderung nach Skalierbarkeit erfüllt. Des Weiteren entsteht durch Nutzung der OWS die Möglichkeit, dass das GIS beliebig durch ein anderes WFS-fähiges GIS ausgetauscht werden kann. Ebenso ist es möglich, dass der Client je nach Geodaten ein anderes GIS verwendet. Für die Abfrage von Vektordaten greift der Client auf den WFS zu. Der GeoServer bietet beim WFS die Möglichkeit die Daten als GML, Shapefile oder GeoJSON wiederzugeben. Als Austauschformat ist hier GML hervorzuheben, da es sich zum einen XML-Anwendung handelt und damit über entsprechende XML-Parser verarbeitet oder über Bibliotheken serialisiert werden kann. Zum anderem ist GML ein OGC- und ISO-Standard.

### 2.5.5 Fazit

Als Datenquelle ist man hier auf OpenStreetMap angewiesen, da es das einzige Projekt ist, welche Geodaten für Europa kostenlos zur Verfügung stellt. Die Datenbestände von OpenStreetMap sind noch nicht vollständig. Hingegen stehen Daten in größeren urbanen Umgebungen bereits sehr detailliert zur Verfügung. Da jedoch ein großer Datenzuwachs zu verzeichnen ist und es keine vorhandenen Alternativen gibt, fällt die Wahl dennoch auf OpenStreetMap. Die Daten können aus OpenStreetMap nur als OSM-Datei exportiert werden. Da diese Daten aber auch als Shapefile zur freien Verwendung vorlagen, konnte dies kompensiert werden. Ebenso ist die Verwendung von Shapefiles als Eingabeformat sinnvoll, da es sich um einen Quasi-Standard handelt, der von allen GIS unterstützt wird. Die Verwendung von aktuellen Standards aus der Geoinformatik, wie sie von der OGC entwickelt werden, bietet sich an. Durch diese Standards war es möglich eine hohe Interoperabilität dieser GDI zu gewährleisten. Außerdem wird dadurch eine hohe Akzeptanz und Benutzbarkeit bei zukünftigen Nutzern erreicht. Da auch die betrachteten GIS diese Schnittstellen benutzen, war die Wahl eines konkreten GIS nicht eindeutig zu bestimmen. Hier ist der GeoServer als Referenzimplementierung und mit der einfacheren Installation jedoch im Vorteil. Da hier ein WFS als Schnittstelle zum Client angeboten wird, ist es durchaus möglich das GIS beliebig gegen ein anderes WFS-fägiges GIS auszutauschen. Des Weiteren ist der GeoServer über ein WFS-Client in der Lage ein weiteres GIS als Datenquelle einzubinden, in dem der GeoServer die WFS Anfragen tunnelt. Somit könnten z.B. zukünftige Projektpartner ihre Geodaten als WFS bereitstellen und zusätzlich im GeoServer eingebunden werden. Alternativ wäre es auch möglich den WFS beim Client direkt zu nutzen, da auch hier ein Standard eingesetzt wird. Man erhält somit eine lose gekoppelte und interoperable Geodateninfrastruktur, die es erlaubt Komponenten auszutauschen oder hinzuzufügen, ohne das dies andere Komponenten beeinflusst. Dies soll in Abbildung 2.2 dargestellt werden.

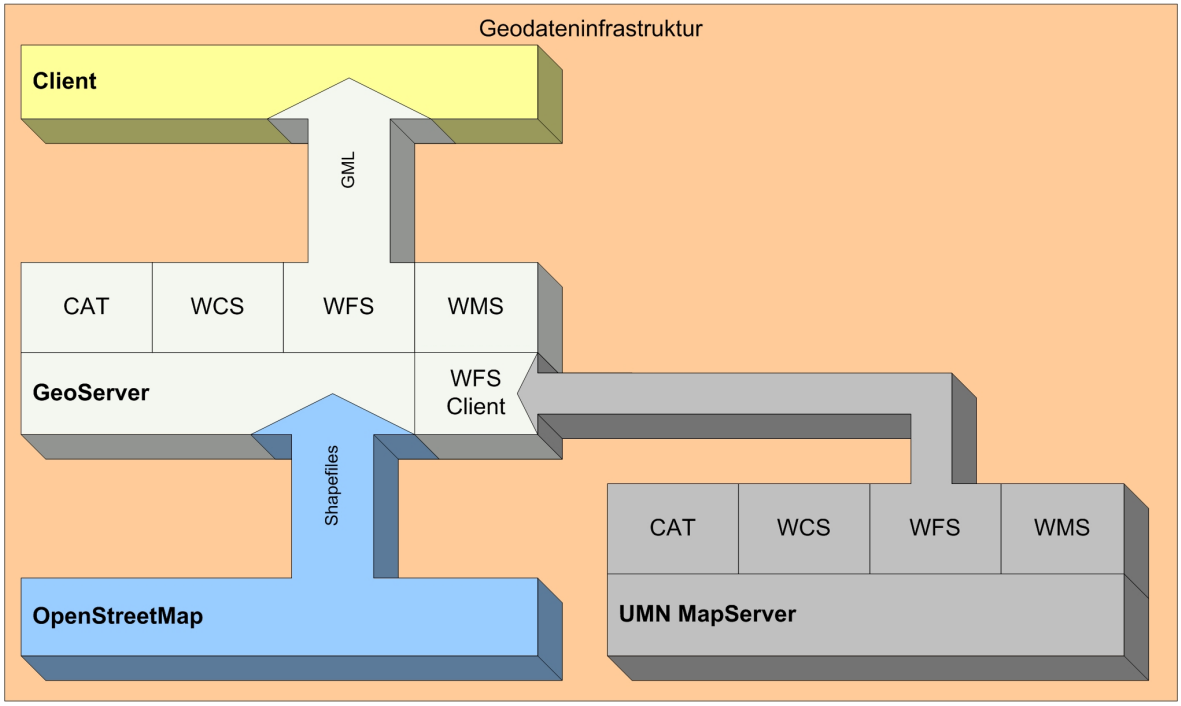


*Abbildung 2.2: Die GDI mit den gewählten Komponenten und zusätzlichem GIS*

# 3 Routing für Fußgänger

Eine Teilaufgabe der Navigation ist es, eine Route zwischen zwei Orten zu finden. An die Berechnung einer Route werden funktionale und nichtfunktionale Anforderungen gestellt. Auf Grund der verbreiteten Verwendung von Navigationsanwendungen für Kraftfahrzeuge eignen sich die üblichen Anforderungen, wie das Bevorzugen von Autobahnen oder Anzeigen von Spurwechseln, nicht für eine Navigation für Fußgänger. Entsprechend wird daher auf Anforderungen für das Routing für Fußgänger eingegangen. Dieses Kapitel schafft die theoretischen Grundlagen, die für die Entwicklung eines Routings für Fußgänger nötig sind. Dazu werden Probleme und Lösungsmöglichkeiten bei der Berechnung der Route betrachtet, die durch die zur Verfügung stehenden Daten und deren Ausgabe beeinflusst werden.

## 3.1 Routenberechnung

Um bei einer Navigation die kürzeste Route auszugeben, muss diese vorher berechnet werden. Für die Berechnung einer Route müssen verschiedene Kriterien betrachtet werden. Neben der Wahl eines geeigneten Algorithmus, der eine Route berechnen kann, spielen Art und Qualität der verfügbaren Daten eine Rolle und wie diese von einem Algorithmus verwendet werden können. Im Folgenden wird daher auf die verfügbaren Routing-Daten, deren Fehler und die Übertragung in einen Graphen eingegangen. Abschließend werden zwei Graphenalgorithmen betrachtet und verglichen, die es erlauben den kürzesten Weg zwischen zwei Knoten zu berechnen.

### 3.1.1 Routing-Daten

Als Grundlage für die Berechnung der Route dienen Straßendaten. Wie in Kapitel 2 beschrieben, werden diese Daten über ein Web Feature Service (WFS) vom Geoinformationssystem (GIS) abgefragt. Als Ergebnis steht der Routing-Berechnung somit eine Menge von Straßen zur Verfügung. Naiv würde man eine Route als eine Liste von Straßen beschreiben: Jägerstraße, Am Schützenplatz, Escherweg. Diese Betrachtung ist jedoch nicht praktikabel, da eine Route nicht zwingend jede Straße von Anfang bis Ende beinhaltet. Aus diesem Grund sind Wegpunkte nötig. Man kann hierbei zwischen zwei Arten von Wegpunkten unterscheiden. Zum einen dienen Wegpunkte um die Richtung der Route anzuzeigen. Zum anderen dienen Wegpunkte als Schnittpunkte zweier oder mehr Straßen. Hierbei spiegeln diese Schnittpunkte dann Kreuzungen oder Übergänge zu einer anderen Straße wider. Diese Wegpunkte müssen nicht separat von dem GIS abgefragt werden, da sie bereits durch die geometrische Darstellung einer Straße vorliegen. Die geometrische Darstellung einer Straße besteht aus einer geordneten Liste von Koordinatenpaaren. Die Qualität dieser Koordinatenpaare ist stark davon abhängig, wie genau die Aufzeichnung einer Straße und ihrer Koordinaten erfolgt. Die verwendeten Daten stammen, wie in Kapitel 2 beschrieben, aus dem freien Projekt OpenStreetMap. Da bei OpenStreetMap die Aufzeichnung der Daten von vielen und nicht professionellen Personen vorgenommen wird, gibt es einige Fehler in den Daten, die berücksichtigt werden müssen. Das Modell von OpenStreetMap geht davon aus, dass zwei Straßen dann miteinander verbunden sind, wenn sie das selbe Koordinatenpaar benutzt. Abbildung 3.1 zeigt hierzu ein Beispiel, bei dem das Koordinatenpaar 3 von beiden Straßen verwendet wird.

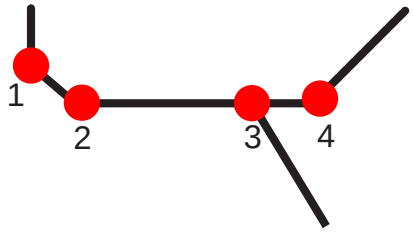


*Abbildung 3.1: Geometrische Darstellung zweier Straßen*

Die Daten von OpenStreetMap sind jedoch nicht durchgehend korrekt angelegt. Hierbei kann man zwei Fehlertypen unterscheiden. Ein Fehlertyp ist in Abbildung 3.2 abgebildet. Hierbei überschneiden sich zwei Straßen, besitzen aber nicht das selbe Koordinatenpaar.

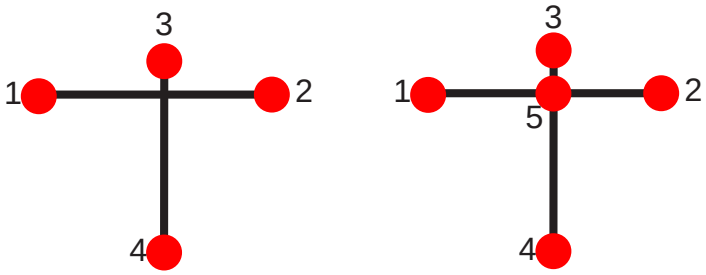


*Abbildung 3.2: Möglicher Fehler durch Überschneidung von Straßen bei OpenStreetMap-Daten*

Ein weiterer Fehlertyp betrachtet eine Straße, die nicht ganz bis an die zweite Straße heranreicht. Auch hier sind die Straßen nicht direkt miteinander verbunden, wie in Abbildung 3.3 zusehen ist. Anhand dieser beiden Fehlertypen kann man zwei Möglichkeiten in Betracht ziehen. Zum einen ist es möglich, dass die beiden Straßen real auch nicht verbunden sind und somit korrekte Daten in OpenStreetMap vorliegen. Zum anderen ist es möglich, dass die Straßen in der Realität verbunden sind, die Daten aber nicht korrekt in OpenStreetMap angelegt wurden. Bei Betrachtung des Fehlers mit Überschneidung würde man darauf schließen, dass die beiden Straßen miteinander verbunden sind, weil sie sich überschneiden und lediglich kein korrespondierender Wegpunkt zu der zweiten Straße angelegt wurde. Diese Beobachtung ist jedoch falsch. So kann z.B. eine Straße ein Tunnel sein, der unter der anderen Straße führt oder es kann entsprechend eine Brücke sein. Aus diesem Grund darf bei der Abfrage des Web Feature Services auch kein Filter eingesetzt, der nur solche Straßen wiedergibt, die sich geometrisch überschneiden um eventuell unerreichbare Straßen aus der Ergebnismenge zu entfernen. Eine automatische Korrektur der Daten bei der man den Schnittpunkt zweier Straßen ergänzt, wäre somit nicht möglich, da der Fehlertyp nicht eindeutig ist. Bei Fehlern mit

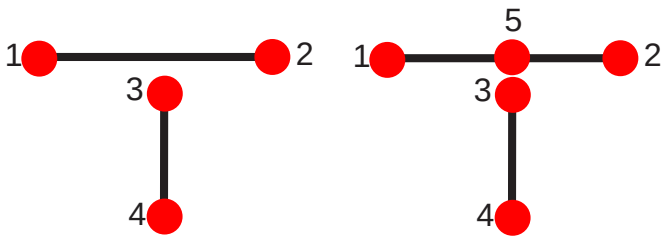


*Abbildung 3.3: Möglicher Fehler durch Berührung von Straßen bei OpenStreetMap-Daten*

Berührung könnte die Vermutung zulassen, dass die Straße bei der Dateneingabe ungenau angelegt wurde und daher Wegpunkt 3 nicht ganz an die zweite Straße heranreicht. Davon kann jedoch nicht explizit ausgegangen werden. Hier ist es z.B. möglich, dass eine Mauer zwischen den beiden Straßen existiert. Auch hier kann man keine automatische Korrektur der Daten durchführen.

Da eine automatische Korrektur nicht möglich ist, da ein Fehlertyp nicht eindeutig als Fehler bestimmt werden kann, muss für die Routenberechnung die Annahme getroffen werden, das OpenStreetMap stets korrekte Daten liefert. Dies bedeutet, dass zwei Straßen nur dann verbunden sind, wenn sie exakt den selben Wegpunkt benutzen. Obwohl OpenStreetMap versucht existierende Fehler zu beheben, muss entsprechend berücksichtigt werden, dass auf Grund der möglichen Fehler in den Daten nicht immer der optimale bzw. kürzeste Weg berechnet werden kann.

### 3.1.2 Routing-Algorithmen

Die Berechnung einer Route versucht neben einer Verbindung von Startpunkt zu Zielpunkt auch verschiedene Kriterien zu erfüllen. Bei Navigationsanwendungen ist dies meist, die schnellste oder kürzeste Route zu finden. Die kürzeste Route berechnet sich aus der Länge der Wege. Die schnellste Route berechnet sich bei einer Routenführung für Fahrzeuge aus der erlaubten Geschwindigkeit und der Länge des Weges [EGW05]. Da sich ein Fußgänger immer mit der selben Geschwindigkeit fortbewegt, ist hier der kürzeste gleich dem schnellsten Weg. Die Anforderung an ein Algorithmus besteht demnach darin den kürzesten Weg zwischen zwei Punkten zu finden. In der Mathematik bezeichnet man dieses Problem als ”Berechnung kürzester Weg in einem Graphen“. Dabei wird das Straßennetz als ein ungerichteter Graph betrachtet, dessen Kanten aus den Straßen bestehen und die Knoten die Kreuzungen zwischen den Straßen widerspiegeln. In der Graphentheorie wird das Problem der kürzesten Wege in drei Kategorien unterschieden [Sch01]:

- **Single Pair Shortest Path Problem (SPSP):** Zu einem Startknoten wird der kürzeste Weg zu einem Zielknoten gesucht.

- **Single Source Shortest Path Problem (SSSP):** Zu einem Startknoten werden die kürzesten Wege zu allen Knoten des Graphen gesucht.

- **All Pairs Shortest Path Problem (APSP):** Zwischen allen Knotenpaaren werden die kürzesten Wege gesucht.

Die Berechnung einer Route ist ein SPSP-Problem, da der kürzeste Weg von einem Startpunkt zu einem Zielpunkt gefunden werden soll. Obwohl dies ein SPSP-Problem ist, wird hier das SSSP-Problem betrachtet, da zurzeit kein Algorithmus bekannt ist, der das SPSP-Problem in einer geringeren Zeit als ein SSSP-Algorithmus löst [Sch01]. Um ein SSSP-Problem zu lösen, muss ein Kantengewicht, meist die Länge oder Entfernung, vorliegen. Zwei Algorithmen, die das SSSP-Problem lösen sind der Dijkstra- und der A*-Algorithmus.

Als Grundlage dieser Algorithmen dient ein Graph. Dazu muss das vorliegende Straßennetz in einen Graphen überführt werden. Da die Wegpunkte einer Straße genutzt werden um den Fußgänger zu führen, werden diese als Knoten abgebildet. Sind zwei Wegpunkte durch eine Straße verbunden, so wird dies durch eine Kante im Graphen dargestellt. Abbildung 3.4 zeigt eine Überführung eines Kartenausschnitts in einen Graphen. Der Graph ist ungerichtet, da ein Fußgänger immer in die entgegengesetzte Richtung gehen kann. Dies gilt z.B. auch für Einbahnstraßen, bei dem ein KFZ nur in eine Richtung fahren dürfte.

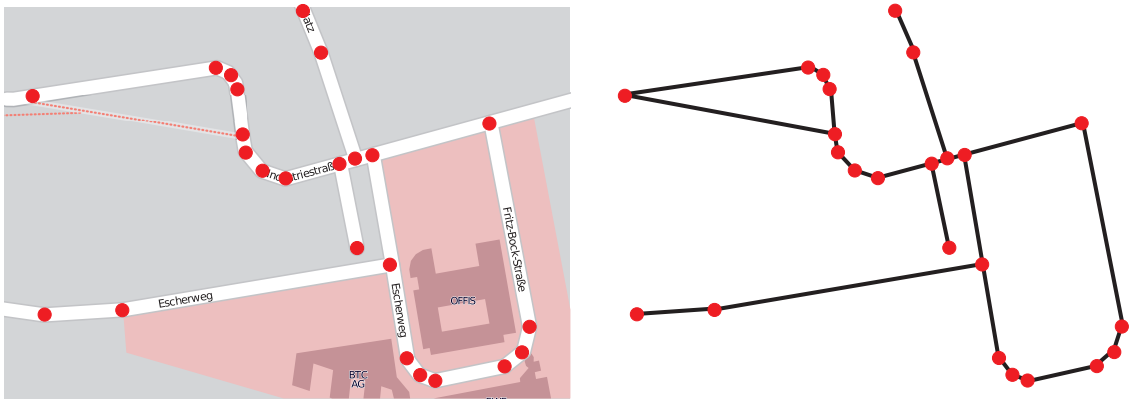


*Abbildung 3.4: Kartenausschnitt mit allen Wegpunkten und der zugehörige Graph*

Da der kürzeste Weg berechnet werden soll, braucht ein SSSP-Algorithmus ein Kantengewicht. Dieses Kantengewicht entspricht der Entfernung zwischen zwei Wegpunkten. In Abbildung 3.5 sind die Kanten entsprechend gewichtet[1]. Der in Abbildung 3.5 dargestellte Graph kann nun als Eingabe für einen SSSP-Algorithmus verwendet werden. Wie man dem Graphen entnehmen kann besteht dieser bei diesem Kartenausschnitt bereits aus 24 Knoten und 25 Kanten. Da Geschwindigkeit und Speicherverbrauch von SSSP-Algorithmen abhängig von der Anzahl der Knoten sind, bietet es sich an den Graph durch Zusammenfassen von Knoten zu optimieren. Dabei müssen vier Voraussetzungen betrachtet werden.

1. Es dürfen keine Wegpunkte gelöscht werden, die als Start- oder Endpunkt einer Route dienen. Diese Wegpunkte müssen erhalten bleiben, da ein SSSP-Algorithmus diese als Eingabe benötigen und diese im Graph vorhanden sein müssen.

[1] Die Streckenlängen sind Näherungswerte

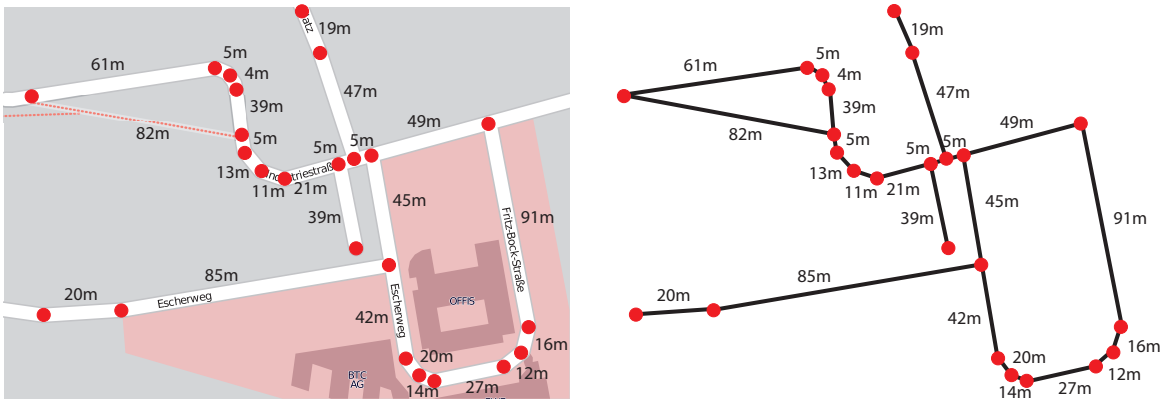


*Abbildung 3.5: Kartenausschnitt und Graph mit Kantengewichten*

2. Es dürfen keine Wegpunkte gelöscht werden, die als Kreuzung dienen. Diese Eigenschaft ist nötig, damit Verknüpfungen zwischen verschiedenen Straßen erhalten bleiben. Zusätzlich wäre das Zusammenfassen von Wegpunkten, die mehr als ein Nachfolger haben nicht deterministisch.

3. Das Kantengewicht muss erhalten bleiben. Das Kantengewicht darf sich nicht durch zusammenfassen der Wegpunkte verändern, sonst würde der berechnete kürzeste Weg nicht mit dem tatsächlich kürzesten Weg übereinstimmen.

4. Es darf nur eine Kante zwischen zwei Wegpunkten geben. Gibt es mehr als eine Kante zwischen zwei Punkten, so ist der Weg nicht mehr deterministisch.

Damit die erste Voraussetzung erfüllt wird, stehen zwei Möglichkeiten zur Verfügung. Bei der ersten Variante werden Start- und Endpunkte nicht gelöscht. Dies impliziert jedoch, dass der Graph erst nach einer Routeneingabe erstellt wird, da erst zu diesem Zeitpunkt Start- und Endpunkt bekannt sind. Eine weitere Variante wäre das Löschen von Start- und Endpunkt. Somit ist der Graph unabhängig von Start- und Zielpunkt und kann schon vor einer Routeneingabe berechnet werden. Damit bei einer Routenberechnung Start und Ziel bestimmt werden kann, muss zusätzlich der Weg zwischen dem Start- bzw. Endpunkt und dem nächstgelegenen zusammengefassten Knoten im Graphen berechnet werden. Die zweite Voraussetzung kann durch die Bedingung erfüllt werden, in dem ein Wegpunkt nicht gelöscht wird, wenn er mindestens zu zwei Straßen gehört. Bei der dritten Voraussetzungen muss das neue Kantengewicht durch die Summe der zusammengefassten Kantengewichte bestimmt werden. Tritt bei der Zusammenfassung auf, dass es bereits eine Kante zwischen den beiden Wegpunkten gibt, so wird die Kante mit der geringeren Distanz verwendet, so dass die vierte Voraussetzung gegeben ist. Unter Verwendung der zweiten Variante bei Voraussetzung 1 zeigt Abbildung 3.6 die zusammengefassten Wegpunkte. Wie der Abbildung 3.6 entnommen werden kann, werden nun 11 statt 24 Knoten genutzt, welches bereits in diesem Ausschnitt eine Optimierung darstellt. Durch die Bedingung, dass ein Wegpunkt nicht gelöscht werden darf, wenn er zu mehr als einer Straße gehört, wurden nicht nur Kreuzungen erhalten, sondern auch Übergänge von einer Straße zur nächsten Straße, wie man Abbildung 3.6 anhand des Beispiels Escherweg und Fritz-Bock-Straße entnehmen kann. Der nun vorliegende Graph kann als Eingabe des SSSP-Algorithmus dienen. Als zusätzliche Eingabe benötigt ein SSSP-Algorithmus einen Start- und Endpunkt. Zur besseren Visualisierung werden die

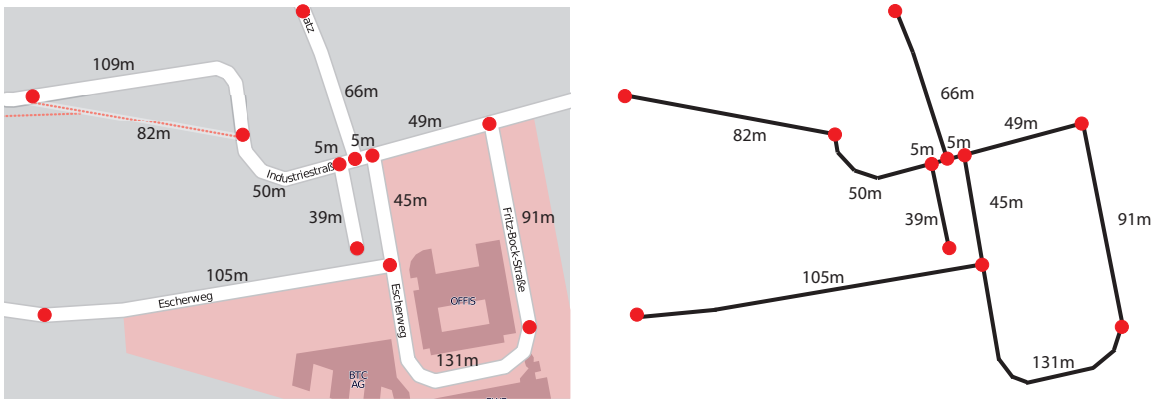


*Abbildung 3.6: Kartenausschnitt und Graph mit zusammengefassten Kanten*

Knoten eines Graphen nummeriert und absteigend dargestellt. Der oberste Knoten ist der Startknoten. Dies wird in Abbildung 3.7 dargestellt. Gesucht wird in diesem Beispiel der kürzeste Weg von Kno-

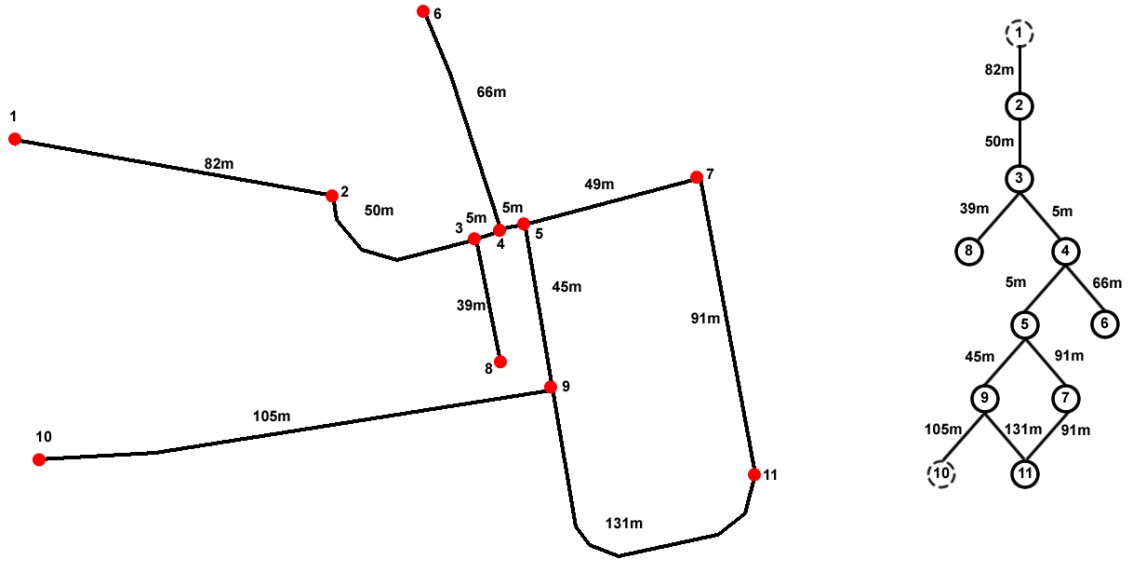


*Abbildung 3.7: Graph des Kartenausschnitts und zugehörigem vereinfachten Graphen*

ten 1 zu Knoten 10. Eine mögliche Strategie, die von verschiedenen Graphenalgorithmen eingesetzt wird, ist *Traversierung durch Färbung*. Zur vereinfachten Anschauung bekommt jeder Knoten eine Farbe zugewiesen. Die Färbung gibt Aufschluss über den Zustand der Knoten. Man betrachtet dabei drei Färbungen:

- **Schwarz:** Alle Knoten die bereits untersucht wurden.
- **Grau:** Alle Knoten, die auf einen Besuch warten bzw. gerade betrachtet werden.
- **Weiß:** Alle übrigen Knoten.

Zu Anfang sind alle Knoten weiß. Bei der Traversierung durch Färbung wird nun ein Knoten aus der weißen Menge genommen und als grau markiert und auf seine Eigenschaften untersucht und je nach Algorithmus modifiziert. Nachdem der Knoten betrachtet wurde wird er zu der schwarzen Menge hinzugefügt. Nacheinander werden nun alle Knoten der weißen Menge gegebenenfalls über den grauen Zustand in die schwarze Menge überführt. Ein Algorithmus ist beendet, wenn es keine weiße Knoten mehr gibt. Die schwarze Menge enthält nun alle traversierten Knoten. Zwei Algorithmen, die diese Traversierung durch Färbung benutzen, sind der Dijkstra- und der A*-Algorithmus [Pep].

#### 3.1.2.1 Der Dijkstra-Algorithmus

Der Algorithmus von Dijkstra wurde 1959 veröffentlicht und löst das Single Source Shortest Path Problem. Dabei betrachtet Dijkstra positive Kantengewichte, die als Entfernung interpretiert werden. Zentrale Idee von Dijkstra war es den Graphen in einen neuen Graphen zu überführen, in dem man nacheinander immer die aktuelle Entfernung zum Startknoten mit den Entfernungen der Nachbarknoten vergleicht und als nächsten Knoten den mit der minimalen Entfernung wählt. Der Dijkstra-Algorithmus gehört somit zu der Klasse der Greedy-Algorithmen, die sich dadurch auszeichnen, dass sie zu einem Zeitpunkt den Folgezustand wählen, der wahrscheinlich den größten Gewinn bringt [Sch01]. Der Dijkstra-Algorithmus gliedert sich in folgende Schritte auf:

- Schritt 1: Alle Knoten werden in die weiße Menge aufgenommen und bekommen ihre momentane Entfernungsschätzung. Diese Entfernungsschätzung ist für den Startknoten 0 und für alle anderen Knoten $\infty$.

- Schritt 2: Es wird der Knoten aus der weißen Menge ausgewählt der die kleinste Distanz hat (y). Beim ersten Durchlauf ist dies der Startknoten (x) selbst. Dieser markierte Knoten y wird nun der schwarzen Menge hinzugefügt und aus der weißen Menge entfernt. Die weiße Menge kann z.B. als Prioritätsliste gesehen werden.

- Schritt 3: Alle Nachbarknoten z von y werden relaxiert. Das heißt, dass die Summe der Distanz vom Startknoten x nach y und die Distanz von y nach z berechnet wird und dem Knoten z in der weißen Menge hinzugefügt wird.

- Schritt 4: Wiederhole Schritt 2 und 3, bis es keinen Knoten mehr in der weißen Menge gibt. In diesem Schritt ist anzumerken, dass der Dijkstra-Algorithmus das SSSP-Problem löst und formal die Distanz zu allen Knoten berechnet. Im Anwendungskontext einer Routenberechnung, ist es jedoch sinnvoll, den Algorithmus abzubrechen, sobald der Endknoten als markierter Knoten y untersucht werden soll, da bereits hier der kürzeste Pfad zwischen Start- und Endknoten vorliegt und die Entfernungen zu weiteren Knoten irrelevant ist [Pep][Sch01].

Für das im vorigen Abschnitt genannte Beispiel ist Knoten 1 der Startknoten und bekommt im ersten Schritt die Distanz 0. Alle anderen Knoten bekommen als Distanz $\infty$, wie der linke Graph in Abbildung 3.8 verdeutlicht. Als erster Knoten wird Knoten 1 markiert, da er mit 0 die geringste Distanz hat. Knoten 1 hat mit Knoten 2 ein Nachbarknoten. Da hier der Abstand von Knoten 1 zu Knoten 2 mit 82m kleiner als die Entfernungsschätzung $\infty$ von Knoten 2 ist, wird dem Knoten 2 als Distanz zum Startknoten 82m hinzugefügt. Dies wird im mittleren Graph von Abbildung 3.8 dargestellt. Im nächsten Schritt wird wieder jener Knoten aus der weißen Menge gewählt, der die geringste Distanz

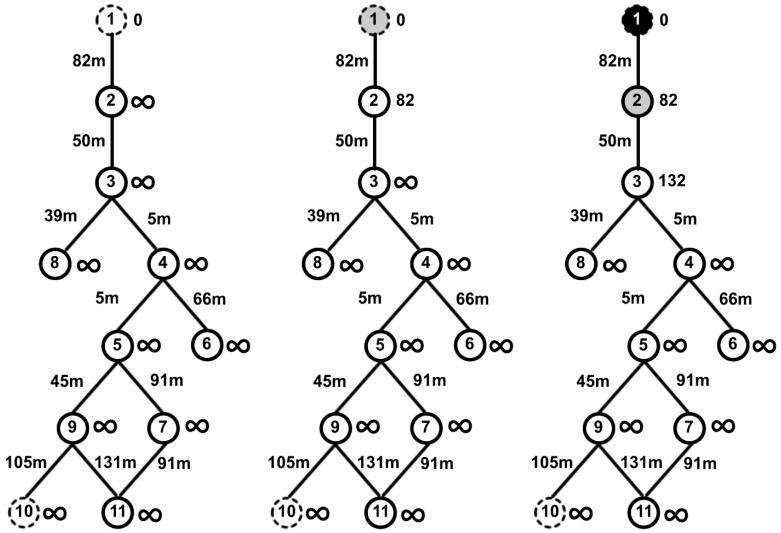


*Abbildung 3.8: Schritt 1 bis Schritt 3 des Dijkstra-Algorithmus*

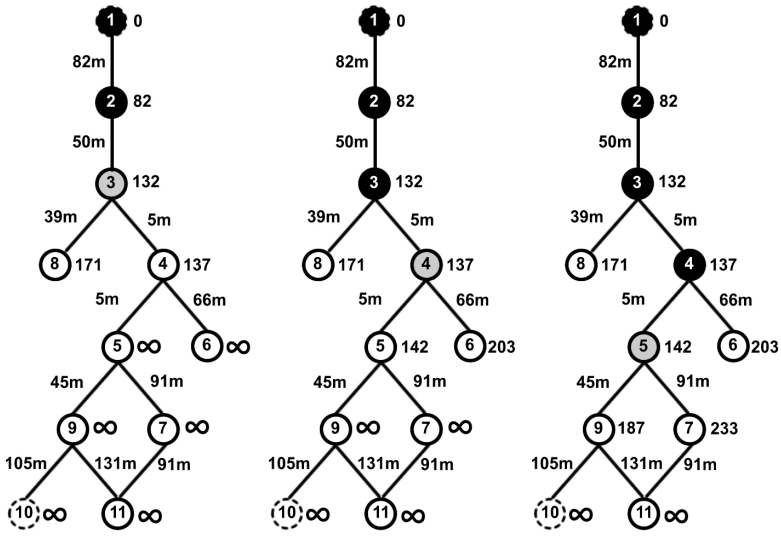


*Abbildung 3.9: Wiederholung von Schritt 2 bis Schritt 3 für Knoten 3, 4 und 5*

aufweist. Dies ist Knoten 2 mit einer Distanz von 82m. Der rechte Graph in Abbildung 3.8 zeigt, dass nun Knoten 3 als Nachbarknoten von Knoten 2 relaxiert wird und die Distanz $82 + 50 = 132$ erhält.

Wiederum ist nun Knoten 3 der Knoten mit der minimalen Distanz. Die Nachbarknoten 8 und 4 von Knoten 3 werden somit relaxiert. In der nach Distanz geordneten Prioritätsliste $(4[137], 8[171], 5[\infty], 6[\infty], ...)$ steht nun Knoten 4 am Anfang und wird somit zunächst betrachtet, wie auch Abbildung 3.9 im mittleren Graph zeigt. Daraufhin werden Knoten 5 und 6 relaxiert. Knoten 5 ist jener mit minimaler Distanz und wird als nächstes markiert und betrachtet.

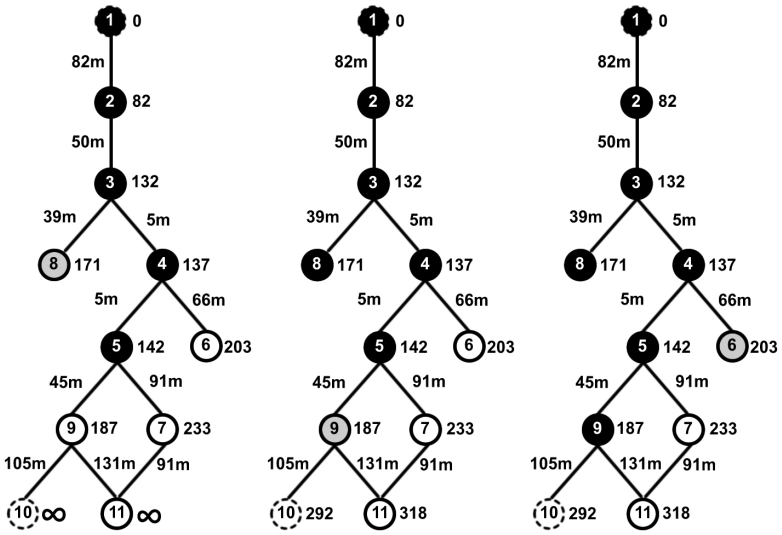


*Abbildung 3.10: Wiederholung von Schritt 2 bis Schritt 3 für Knoten 8, 9 und 6*

Die Prioritätsliste $(8[171], 9[187], 6[203], 7[233], 10[\infty], 11[\infty])$ zeigt nun, dass Knoten 8 als nächstes betrachtet wird. Dieser hat keine Nachbarknoten in der weißen Menge. Als nächstes wird demnach Knoten 9 betrachtet und Knoten 10 und Knoten 11 werden relaxiert, so dass man die Prioritätsliste $(6[203], 7[233], 10[292], 11[318])$ und den mittleren Graph von Abbildung 3.10 erhält. Folglich wird Knoten 6 betrachtet, der keine Nachbarknoten besitzt.

Der linke Graph in Abbildung 3.11 zeigt, dass nun Knoten 7 betrachtet wird. Knoten 7 hat zwar mit Knoten 11 einen Nachbarknoten in der weißen Menge, jedoch ist die Distanz über Knoten 7 mit $233 + 91 = 324$ geringer als die bei Knoten 11 vorhandene Distanz von 318m. Somit wird Knoten 11 nicht neu relaxiert. In der Prioritätsliste befinden sich nun noch $(10[292], 11[318])$. Somit wird nun Knoten 10 betrachtet. Dies ist der Endknoten und der Algorithmus kann hier abgebrochen werden. Das Ergebnis des Dijkstra-Algorithmus ist ein Baum, bei dem zu jedem Knoten der kürzeste Weg zur Wurzel bzw. zum Startknoten ermittelt werden kann. Um nun von einem Endknoten den kürzesten Weg zu bestimmen wandert man ausgehend von Endknoten den Baum hinauf bis zur Wurzel. Der bestrittene Pfad, der aus den besuchten Knoten bzw. Kanten besteht, ergibt den kürzesten Weg vom Start- zum Endknoten.

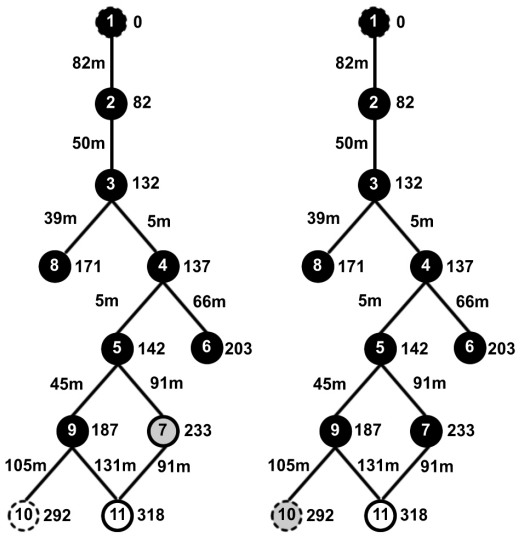


*Abbildung 3.11: Wiederholung von Schritt 2 bis Schritt 3 für Knoten 10. Endknoten gefunden.*

Bei einer Verwendung einer einfachen Liste zur Verwaltung der weißen Nachbarknoten beträgt die Zeitkomplexität $\mathcal{O}(n^2+m)$, wobei $n$ der Anzahl der Kanten und $m$ der Anzahl der Knoten entspricht. Die Komplexität verringert sich durch Verwendung eines Heaps auf $\mathcal{O}(n \cdot \log(n) + m)$ [Sch01].

#### 3.1.2.2 Der A*-Algorithmus

Der A*-Algorithmus ist eine Erweiterung des Dijkstra-Algorithmus und wurde 1968 von Hart, Nilsson und Raphael in [HNR68] veröffentlicht. Der A*-Algorithmus gehört zu den heuristischen Algorithmen, da eine Schätzfunktion verwendet wird. Die Heuristik ersetzt die Entfernungsschätzung des Dijkstra-Algorithmus. Wie beim Dijkstra-Algorithmus beschrieben wird diese verwendet um die Prioritätsliste zu sortieren. Werden bei dem Dijkstra-Algorithmus die Nachbarknoten relaxiert (vgl. Schritt 3 des Dijkstra-Algorithmus), dann wird lediglich die Distanz zum Startknoten berechnet. Diese Berechnung wird bei dem A*-Algorithmus durch eine Funktion $f(x) = g(x) + h(x)$ ersetzt [HNR68]. Dies bedeutet, dass für einen Knoten x die Entfernungsschätzung $f(x)$ durch die Summe der bisherigen Distanz $g(x)$ und einer Heuristik $h(x)$ gegeben ist. $g(x)$ entspricht der Distanz vom Vorgängerknoten zum Startknoten zuzüglich der Distanz der aktuellen Kante. Dies entspricht genau dem Dijkstra-Algorithmus und im Vergleich wäre bei dem Dijkstra-Algorithmus $h(x) = 0$ [Sch01]. Die Heuristik $h(x)$ gibt die geschätzten Kosten an, die zum erreichen des Zielknotens benötigt werden. Bei einer Routenberechnung könnte dies z.B. die Luftlinie zwischen Knoten und Zielknoten sein [HNR68]. Die Heuristik darf jedoch nicht beliebig gewählt werden. Eine Heuristik für den A*-Algorithmus muss entweder zulässig oder monoton sein.

- **Zulässige Heuristik:** Ein Heuristik heißt zulässig, wenn die Kosten nie überschätzt werden. Dies bedeutet, dass die geschätzten Kosten von einem Knoten x zum Zielknoten nie größer sein dürfen,

als die tatsächlichen Kosten. Diese Eigenschaft wird unter anderem durch die Luftlinie gegeben, da eine die tatsächliche Distanz zum Ziel nie geringer sein kann als die direkte Luftlinie.

- **Monotone Heuristik:** Eine monotone Heuristik ergänzt die notwendige Bedingung der Zulässigkeit mit der hinreichenden Bedingung, dass die Heuristik immer kleiner sein muss, als die Summe der Distanz zum Nachbarknoten und der Heuristik des Nachbarsknoten. Für einen Knoten x und seinem Nachbarknoten y muss stets $h(x) \leq d(x, y) + h(y)$ gelten, wobei $d(x, y)$ die Kosten von x nach y angibt. Hier wäre die Luftlinie zwischen zwei Orten auch eine monotone Heuristik.[Sch01][HNR68]

Die Zeitkomplexität ist abhängig von der gewählten Heuristik. Im schlechtesten Fall liegt keine Heuristik vor, welches dann dem Dijkstra-Algorithmus entspricht und somit auch eine Zeitkomplexität von $\mathcal{O}(n \cdot \log(n) + m)$ entspricht. Bei der besten Heuristik liegt der Aufwand nach [Sch01] bei $\mathcal{O}(n + m)$.

#### 3.1.2.3 Bewertung der Algorithmen

Sofern eine Verwendung einer Heuristik möglich ist, ist der A*-Algorithmus in der Zeitkomplexität schneller als der Dijkstra-Algorithmus. Bei einer Vergleich des Wegenetzgraphen von Bonn errechneten [PSK04] bei ca. 30.000 Knoten und 60.000 Kanten, dass der A*-Algorithmus fünf- bis sechsfach schneller ist als der Dijkstra-Algorithmus. Die Implementierung des A*-Algorithmus unterscheidet sich lediglich durch die Erweiterung der Heuristik von dem Dijkstra-Algorithmus. Bei der Verwendung einer Heuristik muss diese zusätzlich für jeden betrachteten Knoten berechnet werden, welches im Vergleich zum Dijkstra-Algorithmus einen höheren Rechenaufwand bedeutet. Wählt man z.B. die Luftlinie mittels euklidischen Abstand als Heuristik, so ist der A*-Algorithmus laut [HNR68] dennoch schneller, da auf Grund der Heuristik wesentlich weniger Knoten besucht werden und damit auch für weniger Knoten eine Heuristik berechnet werden muss. Beim Dijkstra-Algorithmus muss das Ziel vorher nicht bekannt sein, da die Auswahl der Knoten anhand der Restkosten geschieht. Der A*-Algorithmus muss hingegen das Ziel kennen, da es für die Heuristik notwendig ist. Dijkstra- und A*-Algorithmus haben den Nachteil, dass sie alle Knoten im Speicher halten müssen. Es gibt weitere auf dem A*-Algorithmus basierende Algorithmen, wie z.B. IDA* oder MA*, die versuchen dieses Speicherproblem zu lösen.

## 3.2 Routenführung

Fußgänger können sich frei, auch außerhalb des Straßennetzes, bewegen. Die Routenführung, bei dem der Nutzer auf der berechneten Route gehalten werden soll, stellt somit eine besondere Aufgabe dar. Ein Fußgänger befindet sich jedoch nicht zwangsweise auf einem bekannten Wegpunkt, wenn er eine Routenführung starten möchte, so dass hier eine Verknüpfung zwischen Standort und nächstem Wegpunkt geschaffen werden muss. Ebenso kann es durch Messfehlern vorkommen, dass sich ein Fußgänger nicht exakt auf der angegebenen Route bewegt, so dass hier ein Map Matching eingesetzt werden muss. Hierbei spielt auch das Erreichen des nächsten Wegpunktes eine wesentliche Rolle, welches dann unter Umständen auch eine Neuberechnung der Route auslösen kann. Im Folgenden werden vier Problemebereiche betrachtet, die sich mit den genannten Problemen bei der Routenführung für Fußgänger beschäftigen.

### 3.2.1 Festlegen des Startknotens

Wie in Kapitel 3.1 beschrieben, wird für die Berechnung einer Route ein Graph aus den vorhandenen Wegpunkte verwendet. Um eine Route auf diesem Graphen zu berechnen, muss der Startknoten im Graphen vorhanden sein. Dieser Startknoten entspricht der aktuellen Position. In der Praxis ist die Wahrscheinlichkeit gering, dass die aktuelle Position exakt mit einem bekannten Wegpunkt des Straßennetzes übereinstimmt. Wie in Abbildung 3.12 verdeutlicht, kann sich der Benutzer z.B. abseits des Straßennetzes befinden, oder er befindet sich zwar auf einer Straße, steht jedoch etwas vom nächsten Wegpunkt entfernt.

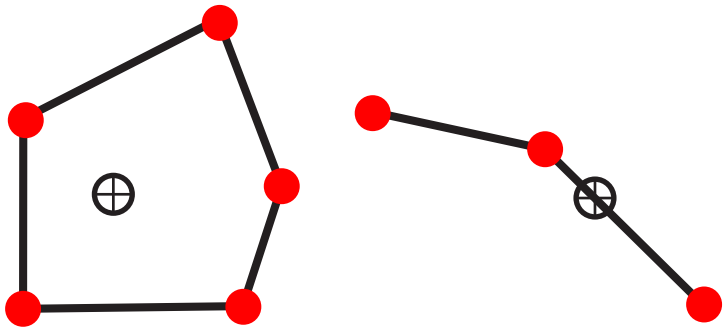

*Abbildung 3.12: Mögliche Positionen einer Person im Straßennetz*

Um nun eine Route berechnen zu können, ist es nötig, dass die aktuelle Position als Wegpunkt zum Graphen hinzugefügt wird. Dazu muss der aktuelle Wegpunkt mit einem bekannten Wegpunkt verknüpft werden. Hierbei entsteht die Frage, welches der beste bekannte Wegpunkt ist, mit dem der aktuelle Wegpunkt verknüpft werden kann. Wie Abbildung 3.13 zeigt, ist der nächstgelegene Wegpunkt nicht zwingend die beste Wahl.

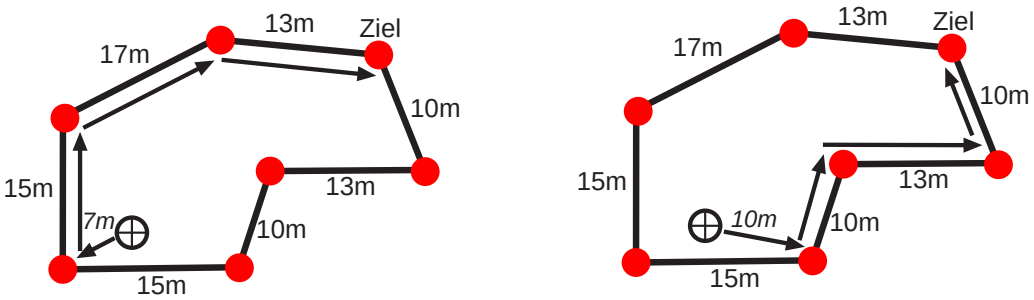


*Abbildung 3.13: Vergleich zwischen Verwendung des nächst- und übernächstgelegenen Wegpunktes*

In der linken Abbildung wird der nächstgelegene Wegpunkt mit 7m Entfernung genommen. Der kürzeste Weg entspricht bei dieser Variante einer Länge von 52m. In der rechten Abbildung wird der Wegpunkt mit der zweitgrößten Entfernung gewählt. Wie zu sehen ist, hat der kürzeste Weg eine Länge von 43m. Es entsteht nun die Frage, wie der optimale Wegpunkt in der Nähe der aktuellen Position gefunden werden kann. Jede aktuelle Position befindet sich zwangsweise innerhalb eines Polygons

aus Straßenzügen. Eine Möglichkeit wäre die Verknüpfung der aktuellen Position mit allen Wegpunkten des umschließenden Polygons. Im schlechtesten Fall würde der Graph um mehrere hundert Kanten erweitert werden. Da der Kürzeste-Wege-Algorithmus abhängig von der Anzahl der Kanten ist, würde dieses Verfahren eine enorme Laufzeitsteigerung verursachen. Um die Laufzeit gering zu halten, bietet es sich an, nur eine Auswahl an Wegpunkten zu betrachten. Hierzu gibt es zwei Varianten. Zum einen ist es möglich einen Schwellwert einzuführen, so dass nur solche Wegpunkte betrachtet werden, deren Abstand innerhalb des Schwellwertes liegen. Die Bestimmung eines konkreten Schwellwertes erachtet sich jedoch als schwierig, da aktuelle Position und der nächstgelegene Wegpunkt je nach Situation verschieden große Entfernungen haben können. Als Lösung kann man z.B. die ersten drei Wegpunkte betrachten. Alternativ geht man davon aus, dass der nächstgelegene Wegpunkt eine Entfernung von 100% hat und setzt als Schwellwert dann eine entsprechende Entfernung von z.B. 120% fest. Obwohl dieses Schwellwertverfahren eine mögliche Lösung für das Problem darstellt, ist es in der Praxis jedoch ungeeignet. Abbildung 3.14 zeigt, dass die Berechnung einer Route zwar möglich ist, da eine Verknüpfung zwischen einem bekannten Wegpunkt und der aktuellen Position besteht, jedoch ist die Führung des Benutzer zum nächsten Wegpunkt nicht optimal.

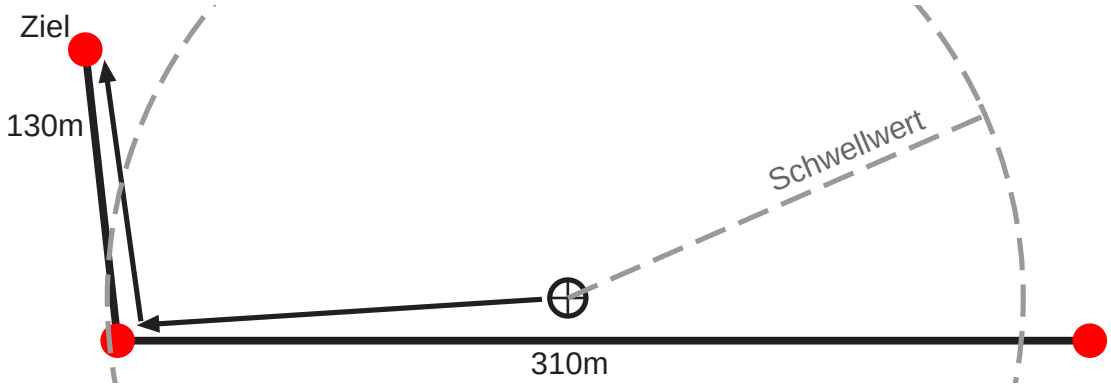


*Abbildung 3.14: Verwendung des Schwellwertverfahrens und nicht optimaler Routenführung*

Wenn die Routenführung einen solchen Wegpunkt verwendet, dann könnte der Benutzer durch die Anzeige des nächsten Wegpunktes irritiert sein. Er sieht zwar die nächste Straße, wird aber in eine andere Richtung geschickt. In diesem Fall müsste der Benutzer z.B. 150m schwer zugängliches Gelände durchqueren um zum ersten bekannten Wegpunkt zu gelangen, obwohl sich eine befestigte Straße in unmittelbarer Nähe befindet. In der Praxis wird daher auf ein solches Schwellwertverfahren verzichten. Anstelle des Schwellwertverfahrens wird die Projektion der aktuellen Position auf die nächstgelegene Straße praktiziert.

### Euklidische Projektion der aktuellen Position

Im rechten Winkel zu der aktuellen Position wird ein neuer Wegpunkt auf der nächstgelegenen Straße projiziert, wie in Abbildung 3.15 skizziert. Durch diese Variante wird zum einen sichergestellt, dass der Fußgänger einen kurzen Weg zum bekannten Straßennetz hat und zum anderen wird der Graph nicht durch viele überflüssige Kanten erweitert, welches sich positiv auf die Berechnungszeit des Kürzesten-Wege-Algorithmus auswirkt. In Abbildung 3.15 wird für die Projektion des Punktes die Lotrechte verwendet. Bei genauerer Betrachtung fällt jedoch auf, dass diese Vorgehensweise nur

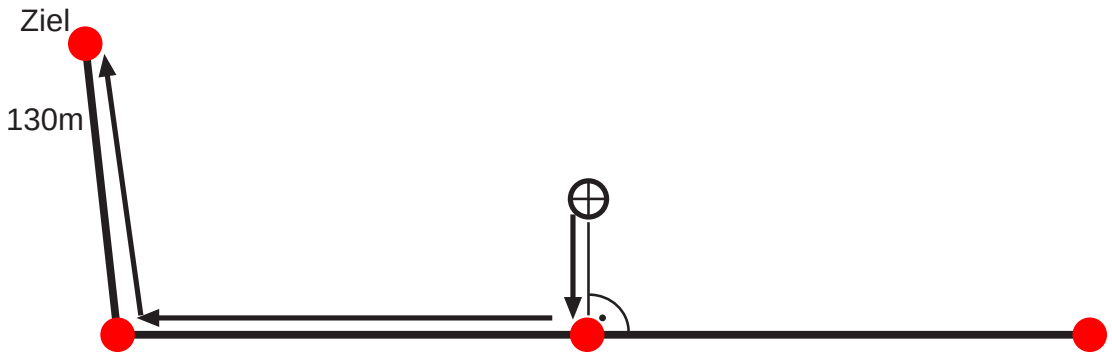


*Abbildung 3.15: Projektion der aktuellen Position auf die nächstgelegene Straße*

einige Fälle abdeckt, da es sich hier nicht um den Abstand zu einer Geraden, sondern zu einer Strecke handelt. Wie in Abbildung 3.16 gezeigt wird, ist die Berechnung über die Lotrechte nicht möglich, wenn der Punkt C außerhalb des lotrechten Bereichs der Strecke $\overline{AB}$ liegt.

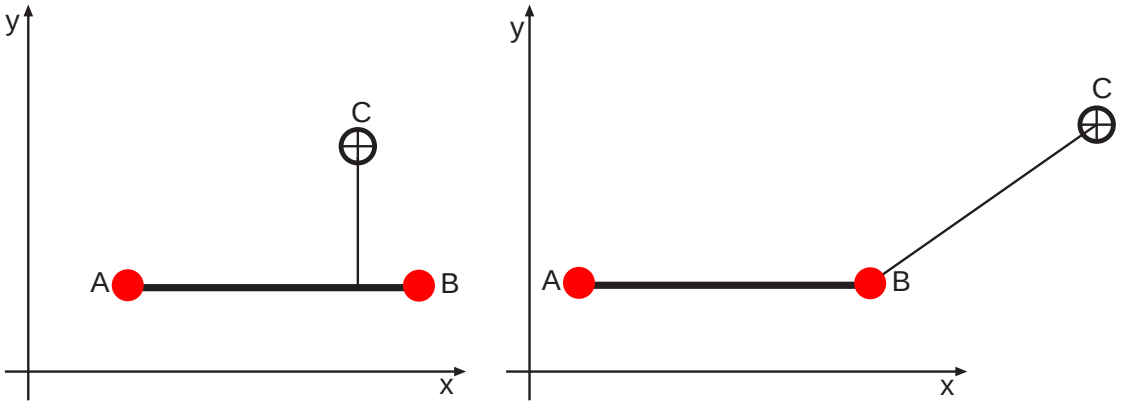


*Abbildung 3.16: Projektion mit innerhalb und außerhalb lotrecht liegender Position*

Für die Berechnung des Projektionspunktes muss demnach eine Fallunterscheidung getroffen werden. Dazu wird im zweidimensionalen Vektorraum folgende Vorgehensweise verwendet:

1. Zum Vektor $\overrightarrow{AB}$ wird die Gerade $g$ durch die Punkte $A$ und $B$ bestimmt: $g : \overrightarrow{x} = A + \lambda \cdot \overrightarrow{AB}$.
2. Der Schnittpunkt $L$ der Geraden $g$ und der Lotrechten von $g$ durch $C$ wird berechnet. Dieser wird auch als Lotfußpunkt bezeichnet. Er berechnet sich durch $E : \overrightarrow{AB} \cdot (\overrightarrow{x} - C) = \overrightarrow{AB} \cdot ((A + \lambda \cdot \overrightarrow{AB}) - C) = 0$. Hieraus lässt sich $\lambda$ bestimmen, welches eingesetzt in $g$ wiederrum $L$ ergibt.
3. Der Vektor $\overrightarrow{AL}$ ist ein Vielfaches des Vektors $\overrightarrow{AB}$. Es gilt demnach $\overrightarrow{AL} = k \cdot \overrightarrow{AB}$ mit $k \in \mathbb{R}$.
4. Ist nun $L$ innerhalb des Vektors $\overrightarrow{AB}$, dann gilt $0 \leq k \leq 1$ und es wird der Lotfußpunkt $L$ verwendet. Liegt $L$ außerhalb von $\overrightarrow{AB}$ wird entweder $A$ oder $B$ verwendet. Für den gesuchten Projektionspunktes $P$ gilt dann:

$$P = \begin{cases} A, & \text{wenn } k < 0 \\ L, & \text{wenn } 0 \leq k \leq 1 \\ B, & \text{wenn } k > 1 \end{cases}$$

Abbildung 3.17 zeigt die drei möglichen Fälle der Fallunterscheidung.

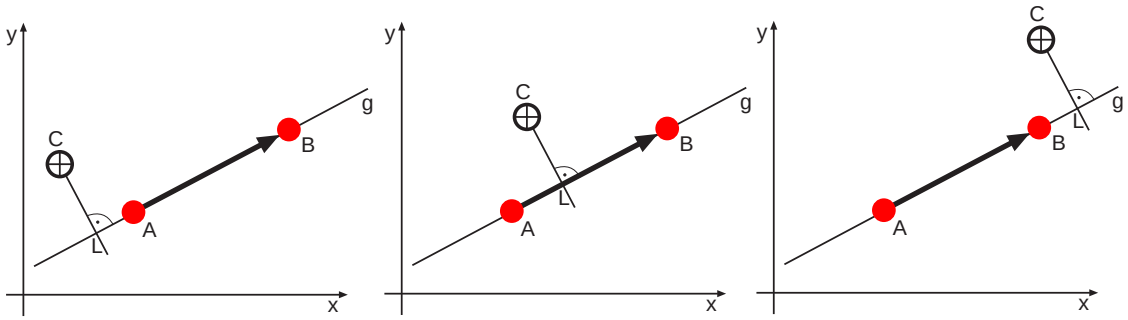


*Abbildung 3.17: Mögliche Fälle bei der euklidische Lotfußpunktbestimmung im 2D-Vektorraum*

## Sphärische Projektion der aktuellen Position

In der geographischen Anwendung ist eine euklidische Projektion nicht verwendbar, da es zum einen keinen Ursprung gibt und zum anderen muss die sphärische Form der Koordinatenprojektion beachtet werden. Die aktuelle Position C muss demnach auf einem Bogen zwischen A und B projiziert werden, wie Abbildung 3.18[2] zeigt.

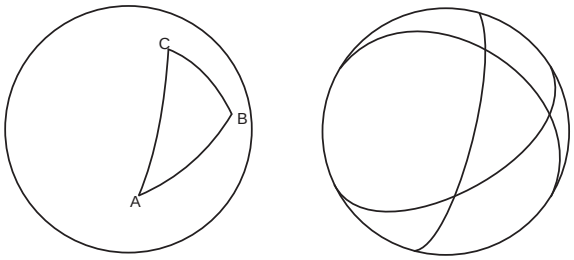


*Abbildung 3.18: Drei Punkte definieren ein sphärisches Dreieck auf der Kugel*

Die Bögen zwischen zwei Punkten sind durch die Großkreise der Kugel definiert. Ein Großkreis beschreibt den größtmöglichen Kreis, dessen Mittelpunkt mit dem Mittelpunkt der Kugel zusammenfällt. Durch den Mittelpunkt der Kugel und zwei Punkten, wie z.B. A und B ist ein Großkreis genau definiert, wie die rechte Zeichnung von 3.18 verdeutlichen soll. Für die Berechnung des Lotfußpunktes L wird jener Großkreis gesucht, der C schneidet und senkrecht zum Großkreis durch und A und B steht. Dies zeigt Abbildung 3.19.

[2] In Anlehnung an [Kom08]

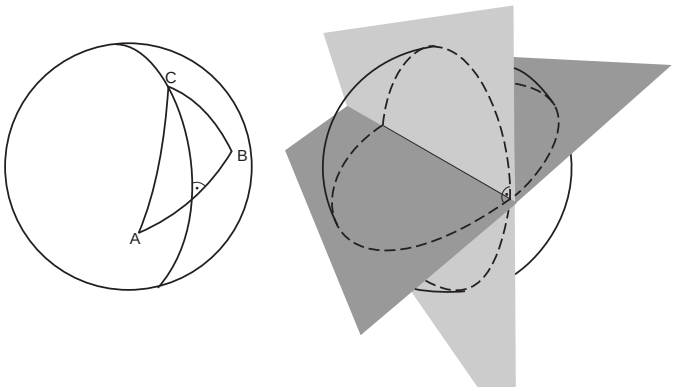


*Abbildung 3.19: Großkreise durch die Punkte und deren Ebenen*

Die zweite Grafik in Abbildung 3.19 zeigt die Grundidee für die Berechnung des Lotfußpunktes. Hierzu werden die Großkreise als zwei Ebenen im dreidimensionalen Vektorraum betrachtet. Der Ursprung entspricht dem Mittelpunkt der Kugel. Eine Ebene wird allgemein mit der Punktrichtungsgleichung $E : \vec{x} = \vec{o} + r \cdot \vec{p} + s \cdot \vec{q}$ mit $r, s \in \mathbb{R}$ angegeben. Der Vektor $\vec{o}$ gibt den Stützvektor an. Da beide Ebenen durch den Ursprung gehen ist dieser null und kann demnach gestrichen werden:

$$E : \vec{x} = r \cdot \vec{p} + s \cdot \vec{q} \tag{3.1}$$

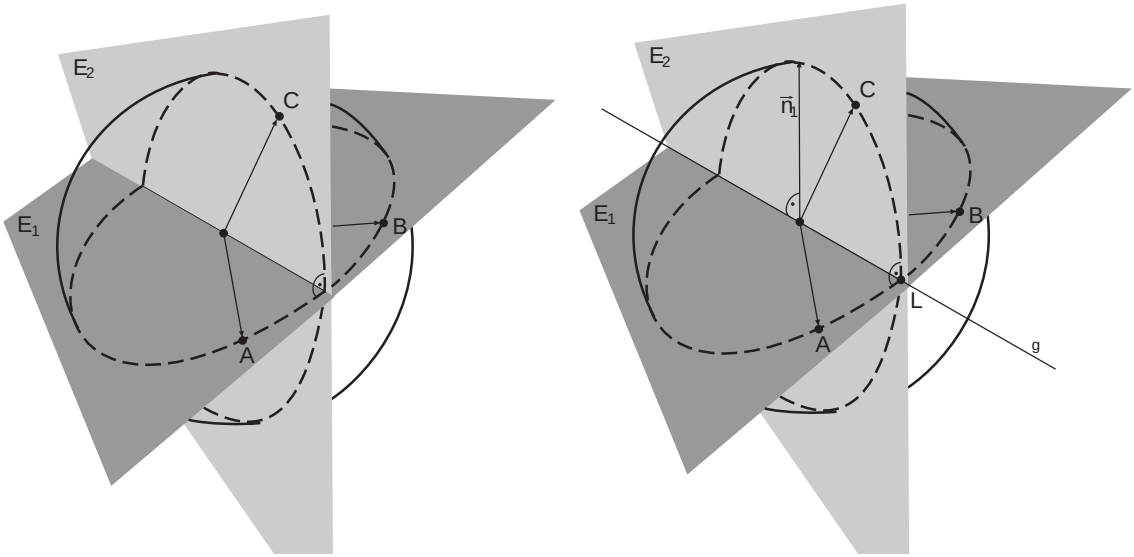


*Abbildung 3.20: Kugel im Vektorraum, deren Vektoren und Darstellung der Normalvektoren der Großkreisebenen*

Wie Abbildung 3.20 zeigt liegen die Punkte A und B in der Ebene $E_1$. Die Ebene kann daher wie folgt beschrieben werden:

$$E_1 : \vec{x} = r \cdot \vec{A} + s \cdot \vec{B} \tag{3.2}$$

Die Ebene $E_2$ wird zum einen durch den Punkt C und zum anderen durch den rechten Winkel zur Ebene $E_1$ bestimmt. Der rechte Winkel kann durch den Normalenvektor $\vec{n_1}$ der Ebene $E_1$ beschrieben werden. Er errechnet sich aus dem Kreuzprodukt von $\vec{A}$ und $\vec{B}$. Entsprechend gilt somit für diese Ebene:

$$E_2 : \vec{x} = t \cdot \vec{C} + u \cdot \vec{n_1} = t \cdot \vec{C} + u \cdot (\vec{A} \times \vec{B}) \tag{3.3}$$

Der Schnitt von $E_1$ und $E_2$ ergibt eine Gerade $g$. Auf dieser Schnittgeraden liegt der Lotfußpunkt $L$, wie Abbildung 3.20 zeigt. Die Schnittgerade wird allgemein durch $g : \vec{x} = \vec{o} + v \cdot \vec{r}$ mit $v \in \mathbb{R}$ beschrieben. Da auch diese Gerade durch den Ursprung bzw. dem Mittelpunkt der Kugel geht, ist auch hier der Stützvektor $\vec{o}$ zu vernachlässigen und es gilt $g : \vec{x} = v \cdot \vec{r}$. Der Vektor $\vec{r}$ dieser Geraden steht orthogonal zum Normalenvektor $\vec{n_1}$ der Ebene $E_1$ und zum Normalenvektor $\vec{n_2}$ der Ebene $E_2$. $\vec{n_1}$ ist bekannt. $\vec{n_2}$ lässt sich über das Kreuzprodukt der beiden Vektoren der Ebenengleichung von $E_2$ berechnen, so dass $\vec{n_2} = \vec{C} \times \vec{n_1}$. Da nun der Vektor $\vec{r}$ orthogonal zu $\vec{n_1}$ und $\vec{n_2}$ steht, gilt $\vec{r} = \vec{n_1} \times \vec{n_2}$. Für die Schnittgerade gilt demnach:

$$g : \vec{x} = v \cdot \vec{r} = v \cdot (\vec{n_1} \times \vec{n_2}) = v \cdot ((\vec{A} \times \vec{B}) \times (\vec{C} \times (\vec{A} \times \vec{B}))) \tag{3.4}$$

Die Schnittgerade hat nun zwei Durchstoßpunkte auf der Kugeloberfläche. Einer dieser Durchstoßpunkte entspricht dem Lotfußpunkt L und der andere Punkt L' liegt auf der gegenüberliegenden Seite von L. Da das verwendete Kreuzprodukt nicht kommutativ, sollte die Reihenfolge bzw. die Lage von A und B sich auf den Vektor $\vec{r}$ der Geraden auswirken, so dass der Vektor $\vec{r}$ entweder in Richtung von L oder in Richtung von L' zeigt. Da jedoch $v \cdot ((\vec{A} \times \vec{B}) \times (\vec{C} \times (\vec{A} \times \vec{B}))) = v \cdot ((\vec{B} \times \vec{A}) \times (\vec{C} \times (\vec{B} \times \vec{A})))$ gilt, zeigt der Vektor $\vec{r}$ immer in Richtung des gesuchten Lotfußpunktes L. Für die genaue Berechnung des Lotfußpunktes im dreidimensionalen Vektorraum muss nun der Einheitsvektor durch $\vec{r_0} = \frac{\vec{r}}{|\vec{r}|}$ bestimmt werden. Durch Streckung von $\vec{r_0}$ auf die Länge des Erdradius $r_e$ kann die genaue Lage von L bestimmt werden. Entsprechend gilt $L = r_e \cdot \vec{r_0}$.

Da die Punkte A, B und C nur als Longitude/Latitude-Paar vorliegen, müssen diese in das sphärische Koordinatensystem umgerechnet werden. Für einen Punkt X mit Latitude $Lat_X$ und Longitude $Lon_X$ gilt die Umrechnung [GS04]:

$$\vec{X} = \begin{pmatrix} X_x \\ X_y \\ X_z \end{pmatrix} = \begin{pmatrix} r_e \cdot \cos(Lat_X) \cdot \cos(Lon_X) \\ r_e \cdot \cos(Lat_X) \cdot \sin(Lon_X) \\ r_e \cdot \sin(Lat_X) \end{pmatrix} \tag{3.5}$$

Der Lotfußpunkt liegt nach der Berechnung hingegen nur als Vektor vor. Für die Umrechnung des Lotfußpunktes L in ein Paar aus Latitude $Lat_L$ und Longitude $Lon_L$ rechnet man nach [GS04]:

$$Lat_L = \arctan(\frac{L_z}{\sqrt{L_x^2 + L_y^2}}) \text{ und } Lon_L = \arctan(\frac{L_y}{L_x}) \tag{3.6}$$

Anzumerken ist, dass die Vektoren $\vec{A}$, $\vec{B}$ und $\vec{C}$ linear unabhängig sein müssen, so dass die Normalenvektoren bestimmt werden können. Ferner müssen die Punkte A, B und C paarweise disjunkt sein. Ebenso dürfen zwei Punkte nicht auf der selben Geraden liegen, wenn diese Gerade durch den Mittelpunkt geht. Dies bedeutet, dass einer der Punkte nicht auf der gegenüberliegenden Seite eines

anderen Punktes liegen darf. Im Kontext einer Navigationsanwendung für Fußgänger befinden sich alle Punkte in unmittelbarer Nähe, da hier die Straße zu einer aktuellen Position bestimmt werden soll. Entsprechend sind die Punkte A, B und C im Normalfall auf der selben Halbkugel.

Wie auch bei der euklidischen Projektion (vgl. Abbildung 3.17) kann der Lotfußpunkt L auch bei der sphärischen Projektion außerhalb der Strecke zwischen A und B liegen. In der sphärischen Projektion liegt L auf dem Großkreis durch A und B. Hier muss geprüft werden, ob L innerhalb des Großkreisbogens zwischen A und B liegt, oder außerhalb. Dazu können die Distanzen zwischen den Punkten A, B und C betrachtet werden. Sei $d(A, B)$ die sphärische Distanz zwischen A und B. Wenn $d(A, L) + d(B, L) = d(A, B)$, dann liegt L innerhalb des Großkreisbogens zwischen A und B. Entsprechend ist L der neue Wegpunkt. Gilt im Gegensatz dazu $d(A, L) + d(B, L) > d(A, B)$, dann liegt der Punkt außerhalb. Vergleicht man in diesem Fall $d(A, L)$ mit $d(B, L)$, ist je nach kürzerem Abstand zu L entweder A oder B der neue Wegpunkt. Allgemein gilt:

$$P = \begin{cases} A, & \text{wenn } d(A,L) + d(B,L) > d(A,B) \text{ und } d(A,L) \leq d(B,L) \\ L, & \text{wenn } d(A,L) + d(B,L) = d(A,B) \\ B, & \text{wenn } d(A,L) + d(B,L) > d(A,B) \text{ und } d(A,L) > d(B,L) \end{cases} \tag{3.7}$$

### Erweiterung des Graphen mit der aktuellen Position

Sowohl euklidisch, als auch sphärisch kann der Lotfußpunkt und damit der neue Wegpunkt auf der nächstgelegenen Straße bestimmt werden. Damit dieser Punkt in einem Graphen benutzt werden kann, muss er entsprechend hinzugefügt werden.

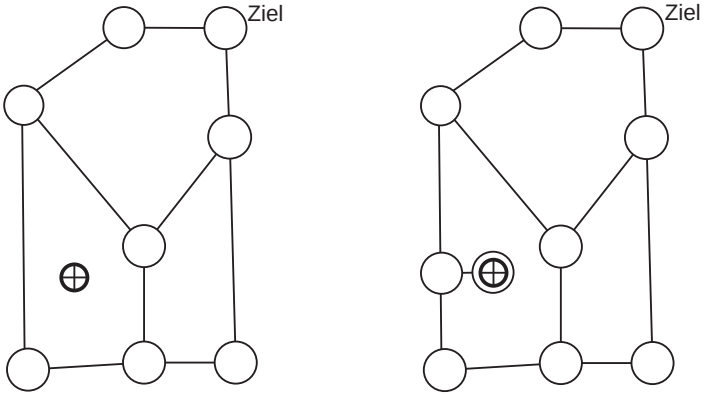


*Abbildung 3.21: Erweiterung des Graphen durch den neuen Wegpunkt und der aktuellen Position*

Abbildung 3.21 verdeutlicht, dass zum einen die aktuelle Position und zum anderen der neue Wegpunkt als Knoten zum Graphen hinzugefügt wird. Der neue Wegpunkt wird zwischen zwei bereits bestehenden Wegpunkten eingeschoben, so dass der neue Wegpunkt entsprechende Referenzen zu den verfügbaren Wegpunkten erhält. Der Graph enthält somit die aktuelle Position als Startknoten und kann durch einen Algorithmus den kürzesten Weg auf diesem Graphen berechnen. Die Verwendung der Projektion erlaubt es den Benutzer auf kürzestem Wege zum bekannten Straßennetz zu führen, um

von dort aus eine kontrollierte Routenführung durchführen zu können. Als Erweiterung wäre es möglich, die aktuelle Position mit allen Kanten des umgebenden Polygons zu verknüpfen. Hierbei muss jedoch beachtet werden, dass es beliebig viele Kanten geben kann, die den Graphen entsprechend erheblich vergrößern könnten. Dies hat wiederum einen direkten Einfluss auf die Geschwindigkeit des Kürzesten-Wege-Algorithmus.

### 3.2.2 Map Matching

Der Standort eines Fußgängers wird mittels einer satellitengestützten Positionsbestimmung festgestellt. Heutzutage benutzt man hierzu das globale Navigationssatellitensystem (GNSS) des US-Verteidigungsministerium, welches offiziell den Namen Navigational Satellite Timing and Ranging - Global Positioning System (NAVSTAR-GPS) trägt. NAVSTAR-GPS wird im allgemeinen Sprachgebrauch kurz als GPS bezeichnet. GPS wurde 1973 vom US-Verteidigungsministerium in Auftrag gegeben und dient zur globalen Positionsbestimmung. Das Problem von GPS ist die Genauigkeit dieser Positionsbestimmung. Bei der vollständigen Inbetriebnahme im Jahr 1992 hatte ein GPS-Signal laut einem Bericht des US-Verteidigungsministeriums eine Messabweichung von 100m in 95% der Messungen [Man98]. Da GPS dem US-Militär vorbehalten sein sollte, wurde für den allgemeinen Gebrauch eine künstliche Ungenauigkeit eingeführt. Im Jahr 2000 wurde die künstliche Ungenauigkeit abgeschaltet, so dass die Abweichung auf 10m verringert wurde [SW06]. Das Differential-GPS (DGPS), welches mehrere GPS-Empfänger und zusätzliche Referenzstationen benutzt, erreicht je nach Gerätequalität eine Genauigkeit von 0.01-5m [Man98].

Die Ungenauigkeit führt dazu, dass gemessene Positionen nicht immer mit den Wegpunkten einer Straße zusammenfallen. Für die Routenführung ist es jedoch von belangen zu wissen, auf welcher Straße bzw. auf welcher Kante die Person sich befindet. Abbildung 3.22 zeigt eine solche Messungenauigkeit während ein Fußgänger von Wegpunkt A nach B geht.

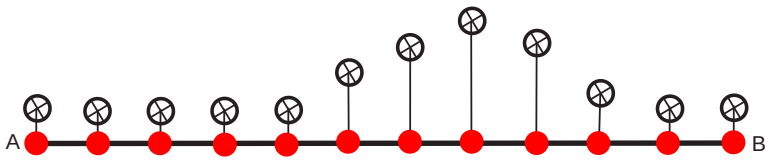


*Abbildung 3.22: Messfehler bei GPS entlang einer geraden Straße*

Betrachtet man nur eine Straße, können die gemessenen Wegpunkte auf die Straße abgebildet werden, so dass der Messfehler korrigiert werden kann. Dies soll ebenfalls in Abbildung 3.22 gezeigt werden, in dem die gemessenen Positionen als Wegpunkte auf die Straße *gematcht* werden. Im Falle einer Straße kann dieses Matching eindeutig durchgeführt werden, in dem z.B. jede gemessene Position im rechten Winkel auf die Straße projiziert wird. Ein komplexes Straßennetz erweist sich jedoch als Hürde für ein solches Matching. Wie Abbildung 3.23 verdeutlichen soll, ist die Straße, auf der ein Punkt gematcht werden soll, nicht immer eindeutig.

Neben solchen parallelen Straßen stellen auch Kreuzungen eine Herausforderung, da ein Messfehler z.B. die Vermutung offen lässt, ob ein Fußgänger nun abgebogen ist oder nicht. Ein weiteres Problem, welches in die selbe Problemklasse fällt, entsteht nicht durch Messfehler, sondern durch

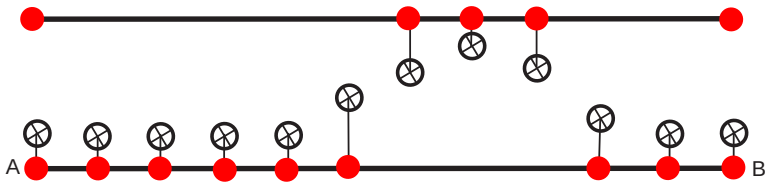


*Abbildung 3.23: Fehlerhaftes Map Matching*

die vorhandenen Datensätze. Die Wegpunkte, die eine Straße definieren, liegen immer in der Mitte einer Straße. Eine Straße kann aber mehrere Fahrspuren enthalten, so dass der Fußgänger nicht direkt entlang der Wegpunkte geht. Im einfachsten Fall geht der Fußgänger parallel zur Straße auf einem Fußweg. Jedoch muss bereits hier die Position des Fußgängers auf die eigentliche Straße abgebildet werden, wie Abbildung 3.24 verdeutlichen soll.

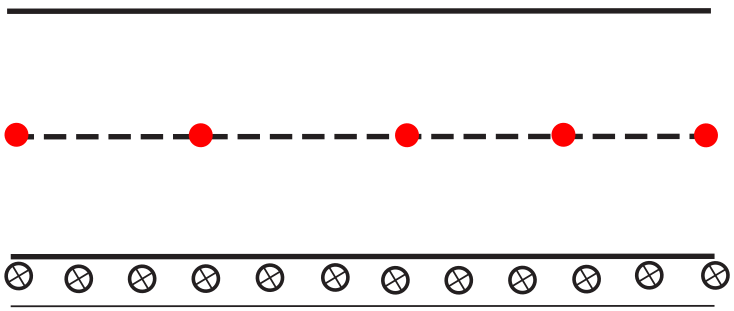

*Abbildung 3.24: Vergrößerte Darstellung einer Straße und Fortbewegung auf einem Fußweg*

Die Fortbewegung des Fußgängers stellt ein weiteres Problem. Der Fußgänger ist im Gegensatz zu einem Kraftfahrzeug nicht auf eine Fahrbahn angewiesen. Entsprechend kann er z.B. an Straßenecken diese schneiden. Hierzu zählen auch Abkürzungen, die ein ortskundiger Fußgänger kennt, aber nicht in den Datensätzen vorhanden sind. Hierzu zählen auch implizit vorhandene Wege, wie z.B. Bahnhöfe oder Einkaufszentren mit mehr als einem Ausgang. Einige Beispiele sind in Abbildung 3.25 dargestellt.

Das Abbilden von gemessenen Positionen auf eine bekannte Straße wird als Map Matching oder auch Karteneinpassung bezeichnet [WR06]. Dabei wird versucht aufgezeichnete Daten mit der digitalen Karten abzugleichen. Dazu wird die digitale Referenz verwendet, die mit der höchsten Wahrscheinlichkeit zu den aufgezeichneten Karten passen. In der klassischen Kraftfahrzeugnavigation setzt man hierzu zusätzlich Dead Reckoning ein [Web99]. Dead Reckoning bezeichnet eine aktuelle Ortsbestimmung durch Abgleich von Zeit, Geschwindigkeit und Himmelsrichtung. Diese Daten werden zusätzlich mit den Daten aus der GPS-Ortsbestimmung gekoppelt. Somit erhält man genauere Daten, die ein Map Matching wesentlich erleichtern [Web99]. Obwohl z.B. Schrittzähler oder Kompasse eingesetzt werden könnten, ist bei einer Fußgängernavigation ein Dead Reckoning mit Geschwindigkeit und Himmelsrichtung schwer umsetzbar, da zusätzliche Geräte notwendig sind. Für das Map Mat-

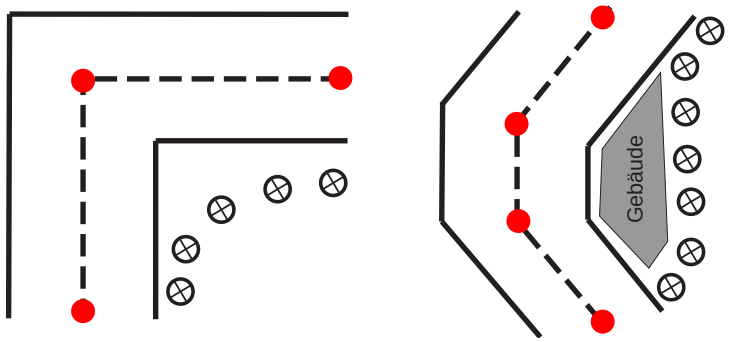


*Abbildung 3.25: Schneiden einer Ecke und Nutzen einer Abkürzung*

ching stehen somit nur die ungenaueren GPS-Daten zur Verfügung. Ein solches Map Matching wird in drei Schritten durchgeführt:

1. Bestimmung der Position
2. Festlegen des Suchbereichs
3. Anwendung von Matching-Techniken

Als Ergebnis erhält man den logischen Bezug zwischen der aktuellen Position und der digitalen Karte. Dieser ist nötig, damit die Routenführung erreichte und kommende Wegpunkte bestimmen und anzeigen kann.

Die Bestimmung der Position kann hier als gegeben betrachtet werden. Der Suchbereich kann hingegen mittels eines Schwellwertes (vgl. Abbildung 3.14) festgelegt werden. Ebenso gibt die Historie Auskunft über den Suchbereich. So können bereits besuchte Straßen aus der Ergebnismenge des Suchbereichs entfernt werden.

### Map Matching mit Projektionsverfahren

Eine einfache Matching-Technik versucht die Straße zuzuordnen, die am nächsten gelegen ist. Dieses kann mit einem Projektionsverfahren, wie im vorigen Abschnitt beschrieben, durchgeführt werden. Dabei wird jene Straße genommen, die am nächsten liegt. Es gibt dabei unentscheidbare Fälle, bei dem die Distanz zu zwei oder mehr Straßen exakt gleich ist. Hier kann allein aus der Position nicht entschieden werden, welche Straße die bessere Wahl ist. Eine Möglichkeit diese Entscheidung zu vereinfachen ist die Verwendung der Bewegungsrichtung. Die NMEA-Daten, welche als GPS-Rohdaten beschrieben werden können, liefern auch Bewegungsrichtungen. Diese Bewegungsrichtung kann mit dem Azimuth einer Straße verglichen werden. Entsprechend wird die Straße gewählt, dessen Azimuth am nächsten zum Azimuth der Bewegungsrichtung ist [HS98]. Vorteil dieses Verfahrens ist die Unabhängigkeit von den eigentlichen Positionen, da hier Winkel verwendet werden.

### Map Matching mit Profilen

Neben der Verwendung der Bewegungsrichtung wird in der Praxis oft die Historie der Positionen mit der Karte verglichen. Dazu wird ein aus diesen Punkten bestehendes Straßenstück erstellt. Mit einer

Matching-Technik wird dann geprüft, wie gut dieses Straßenstück in die digitale Straße passt. Wenn dieses Straßenstück in der aktuellen Karte gematcht werden kann, dann kennt man die aktuelle Position in Bezug auf die digitale Karte, welches Ziel des Map Matchings ist. Eine Matching-Technik, die dieses Verfahren einsetzt wird in der Literatur als Map Matching mit Profilen angegeben. Hierbei wird ein Profil der zurückgelegten Strecke angelegt und als Ist-Funktion bezeichnet. Diese Daten ergeben sich aus diskret gemessenen Werten, die die Winkeländerungen angeben. Für einen Kantenzug der digitalen Karte erstellt man daraufhin ein weiteres Profil. Dazu wird die Bogenlänge, also die Distanz zwischen zwei Wegpunkten aus der digitalen Karten, bestimmt und stellt die Soll-Funktion dar. Abbildung 3.26 soll dies verdeutlichen. Mit einer Kreuzkorrelation kann man daraufhin einen

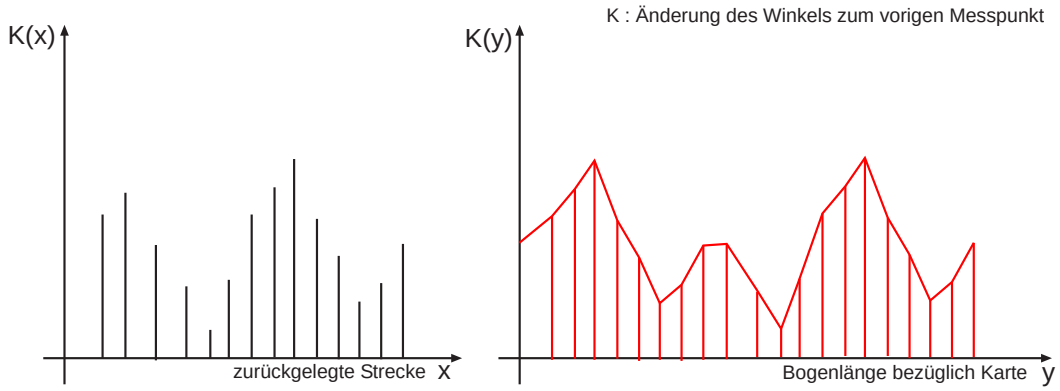


*Abbildung 3.26: Beispiel für Map Matching mit Profilen*

Zusammenhang zwischen diesen beiden Kurven berechnen. Durch diesen Vergleich von Soll und Ist kann eine Aussage gemacht werden, ob die beiden Profile ähnlich sind oder nicht [Czo00]. Zusätzlich lässt sich dazu ein Kalman-Filter anwenden. Idee des Kalman-Filters ist es, dass zusätzlich weitere Beobachtungen, wie z.B. alte Profile, in die Schätzung einfließen, damit Messfehler beseitigt werden. Eine praktische Anwendung des Kalman-Filters ist z.B. die rekursive Parameterschätzung [Czo00]. Dazu werden mehrere Gruppen von Messungen so geglättet, als wären sie eine Messung. Die Glättung wird mit einer allgemeinen, nichtlinearen Beobachtungsgleichung errechnet. Die Beobachtungsgleichung wird bei jeder Gruppe neu angepasst und dient dann als Eingabe für die nächste Gruppe. Sie ist somit rekursiv. Vorteil dieser Beobachtungsgleichung ist, dass sie die Parameter von vorigen Gruppen nicht kennen muss. Dies verringert unter anderem den Rechenaufwand.

### Map Matching auf Koordinatenebene

Neben dem Map Matching mit Profilen gibt es noch das Map Matching auf Koordinatenebene [Czo00]. Hierbei wird die beobachtete Straße in konstante Linienelemente unterteilt und entspricht der Soll-Messstrecke, wie z.B. alle 2m. Die aufgezeichnete Strecke wird als Ist-Strecke bezeichnet und wird ebenfalls in konstante Linienelemente gleicher Länge unterteilt. Man erhält durch die Unterteilung jeweils eine Menge von Punkten. Diese Punkte dienen nun zum Vergleich von Soll und Ist. Eine einfache Form wäre hier die Translationseinpassung, bei der lediglich eine vektorielle Verschiebung der Punkte betrachtet wird. Da hierbei aber keine Messabweichungen möglich sind, wird eine Ähnlichkeitstransformation durchgeführt. Hierbei werden auch Rotation und Maßstab beachtet. Zur Beurtei-

lung, ob ein Soll-Punkt zu einem Ist-Punkt passt kann der Helmertsche Punktfehler errechnet werden [Czo00].

Die beschriebenen Map Matching-Techniken werden in erster Linie bei der Fahrzeugnavigation eingesetzt und beachten demnach nicht viele Sonderfälle, die bei der Fußgängernavigation auftreten. Hierbei stechen vor allem beschriebene Abkürzungen oder die vielen Varianten, wie man eine Kreuzung überqueren kann, hervor. Je besser ein Map Matching-Verfahren ist, desto höher ist auch der Rechenaufwand, der dafür aufgewendet werden muss.

### 3.2.3 Erreichen und Anzeigen des nächsten Wegpunktes

Das Erreichen des nächsten Wegpunktes einer vorgegebenen Route besteht aus zwei Aufgaben. Zum einen muss berechnet werden, wann ein Wegpunkt erreicht wurde und zum anderen muss es eine geeignete Anzeige für den nächsten Wegpunkt geben.

#### Erreichen eines Wegpunktes

Bei der Routenführung soll der Fußgänger von einem Wegpunkt zum nächsten Wegpunkt geleitet werden. Dazu muss die Navigationsanwendung wissen, wann ein Wegpunkt erreicht wurde, um in Folge dessen den nächsten Wegpunkt anzeigen zu können. Wie schon im vorigen Abschnitt beschrieben, liegt ein Wegpunkt nicht zwingend auf dem Weg, den ein Fußgänger benutzt (vgl. Abbildung 3.24). Entsprechend ist es nicht praktikabel zu prüfen, ob die Koordinaten eines Wegpunktes mit den Koordinaten der aktuellen Position übereinstimmen, um so das Erreichen eines Wegpunktes festzustellen. Um dennoch die aktuelle Position mit dem digitalen Kartenmaterial abzugleichen wird das zuvor beschriebene Map Matching angewandt. Dabei wird ein Abbild der zurückgelegten Strecke auf die vorhandene digitale Karte abgebildet. Dadurch kann dann wiederum festgestellt werden, welche Wegpunkte bereits erreicht wurden. Entsprechend kann auch der nächste Wegpunkt festgestellt werden. Abbildung 3.27 soll dieses Verfahren zeigen.

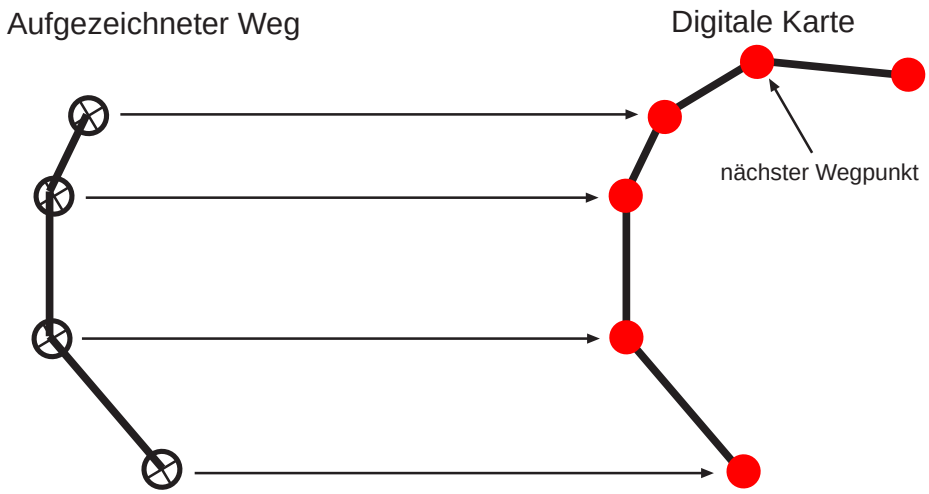


*Abbildung 3.27: Map Matching zur Berechnung des nächsten Wegpunktes*

Ein Nachteil bei der Nutzung von Map Matching ist neben der aufwendigen Berechnung die Tatsache, dass stets der zurückgelegte Weg aufgezeichnet werden muss. Zusätzlich muss die digitale Karte für eine Route vorliegen. Es kann demnach nicht der Kürzeste-Wege-Graphen, bei dem die Knoten den Wegpunkten entsprechen, durchlaufen werden. Eine andere Technik, bei dem nur die Wegpunkte

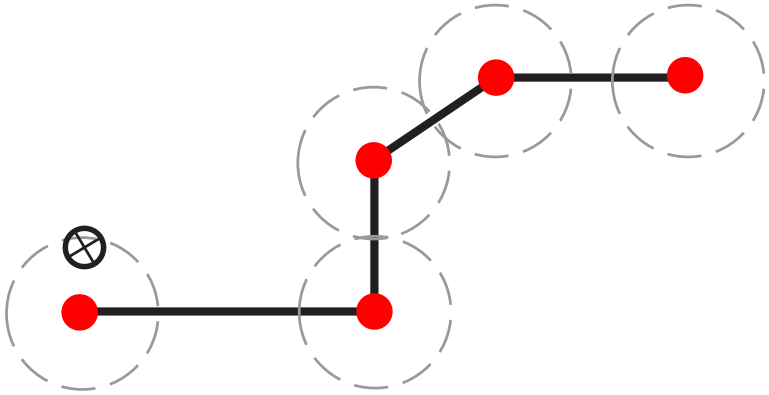

*Abbildung 3.28: Erreichen eines Wegpunktes bei Wegpunkten mit Schwellwert*

benutzt werden, benutzt einen Schwellwert. Hierbei gilt ein Wegpunkt als erreicht, wenn die Distanz zwischen Fußgänger und Wegpunkt innerhalb des Schwellwertes liegt. Abbildung 3.28 zeigt die Verwendung eines Schwellwertes. Für eine gute Routenführung muss ein geeigneter Schwellwert bestimmt werden. Wird der Schwellwert zu groß gewählt, könnten dadurch Irritationen entstehen oder es wird ein falscher Weg angezeigt. Wird der Schwellwert hingegen zu klein gewählt, kann es sein, dass der Fußgänger den Wegpunkt entweder nicht erreichen kann. Dabei könnte es z.B. passieren, dass ein Fußgänger in die Mitte einer mehrspurigen Kreuzung geleitet wird, damit der Wegpunkt erreicht werden kann. Abbildung 3.29 stellt jeweils ein Beispiel vor.

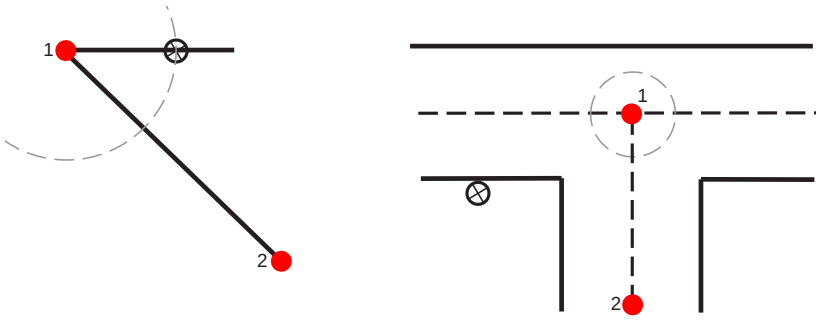


*Abbildung 3.29: Beispiel für ein zu groß und ein zu klein gewählter Schwellwert*

Das erste Beispiel der Abbildung zeigt einen zu großen Schwellwert, bei dem der Fußgänger den Wegpunkt 1 erreicht hat. Da in dieser Situation der Wegpunkt 1 nun als erreicht markiert wird, wird

Wegpunkt 2 als nächsten Wegpunkt angezeigt. Aber im Gegensatz zum alten Weg, liegt der Wegpunkt 2 nun in der anderen Richtung. Dies verwirrt unter Umständen den Benutzer. Insbesondere, wenn zwischen der aktuellen Position und Wegpunkt 2 sich z.B. ein Gebäude oder ein See befindet. Das zweite Beispiel zeigt eine Kreuzung bei der ein Schwellwert zu klein gewählt wurde. Hier müsste der Fußgänger die Fahrbahn betreten, damit der Wegpunkt 1 als erreicht markiert wird. Ist Wegpunkt 2 der nächste Wegpunkt auf der Route, dann müsste der Fußgänger zusätzlich noch einen Umweg nehmen, damit der Wegpunkt erreicht wird. Einen optimalen Schwellwert zu bestimmen ist demnach schwierig, da jede Straße und jeder Weg eine andere Struktur hat. Ebenso besteht bei der Fußgängernavigation das Problem, dass der Fußgänger nicht zwingend auf der bekannten Straße läuft. Um jedoch einen annähernd guten Schwellwert zu bestimmen, kann man auf die Datensätze der Straße zurückgreifen. Bei den Datensätzen von OpenStreetMap steht z.B. der Straßentyp zur Verfügung. Aus diesem kann dann eine ungefähre Breite der Straße bestimmt werden. Ein Fahrstreifen in Deutschland ist ungefähr 4m Breit, so dass eine Kreisstraße mit zwei Fahrstreifen eine ungefähre Breite von 8m hat. Der 1m breite Fußgängerweg befindet sich parallel zu der Straße mit 1m Abstand. Liegt nun der Wegpunkt in der digitalen Version dieser Straße in der Mitte der Fahrbahn, so muss der Schwellwert hier mindestens 5-6m sein, damit der Fußgänger den Wegpunkt erreichen kann, ohne dass er den Fußweg verlässt. Abbildung 3.30 veranschaulicht diesen Sachverhalt.

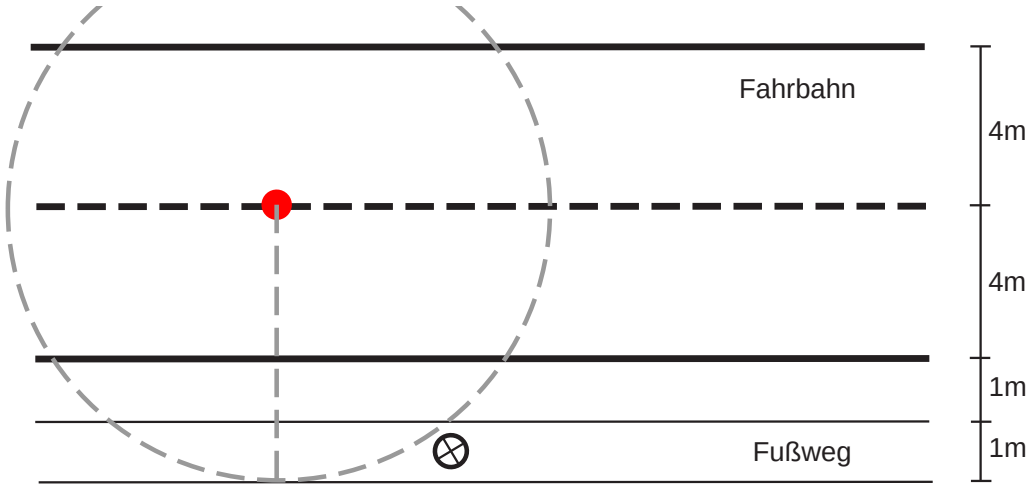


*Abbildung 3.30: Zweistreifige Straße mit Fußweg und minimalem Schwellwert*

Analog kann dies auch für mehrspurige Straßen angepasst werden. Ebenso sind minimale Fahrstreifenbreiten je Straßentyp in Deutschland vorgeschrieben. Wird dieses Verfahren verwendet, setzt dies natürlich vorraus, dass zum einen Straßentypus bekannt und falls bekannt dieser auch korrekt ist. Zusätzlich sollte der Schwellwert ein wenig höher sein, da ein Fußweg nicht zwingend direkt an der Fahrbahn oder auch mehr als 1m von der Fahrbahn entfernt sein kann.

### Anzeigen des nächsten Wegpunktes

Solange ein Wegpunkt noch nicht erreicht wurde, muss dieser dem Benutzer angezeigt werden, damit dieser weiß in welche Richtung er gehen muss. Auf Grund der vorher beschriebenen Problematiken, wie sie z.B. in Abbildung 3.29 dargestellt wurden, kann eine Anzeige des nächsten Wegpunktes ein Problem darstellen. Wählt man z.B. nur einen Richtungspfeil, der in Richtung des nächsten Wegpunk-

tes zeigt, kann dieses eine falsche Route suggerieren. Abbildung 3.31 soll darstellen, wie der nächste Wegpunkt für verschiedene Positionen mit einem Richtungspfeil angezeigt wird.

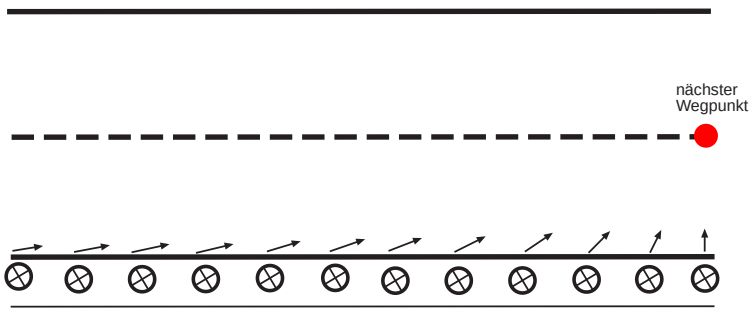


*Abbildung 3.31: Anzeige der Richtung zum nächsten Wegpunkt bei verschiedenen Positionen*

Eine bessere Lösung kann über die Korrektur des Richtungswinkel erreicht werden. Dazu wird die Strecke A zwischen dem alten Wegpunkt 1 und dem nächsten Wegpunkt 2 benötigt. Des Weiteren wird der Punkt markiert, bei dem der Benutzer den Wegpunkt 1 erreicht hatte. Zusammen mit der aktuellen Position ergibt dies die Strecke B. Durch die markierte Position und durch die aktuelle Position wird jeweils eine Parallele zu A gelegt. Die Parallele A' durch den markierten Punkt gibt den Weg an, den die Person hätte gehen sollen. Somit gibt der Winkel zwischen A' und B die Abweichung an. Dieser Winkel wird genutzt, um die Richtung bei der aktuellen Position zu korrigieren. Abbildung 3.32 zeigt ein Beispiel dazu.

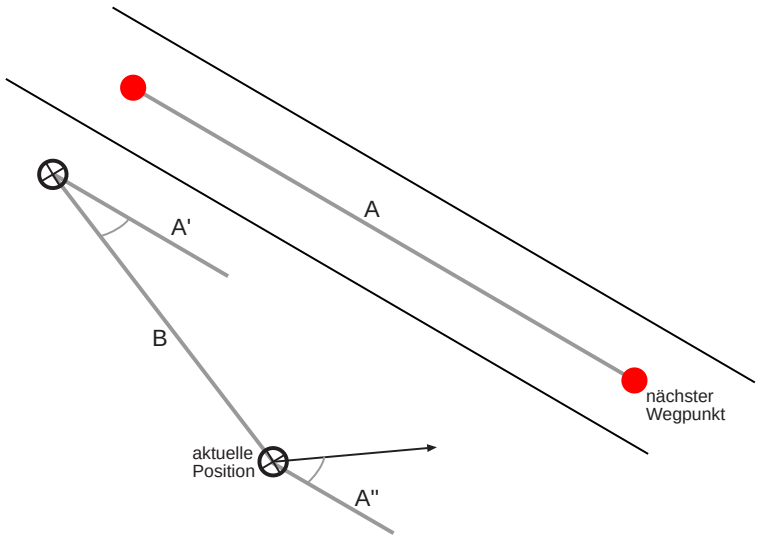


*Abbildung 3.32: Differenz der Winkel als Richtungsangabe*

Der Vorteil dieser Richtungsangabe ist, dass die Richtung eher mit dem eigentlichen Weg übereinstimmt, als beim dem anderen Verfahren. Verfolgt man genau die Richtung dieser Richtungsangabe, so wird die Richtung parallel zur Straße angezeigt, da der Korrekturwinkel 0 ist. Wird hier z.B.

das Schwellwertverfahren eingesetzt, um den nächsten Wegpunkt zu erreichen, so erreicht man den nächsten Wegpunkt auch innerhalb des Schwellwertes, obwohl man nicht direkt in Richtung dieses Wegpunktes geleitet wird. Die linke Grafik in Abbildung 3.33 zeigt dazu ein Beispiel. Nachteil dieses Verfahrens ist, dass ein zu großes Abweichen von der Route eine Richtung erzeugt, die von dem nächsten Wegpunkt wegführt, wie das rechte Beispiel in Abbildung 3.33 zeigt.

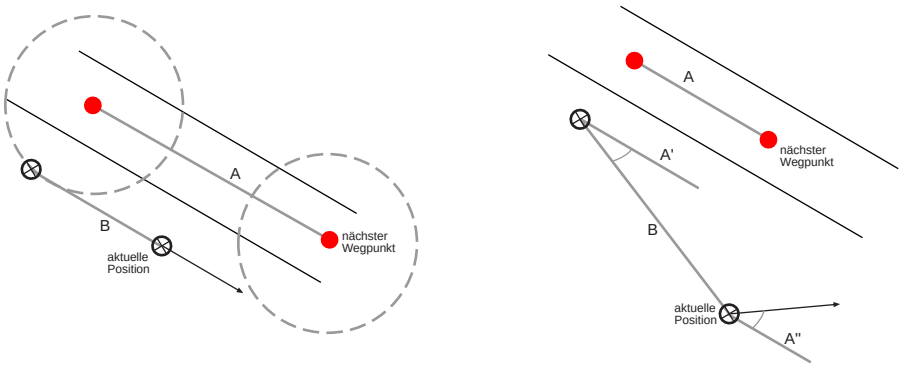


*Abbildung 3.33: Differenz der Winkel als Richtungsangabe*

Bei diesem Verfahren ist es demnach nötig, dass der Fußgänger der Richtungsangabe möglichst genau folgt. Sobald er von der vorgegebenen Richtung abweicht, wird die Routenführung mit diesem Verfahren nicht mehr funktionieren.

### 3.2.4 Neuberechnung einer Route

Kommt der Fußgänger gewollt oder ungewollt von der Route ab oder hat er eine Abkürzung benutzt, so ist es notwendig, dass die aktuelle Routenführung abgebrochen wird und eine Neuberechnung der Route ausgelöst wird. Wie auch in den vorigen Abschnitten beschrieben, stellt auch hier die Bewegungsfreiheit eines Fußgängers eine besondere Herausforderung dar. So soll eine Neuberechnung nicht ausgelöst werden, wenn der Fußgänger auf Grund der baulichen Struktur des Fußweges zu weit vom Wegpunkt entfernt ist. Ein Verfahren, dass die Distanz zum nächsten Wegpunkt betrachtet und eine Route neuberechnet, wenn die Distanz nicht abnimmt, sondern zunimmt, ist daher nicht geeignet. Im Gegensatz dazu kann die Bewegungsrichtung betrachtet werden. Weicht die Bewegungsrichtung z.B. mehr als 100 Grad von der Richtung zum nächsten Wegpunkt ab, so geht die Person in die falsche Richtung und es wird eine Neuberechnung ausgelöst. Hierbei sollte nicht direkt bei der ersten Messung eine neue Route berechnet werden, sondern erst, wenn der Fußgänger eine bestimmte Strecke oder Zeit in die falsche Richtung gelaufen ist. Zusätzlich kann man das Verfahren ergänzen, in dem man sowohl den besuchten Teil, als auch den noch offenen Teil der Route betrachtet. Beobachtet man die bereits besuchten Wegpunkte und der Fußgänger erreicht wieder einen solchen Wegpunkt, so ist es möglich, dass die Route verlassen wurde. Beobachtet man zukünftige Wegpunkte, so können unter Umständen Abkürzungen erkannt werden, wenn der Fußgänger einen Wegpunkt auslässt und den übernächsten Wegpunkt erreicht.

## 3.3 Fazit

Im Folgenden werden zu den oben genannten Problemen eine Lösung aus den möglichen Lösungsansätzen ausgewählt. Die ausgewählten Techniken sind die Anforderungen für die Navigationsanwendung.

### Daten

Für die Berechnung des kürzesten Weges, müssen die Daten zuerst in einen Graphen überführt werden. Dabei ist auch entscheidend, wie die Daten interpretiert werden sollen. So sieht es OpenStreetMap vor, dass nur Straßen mit einander verbunden sind, wenn sie den selben Wegpunkt benutzen. Da die Daten von nicht-professionellen Personen erfasst werden, sind Verknüpfungen zwischen Straßen nicht immer korrekt angelegt. Eine automatische Korrektur ist hierbei jedoch nicht möglich, da fehlerhafte Daten nicht von korrekten Daten eindeutig unterschieden werden können. Es wird demnach die Annahme getroffen, dass OpenStreetMap nur korrekte Daten liefert. Entsprechend folgt **keine Korrektur der Daten**.

### Routing-Algorithmus

Auf den Daten wird eine Route berechnet, in dem die Daten vorher in einen Graphen überführt wurde. Mit diesem Graphen kann das Kürzeste-Wege-Problem gelöst werden. Es gibt zu diesem Single Source Shortest Path Problem (SSSP) bereits effektive Algorithmen. So ist der Algorithmus von Dijkstra ein einfacher Algorithmus, der mit geeigneten Datenstrukturen eine Zeitkomplexität von $\mathcal{O}(n \cdot \log(n) + m)$ aufweisen kann. Eine Erweiterung des Dijkstra-Algorithmus ist der A*-Algorithmus. Dieser Algorithmus betrachtet zusätzlich eine Heuristik, wie z.B. die Luftlinie zwischen zwei Orten. Sofern die Berechnung der Heuristik nicht extrem Aufwendig ist, ist der A*-Algorithmus schneller als der Dijkstra-Algorithmus. Da der A*-Algorithmus eine Erweiterung des Dijkstra-Algorithmus ist, muss der Dijkstra-Algorithmus an einer Stelle nur durch die Heuristik ergänzt werden. Die Luftlinie zwischen zwei Orten bzw. Wegpunkten ist eine Heuristik im Sinne des A*-Algorithmus. Entsprechend wird hier der **A*-Algorithmus** verwendet, da durch wenig Aufwand eine große Geschwindigkeitssteigerung erreicht werden kann.

### Berechnung des Startpunktes

Vor der Berechnung einer Route, muss die aktuelle Position zum Straßennetz hinzugefügt werden, da sonst kein Startknoten vorliegt. Eine Lösung, bei dem die aktuelle Position mit dem nächstgelegenen Wegpunkt verknüpft wird, liefert nicht das beste Ergebnis. Eine weitere Möglichkeit besteht darin, dass die aktuelle Position lotrecht auf die nächstgelegene Straße projiziert wird und der Projektionspunkt einem neuen Wegpunkt entspricht. Eine einfache Variante zur Berechnung des Projektionspunktes ist die euklidische Berechnung. Diese kann jedoch zu Fehlern führen, da die Kugelkrümmung der Erde nicht beachten wird. Entsprechend wird dies durch die **sphärische Lotpunktberechnung** kompensiert. Die sphärische Betrachtung ist somit das einzig korrekte Verfahren für die Berechnung des Projektionspunktes. Nach der Berechnung werden Projektionspunkt und aktuelle Position dem Graphen hinzugefügt und miteinander verbunden. Der Wegpunkt des Projektionspunktes wird mit den Wegpunkten verknüpft, die zu der Straße gehören, auf die der Projektionspunkt projiziert wurde. Mit diesem Graphen kann dann von der aktuellen Position aus der kürzeste Weg berechnet werden.

### Map-Matching

Wenn der kürzeste Weg berechnet worden ist, soll der Nutzer von der Navigationsanwendung entlang der Route geleitet werden. Dazu ist es nötig, dass die Anwendung einen Bezug zwischen digitaler Karte und aktueller Position hat. Hierzu werden in der klassischen Navigation Map Matching Techniken eingesetzt, die unter anderem auch Messfehler in der Positionsbestimmung korrigieren können. Map Matching-Techniken wurden für die Kraftfahrzeugnavigation entwickelt. So ist ein Map-Matching bei einem Fußgänger wesentlich schwieriger, da die Bewegungsfreiheit sehr viel größer ist. Die grundsätzliche Verwendung von Map Matching ist sinnvoll, jedoch gibt es keine geeigneten Verfahren für Fußgänger. Es wird daher **kein Map-Matching** verwendet.

### Erreichen eines Wegpunktes

Bewegt sich der Fußgänger entlang einer Route, so soll geprüft werden, ob er den nächsten Wegpunkt erreicht hat und wo der nächste Wegpunkt ist, zu dem er gehen muss. Um dies zu können, ist es nötig zu wissen, wann ein Wegpunkt erreicht wurde. Hier erweist sich das **Schwellwertverfahren** als geeignete Lösung. Die Bestimmung eines geeigneten Schwellwertes ist jedoch schwierig. Hierbei können jedoch zusätzliche Daten, wie Informationen über den Straßentyp, helfen. Als einen konkreten Wert bietet sich die Summe aus Straßenbreite und der GPS-Abweichung an.

### Anzeigen des nächsten Wegpunktes

Somit kann ein Wegpunkt als erreicht erkannt werden und der nächste Wegpunkt wird angezeigt. Die Anzeige des nächsten Wegpunktes stellt eine weitere Aufgabe dar. Hier ist es zum einen möglich direkt auf den Wegpunkt zu zeigen oder einen Korrekturwinkel einzusetzen. Setzt man einen Korrekturwinkel ein, kann es sein, dass die Routenführung ab einem Zeitpunkt nicht mehr korrekt ist, wenn sich der Benutzer zu weit von der Route entfernt hat. Da dies gerade bei Fußgänger häufig vorkommen kann, kann hier nur die direkte Richtungsanzeige zum nächsten Wegpunkt verwendet werden. Es wird somit die **Richtung des Wegpunktes von der aktuellen Position aus** angezeigt. Wendet sich der Benutzer zu weit von der Route ab, so soll dies eine Neuberechnung der Route auslösen. Dies wird z.B. durch eine Abweichung von mehr als 100 Grad von der vorgegebenen Route ausgelöst. Auf Grund der baulichen Struktur einer Straße darf nicht sofort davon ausgegangen werden, dass sich der Nutzer von der Route entfernt hat. Somit wird eine Neuberechnung erst ausgelöst, wenn sich der Nutzer mehr als 100 Grad für zwei Minuten von der Route abgewandt hat.

### Zusammenfassung

Auf Grundlage der Straßendaten kann somit eine Routenführung bereitgestellt werden. Dazu werden die Straßendaten in einen Graphen überführt. Ergänzt mit der aktuellen Startposition und einem sphärisch berechneten Projektionspunkt, dient der Graph als Eingabe für den A*-Algorithmus, um den kürzesten Weg zu berechnen. Dieser kann dem Nutzer dann angezeigt werden, in dem die Richtung zum nächsten Wegpunkt angezeigt wird. Das Erreichen eines Wegpunktes wird über ein Schwellwertverfahren entschieden. Bewegt sich die Person jedoch zu weit von der Route weg, so wird eine Neuberechnung der Route ausgelöst.

# 4 Entwicklung der Navigationssoftware

Die im vorigen Kapitel vorgestellten Techniken sollen in einer Navigationssoftware realisiert werden. Dieses Kapitel beschreibt diese Implementierung. Für die Implementierung wird vorerst eine Anforderungsdefinition durchgeführt. In dieser werden die Akteure und Funktionen des Systems identifiziert, miteinander in Verbindung gesetzt und sukzessive verfeinert. Die Anforderungsdefinition beschreibt Zweck und Umfang des Systems. Die daraus entstehenden Anwendungsfälle fließen daraufhin zusammen mit den nichtfunktionalen Anforderungen in den Systementwurf ein. Unter Berücksichtigung und Vorstellung des vorhandenen Systems, dem NICCIMON-Framework, wird die Benutzerschnittstelle anhand von skizzierten Grafiken erläutert. Als Ergebnis dieser aufbauenden Betrachtung stehen am Ende Klassendiagramme für die konkrete Implementierung zur Verfügung.

## 4.1 Anforderungsdefinition

Dieser Abschnitt beschreibt die Anforderungen, die Merkmale und Einschränkungen, die die Software besitzen bzw. erfüllen soll. Für die Ermittlung der Anforderungen werden Anforderungstechniken eingesetzt. Die hier verwendete Technik lehnt sich an Brügge und Dutoit [BD04] an. Ausgehend von einigen konkreten Beispielszenarien und deren Akteuren werden Anwendungsfälle definiert, die als funktionale Anforderungen dienen. Ergänzt werden die funktionalen Anforderungen durch nichtfunktionale Anforderungen. Dieser werden, wie in [BD04] beschrieben, in Anlehnung an ISO 9126 identifiziert und beschrieben.

### 4.1.1 Identifizierung von Akteuren

Akteure im Sinne dieses System sind jegliche Entitäten, die mit diesem System interagieren. Entsprechend beschränkt sich die Auswahl der Akteure nicht nur auf den Benutzer des Systems. So interagiert das System auch mit dem Geoinformationssystem, welches die Straßendaten liefert. Ebenso braucht das System ein Positionssignal, um die aktuelle Position zu bestimmen.

- **Benutzer:** Der Benutzer ist die Person, die die Navigationsanwendung verwendet. Er ist demnach der Akteur, der die Hauptfunktionen des Systems nutzt. Er erstellt Routen und lässt sich diese anzeigen

- **Geoinformationssystem:** Das Geoinformationssystem beinhaltet alle nötigen Daten, wie z.B. Straßen, die für ein Routing notwendig sind. Das System greift somit auf das Geoinformationssystem zurück und interagiert mit diesem.

- **Positionssignal:** Das Positionssignal wird benötigt, damit die aktuelle Position festgestellt werden kann. Dieses befindet sich auch außerhalb des Systems und muss abgefragt werden.

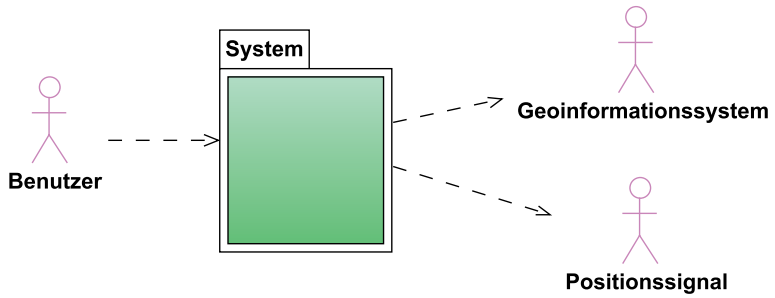


*Abbildung 4.1: Akteure des Systems*

### 4.1.2 Identifizierung von Szenarien

Szenarien beschreiben typische Funktionen an einem konkreten Beispiel. Sie sollen die Beschreibung von Anwendungsfällen nicht ersetzen, sondern diese zum besseren Verständnis ergänzen. Im Folgenden werden drei Szenarien, die die Hauptfunktionen illustrieren, beschrieben.

#### Szenario 1 - Routeneingabe

| | |
|---|---|
| *Akteur-Instanzen* | Marvin: Benutzer<br>GPS-Signal: Positionssignal |
| *Ereignisfluss* | **1.** Marvin befindet sich in der Uni. Er möchte von dort aus zum OFFIS laufen, kennt den Weg aber nicht.<br>**2.** Marvin startet die Navigationsanwendung. Dort gibt er seine aktuelle Position als Startadresse und die Adresse vom OFFIS als Ziel ein. Er klickt auf “Route berechnen”.<br>**3.** Die Navigationsanwendung berechnet anhand der eigenen Position bzw. dem aktuellen GPS-Signal und den vorhandenen Daten die Route und gibt diese auf dem Bildschirm aus. |

#### Szenario 2 - Kartenmaterial abrufen

| | |
|---|---|
| *Akteur-Instanzen* | Marvin: Benutzer<br>GeoServer: Geoinformationssystem |
| *Ereignisfluss* | **1.** Marvin möchte die Navigationsanwendung in Hamburg benutzen. Er sieht aber, dass die Navigationsanwendung noch keine Daten für Hamburg hat.<br>**2.** Marvin startet die Navigationsanwendung und wählt “Kartenmaterial verwalten”. In dieser Übersicht sieht er, dass er bisher nur die Sraßendaten für Oldenburg geladen hat. Er klickt dann auf “Neues Kartenmaterial hinzufügen”.<br>**3.** Marvin gibt Hamburg ein klickt auf “Kartenmaterial laden”.<br>**4.** Die Navigationsanwendung verbindet sich mit dem GeoServer und lädt die Daten für Hamburg herunter. Die Verwaltung des Kartenmaterials wird angezeigt und enthält nun zusätzlich einen Datensatz Hamburg. |

### Szenario 3 - Routenführung

| | |
|---|---|
| *Akteur-Instanzen* | Marvin: Benutzer<br>GPS: Positionssignal |
| *Ereignisfluss* | **1.** Marvin hat sich eine Route berechnen lassen. Diese möchte er nun ausgegeben bekommen.<br>**2.** Auf einem Bildschirm wird ihm eine Karte angezeigt. Ergänzt wird es durch die Richtung und Position eines Wegpunktes, zu dem er gehen soll.<br>**3.** Marvin folgt den Anweisungen der Navigationsanwendung. Er erreicht den ersten Wegpunkt. Das Programm zeigt ihm nun wiederrrum den nächsten Wegpunkt an. Marvin geht weiter.<br>**4.** Marvin geht solange weiter von Wegpunkt zu Wegpunkt, bis ihm die Navigationsanwendung sagt, dass das Ziel erreicht wurde.<br>**5.** Die Routenführung ist beendet. |

#### 4.1.2.1 Identifizierung von Anwendungsfällen

Ein Szenario, wie im vorigen Abschnitt beschrieben, ist eine Instanz eines Anwendungsfalls. Die Identifizierung des zugehörigen Anwendungsfalls definiert somit sämtliche Szenarien bzw. Instanzen dieses Anwendungsfalls. Als Grundlage für die Herleitung dienen die oben beschriebenen Szenarien. Die Szenarien werden entsprechend abstrahiert und durch Bedingungen ergänzt.

### Kartenmaterial anzeigen

Dieser Anwendungsfall beschreibt die Anzeige des Kartenmaterials, die bereits im System gespeichert sind. Kartenmaterial entspricht einer Menge von Straßen, die zu einer Adresse oder einem Ort zugeordnet sind.

| | |
|---|---|
| *Akteure* | Benutzer |
| *Ereignisfluss* | **1.** Der Benutzer aktiviert die “Kartenmaterial anzeigen”-Funktion auf seinem PDA.<br>**2.** Das System zeigt daraufhin eine Liste für den Benutzer an.<br>**3.** Der Benutzer kann in der Liste sehen, welches Kartenmaterial vorhanden sind.<br>**4.** Der Benutzer klickt auf einen Datensatz.<br>**5.** Das System zeigt Details zu dem ausgewählten Kartenmaterial an. |
| *Anfangsbedingungen* | Zum Anzeigen von Details ist mindestens ein Datensatz nötig. |
| *Abschlussbedingungen* | |
| *Qualitätsanforderungen* | |

### Kartenmaterial löschen

Kartenmaterial muss gelöscht werden können, damit unter anderem Speicherplatz gespart werden kann. Diese Funktion wird von diesem Anwendungsfall dargestellt.

| *Akteure* | Benutzer |
|---|---|
| *Ereignisfluss* | **1.** Der Benutzer aktiviert die “Kartenmaterial anzeigen”-Funktion auf seinem PDA.<br>**2.** Das System zeigt daraufhin eine Liste für den Benutzer an.<br>**3.** Der Benutzer kann in der Liste sehen, welches Kartenmaterial vorhanden ist.<br>**4.** Der Benutzer klickt auf einen Datensatz mit Kartenmaterial.<br>**5.** Das System zeigt Details zu dem ausgewählten Kartenmaterial an.<br>**6.** Der Benutzer klickt auf “Kartenmaterial löschen”.<br>**7.** Das System löscht das ausgewählte Kartenmaterial<br>**8.** Das System zeigt dem Benutzer eine neue Liste ein, in dem das gelöschte Kartenmaterial entfernt wurde. |
| *Anfangsbedingungen* | Es ist mindestens ein Datensatz mit Kartenmaterial vorhanden. |
| *Abschlussbedingungen* | Das Kartenmaterial ist (im Dateisystem) gelöscht. |
| *Qualitätsanforderungen* | |

### Kartenmaterial von Geoinformationssystem laden

Ist zu einem Zielgebiet noch kein Kartenmaterial vorhanden, so muss dieses in das System geladen werden. Nach Eingabe einer Adresse werden die Daten von dem Geoinformationssystem abgefragt und heruntergeladen.

| *Akteure* | Benutzer<br>Geoinformationssystem |
|---|---|
| *Ereignisfluss* | **1.** Der Benutzer aktiviert die “Kartenmaterial anzeigen”-Funktion auf seinem PDA.<br>**2.** Das System zeigt daraufhin eine Liste für den Benutzer an.<br>**3.** Der Benutzer kann in der Liste sehen, welches Kartenmaterial vorhanden ist.<br>**4.** Der Benutzer klickt auf “Neues Kartenmaterial laden”.<br>**5.** Das System öffnet ein Eingabeformular.<br>**6.** Der Benutzer gibt eine Adresse in das Eingabeformular ein und schickt es ab.<br>**7.** Das System empfängt die Formulardaten.<br>**8.** Das System schickt die Adressdaten an das Geoinformations-system.<br>**9.** Das Geoinformationssystem schickt das zu den Formulardaten passende Kartenmaterial an das System.<br>**10.** Das System speichert die Daten ab und zeigt dem Benutzer eine Liste an, in dem das neue Kartenmaterial hinzugefügt wurde. |

| | |
|---|---|
| *Anfangsbedingungen* | Die Konfigurationsdaten für das Geoinformationssystem wurden gesetzt und die Daten sind valide. |
| *Abschlussbedingungen* | Das empfangene Kartenmaterial ist (im Dateisystem) gespeichert. |
| *Qualitätsanforderungen* | |

### Ziel eingeben

Möchte der Benutzer eine neue Routenführung starten, so muss er vorher seine Zieladresse eingeben. Dieser Fall wird hierdurch spezifiziert.

| | |
|---|---|
| *Akteure* | Benutzer<br>Positionssignal |
| *Ereignisfluss* | **1.** Der Benutzer aktiviert die "Neues Ziel"-Funktion auf seinem PDA.<br>**2.** Das System zeigt daraufhin ein Eingabeformular.<br>**3.** Der Benutzer gibt eine Adresse in das Eingabeformular ein und schickt es ab.<br>**4.** Das System empfängt die Formulardaten.<br>**4a.** Das System empfängt die aktuelle Position vom Positionssignal<br>**5.** Das System errechnet eine Route zwischen der Adresse und der aktuellen Position.<br>**6.** Das System gibt eine "Route berechnet"-Meldung an den Benutzer aus.<br>**7.** Das System starten die Routenführung (Anwendungsfall "Karte und Routenwegpunkte anzeigen") |
| *Anfangsbedingungen* | Es sind Datensätze für die eingegebene Adresse vorhanden. |
| *Abschlussbedingungen* | Route ist berechnet |
| *Qualitätsanforderungen* | Errechnet die kürzeste bzw. schnellste Route |

### Karte und Routenwegpunkte anzeigen

Wenn eine Route berechnet wurde, dann wird sie dem Benutzer angezeigt. Dieser Anwendungsfall beschreibt die Anzeige der Routenführung. Bei der Routenführung wird anhand der aktuellen Position fortlaufend geprüft ob Wegpunkte erreicht werden. Ist ein Wegpunkt erreicht, wird der nächste angezeigt. Dieser Anwendungsfall ist beendet, wenn das Ziel erreicht wurde.

| | |
|---|---|
| *Akteure* | Benutzer<br>Positionssignal |
| *Ereignisfluss* | **1.** Das System hat eine Route berechnet.<br>**2.** Das System zeigt dem Benutzer eine Karte an. Die Route ist farblich hervorgehoben. Ebenso der nächste Wegpunkt.<br>**3.** Das Positionssignal wird an das System geschickt.<br>**4.** Das System zeigt die aktuelle Position auf der Karte an.<br>**5.** Schritt 3.-4. wird fortlaufend parallel ausgeführt. |

| | |
|---|---|
| | **6.** Der Benutzer geht zum nächsten Wegpunkt<br>**7.** Das System erkennt das Erreichen des Wegpunktes und zeigt den darauffolgenden Wegpunkt dem Benutzer an.<br>**8.** Wiederholung von 6.-7., bis der Endpunkt erreicht wurde.<br>**9.** System erkennt, dass das Ziel erreicht wurde und geht in den Anwendungsfall "Zielankunft anzeigen" über. |
| *Anfangsbedingungen* | Route wurde erfolgreich berechnet. |
| *Abschlussbedingungen* | Benutzer hat Ziel erreicht. |
| *Qualitätsanforderungen* | Wegpunkte sind erreichbar.<br>Taktile Ausgabe ist möglich. |

### Zielankunft anzeigen

Nachdem der Benutzer das Ziel erreicht hat, ist die Routenführung beendet. Die Zielankunft soll dem Benutzer angezeigt werden, wie durch diesen Anwendungsfall beschrieben wird.

| | |
|---|---|
| *Akteure* | Benutzer |
| *Ereignisfluss* | **1.** Das System hat erkannt, dass das Ziel erreicht wurde.<br>**2.** Der Benutzer bekommt eine Meldung, dass das Ziel erreicht wurde.<br>**3.** Der Benutzer bekommt die Möglichkeit eine neue Routenführung zu starten. |
| *Anfangsbedingungen* | Das Ziel wurde erreicht. |
| *Abschlussbedingungen* | |
| *Qualitätsanforderungen* | |

### Einstellungen ändern

Dem Benutzer soll die Möglichkeit gegeben werden, dass er Einstellungen ändern kann. Ihm werden dazu alle Einstellungsmöglichkeiten angezeigt.

| | |
|---|---|
| *Akteure* | Benutzer |
| *Ereignisfluss* | **1.** Der Benutzer aktiviert die "Einstellungen"-Funktion auf seinem PDA.<br>**2.** Das System zeigt daraufhin alle Einstellungsmöglichkeiten an.<br>**3.** Der Benutzer ändert die Einstellungen und speichert diese.<br>**4.** Das System empfängt die Änderungen.<br>**5.** Das System zeigt eine Meldung über Erfolg oder Misserfolg.<br>**6.** Das System zeigt das Hauptmenü. |
| *Anfangsbedingungen* | |
| *Abschlussbedingungen* | Einstellungen sind gespeichert |
| *Qualitätsanforderungen* | Einstellungen, welche Formal nicht korrekt sind, sind nicht möglich. Der Benutzer wird darauf hingewiesen. |

### 4.1.3 Identifizierung von Beziehungen zwischen Akteuren und Anwendungsfällen

Aufbauend auf der Identifizierung von Anwendungsfällen werden die Kommunikations-Beziehungen zwischen den Akteuren und den Anwendungsfällen identifiziert. Im Anwendungsfalldiagramm 4.2 werden diese Beziehungen dargestellt. Ebenso stellt dieses Diagramm die Beziehungen zwischen den Anwendungsfällen dar. Ein Anwendungsfall “Kartenmaterial löschen” setzt den Anwendungsfall “Kartenmaterial anzeigen” voraus. Analog ist für den Anwendungsfall “Karte und Routenwegpunkte anzeigen” der Anwendungsfall “Ziel eingeben” nötig. Existiert bei dem Anwendungsfall ”Ziel eingeben“ ein benötigter Datensatz nicht, so löst dies den Anwendungsfall “Kartenmaterial von Geoinformationssystem laden” aus.

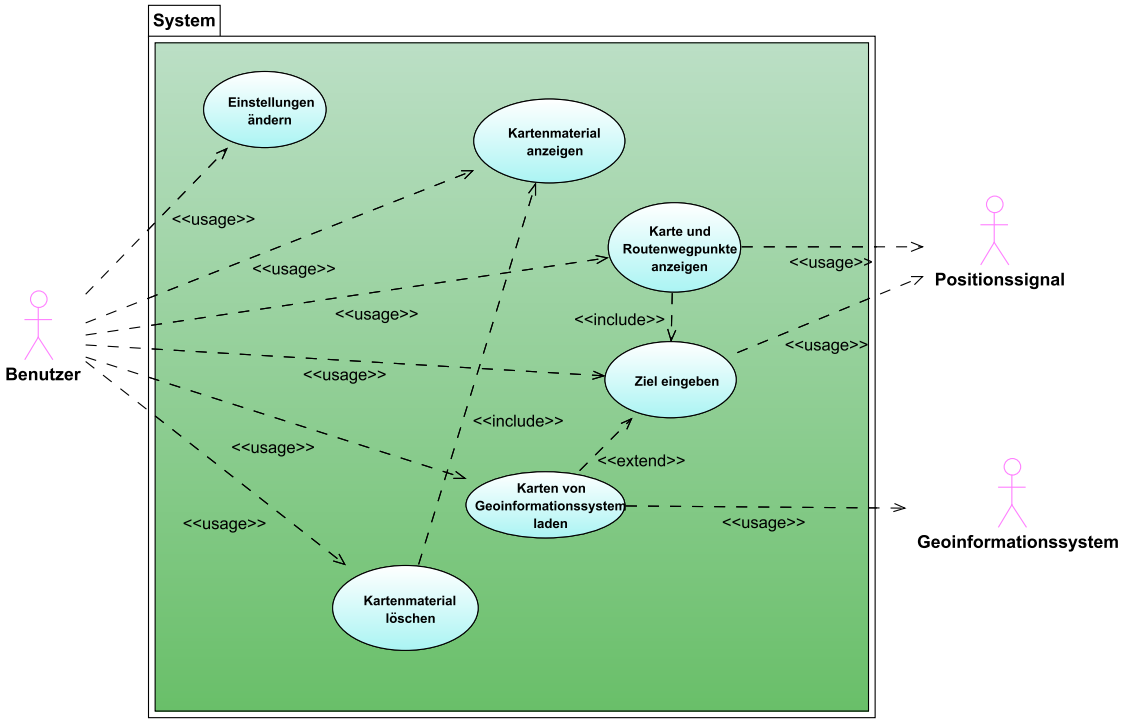


*Abbildung 4.2: Anwendungsfalldiagramm*

### 4.1.4 Identifizierung von nichtfunktionalen Anforderungen

Nichfunktionale Anforderungen betrachten Aspekte des Systems, die nicht direkt zu einem funktionalen Verhalten stehen. Sie beeinflussen jedoch den Benutzer und sein Arbeitsverhalten. Im Folgenden werden Punkte der FURPS+-Methode[1], welche an ISO 9126 angelehnt ist, betrachtet [Eel05][BD04].

[1] Functionality, Usability, Reliability, Performance, Supportability + Design, Implementation, Interface and Physical requirements

### Benutzerfreundlichkeit

Der Benutzer einer Navigationsanwendung ist im Normalfall kein Experte aus dem Bereich der Informatik. Da das Programm auf einem PDA verwendet wird, kann davon ausgegangen werden, dass sich der Kenntnisstand des Benutzers auf die Verwendung von PDAs beschränkt. Unter Umständen sind Erfahrungen aus Navigationsgeräten für Kraftfahrzeuge vorhanden. Ein Benutzer muss demnach das System leicht bedienen können. Das heißt, dass Icons und Texte eindeutig zu identifizieren sind, indem aussagekräftige Bezeichnungen und Symbole verwendet werden.

### Zuverlässigkeit

Zuverlässigkeit ist keine notwendige Bedingung, da es sich nicht um ein sicherheit- oder zeitkritisches System handelt. Dies beinhaltet auch die Robustheit des Systems. Tritt ein Fehler auf, so dass ein Systemneustart erforderlich ist, so wird die aktuelle Position neu ermittelt und die Routenführung kann neu gestartet werden. Zuverlässigkeit und Robustheit sind somit keine notwendigen Anforderungen, sind jedoch für die Benutzerfreundlichkeit förderlich. Sicherheitsaspekte müssen bei diesem System nicht berücksichtigt werden.

### Leistung

Das System soll während der Routenführung die aktuelle Position wiedergeben. Entsprechend darf das System mit der Anzeige der aktuellen Position nicht zu langsam reagieren, so dass z.B. alle drei Sekunden ein *springen* des Positionspunktes von mehreren Metern auf dem Bildschirm erzeugt wird. Das Positionssignal soll daher in einem Intervall von 1000ms erneuert werden. Hierbei handelt es sich jedoch nicht um zeitkritische Aufgaben. Die Berechnung einer Route soll in einer für den Benutzer angemessenen Zeit durchgeführt werden. Hierbei sind jedoch einige Sekunden akzeptabel. Hierbei darf die Benutzeroberfläche nicht blockieren. Die Berechnung soll daher im Hintergrund durchgeführt werden. Die Leistung hängt stark von dem gegebenen System ab. Da es sich hier um mobile Endgeräte handelt, sind die geringeren Resourcen gesondert zu beachten.

### Unterstützung

Die Unterstützung im Sinne der Wartbarkeit betrachtet zukünftige Erweiterungen und Fehlerkorrektur des Systems. Hierbei sollen Schnittstellen und eine modulare Aufbauweise mögliche Erweiterungen und Änderungen erleichtern. Zuzüglich soll eine interne Qualität, wie z.B. Codedokumentation, dies unterstützen.

### Implementierung

Die Implementierung ist für mobile Endgeräte vorgesehen. Das Betriebssystem ist Windows Mobile 6. Diese Plattform stellt besondere Anforderungen an die Implementierung. Bei der Verwendung des .NET Compact Frameworks stehen im Vergleich zum .NET Framework nur einen Teil der Funktionen zur Verfügung. Des Weiteren gibt es Beschränkungen und zusätzliche Optionen durch das NICCIMON-Framework, das verwendet werden muss.

### Schnittstelle

Das System soll mit einem Geoinformationssystem, wie in Kapitel 2.3 beschrieben, interagieren können. Demnach ist eine Schnittstelle zum Web Feature Service des Geoinformationssystem nötig, so dass die Daten abgefragt werden können. Weiterhin müssen Schnittstellen zu anderen NICCIMON-Funktionen geschaffen werden, so das unter anderem das Positionssignal genutzt werden kann, dass von NICCIMON bereitgestellt wird.

### Betrieb, Installation und Rechtliches

Betrieb und Installation kann von jedem Benutzer durchgeführt werden. Dabei soll das System auf ein PDA kopiert werden können und ohne zusätzliche Konfigurationen betriebsbereit sein. Das Programm kann dazu beliebig vervielfältigt werden, wenn es die Lizenzrechte erlauben. Die Lizenzrechte dieser Anwendung werden durch die Lizenzrechte des NICCIMON-Frameworks vorgegeben oder entsprechen denen der verwendeten Komponenten von Drittanbietern.

## 4.2 Entwurf

Die Anforderungsdefinition ist die Grundlage für den Entwurf der Navigationssoftware. Das folgende Kapitel wird auf Grundlage der Anforderungsdefinitionen schrittweise den Entwurf aufbauen. Dazu wird zuerst die Grundlage für den Systementwurf vorgestellt. Dieser beruht auf dem NICCIMON-Framework, dessen Konzept und Struktur dazu vorgestellt wird. Unter Hilfe diesen Wissens wird aus der Anforderungsdefinition heraus ein Objektmodell generiert. In diesem Objektmodell stehen die wichtigsten Entitäten des Systems. Diese dienen daraufhin für die Erstellung der Klassendiagramme.

### 4.2.1 Vorhandenes System

Das vorhandene System beruht auf der Vorgabe, dass die Navigationsanwendung das NICCIMON-Framework nutzen soll. NICCIMON ist ein modulares Framework zum Rapid Prototyping von mobilen, ortsbezogenen Anwendungen. NICCIMON bietet die Möglichkeit Module zu entwickeln, die über das Framework miteinander kommunizieren können. Das Framework basiert auf dem Vermittler-Entwurfsmuster (Mediator-Design-Pattern), so dass die Module über einen Vermittler miteinander kommunizieren können, ohne dass sie direkt aufeinander zugreifen müssen. NICCIMON stellt bereits zwei solcher Module als Basis bereit. Das Positions-Modul ermöglicht den Zugriff auf die aktuelle Position, welches das Positions-Modul über den GPS-Empfänger ermittelt. Als zweites Basismodul steht das Map-Modul zur Verfügung, mit dem eine Umgebungskarte zu der aktuellen Position auf dem Bildschirm ausgegeben werden kann [Wic08]. Abbildung 4.3 zeigt den modularen Aufbau von NICCIMON. Neben den Modulen bietet das Framework verschiedene Funktionen, die den Entwurf einer konkreten Anwendung erleichtern. So gibt es z.B. Funktionen zur Berechnung von sphärischen Distanzen oder Datenstrukturen für ortsbezogene Daten.

Das System wird eine konkrete Implementierung des Vermittlers sein, da dies vom Framework vorgegeben wird. Dabei bindet es die entsprechenden Module ein, so dass das Framework für die Kommunikation zwischen den Modulen vermitteln kann. Da sowohl die aktuelle Position, als auch eine Kartenausgabe nötig ist, wird hier auf die vorhandenen Module zurückgegriffen. Das Positions-Modul schickt über ein *PositionEvent* die aktuelle Position an den Vermittler. Ein Modul, welches

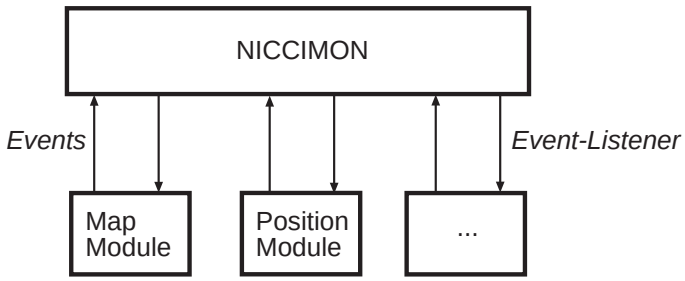


*Abbildung 4.3: Modul-Konzept von NICCIMON*

nun diese Position nutzen möchte, registriert sich für dieses Event beim Vermittler mit einem Event-Listener an und bekommt dann das *PositionEvent*, wenn es vom Positions-Modul ausgelöst worden ist. Dies verwendet unter anderem das Map-Modul, dass die aktuelle Position auf einer Karte darstellt. Die Karteanzeige besteht dabei aus mehreren übereinander liegenden Ebenen, den Layers. So existiert z.B. ein Layer für die aktuelle Position und ein Layer für die eigentliche Karte. Diese Karte kann wiederum ausgetauscht werden. Dies ermöglicht z.B. das wechseln zwischen Straßenkarten und Satellitenbildern. Für diese Anwendung muss das Map-Modul zusätzlich noch um einen Layer für die Routenführung erweitert werden. Ebenso müssen die entsprechenden Ausgaben und Interaktionsmöglichkeiten bereitgestellt werden. Für die Erweiterung dieser Basisfunktionalitäten des Frameworks, stellt NICCIMON Abstrakte Klassen und Interfaces als Schnittstelle zur Verfügung, so dass der modulare Aufbau gewährleistet wird.

### 4.2.2 Objektmodell

Das Objektmodell baut auf den Anwendungsfällen und den Szenarios auf. In den folgenden Schritten dient das Objektmodell als Grundlage für den Entwurf und wird sukzessive weiter aufgebaut. Das Objektmodell besteht aus Entitäts-, Grenz und Steuerungsobjekten. Entitäten repräsentieren Informationen im System, die sich während der Laufzeit verändern können. Grenzobjekte stellen die Interaktion zwischen den Akteuren und dem System bereit. Die Steuerungsobjekte stellen eine abstrakte Repräsentation der Anwendungsfälle dar.

| Entitätsobjekte | Grenzobjekte | Steuerungsobjekte |
|---|---|---|
| Route | Zieleingabeformular | Kartenmaterial verwalten |
| Datensatz | Kartenmaterialanzeige | Ziel eingeben |
| Wegpunkt | Kartenmaterialabruf | Route und Karte anzeigen |
| Adresse | Straßen- und Routenkarte | Einstellungen ändern |
| | Einstellungen | Kürzesten Weg finden |

### 4.2.3 Benutzerschnittstelle

Die Benutzerschnittstelle wird durch die Ausgabe auf dem Bildschirm des mobilen Endgeräts realisiert. Hierbei gibt es je nach Aufgabe verschiedene Ausgaben. Im Folgenden sollen diese Ausgaben durch Mock-ups skizziert werden, so dass die Anforderungen an die Benutzeroberfläche visualisiert werden. Sie sind an den Grenzobjekten des Objektmodells angelegt.

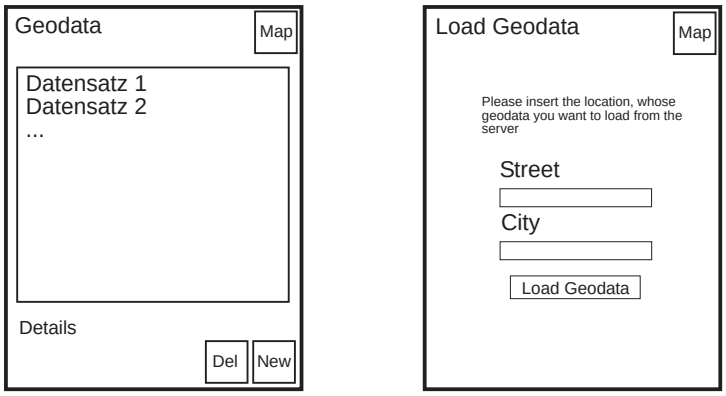


*Abbildung 4.4: Mock-up von Kartenmaterial verwalten und Neues Kartenmaterial laden*

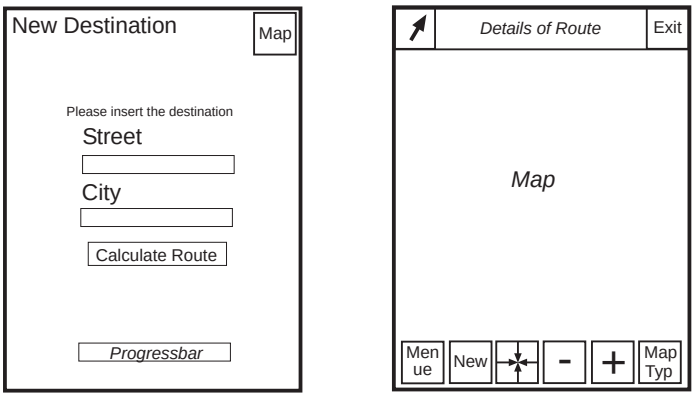


*Abbildung 4.5: Mock-up von Neues Ziel eingeben und Karte und Route anzeigen*

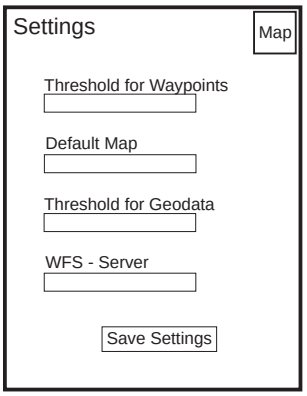


*Abbildung 4.6: Mock-up von Einstellungen*

### 4.2.4 Systementwurf

Zusammen mit der Benutzerschnittstelle bildet das Objektmodell die Grundlage für den Entwurf des Systems. Die darin genannten Entitäten und Objekte finden sich im Folgenden als Komponenten bzw. Klassen wieder. Hierbei wird bereits eine Abgrenzung innerhalb des Systems vorgenommen, so dass die jeweiligen Funktionen und Eigenschaften modular sind. Dies ermöglicht den unabhängigen Einsatz einzelner Module für weitere zukünftige Entwicklungen. Wie sich aus den Anwendungsfällen und dem Objektmodell feststellen lässt, gibt es drei Bereiche, die dem Benutzer angeboten werden. Zum einen gibt es die Verwaltung von Kartenmaterial. Zum anderen gibt es die Routen-Funktion der Anwendung. Des Weiteren gibt es einen zentralen Bereich, in dem alle Funktionen miteinander verknüpft werden. Diese Bereiche sollen entsprechend dem modularen Gedanken von NICCIMON getrennt betrachtet werden. Dazu wird das Laden und Verwalten von Kartenmaterial in einem eigenständigen Modul, dem Vektor-Import-Modul realisiert. Weiter wird die Berechnungen von Routen ebenso als Routing-Modul entworfen, damit dies für weitere Module oder Anwendungen zur Verfügung steht. Die Implementierung des zentralen Bereiches entspricht der konkreten Implementierung des Vermittlers, welches als Routing-Anwendung verwirklicht wird.

#### Vektor-Import-Modul

Das Vektor-Import-Modul stellt die Funktionen für den Import von Geodaten bereit. Das Modul kommuniziert somit mit dem Geoinformationssystem. Es realisiert demnach den in Kapitel 2.3 bzw. 2.5 genannten WFS-Client. Er beinhaltet demnach die Möglichkeit eine WFS-Anfrage an den Server

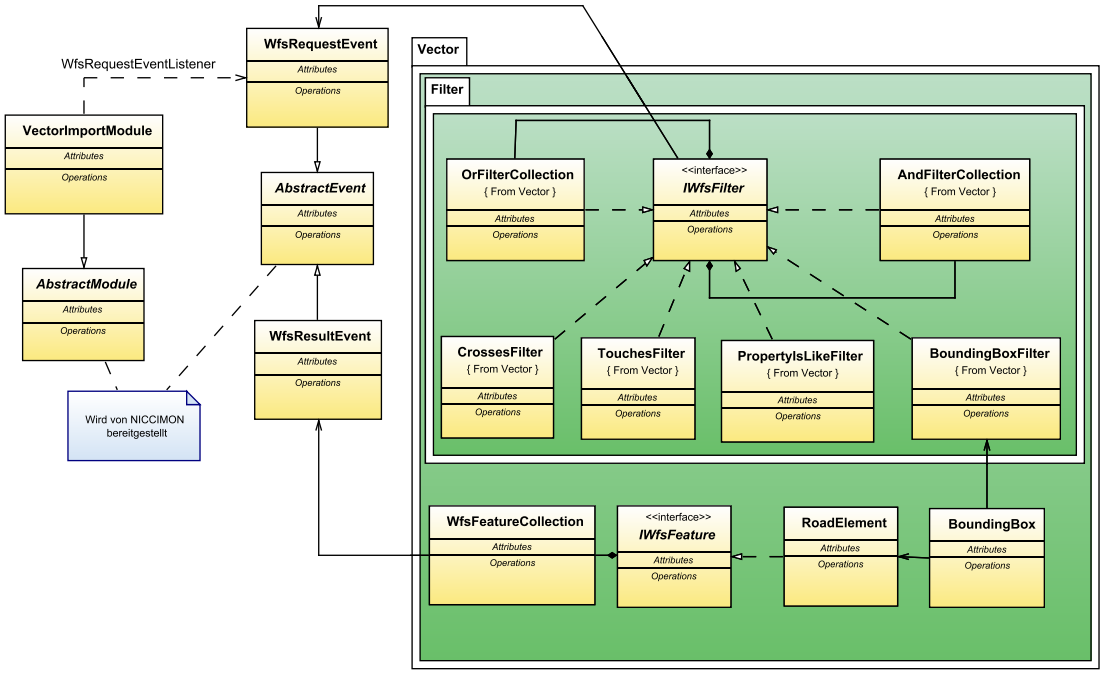


*Abbildung 4.7: Klassendiagramm des Vektor-Import-Moduls*

zu senden und die Ergebnisse zu empfangen. Dazu hört das Vektor-Import-Modul auf ein *WfsRequestEvent*. Ein solches *WfsRequestEvent* beinhaltet verschiedene Filter, die für die Eingrenzung der Ergebnismenge dienen. Nachdem das Vektor-Import-Modul ein *WfsRequestEvent* vom Vermittler empfangen hat, ruft es den entsprechenden Service beim Geoinformationssystem auf und fragt die geforderten Daten ab. Die Antwort des Geoinformationssystems wird daraufhin in eine Objektstruktur übersetzt. Diese Daten werden dann in ein *WfsResultEvent* verpackt und an den Vermittler geschickt. Über den Vermittler kann dann wieder das Modul, welches den *WfsRequestEvent* initiiert hat, das Ergebnis der Abfrage bekommen, in dem es selbst den *WfsResultEvent* abonniert hat. Abbildung 4.7 stellt das Klassendiagramm des Vektor-Import-Moduls dar. Zu sehen sind dort die Abstrakten Klassen, die von NICCIMON bereitgestellt werden und entsprechend von diesem Modul implementiert werden.

## Routing-Modul

Dieses Modul stellt die Methoden zur Berechnung von Routen bereit. Dazu hat das Modul das *RoutingRequestEvent* abonniert. Ein *RoutingRequestEvent* enthält dabei einen Graphen, der wiederrum aus mehreren Knoten und Kanten besteht. Zusätzlich gibt es einen Start- und einen Endknoten. Diese Daten werden dem Kürzesten-Wege-Algorithmus, hier der A*-Algorithmus, übergeben. Dieser errechnet den kürzesten Pfad, die Route zwischen Startknoten und Endknoten, auf dem übergebenen Graphen. Das Ergebnis dieser Berechnung wird als *RoutingResultEvent* an den Vermittler geschickt. Abbildung 4.8 zeigt das Klassendiagramm für das Routing-Modul.

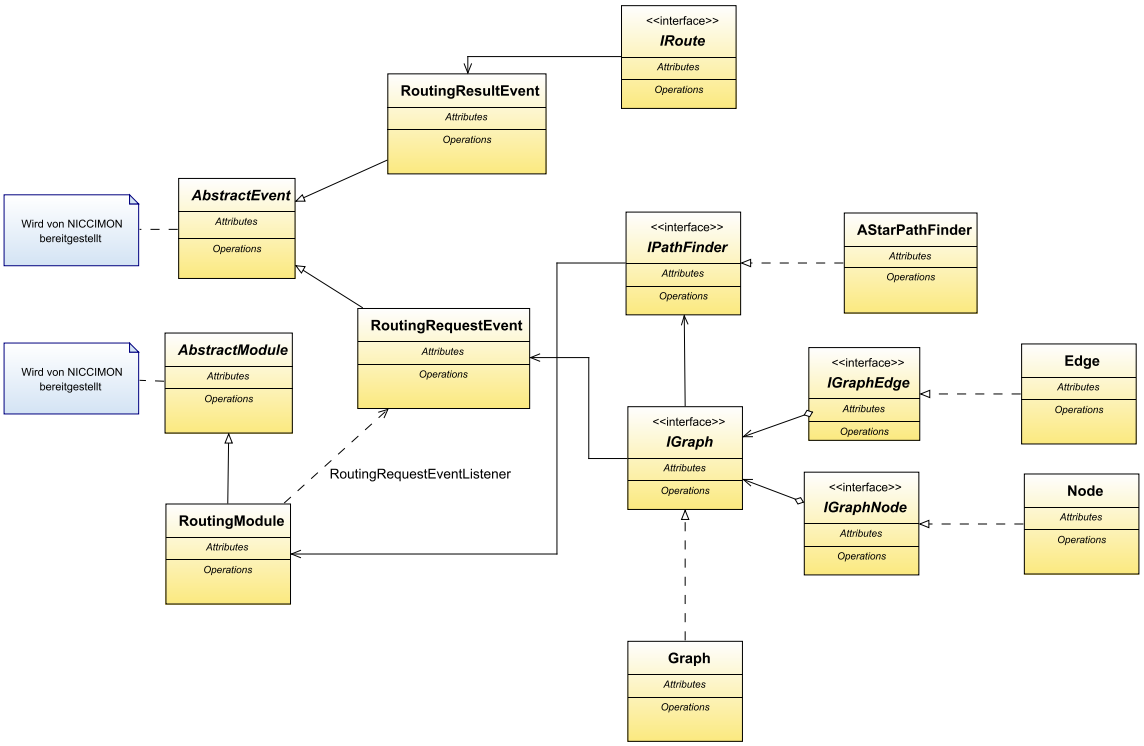


*Abbildung 4.8: Klassendiagramm des Routing-Moduls*

### Routing-Anwendung

Die Routing-Anwendung ist die konkrete Realisierung des Vermittlers. Es implementiert dazu das Interface *IMediator*, welches die Schnittstelle eines Vermittlers in NICCIMON darstellt. NICCIMON bietet hierzu die *AbstractPrototypApplication* als abstrakte Klasse, welches bereits eine rudimentäre Implementierung von *IMediator* ist. Die *RoutingPrototypApplication* lädt und verwaltet die Module der Anwendung. Das Map-Modul, welches als Basismodul schon zur Verfügung steht, reicht für diese Anwendung nicht aus, da unter anderem die Darstellung und Optionen in der Karte erweitert werden müssen. Dazu gibt es das *RasterDataMapModule*, welches eine abstrakte Klasse des Basis-Map-Moduls um entsprechende Funktionalitäten erweitert. Neben der Verwaltung der Module, ist die *RoutingPrototypApplication* für die Darstellung und Funktionen der einzelnen Eingabeformulare, also den Grenzobjekten des Objektmodells, und deren verschiedenen Aufgaben, also den Steuerungsobjekten, zuständig. In Abbildung 4.9 ist das Klassendiagramm zu sehen. Dabei wird visualisiert, dass die Module unabhängig von der Anwendung nur über Events miteinander kommunizieren.

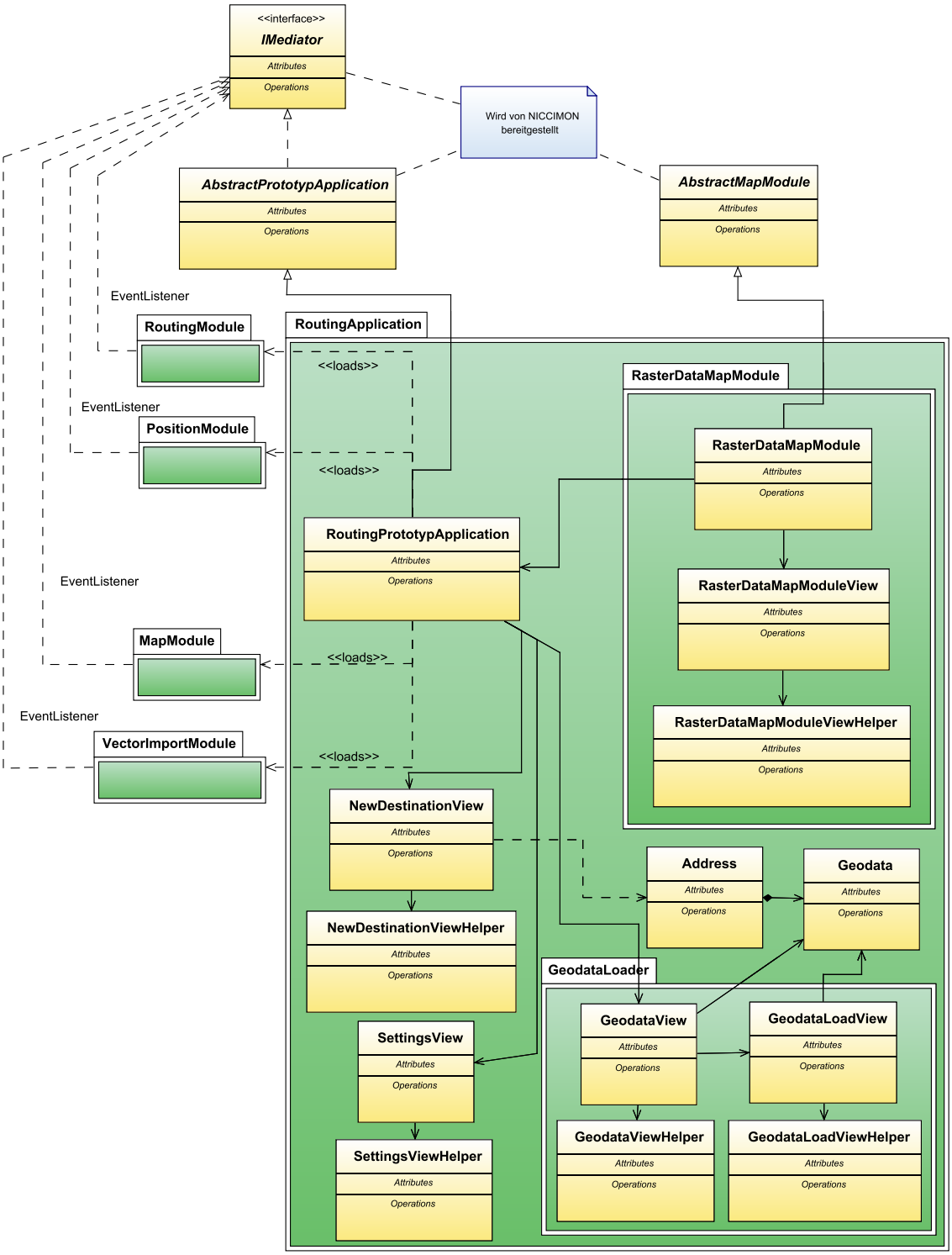


*Abbildung 4.9: Klassendiagramm der Routing-Application*

## 4.3 Die Navigationsanwendung

Im Folgenden wird auf die Entwicklung der Navigationsanwendung eingegangen. Dazu werden die wichtigsten Funktionen der Anwendung vorgestellt. Ausgehend von der Benutzerinteraktion wird auch die interne Funktionweise kurz erklärt. Ergänzend wird die Einrichtung der verwendeten Geodateninfrastruktur erläutert.

### 4.3.1 Kartenmaterial verwalten

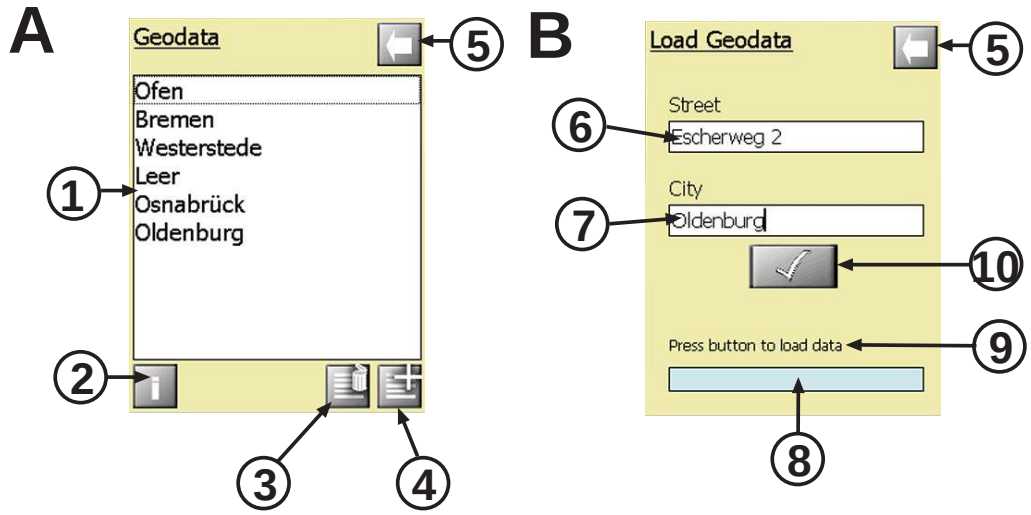


*Abbildung 4.10: Screenshot der Kartenmaterialverwaltung und Laden neuer Daten*

Abbildung 4.10 zeigt die Verwaltung des Kartenmaterials. Dabei teilt sich diese Verwaltung in zwei Teile. Zum einen werden existierende Datensätze angezeigt und zum anderen können neue Datensätze hinzugefügt werden. Dabei besteht die Bildschirmausgabe aus folgenden Funktionen:

1. *Datenliste*: Die Liste der zur Verfügung stehenden Kartenmaterialdatensätze.
2. *Information anzeigen*: Hierüber können Informationen zu dem ausgewählten Kartenmaterialdatensatz angezeigt werden.
3. *Kartenmaterial löschen*: Über diesen Button kann ein ausgewählter Datensatz gelöscht werden.
4. *Neues Kartenmaterial laden*: Dieser Button führt zu der Ansicht “Load Geodata”.
5. *Zurück*: Mit diesem Button gelangt man zurück zur Hauptansicht.
6. *Eingabe der Straße*: Möchte man einen Datensatz laden, so muss hier die Straße angegeben werden.
7. *Eingabe des Ortes*: Möchte man einen Datensatz laden, so muss hier der Ort angegeben werden.
8. *Fortschrittsanzeige*: Dieser Balken zeigt den Fortschritt des Ladens an.

9. *Statustext*: Dieser Text gibt Auskunft über den aktuell durchgeführten Schritt.

10. *Kartenmaterial laden*: Nach dem Straße und Ort eingegeben worden sind, startet die Abfrage der Daten durch Betätigung dieses Buttons.

In der *Datenliste* (1) stehen alle Datensätze, die zur Zeit im System gespeichert sind. Nach der Auswahl eines Datensatzes, können über die Funktion *Information* (2) Details über den ausgewählten Datensatz angezeigt werden. Über den Button *Kartenmaterial löschen* (3) kann der ausgewählte Datensatz gelöscht werden. Der Benutzer muss das Löschen zusätzlich bestätigen. Fehlt ein Datensatz in der Liste, so kann der Benutzer über den Button *Neues Kartenmaterial laden* (4) in die Ansicht *Load Geodata* (B) wechseln. In dieser Ansicht muss eine Straße (6) und ein Ort (7) angegeben werden. Die Anfrage kann nun über den Button (10) abgeschickt werden. Die Anwendung schickt daraufhin eine Anfrage an den Server, bei dem die eingegebene Adresse in ein geographisches Datum übersetzt wird. Ausgehend von diesem Datum wird eine Bounding Box festgelegt. Diese Bounding Box wird als Filter über ein WfsRequestEvent an das Vektor-Import-Modul geschickt. Das Vektor-Import-Modul fragt daraufhin alle Straßen vom GeoServer ab, die sich innerhalb der angegebenen Bounding Box befinden. Das Vektor-Import-Modul schickt die abgerufenen Daten danach als WfsResultEvent über den Vermittler zurück an die Anwendung. Die Anwendung generiert aus der Ergebnisliste einen Graphen, wie in Abschnitt 3.1.2 beschrieben. Dieser Graph wird in eine XML-Struktur übersetzt, die daraufhin komprimiert im Dateisystem gespeichert wird. Damit ist das Laden der neuen Daten beendet und die Anwendung wechselt in die Übersicht für das Kartenmaterial (A). Der Benutzer kann die Ansicht über den Zurück-Button (5) verlassen und gelangt zurück zur Kartenansicht.

### 4.3.2 Neues Ziel & Karte und Route anzeigen

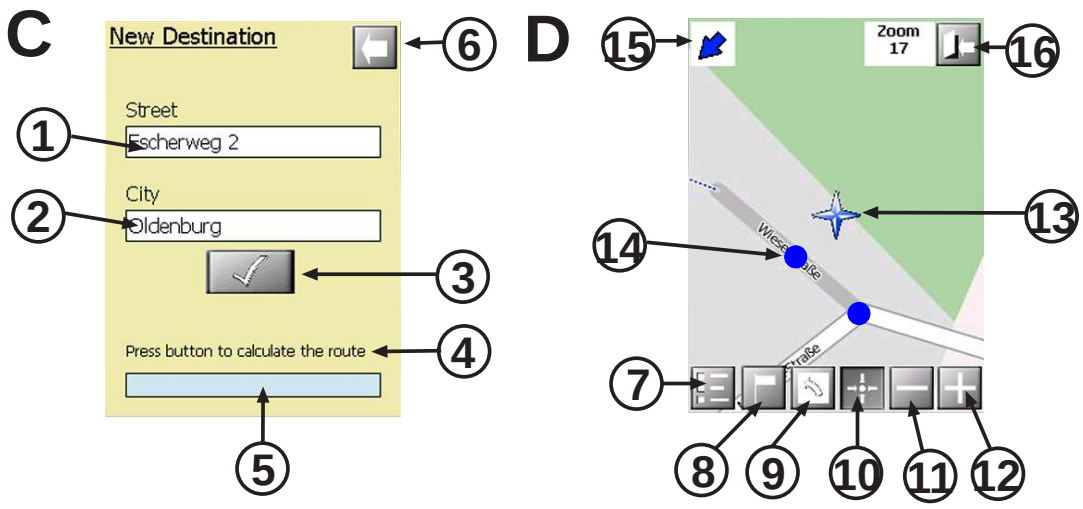


*Abbildung 4.11: Screenshot bei Eingabe eines neuen Ziels und Anzeige der Karte und Route*

In Abbildung 4.11 wird das Routing dargestellt. Hierbei ist es möglich ein neues Ziel einzugeben und die Route berechnen zu lassen. Diese Route wird daraufhin in der Kartenansicht angezeigt. Die Benutzerschnittstelle verfügen dabei über folgende Komponenten:

1. *Eingabe der Straße*: Die Straße des Ziels.
2. *Eingabe des Ortes*: Der Ort des Ziels. Dieser bezieht sich auf einen geladenen Datensatz.
3. *Route berechnen*: Dieser Button löst die Berechnung der Route aus.
4. *Statustext*: Dieser Text zeigt Details über die Berechnungsschritte.
5. *Fortschrittsanzeige*: Dieser Balken zeigt den Fortschritt der Berechnung an.
6. *Zurück*: Mit diesem Button gelangt man zurück zur Hauptansicht (D).
7. *Untermenü*: Dieser Button öffnet das Untermenü. Vom Untermenü aus kann zu der Ansicht des Positions- und Vektor-Import-Moduls gewechselt werden. Außerdem kann hier die Kartenmaterialverwaltung (A) aufgerufen werden.
8. *Neues Ziel*: Diese Funktion wechselt zu der Ansicht (C), bei dem ein neues Ziel eingegeben werden kann.
9. *Karte wechseln*: Wechselt die Karte zwischen einer Straßen- und Satellitenkarte.
10. *Position verfolgen*: Aktiviert bzw. deaktiviert, dass die Kartenansicht auf die aktuelle Position des Benutzers zentriert wird.
11. *Zoom out*: Verkleinert den angezeigten Kartenausschnitt.
12. *Zoom in*: Vergrößert den angezeigten Kartenausschnitt.
13. *Position*: Zeigt die aktuelle Position des Benutzers an.
14. *Wegpunkt*: Visualisiert die Wegpunkte der Route.
15. *Richtung*: Zeigt die Richtung des nächsten Wegpunktes ausgehend von der aktuellen Position an.
16. *Beenden*: Beendet die gesamte Anwendung.

Der Benutzer befindet sich zu Anfang in der Kartenansicht (D). In dieser Ansicht sieht er seine aktuelle Position(13) in der Karte. Möchte der Benutzer eine neue Routenführung starten, so ist dazu die *Route berechnen*(8) zu betätigen. Daraufhin wechselt die Ansicht zur Zieleingabe (C). Dort gibt der Benutzer Straße (1) und Ort (2) des Ziels ein und startet die Berechnung (3). Das System öffnet das Kartenmaterial, welches dem angegebenen Ort zugeordnet ist. Dort liegt, wie im vorigen Abschnitt beschrieben, bereits ein Graph vor. Zu diesem Graphen wird die aktuelle Position als Startknoten hinzugefügt. Dazu wird der in Kapitel 3.2.1 beschriebene sphärische Lotfußpunkt bestimmt. Der daraus entstehende Graph wird dann als RoutingRequestEvent an das Routing-Modul geschickt. Das Routing-Modul errechnet daraufhin mit dem A*-Algorithmus den Kürzesten-Pfad von Startknoten zum Zielknoten und schickt das Ergebnis als RoutingResultEvent zurück. Im unteren Teil kann der Status (4),(5) der Berechnung verfolgt werden. Anschließend wird die Route als Wegpunkte (14) auf der Karte angezeigt. Zusätzlich zeigt ein Pfeil (15) die Richtung zum nächsten Zielpunkt an. Der Benutzer kann nun die berechnete Route verfolgen, in dem er jeweils von Wegpunkt zu Wegpunkt geht, bis er den Zielwegpunkt erreicht hat. Die Routenführung ist dann beendet.

### 4.3.3 Einrichtung der Geodateninfrastruktur

Für die Funktionstüchtigkeit der Navigationsanwendung als Client in der Geodateninfrastruktur (GDI) sind zusätzlich das Geoinformationssystem (GIS) und Daten aus OpenStreetMap notwendig. Als GIS wurde, wie in Kapitel 2 erläutert, der GeoServer gewählt. Für die Installation des GeoServers dient ein Linux-System mit installiertem Java. Nach dem Entpacken des Paketes kann der GeoServer über ein Skript direkt gestartet werden und ist damit einsatzbereit. Über das Archiv[2] der Firma Geofabrik stehen Shapefiles zur Verfügung, welche aus den Daten von OpenStreetMap generiert wurden. Die Shapefiles von Europa wurden dazu heruntergeladen und in das Datenverzeichnis des GeoServers abgelegt. Über das Webinterface des GeoServers wurde daraufhin ein *DataStore* eingerichtet und das Shapefile importiert. Ausgehend von diesem *DataStore* wurde ein *FeatureType* generiert. Die Daten des Shapefiles sind nun als *topp:roads* dem WFS-Server bekannt und können von der Navigationsanwendung abgefragt werden.

[2] `http://download.geofabrik.de/osm/`

# 5 Zusammenfassung und Ausblick

Ziel dieser Arbeit war die Implementierung einer Navigationsanwendung für Fußgänger, die eine standardisierte Geodateninfrastruktur nutzt. Dazu wurden Probleme und Lösungsansätze untersucht, die bei einer Navigation für Fußgänger vorkommen. Diese Probleme lassen sich auf die Mobilität eines Fußgängers zurückführen. Ein Fußgänger kann sich in seiner Umgebung frei bewegen. So ist es ihm möglich sich zwischen bekannten Straßen zu bewegen, Kurven zu schneiden oder Abkürzungen zu nutzen. Auf Grund dieser freien Beweglichkeit sind klassische Techniken aus der Kraftfahrzeugnavigation ungeeignet. Die gesamte Anwendung gliedert sich dabei in drei Komponenten, deren Techniken und Probleme untersucht wurden. Ausgehend von der Geodatenbereitstellung wird eine Berechnung durchgeführt, deren Ergebnis dann in der Routenführung verwendet wird.

Für die Bereitstellung der Geodaten wurde eine Geodateninfrastruktur (GDI) untersucht. Obwohl es kommerzielle Anbieter mit sehr genauem Kartenmaterial gibt, fiel hier die Wahl auf OpenStreetMap, da diese Daten kostenfrei zur Verfügung stehen. OpenStreetMap selbst ist nicht vollständig fehlerfrei, da das Kartenmaterial von derzeit über 55.000 Freiwilligen erfasst wird. Des Weiteren sind zwar größere urbane Umgebungen sehr detailiert verfügbar, aber gerade ländliche Umgebungen sind teilweise gar nicht verzeichnet oder es gibt nur eine Bundes- oder Landstraße. Die Daten von OpenStreetMap wurden im nächsten Schritt in ein Geoinformationssystem (GIS) importiert. Dazu dient das Shapefile, welches ein Quasi-Standard ist, als Austauschformat. Obwohl der UMN MapServer vergleichbar ist, diente hier der GeoServer als GIS, da dieser den Anforderungen entsprach und zusätzlich noch die Referenzimplementation des Web Feature Service (WFS) ist. Dieser WFS ist ein OGC-Standard, der eine standardisierte Schnittstelle für vektorbasierte Geodaten bereitstellt. Dieser Standard wird den Clients in der GDI zur Verfügung gestellt, um Geodaten abzurufen. Einer dieser Clients ist die Navigationsanwendung.

Die Navigationsanwendung fragt das benötigte Kartenmaterial über den WFS vom GIS ab. Die Daten liegen hier als eine Menge von Straßen vor. Jede Straße besteht dabei aus einer Liste von Wegpunkten. Um jedoch den kürzesten Weg zwischen zwei Wegpunkten zu finden, müssen die Wegpunkte miteinander verknüpft sein. Dazu werden die Wegpunkte in einen zusammenhängenden Graphen überführt. Auf diesem Graphen kann nun das Single Pair Shortest Path Problem, bei dem der kürzeste Weg von einem Start- zu einem Zielknoten berechnet wird, gelöst werden. Dazu kann der A*-Algorithmus verwendet werden. Dieser hat im besten Fall eine Laufzeit von $\mathcal{O}(n + m)$ bei $n$ Knoten und $m$ Kanten. Der vom A*-Algorithmus berechnete kürzeste Pfad wird daraufhin für die Routenführung verwendet. Dazu muss die aktuelle Position mit dem Straßennetz verknüpft werden. Dabei wird die aktuelle Position auf die nächstgelegene Straße projiziert. Um die Kugelform der Erde zu berücksichtigen, wird eine sphärische Projektion verwendet. Sowohl aktuelle Position, als auch der projizierte Punkt werden dem Graphen hinzugefügt und miteinander verknüpft. Die Routenführung kann somit gestartet werden. Der Benutzer wird von Wegpunkt zu Wegpunkt geführt. Um zu erkennen, ob ein Benutzer den nächsten Wegpunkt erreicht hat, müssen aktuelle Position und Wegpunkte der Route verglichen werden. Da jedoch das Positionssignal eine Ungenauigkeit von bis zum 100m haben kann, werden Map-Matching-Techniken zur Korrektur eingesetzt, die die aktuelle Position mit der digitalen Route verknüpfen. Map-Matching-Techniken setzen voraus, dass sich der Benutzer nur auf Straßen fortbewegen kann. Da sich Fußgänger jedoch frei, auch zwischen Straßen, bewegen können, können Map-Matching-Techniken aus der Kraftfahrzeugnavigation nicht eingesetzt werden. Um nun zu prüfen, ob ein Fußgänger einen Wegpunkt erreicht hat, wird ein Schwellwert verwendet. Ist die Distanz zwischen aktueller Position und nächstem Wegpunkt kleiner als der Schwellwert, so wird

der Wegpunkt als erreicht erkannt. Die Größe des Schwellwertes hat direkte Auswirkungen auf die Routenführung. Ist der Schwellwert zu groß, so wird ein Wegpunkt zu früh erkannt und der nächste Wegpunkt liegt in einer Richtung, die den Fußgänger irritieren könnte. Wird der Wegpunkt hingegen zu klein gewählt, wird der Fußgänger z.B. auf die Mitte einer Straße geschickt. Eine Möglichkeit einen geeigneten Schwellwert zu bestimmen, ist die Verwendung der Straßenbreite in Verbindung mit der aktuellen Messabweichung des GPS-Signals. Kann ein Wegpunkt nicht erreicht werden, so ist eine Neuberechnung der Route erforderlich. Auch hier ist es schwer, eine konkrete Aussage zu machen, ob ein Fußgänger nun die Route verlassen hat, oder sich nur auf Grund der baulichen Struktur des Fußweges von der Straße bzw. dem Wegpunkt entfernt hat. Aus diesem Grund ist es sinnvoll, dass eine Neuberechnung erst nach Ablauf einer festgesetzten Zeit veranlasst wird. Hat der Fußgänger nun den letzten Wegpunkt der Route erreicht, so ist die Routenführung beendet. Der Benutzer kann daraufhin nach Bedarf eine neue Route berechnen lassen.

Die Entwicklung der Navigationsanwendung setzt eine Auswahl der genannten Lösungsansätze um. Wie in dieser Arbeit festgestellt wurde, konnten nicht alle vorgestellten Verfahren, die in der klassischen Kraftfahrzeugnavigation genutzt werden, bei einer Navigation für Fußgänger direkt verwendet werden. Ein Großteil der bekannten Verfahren setzt voraus, dass nur ein Fortbewegen auf dem bekannten Straßennetz möglich ist, jedoch nicht auf Flächen außerhalb dieses Straßennetzes. Dies trifft für Fußgänger jedoch nicht zu, da diese sich frei bewegen können. Um dennoch eine Routenführung zu ermöglichen, wurden alternative Lösungsansätze verwendet. Diese Lösungsansätze sind daher noch zu verbessern. Durch aufwendige Map-Matching-Techniken wäre es möglich einen erreichten Wegpunkt eindeutiger zu identifizieren. Hierbei ist durchaus “Dead Reckoning” zu erwähnen, bei dem z.B. ein Schrittzähler und ein Kompass mit dem Positionssignal gekoppelt wird, um damit den zurückgelegten Weg aufzuzeichnen. Dieser kann dann wiederum mit der digitalen Karte abgeglichen werden. Dies setzt jedoch voraus, dass auch das digitale Kartenmaterial auf Fußgänger ausgelegt ist. So enthält zwar OpenStreetMap eine Reihe von Fuß- und Radwegen, aber es gibt auch viele implizit vorhandene Wege, wie z.B. Bahnhöfe mit einem Fußgängertunnel und mehreren Ausgängen. Des Weiteren macht das Kartenmaterial keine Aussage über die Güte einer Strecke. Hierbei sollten Fußgänger Straßen nur dort überqueren, wo Zebrastreifen oder Ampeln zur Verfügung stehen. Die Verwendung solcher Eigenschaften würde sich jedoch auch direkt bei der Berechnung des kürzesten Weges bemerkbar machen. Bei der einfachen Wegpunktnavigation, wie sie in dieser Arbeit verwendet wird, leistet der A*-Algorithmus bereits ein gutes Ergebnis. Werden jedoch die Graphen und damit der nötige Speicher zu groß, so kann dies unter Umständen nicht mehr von dem A*-Algorithmus berechnet werden. Hierbei ist unter anderem die begrenzte Kapazität des mobilen Endgerätes zu erwähnen. Die Performanz eines Kürzesten-Wege-Algorithmus ist stark von der Art der gelieferten Daten abhängig. Die von dem GIS abgefragten Daten liefern lediglich eine Menge von Straßen. Hier wäre es von Vorteil, wenn die Straßen bereits Nachbarschaftsbeziehungen enthalten würden, wie es z.B. in dem GDF-Standard der Fall ist. Nach dem Abfragen der Daten müssen in dieser Anwendung die Nachbarschaftsbeziehungen erst aufwendig berechnet werden. Im schlechtesten Fall muss hier jeder Wegpunkt mit jedem anderem Wegpunkt verglichen werden. Ebenso wäre eine Anfrage für das GIS schneller zu realisieren. Derzeit werden die Daten einer Bounding-Box abgerufen. Dies bedeutet, dass nicht zwangsweise alle nötigen Straßen in der Ergebnismenge vorhanden sind. Hier wäre die Verwendung von administrativen Grenzen, wie z.B. Landkreise oder Stadtgebiete eine mögliche Verbesserung. Die entsprechenden Daten müssten dazu aus den OpenStreetMap-Daten extrahiert werden. Hier wäre es alternativ möglich, dass die Daten nicht als Shapefile im GIS abgelegt werden, sondern über eine Geodatenbank, wie z.B. PostGIS, verwaltet werden. Hierbei können die Daten von OpenStreetMap in einer höheren Detaillierung importiert werden, so dass entsprechende

Informationen, wie Grenzen, Orte und Beziehungen zwischen diesen, ebenso zur Verfügung stehen. Zusätzlich würde die Abfragegeschwindigkeit verringert werden, da eine Geodatenbank im Gegensatz zu Shapefiles Indexstrategien zur schnelleren Suche verwendet. Durch die Verwendung des WFS als standardisierte Schnittstelle ist es jedoch möglich, dass sogar das gesamte GIS gegen ein anderes WFS-fähiges GIS ausgetauscht werden kann. Die Navigationsanwendung müsste nur angepasst werden, wenn sich das Feature einer WFS-Antwort ändert. Da dies aber vom Vektor-Import-Modul bewerkstelligt wird, sind keine Änderungen in der gesamten Anwendung durchzuführen. Zusätzlich wurde das Vektor-Import-Modul durch weitere Schnittstellen so definiert, dass das Modul problemlos um weitere Features erweitert werden kann. Dies ermöglicht unter anderem die Nutzung des Vektor-Import-Moduls für andere Projekte. So können damit z.B. Points of Interests per WFS vom GIS abgefragt werden. Ebenso ermöglicht der modulare Aufbau des Routing-Moduls, dass der kürzeste Weg auch auf anderen Graphen berechnet werden kann. So könnte man z.B. berechnen, wie der kürzeste Weg zwischen verschiedenen POIs ist, um so eine Sight-Seeing-Tour zu optimieren. Zusätzlich könnte dazu noch ein taktiler Gürtel (vgl. [HHBP08]) verwendet werden, so dass man der Ausgabe am Bildschirm nicht ständig folgen muss.

# Glossar

Nachfolgend werden verwendete Begriffe dieser Arbeit ergänzend erläutert. Das im Folgenden im Rahmen der Erläuterung verwendete Symbol ∼ bezieht sich jeweils auf den im Einzelnen vorgestellten Begriff, das Symbol ↑ verweist auf einen ebenfalls innerhalb dieses Glossars erklärten Begriff.

**Application Programming Interface** Eine Softwareschnittstelle, auch kurz API, die es dem Programmierer ermöglicht auf die Funktionen hinter der ∼ zuzugreifen.

**Azimuth** Der ∼ ist Winkel zwischen Höhenkreis und Meridian. Bei ebenen Polarkoordinaten beschreibt er den Winkel zwischen der Achse der Koordinaten und dem Radiusvektor.

**Client** Der ∼ ist ein Computer oder ein Programm, der Dienste oder Daten eines ↑Servers in Anspruch nimmt.

**Community** Eine ∼ ist ein virtueller Zusammenschluss von Internet-Nutzern, die ein gemeinsames Thema oder Ziel haben.

**Feature** Ein ∼ ist die kleinste geometrische Einheit in einem ↑Geoinformationssystem und beschreibt dabei eine allgemeine Abstraktion eines Faktes in der realen Welt.

**Framework** Als ∼ wird eine Rahmenstruktur für die konkrete Entwicklung einer Software bezeichnet. Das ∼ gibt dabei die Struktur und damit den Kontrollfluss und die Schnittstellen der Anwendung vor. Oft liefert das ∼ oft verwendetet Funktionen als Bibliothek mit.

**Geoinformationssystem** Ein ∼ , oder auch Geographisches Informationssystem, ist ein System, mit dem raumbezogene Daten erfasst, gespeichert, abgefragt und aktualisiert werden können. Ein ∼ wird oft als ↑Server bereitgestellt.

**Kartenprojektion** Die Darstellung der Erdoberfläche auf einer Karte. Dabei gibt es verschiedene Verfahren, die das Gradnetz der gekrümmten Erdoberfläche auf eine ebene Kartenfläche abbilden.

**Latitude** Mit ∼ wird die geographische Breite bezeichnet. Zusammen mit ↑Longitude bezeichnet es ein geographisches Datum.

**Longitude** Mit ∼ wird die geographische Länge bezeichnet. Zusammen mit ↑Latitude bezeichnet es ein geographisches Datum.

**Navigation** Allgemein umfasst die ∼ die Bestimmung des Standortes und die Berechnung des Kurses.

**NMEA** Die National Marine Electronics Association ist eine US-amerikanische Vereinigung von Schiffselektronikherstellern. Der ∼ -Standard beschreibt die Weitergabe von Positionsdaten zwischen zwei Geräten.

**Server** Der ∼ ist ein Computer in einem Netzwerk, der bestimmte Dienste und Daten für einen ↑Client bereitstellt.

**Service** Als ∼ bezeichnet man die Bereitstellung einer Funktion über Schnittstellen (↑API). Ein Web ∼ stellt dabei einen ∼ über das Internet dar.

**sphärisch** Auf die Himmelskugel bezogen. Die ∼ e Trigonometrie betrachtet die Trigonometrie auf einer Kugeloberfläche.

**Tomcat** Der Apache ∼ ist eine Umgebung zur Ausführung von Java-Programmen auf Web-↑Servern.

**Wiki** Ein ∼ ist eine Internet-Anwendung, die von Nutzern gelesen und geändert werden kann. Die einzelnen Seiten eines ∼ s sind dabei durch Querverweise verbunden und mit Schlagwörtern verknüpft.

# Abkürzungen

| | |
|---|---|
| ALK | Automatisierte Liegenschaftskarte |
| APSP | All Pair Shortest Path Problem |
| API | Application Programming Interface |
| CAD | Computer Aided Design |
| CAT | Catalogue Service |
| DXF | Drawing Interchange Format |
| GDF | Gepgraphic Data Files |
| GDI | Geodateninfrastruktur |
| GIS | Geographisches Informationssystem |
| GML | Geography Markup Language |
| GPS | Global Positioning System |
| GPX | GPS Exchange Format |
| ISO | International Organization for Standardization |
| KML | Keyhole Markup Language |
| LoD | Level of Detail |
| MIF | MapInfo Interchange Format |
| OGC | Open Geospital Consortium |
| OSM | OpenStreetMap |
| OWS | OpenGIS Web Services |
| PND | Personal Navigation Device |
| PNG | Portable Network Graphics |
| POI | Point of Interest |
| RDF | Resource Description Framework |
| SHP | Shapefile |
| SIF | Standard Interchange Format |
| SMIL | Synchronized Multimedia Integration Language |
| SOA | Serviceorientierte Architektur |
| SPSP | Single Pair Shortest Path Problem |
| SSSP | Single Source Shortest Path Problem |
| SVG | Scalable Vector Graphics |
| W3C | World Wide Web Consortium |
| WCS | Web Coverage Service |
| WFS | Web Feature Service |
| WMS | Web Map Service |
| XML | Extensible Markup Language |
| XMML | eXploration and Mining Markup Language |

# Abbildungen

# Literatur


[Bar00] BARTELME, Norbert: *Geoinformatik: Modelle, Strukturen, Funktionen*. 3. Springer, 2000

[BD04] BRÜGGE, Bernd ; DUTOIT, Allen H.: *Objektorientierte Softwaretechnik*. Pearson, 2004

[Czo00] CZOMMER, Renate: *Leistungsfähigkeit fahrzeugautonomer Ortungsverfahren auf der Basis von Map-Matching-Techniken*, Universität Stuttgart, Diss., Dezember 2000

[Eel05] EELES, Peter: *Capturing Architectural Requirements*. `http://www-128.ibm.com/developerworks/rational/library/4706.html`, besucht 02.11.2008, 2005

[EGW05] ERLE, Schuyler ; GIBSON, Rich ; WALSH, Jo: *Mapping Hacks – Tips and Tools for Electronic Cartography*. OReilly, 2005

[Env98] ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE: *ESRI Shapefile Technical Description*. `http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf`, besucht 03.07.2008, 1998

[Env08] ENVIRONMENTAL SYSTEMS RESEARCH INSTITUTE: *ArcIMS - Publish Maps, Data and Metadata on the Web*. `http://www.esri-germany.de/products/arcgis/arcims/index.html`, besucht 26.08.2008, 2008

[Fos04] FOSTER, Dan: *GPX - The GPS Exchange Format*. `http://www.topografix.com/gpx.asp`, besucht 31.07.2008, 2004

[Fri01] FRIEBE, Jörg: *Architekturen für komponentenbasierte Geographische Informationssysteme im Internet*. 1. Logos-Verlag, 2001

[GfK08] GFK GRUPPE, CORPORATE COMMUNICATIONS: *Tragbare Navigationsgeräte auf Erfolgskurs*. `http://www.gfk.com/group/press_information/press_releases/002095/index.de.html`, besucht 18.06.2008, 2008

[GKC07] GRÖGER, Gerhard ; KOLBE, Thomas H. ; CZERWINSKI, Angela: *Candidate OpenGIS CityGML Implementation Specification*. `http://www.opengeospatial.org/standards/gml`, besucht 03.07.2008, 2007

[GS04] GROBSTICH, Peter ; STREY, Gerhard: *Mathematik für Bauingenieure: Grundlagen, Verfahren und Anwendungen*. Vieweg+Teubner Verlag, 2004

[GSL07] GEESEN, Dennis ; SCHULTE-LOH, Gunnar: *Geodaten standardisiert integrieren - Vier Geodatenbanken im Vergleich*. Germany, Universität Oldenburg, Seminararbeit, April 2007

[HHBP08] HEUTEN, Wilko ; HENZE, Niels ; BOLL, Susanne ; PIELOT, Martin: Tactile Wayfinder: A Non-Visual Support System for Wayfinding. In: *NordiCHI*, 2008

[HNR68] HART, Peter E. ; NILSSON, Nils J. ; RAPHAEL, Bertram: A Formal Basis for the Heuristic Determination of Minimun Cost Paths. In: *IEEE Transactions of Systems Science and Cybernetics* ssc-4 (1968), Juli, Nr. 2, S. 100–107

[HS98] HUNG, Pen S. ; SU, Tsui C.: Map-Matching Algorithm of GPS Vehicle Navigation System. In: *Mapping from Space*, 1998

[Inn05] INNOCENTI, Emilio B.: *Towards seamless real-time in-car and personal navigation service delivery: the HIGHWAY integrated Architecture*. 2005

[Ins06] INSTITUT FÜR TECHNOLOGIEABSCHÄTZUNG UND SYSTEMANALYSE: Zukunftsfähige Verkehrspolitik - Ansätze für den Personenverkehr. In: *Technologieabschätzung - Theorie und Praxis* Vol. 15 (2006), Dezember, Nr. 3

[Kie06] KIEHLE, Christian: Lokales abstrakt – Geodaten standardisiert integrieren. In: *iX* 2006 (2006), Dezember, Nr. 12, S. 50–55

[Kom08] KOMPF, Martin: *Entfernungsberechnung*. `http://www.kompf.de/gps/distcalc.html`, besucht 15.10.2008, 2008

[Mac95] MACEACHREN, Alan M.: *How Maps Work – Presentation, Visualization, and Design*. 1. Guilford Press, 1995

[Mac07] MACHA, Adrian: *Routes 2 Niccimon*. Germany, Universität Oldenburg, Individuelles Projekt, Oktober 2007

[Man98] MANSFELD, Werner: *Satellitenortung und Navigation*. 1. Vieweg-Verlag, 1998

[Man03] MANSFELD, Werner: *Satellitenortung und Navigation*. 2. Vieweg-Verlag, 2003

[Map99] MAPINFO CORPORATION: *MapInfo Interchange File (MIF) Format Specification*. `http://www.gissky.com/Download/Download/DataFormat/Mapinfo_Mif.pdf`, 1999

[Mit08] MITCHELL, Tyler: *Web-Mapping mit Open Source-GIS-Tools*. OReilly, 2008

[MS04] MUTSCHLER, Bela ; SPECHT, Günther: *Mobile Datenbanksysteme: Architektur, Implementierung, Konzepte*. Springer-Verlag, 2004

[NH07] NAGEL, Claus ; HÄFELE, Karl-Heinz: Generierung von 3D-Stadtmodellen auf Basis des IFC-Gebäudemodells. In: CLEMEN, Christian (Hrsg.): *Entwicklerforum Geoinformationstechnik 2007 - Junge Wissenschaftler forschen*, Shaker Verlag, 2007, S. 151–165

[NM04] NETELER, Markus ; MITASOVA, Helena: *Open Source GIS: A Grass GIS Approach*. 2. Springer-Verlag, 2004

[Ope08] OPENSTREETMAP: *OpenStreetMap Reference Guide*. `http://wiki.openstreetmap.org/wiki/Main_Page`, besucht 02.11.2008, 2008

[OQS96] OLBRICH, Gerold ; QUICK, Michael ; SCHEIKART, Jürgen: *Computerkartographie*. 2. Springer-Verlag, 1996

[Pep] PEPPER, Peter: *Programmieren lernen - Eine grundlegende Einführung mit Java*

[PHHB08] PIELOT, Matrin ; HENZE, Niels ; HEUTEN, Wilko ; BOLL, Susanne: Evaluation of Continous Direction Encoding with Tactile Belts. In: *HAID*, 2008

[Por07] PORTELE, Clemens: *OpenGIS Geography Markup Language (GML) Encoding Standard*. `http://www.opengeospatial.org/standards/gml`, besucht 31.07.2008, 2007

[PSK04] PLÜMER, Lutz ; SCHMITTWILKEN, Jörg ; KOLBE, Thomas H.: Mobile GIS für die Orientierung von Fußgängern in städtischen Umgebungen. In: *Praktische Kartographie 2004*, Kirschbaum Verlag, 2004

[PZ04] PENG, Zhong-Ren ; ZHANG, Chuanrong: The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS). In: *Journal of Geographical Systems* Vol. 6 (2004), Juni, Nr. 2, S. 95–116

[Rez04] REZAGHOLI, Mohsen: *Prozess- und Technologiemanagement in der Softwareentwicklung: Ein metrikbasierter Ansatz zur Bewertung von Prozessen und Technologien*. Oldenbourg Wissenschaftsverlag, 2004

[Sch01] SCHMID, Wolfgang: *Berechnung kürzester Wege in Straßennetzen mit Wegeverboten*. Germany, Universität Stuttgart, Dissertation, Juli 2001

[Sol08] SOLID EARTH AND ENVIRONMENT GRID: *eXploration and Mining Marktup Language*. `https://www.seegrid.csiro.au/twiki/bin/view/Xmml/WebHome`, besucht 26.08.2008, 2008

[SW06] SCHWIEGER, Volker ; WANNINGER, Lambert: Potential von GPS-Navigationsempfängern. In: *GPS und Galileo - Methoden, Lösungen und neuste Entwicklungen* 49 (2006), S. 221–239

[Web99] WEBSTER, John G.: *The Measurement, Instrumentation, and Sensors Handbook*. CRC Press, 1999

[Wic08] WICHMANN, Daniel ; OFFIS (Hrsg.): *The NiccimonSharp Platform - Developer Documentation for the Platform and the Basic Modules*. Germany: OFFIS, 2008

[Wil06] WILK, Christian: GML – Komplexe Nische. In: *iX* 2006 (2006), dec, Nr. 12, S. 66–68

[Wil08] WILSON, Tim: *OGC KML*. `http://www.opengeospatial.org/standards/kml`, besucht 31.07.2008, 2008

[WR06] WALLENTOWITZ, Henning ; REIF, Konrad: *Handbuch Kraftfahrzeugelektronik: Grundlagen, Komponenten, Systeme, Anwendungen*. Vieweg+Teubner, 2006

# Index

## W



## Z

# Versicherung

Hiermit versichere ich, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel und Quellen verwendet habe.

Oldenburg, den 4. Juni 2009

Dennis Geesen