Bachelorarbeit

Thomas Pfaff

Entwicklung eines PDA-basierten
Indoor-Navigationssystems

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Thomas Pfaff

Entwicklung eines PDA-basierten Indoor-Navigationssystems

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 21. September 2007

Thomas Pfaff

Thema der Bachelorarbeit

Entwicklung eines PDA-basierten Indoor-Navigationssystems

Stichworte

ubiquitär, Indoor-Navigation, ubiquitäre Indoor-Navigation, IMAPS, Innenraum Routenplaner

Kurzzusammenfassung

Ziel dieser Arbeit ist die Entwicklung eines PDA-basierten Indoor-Navigationssystems auf der Grundlage der Positionsbestimmung mittels IMAPS und dem „Innenraum Routenplaner“. Der Schwerpunkt der Arbeit liegt auf der Präsentation des Indoor-Navigationssystems.

Thomas Pfaff

Title of the paper

Development of a PDA-based indoor route guidance system

Keywords

ubiquitous, indoor navigation, indoor navigation system, indoor route guidance, ubiquitous indoor navigation, IMAPS, Innenraum Routenplaner

Abstract

The aim of this thesis is the development of a PDA-based indoor route guidance system based on positioning with IMAPS and routing assisted by "‘Innenraum Routenplaner"’. The main focus of this paper is on presenting the indoor route guidance system.

Danksagung

Zunächst möchte ich mich recht herzlich für die optimale Betreuung durch Kai von Luck und meiner einstigen Betreuerin Birgit Wendholt bedanken: Euch habe ich es zu verdanken, dass ich meinen Abschluss machen konnte. Danke für diese Zeit, die Mühe und die aufbauenden Wort!
Dann möchte ich mich gleichfalls recht herzlich bei meinem Zweitgutachter Herrn Prof. Dr. rer. nat. Gunter Klemke für die spontane Zusage als Gutachter bedanken. Vielen herzlichen Dank dafür!

Und dann möchte ich mich aufs tiefste vor meine Eltern verneigen, die mir, egal welchen Weg ich einschlug, immer beiseite standen. Ich liebe Euch!

Mein bester Freund Markus soll auch nicht leer ausgehen, denn er war und ist immer in der Lage meine Stimmung aufzuhellen. Bitte weiter so!

Mein Dank gilt auch den beiden Ärzten, Th. Volkens und H. Brocks, die mich wieder zurück ins normale Leben geholt haben und mir damit das ermöglichten, was nach dem Studium mein Leben sein wird.

Weiterhin gilt mein Dank auch Tatjana, die jetzt leider wieder im Ausland wohnt, sowie Catrin und Georg, die mir mit bestem Deutsch beiseite standen. Ich danke Euch für die geopferte Zeit.

So, liebes Schwesterherz, jetzt bist du an der Reihe abzuschließen!

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung

Indoor-Navigationssysteme sind in den vergangenen Jahren des auslaufenden und des beginnenden Jahrtausends nicht nur ein Schwerpunkt des Forschungsinteresses geworden, sondern können sich, insbesondere durch das explosive Wachstum von *location based services* (LBS) in der Industrie [(20)], sowie auf dem privaten Sektor, als überaus wichtige Entwicklung herausstellen. Dabei versucht dieses Gebiet die bisher unerschlossene Navigationslandschaft in Gebäuden zu erschließen. Versuche, das sich bewährte Outdoor-Positioning Konzept GPS zu nutzen, scheitern unter anderem an der zu kurzen Akquirierungszeit[1] [(17)].

Diese Art der Navigation baut sich auf einer kabellosen Vernetzung von Sende- und Empfangseinheit auf. Dieses zusammen mit den verschiedenen Ansätzen zur Prozessabarbeitung im Hintergrund, wie die Positionsbestimmung oder die Routenberechnung aufgrund einer Datenbasis, reihen Indoor-Navigationssysteme in die Liga ubiquitärer Systeme ein. Somit entspricht ein solches System voll und ganz dem, was Mark Weiser [(68)] in die Zukunft schauend erdachte:[..]In the 21st century the technology revolution will move into the everyday, the small and the invisible. [..]. Für die Umsetzung des Größen- und Alltags-Charakters dieser Vision können mobilen Kleingeräten (z. B. ein PDA) exzellent verwendet werden. Mobile Geräte werden auch in Systemen verwendet, die Aufgrund des Gedankengutes von (68) bereits umgesetzt worden sind und Anwendung finden. Zu nennen sind in diesem engeren Zusammenhang beispielsweise Outdoor-Navigationssysteme.

Navigationssysteme für den Indoor-Bereich sollen dem Benutzer die Möglichkeit zur autonomen Navigation bieten, die mit Hilfe einer Programmiersprache und einer geeigneten Infrastruktur zu jedem Zeitpunkt die Position auf einem kleinen mobilen Gerät, beispielsweise einem PDA, anzeigen können. Darüber hinaus soll dem Benutzer der Komfort geboten werden, sich eine Route von einem Startpunkt zu einem Zielpunkt unsichtbar im Hintergrund berechnen und anzeigen zu lassen.

Die Forschungen auf dem Gebiet der Indoor-Navigation teilen sich in viele verschiedene Gebiete auf. Hierzu gehört die Bestimmung der Position mittels vorhandener oder noch zu

[1] Eine Signal muss innerhalb der Akquirierungszeit von 20ms gefunden werden. Nach Ablauf dieser Zeit wird von der Senderseite stets ein Navigationsdatenbit gesendet. Um innerhalb dieser Zeit Signale auszuwerten, wurden zwar Methoden vorgestellt, die innerhalb dieser Zeit Positionsdaten berechnen können, jedoch lassen sie sich nicht mit kleinen Geräten auf engstem Raum durchführen.

installierender Infrastruktur (Sender/Empfänger) [(28)] u. a., die Forschung auf dem Gebiet der Landmarken [(18)] u. a., die Forschungen auf dem Gebiet der Anzeige und der Darstellung [(9)] u. a., die Entwicklung neuer und leistungsfähigerer mobiler Hardware, wie mobil Telefone aber auch GPS-Empfänger [(35)] u. a. und die Entwicklung von Software, um die neuen Techniken vereint zu nutzen und zu einer virtuellen Karte zu transformieren. Neben diesen technischen Forschungsgebieten gibt es auch Forschungsschwerpunkte in der Psychologie des Menschen, die sich mit der Fähigkeit des Menschen, sich in seiner Umwelt selbständig zu navigieren, befassen.

1.1. Motivation

Für die Zukunft stellt sich die Frage, welche mobilen Lösungen, außer dem Befragen von ortskundigen Personen zur Orientierung in großen und unübersichtlichen Gebäuden, es geben wird. Dies gibt Anlass, einen Beitrag zur Entwicklung eines Indoor-Navigationssystems zu leisten. Im Mittelpunkt dieser Arbeit soll die Visualisierung von Gebäudekarten stehen, um mit Hilfe von IMAPS eine Routenplanung durchführen zu können. Des weiteren soll geklärt werden, welche Elemente ein Indoor-Navigationssystem Anzeigen sollte, damit die Navigation anhand eines mobilen Gerätes komfortabel und übersichtlich bleibt.

Ganz allgemein sind Navigationssysteme in der Lage, computergestützt, die kürzeste, schnellste oder schönste Verbindung innerhalb einer Infrastruktur, bestehend aus Sendern und Empfängern, zu berechnen und dies dem Benutzer akustisch oder visuell darzustellen. Im Outdoor-Bereich finden sich Navigationssysteme vor allem im Straßenverkehr - dafür haben sie sich in den letzten Jahrzehnten etabliert - aber aktuell auch bei Anwendungen der Fußgänger- oder Fahrradnavigation. Diese Systeme sind vor allem durch den zunehmenden Gebrauch von kleinen mobilen Geräten wie PDAs oder Handys möglich geworden. Die angesprochenen Systeme haben unabhängig von ihren Benutzergruppen die verwendete Datenbasis (Kartenmaterial, Navigationsinformationen, Gebäudedaten usw.) zur Gemeinsamkeit. Auf deren Grundlage erfolgt das Routing und die Zusammenstellung von Navigationsanweisungen, wie beispielsweise die Vorgabe der Richtung [(18)].

Die vorliegenden Daten müssen zunächst für den jeweiligen Zweck festgestellt und modelliert werden. Im Indoor-Bereich sollten beispielsweise zu erst die Etage vermessen, wichtige Gegenstände erfasst und eine maßstabstreue Karte angefertigt werden. Da ein Navigationssystem Zusatzinformationen, die bei der normalen Navigation nicht vorhanden wären, erlaubt darzustellen, werden auch so genannte *Points of Interest* (POI) in der Datenbasis eingepflegt. POIs haben keine direkte Verwendung für die eigentliche Wegbeschreibung, bieten aber Hinweise, die dem Komfort zu Gute kommen. Sie weisen beispielsweise auf Toiletten, Geschäfte oder auch Restaurants hin. Die Darstellung von Weginformation erfolgt wie bei der Indoor- als auch bei der Outdoor-Navigation üblicherweise in einer reduzierten Grafik

und akustischer Sprachanweisung, Kapitel 3.3, kann aber auch in Form einer Karte, graphisch darübergelegten Routen, Landmarken oder POIs und textuellen Wegbeschreibungen erfolgen, siehe Kapitel 3.2 und 3.3. Eine digitale Wegbeschreibung wird aus Navigationsdaten, beispielsweise Start und Ziel, Kapitel 3.5, ohne das Zutun des Benutzers generiert. Eine Routenbeschreibung erfolgt aufgrund der generierten Daten abschnittsweise und ist an den Routenfortschritt geknüpft. Eine übliche textuelle Beschreibung setzt sich aus den Bausteinen „Ortsbezeichnung“, „Abzweigungsvorgang“ sowie „metrischer Distanz“ zum nächsten Knoten zusammen: „Leave Lokstedter Steindamm and head straightforward onto Osterfeldstrasse. Stay on for 886 m.“ (Auszug aus einer Routenanweisung von [(1)])

Aufgrund fehlerhafter Beschreibungen durch die Datenbasis, ungenauer Positionsbestimmung oder automatisch erzeugter Wegbeschreibung kann es zu Fehlern bei der Berechnung, infolge geänderten Verkehrssituationen, aber auch zu einer Fehlinterpretation der Anweisung, Fehlentscheidungen in der Navigation oder einer Irritation des Anwenders durch die Form der Wegbeschreibung kommen. Dies kann eine Folge des strukturellen Aufbaues der Karte, des Inhaltes der Karte oder der fehlerhaften Weganweisung sein, die nicht der menschlichen Erfordernis entsprach. In der Fahrzeugnavigation wird versucht solche fehlerhaften Auswirkungen zum einen mittels Sensoren (Wegmesser, inertial Messeinheiten, GPS) und zum anderen mittels Matching-Verfahren[2] entgegen zu wirken. Im Innenbereich von Gebäuden herrschen andere Bedingungen als im Straßenverkehr. Dort bestehen aufgrund der Ungenauigkeit der Messung und der Orientierungssituation erschwerte Bedingungen, die nicht mit den Ansätzen der Fahrzeugnavigation gelöst werden können. Daher besteht auch auf diesem Sektor Forschungsbedarf, wobei es darum geht, ähnliche Ansätze für den Indoor-Bereich zu entwickeln.

Aufgrund von Messungenauigkeiten und Orientierungsschwierigkeiten kann der Benutzer seine Navigation möglicherweise nicht zufriedenstellend oder fehlerfrei durchführen. Dies kann wiederum dazu führen, dass der Benutzer frustriert reagiert und die Navigation abbricht. Eine solche Situation sollte Anlass geben Indoor-Navigationssysteme an die Bedürfnisse und Erwartungen des Nutzers anzupassen, und diese einfacher und intuitiver zu gestalten.

Zur Entwicklung eines Indoor-Navigationssystems sind kognitive Aspekte der Wegfindung[3] beim Menschen [(18)] und Kommunikationsverfahren von Navigationsinformationen zwingend erforderlich, sofern die Systeme die menschliche Navigation „imitieren“ sollen. Ein aktueller Schwerpunkt in der Forschung liegt darin, den zu Grunde liegenden Bauplan für menschliche Wegbeschreibung mit seinen notwendigen Elementen und Strukturen zu entschlüsseln und in Form einer so genannten *Ontologie*[4] explizit sichtbar zu machen [(18)].

[2] Ein PKW befindet sich auf der Autobahn, weil es sich nur dort entsprechend schnell fortbewegen kann
[3] unbewusstes Erlernen der Wegfindung
[4] Die Ontologie, die Seinslehre. Gemeint ist die Seinslehre der menschlichen Navigation

1.2. Zielsetzung der vorliegenden Arbeit

Da sich Aufgrund von Indoor-Navigationssystemen, beispielsweise zur Bergung von Eingeschlossenen während eines Brandes [(6)], die menschliche Navigation in unübersichtlichen Gebäuden ergänzen lässt, ist die Bereitstellung solcher Systeme in den Vordergrund gerückt. Neben den Entwicklungen auf dem Gebiet der Infrastruktur solcher Systeme bietet sich auch auf dem Gebiet der visuellen Darstellung die Notwendigkeit zur Forschung, um somit die Umgebung auf der Anzeige detailgetreu darzustellen. Zu diesem Forschungsgebiet existieren einige verschiedene Ansätze und Methoden, die im Kapitel 3.3 näher gebracht werden sollen.

Zur Unterstützung der menschlichen Navigation aufgrund einer Datenbasis müssen im Indoor-Bereich folgende Aspekte Berücksichtigung finden: die Bereitstellung einer Infrastruktur zur Erzeugung, Übermittlung und den Empfang von Sensordaten, die Bereitstellung der Gebäudekarten und die Routenberechnung, die Hardware zur Darstellung der Karten und Routen und die Programmiersprache, um mit deren Hilfe aufgrund der Datenbasis ein Software-Model für die Hardware zu generieren. Um dies zu erreichen, muss die Infrastruktur eine überzeugende Genauigkeit der Positionsbestimmung aufweisen, da in Gebäuden, anders als im Freien, mit schmalen Gängen zu rechnen ist. Die Gebäudedaten müssen maßstabsgetreu erfasst und in eine Datenbank abgelegt werden, um abrufbereit zur Verfügung zu stehen und dies sowohl am Tage als auch in der Nacht. Hardwareseitig werden PDAs verwendet, da sie dem Aspekt der Mobilität entsprechen und dazu klein und handlich sind. Softwareseitig ist die Programmiersprache dafür verantwortlich, die verschiedenen Aspekte zu vereinen und daraus ein virtuelle Darstellung des umgebenden Raumes zu erzeugen. Die Art der Darstellung soll sich an verschiedene funktionierenden Indoor- und Outdoor-Navigationssystemen orientieren, da davon auszugehen ist, dass sich diese in unterschiedlichen Punkten an die erlernte menschliche Navigation anlehnen.

Die vorliegende Arbeit geht davon aus, dass die Infrastruktur zur Positionsbestimmung funktionsfähig ist. Daher wird es als möglich vorausgesetzt, ein Indoor-Navigationssystem auf der Basis von Sende- und Empfangsmodulen zu entwickeln. Desweiteren gilt die entwickelte Anwendung zur späteren Navigation auf der Hardware als vorinstalliert. Der Karten-Service und die Routenberechung die durch einen Server ausgeführt wird, ist ebenfalls als funktionierend vorausgesetzt.

Die Entwicklung der Arbeit setzt auf schon bestehenden Systemen auf, daher werden diese im Verlaufe der Arbeit vorgestellt. Da eine vollständige Erarbeitung der visuelle Darstellung sehr viel länger andauern würde, als es die Bearbeitungszeit erlaubt, soll ein Vergleich verschiedener Navigationssystemen dazu genügen, die wichtigsten für die Navigation erforderlichen Aspekte aufzudecken und in die Anwendung zu integrieren. Auf dieser Grundlage soll ein System entstehen, dass sich zur Indoor-Navigation eignet und verschiedene

Techniken in sich vereint. Mittels diesen soll das entwickelte Indoor-Navigationssystem nicht an eine Gebäudedatenbasis gebunden sein, sondern auch für weitere Gebäudeinformationen oder Gebäude gebraucht werden können. Anhand der Studie von Technologien zur Positionsbestimmung, der Betrachtung heutiger Navigationssoftware und heutiger Indoor-Navigationssysteme, soll vermittelt werden, dass es möglich ist, ein prototypische Entwicklung eines an mobile Geräte angepassten Indoor-Navigationssystems zu erarbeiten.

Kernstück dieser Arbeit ist daher die Navigation und die Darstellung von Karten, sowie die Darstellung einer zuvor berechneten Route, auf der sich ein Benutzer in das von ihm ausgewählte Ziel bewegt. Um die Wahl der Verwendung der Infrastruktur, welche die Positionsbestimmung vornimmt, zu rechtfertigen, wird diese anderen Infrastrukturen gegenübergestellt. Mit einer Beschreibung der Darstellungsarten von Karten soll zudem herausgestellt werden, welche der vorgestellten am geeignetsten ist. Für die spätere Darstellung ist es ebenfalls sehr wichtig, welche Informationen bei der Navigation auf der Anzeige zu sehen sein sollen, dies sollte daher ebenfalls erörtert werden.

1.3. Gliederung der Arbeit

Kapitel 2 erläutert zunächst die Wegfindung beim Menschen und die Grundlagen digitaler Ortung und Positionsbestimmung. Diese Grundlagen dienen der Vermittlung von Verständnis, das bei den darauf folgenden Verfahren und Techniken benötigt wird. Nach der Vorstellung der Grundlagen zur Positionsbestimmung erfolgt die Beschreibung einer geeigneten Programmiersprache, die als Grundlage zur Implementierung des Indoor-Navigationssystems dient.

In Kapitel 3 werden vergleichbare Arbeiten und Studien vorgestellt und auf die Verwendung für die vorliegende Arbeit analysiert. Zu Anfang werden *Location Based Services* (LBS) erläutert, die als Grundlage der folgenden beiden Kapitel der Outdoor-Navigationssoftware und Indoor-Navigationssysteme dienen. Diese beiden Kapitel beleuchten die Datenbasis, die bei Informationsvisualisierung auf der verwendeten Hardware genutzt wird. Nach der Beleuchtung bestehender Systeme kann auf die Informationsdarstellung auf einem mobilen Gerät Bezug genommen werden. Ein solches Gerät dient zur Anzeige der Anwendung dieser Arbeit.

Ziel des Kapitels 4 ist es, die aus dem Kapitel 3 gewonnenen positiven sowie negativen Erfahrungen in das zu erstellende Indoor-Navigationssystem einfließen zu lassen. Dazu wird zunächst anhand eines Beispiel-Szenarios durchgespielt (Kapitel 4.1), welche grundlegenden Anforderungen an ein Indoor-Navigationssystem zu stellen sind. Die gewonnen Anforderungen werden anschließend in Kapitel 4.2 präzisiert und festgehalten. Darauf folgend werden aus diesen die funktionalen Anforderungen konkretisiert. Ebenso werden die

nicht-funktionalen Anforderungen an das System gestellt. Diese Thematik teilt sich in zwei Bereiche: Der Bereich der nicht-funktionalen Anforderungen der Anwendung und die der nicht-funktionalen Anforderung der Benutzerführung und der Karte. Abschließend wird das Kapitel zusammengefasst.

In Kapitel 5 wird die Anwendung des Systems in groben Zügen entworfen. Es beginnt mit der Darstellung einer Nachvollziehbarkeitsmatrix und im Anschluss daran wird die Präsentationsschicht einem Design-Pattern zugeordnet. Danach erfolgt in groben Zügen die Gestaltung der Anwendungsschicht. Ist dies geschehen, kann die Gestaltung der Benutzerschnittstelle und abschließend die Gestaltung des Models des Indoor-Navigationssystems erfolgen.

Im Kapitel 6 wird der Feinentwurf und die Realisierung vorgenommen. Dazu wird zuerst auf die Hard- und die Software explizit eingegangen, um deren spezifische Bedeutung für den Entwurf zu erläutern. Im Anschluss daran wird der Indoor-Navigationsclient, dessen Schnittstelle, die verwendeten Entwurfsmuster und das spezielle XML-Parsing vorgestellt. Um die Realisierung zu ergänzen werden die Anwendungsfälle in Sequenzdiagrammen festgehalten und dargestellt. Anschließend erfolgt eine kurze Zusammenstellung dessen, was erreicht wurde und was nicht erreicht wurde. Zum Abschluss dieses Kapitels erfolgt eine Zusammenfassung.

Das letzte Kapitel 7 erfaßt schließlich das, was in Bezug auf die Aufgabenstellung realisiert wurde und welche Erkenntnisse sich daraus entwickelt haben. In den Ausblicken soll abschließend Anregung zu ergänzenden Verfahren, Methoden und Visualisierungstechniken gegeben werden.

Im Anhang A wird in Anhang A.1 ein Beispiel-Code zu einer SVG-Grafik gezeigt. Und in Anhand A.2 ist der Pseudo-Code der eigens erstellten Parser zu sehen.

2. Grundlagen

Dieses Kapitel beschäftigt sich mit den Grundlagen rund um die Positionsbestimmung, d.h. mit und ohne heutiger Ortungsverfahren und Techniken. Begonnen wird in Kapitel 2.1 mit der Beschreibung der Wegefindung des Menschen. Daraufhin erfolgt im Kapitel 2.2 die Betrachtung der automatischen Positionsbestimmung, wobei verschieden Klassen, Techniken und Methoden zu diesem Thema vorgestellt werden. In Kapitel 2.3 wird anhand des gewonnenen Wissens auf die automatische Poitionsbestimmung im Indoor-Bereich eingegangen.

Zu den Grundlagen dieser Arbeit zählt ebenfalls die Eingrenzung der Programmiersprache. Zu diesem Zweck erfolgt in Kapitel 2.4 eine Einführung zu diesem Thema, da in Kapitel 4 auf die Programmiersprache Bezug genommen wird.

2.1. Wegfindung beim Menschen

Wir Menschen setzen uns tagtäglich mit der Wegfindung auseinander, sei es beim Autofahren, beim Shopping oder in den eigenen vier Wänden. Das Ziel der Wegfindung besteht darin, den Weg von einem Standort zu einem Zielort zu finden. Damit sich der Mensch anhand seiner Sinne navigieren kann, nutzt er unbewusst das Wissen über den Raum und sein Wahrnehmung. Daher wird die Wegfindung zunächst abgekoppelt von der Wegfindung in einer virtuellen Realität oder auf einer Karte betrachtet.

Neben der menschlichen Wahrnehmung ist auch dessen räumliches Vorstellungsvermögen bei der Wegfindung von Bedeutung. Sowohl Wahrnehmung über die Sinnesorgane als auch der Intelligenzfaktor räumliches Vorstellungsvermögen [(49)] sind von Mensch zu Mensch unterschiedlich stark ausgeprägt.

Die räumliche Wahrnehmung des Menschen beruht auf drei Eigenschaften:

1. Aus Landmarken, die aus sich hervorstechenden Referenzpunkten der Umgebung gebildet werden. Daher sollten Landmarken möglichst unterschiedlich sein, damit sie fehlerfrei genutzt werden können [(46)].
2. Wissen über die Route, d.h. Wissen darüber, dass die Route aus aneinandergereihten Landmarken besteht.

3. Wissen über die Struktur und die Umgebung, dass es erlaubt Landmarken zu erkennen.

Sind die aufgeführten Möglichkeiten zur Wegfindung ausgeschöpft, dient die verbale Kommunikation als weiteres Hilfsmittel zur weiteren Navigation [(62)].

Daraus ist ersichtlich, dass zu einer erfolgreiche Wegfindung gewisse Anhaltspunkte nötig sind. Ein Großteil der Literatur zu diesem Thema, unter anderem auch die von (46), beschäftigt sich mit der so genannten kognitiven Präsentation des Raumes. Die kognitive Präsentation bestimmt, wie Menschen geografische Informationen aufnehmen, verarbeiten und für die Entscheidung nutzen. Somit werden die zuvor genannten Anhaltspunkte mental, anhand so genannten kognitiver Karten[1] „cognitive maps“ bewertet, interpretiert und schließlich wiedererkannt. (46) erläutert „cognitive maps“ folgendermaßen: „[...]a mental representation that corresponds to people’s perceptions of the real world[...]“.

Ein weiterer wichtiger Faktor für den Menschen bei der Navigation ist, das vorhandene „Wissen in der Welt“ in Form von Symbolen (Richtungsweiser, Verkehrsschilder, Straßennamen usw.) zu nutzen. Dies ist beispielsweise bei der verbalen Erläuterung eines Weges wichtig: „[...]gehen Sie bis zum Schild xy und folgen dann der Anweisung[...]“.

2.2. Automatisierte Positionsbestimmung

Da sich während einer Navigation innerhalb von Gebäuden die eigene Position ändert oder ändern kann, ist es erforderlich, Verfahren zu verwenden, die ein Objekt zu einem Ort in Beziehung stellen. Durch diesen Ortsbezug ist es dann möglich verschiedene sinnvolle Anwendungen rund um das Thema Positionierung zu entwickeln. Beispielhaft sind dazu im Vorgriff auf das Kapitel 3.1 (LBS) die Navigation im Outdoor-Bereich, Interaktive Informationsstände (an Flughäfen, in Einkaufzentren, POIs), Mobile-Computing (GSM, SMS, MMS, Friend-Finder) aber auch Routenplanung und ortsbezogene Gebühren (Maut, Zoll) zu nennen, die eine korrekte Positionsbestimmung voraussetzen. Die Verfahren, die heute zur Positionsbestimmung verwendet werden, sind in Kapitel 2.2 erläutert. Zunächst sind noch einige Begriffe rund um die Positionsbestimmung zu beschreiben, die in diesem Zusammenhang wichtig erscheinen.

[1] Kognitive Karten entsprechen geistigen Repräsentationen (Bilder), die der Wahrnehmung des einzelnen Menschen entsprechen

Begriffe rundum Ortung und Positionierung

In der Literatur werden für die Bestimmung des Aufenthaltsortes eines Objektes drei verschiedene Stichworte genutzt: Lokalisation, Positionierung und Ortung. Daher ist zunächst unklar, welcher Begriff richtig ist und im Kontext der Indoor-Navigation verwendet werden kann.

- Lokalisation

  Lokalisation wird immer dann verwendet, wenn Menschen mit Hilfe ihres Sinnesorgans Ohr versuchen, die Position eines schallaussendenden Objektes zu bestimmen.

  Beim natürlichen Hören und beim Stereo-Hören bezeichnen wir häufig die Richtungsbestimmung inkorrekt mit *Ortung*; wir orten jedoch nicht aktiv unter Aussenden von Wellen, wie bei der Echoortung der Funknavigation oder wie es die Fledermäuse tun. Für die richtige Bestimmung der Schalleinfallsrichtung ist besser der Fachbegriff Lokalisation für das Richtungshören zu verwenden. Wir Menschen lokalisieren also beim Hören, um die Hör-Ereignisrichtung festzustellen. [(24)]

- Ortung

  Alle Fach-Lexika weisen deutlich darauf hin, dass das Orten sich begrifflich auf die Radar-Peilung bezieht, wenn ein Signal ausgesendet wird, dessen schwaches, vom Hindernis reflektiertes Signal, zur Richtungs- und Entfernungsmessung herangezogen wird [(50)].

- Positionierung

  Die Positionierung erfolgt demnach bei der Lokalisierung durch Wahrnehmung (passiv) und bei der Ortung durch Aussenden und Empfangen (aktiv) von elektromagnetischen Wellen.

  Grundsätzlich bedarf es bei der Bestimmung einer Position der Projektion des Objektes in ein dreidimensionales kartesisches Koordinatensystem. In einem solchen System ist die Position des Objekts durch drei Koordinaten (x,y,z) eindeutig beschrieben. Einige der im folgenden beschriebenen Verfahren nutzen eine solche Projektion, andere die Projektion in ein kartesisches Kugelkoordinatensystem.

Aus diesen Erläuterungen folgt, die Bezeichnung *Lokalisation* nicht weiter zu verwenden.

Begriffe, die zur Klärung von Sachverhalten nötig sein können, werden nun vorgestellt und erklärt.

Signaldämpfung: Signaldämpfung bedeutet die Abnahme der Feldstärke bei Zunahme der Entfernung von der Quelle. Unter Normalbedingungen in einem Raum auf der Erde wird das Signal zusätzlich gedämpft, reflektiert oder gestreut.

Genauigkeit: Der Begriff Genauigkeit ist für diese Arbeit von Bedeutung, da im folgenden die verschiedenen Positionsbestimmungsverfahren anhand ihrer Genauigkeit gemessen wurden.

Die Genauigkeit ist definiert als der Positionierungsfehler, der entsteht, wenn die Positionsschätzung nicht mit der realen Position eines Objektes übereinstimmt [(39)]. In diesem speziellen Fall ist die Genauigkeit ebenfalls abhängig von der Signaldämpfung.

Ist die Genauigkeit etwa mit 2m in 95% aller Messungen angegeben, so bedeutet dies, dass sich um die reale Position 95% der Schätzungen innerhalb des Radius von 2m befinden.

Rauschen: Rauschen entsteht durch Bewegung von Objekten und wird unwiderruflich den Sensordaten hinzugefügt. Genauer gesagt resultiert das Rauschen aus der Beschleunigung und Richtungsänderung eines Objektes.

Basistechniken zur Ortung (Indoor/Outdoor)

Die im folgenden Kapitel besprochenen Klassen und Methoden zur Positionsbestimmung bauen auf Basistechniken auf.
Diese sind:

- *Cell of Origin (COO)*

  Ist die Klasse von Positionierungssystemen durch eine Zellstruktur aufgebaut, so lassen sich dadurch Rückschlüsse auf die Position eines Gerätes innerhalb der Zellstruktur ziehen. Hintergrund: In jeder Zelle wird ein Signal einer bestimmten Frequenz ausgestrahlt. Der Empfänger innerhalb einer Zelle meldet sich über diese Frequenz bei dem Sender der Zelle an. Durch diese Anmeldung kann herausgefunden werden in welcher Zelle sich ein Empfänger aufhält. Dies ist möglich, da die Daten der Anmeldung auf einem Server temporär gespeichert wurden. Das Verfahren wird in GSM-Netzen verwendet, in denen die Genauigkeit von der Größe und Form der Zelle abhängig ist.

- *Time of Arrival (TOA), Time Difference of Arrival (TDOA)*

  Hierbei wird der Zeitunterschied zwischen Aussenden und Empfangen des gleichen Signals gemessen. Durch diese zeitliche Differenz kann auf die Entfernung geschlossen werden, da die Ausbreitungsgeschwindigkeit[2] des Signals bekannt ist.

[2] Elektromagnetische Wellen haben eine Ausbreitungsgeschwindigkeit von 300.000 km/s. Ultraschall-Wellen breiten sich in Luft mit 330 m/s und in Wasser mit 1480 m/s aus.

- *Angel of Arrival (AOA)*

  Durch einen Satz von Antennen mit Richtungscharakteristik ist es möglich, die Richtung zu ermitteln, aus der ein Signal eintrifft. Dieses Verfahren eignet sich zur Kombination mit anderen Verfahren, beispielsweise mit TOA.

- *Messung der Signalstärke*

  Das Verfahren beruht auf der Signaldämpfung. Bekommt ein Empfänger ein Signal, dessen Signalstärke vor Ausbreitung bekannt ist, so misst er die eingehende Signalstärke. Aus der Differenz der Ausgangs-Signalstärke zu gemessener Signalstärke kann statistisch auf die wahrscheinlichste Entfernung geschlossen werden.

- *Auswertung von Videodaten*

  Durch Vergleichen mehrerer Aufnahmen aus verschiedenen Blickwinkeln ist es möglich, auf die Entfernung eines Objektes zu schließen. Das Verfahren ist daher ein besonderes AOA.

Klassen und Methoden zur Positionsbestimmung (Indoor/Outdoor)

Positionierungsverfahren, wie GPS oder Indoor-Positionierungsverfahren, setzen das Vorhandensein eines Sensorsystems voraus. Dieses besteht wiederum aus einem oder mehreren Sender/-n bzw. Empfänger/-n wodurch mit Hilfe verschiedenartiger Methoden der Signalverarbeitung die Bestimmung der Position von Objekten ermöglicht. Die Methoden, in denen die besprochenen Basistechniken verwendet werden, lassen sich in zwei Klassen einteilen:

- *Tracking*

  Tracking ist die Positionsbestimmung eines Objektes durch ein Sensorsystem. Das System, das die Sensoren überwacht, muss gleichfalls dafür sorgen, dass die Position an den Benutzer weitergeleitet wird. Dieses Verfahren schont die Ressourcen des Client.

- *Positioning*

  Beim Positioning ermittelt das System des Benutzers die Position über Sender oder Barken. Beim Positioning wird ein fixer Ort als Position herangezogen. Dadurch entfällt das störende Rauschen. Dieses Verfahren schont die Ressourcen des Client nicht.

Sowohl die Klasse der Trackingsysteme als auch die der Positioningsysteme verwenden verschiedene Methoden zur Ortung, um die Position eines Objektes innerhalb eines Empfangsreichweite zu bestimmen. Anhand dieser beiden Klassen lässt sich auch eine Einteilung der folgenden Methoden in Infrastruktur-basierte- oder Endgeräte-basierte-Verfahren vornehmen. Eine solche Unterteilung kann während der Positionierungsphase stattfinden [(39)]. Die beiden Varianten zusammengenommen ergänzen sich zu dem so genannten hybriden Verfahren.

- Infrastruktur-basiert/Endgerät-assistiert (Tracking)
- Infrastruktur-assistiert/ Endgerät-basiert (Positioning)

Die Verwendung einzelner Basistechniken zur Ortung bzw. deren Kombination definiert unterschiedliche Methoden zur Positionsbestimmung. Zu diesen Methoden gehören:

- *Nachbarschaftserkennung*

  Nachbarschaftserkennung ist eine vergleichsweise einfache Methode zur Positionsbestimmung. Dabei wird lediglich eine zu ermittelnde Position mit einem Referenzpunkt, dem nächst gelegenen, gleichgesetzt bzw. diesem zugeordnet. Somit werden die Entfernungen zu den Endgeräten geschätzt. Diese Schätzungen liegen innerhalb der Kommunikationsreichweite, die abhängig von der verwendeten Funktechnologie sind.
  Bei der Nachbarschaftserkennung wird zwischen Nahbereichsnachbarschaft und Fernbereichsnachbarschaft unterschieden. In Nahbereichsnachbarschaft befinden sich Endgeräte, die im Sendekreis eines Access Points sind. Fernbereichsnachbarschaft besteht zwischen Endgeräten, die sich in den Überlappungsgebieten der Sendekreise zweier oder mehrere Access Points befinden. Der Positionierungsfehler ist gleich der halben Kommunikationsreichweite.

- *Entfernungspeilung*

  Entfernungspeilung ist eine Methode, durch die die Position eines Objektes anhand mehrerer Referenzpunkte bestimmt werden kann [(39)] und basiert auf dem Theorem von Pythagoras. Eine Position hat, entsprechend der Anzahl der Referenzpunkte, verschiedene Entfernungen zu den Referenzpunkten. Diese Entfernungen lassen eine Peilung Zustandekommen. Bei der Entfernungspeilung spielt daher die Signaldämpfung eine entscheidende Rolle. Neben der Entfernung ist auch die Sendestärke des Senders, die Sensibilität und die Rechenleistung des Empfängers bei der Messung von ausschlaggebender Bedeutung. Reflektion, Streuung und Absorption setzen die Messgenauigkeit durch Behinderung der Signalausbreitung herab. Positionierungsfehler ergeben sich bei dieser Methode infolge der Genauigkeit der gemessenen Entfernungen.

- *Karten mit Radio-Fingerabdrücken*

  Das Verfahren gliedert sich in zwei Phasen. In der ersten, der Trainingsphase (auch Online-Phase genannt) werden an zuvor definierten Punkten eines Raumes elektromagnetische Eigenschaften eingehender Signale gemessen und gesammelt. Diese Eigenschaften werden auch als Radio-Fingerabdrücke bezeichnet. Gespeichert[3] werden diese „Fingerabdrücke“ in einer Datenbank und dienen dem späteren Zugriff in der zweiten Phase, der Positionierungsphase.
  In der Positionierungsphase, auch unter Offline-Phase bekannt, werden an einer Position aktuelle elektromagnetische Eigenschaften gemessen. Diese werden mit den Eigenschaften, die in der Trainingsphase gewonnen wurden, verglichen. Sobald zu einer gemessenen elektromagnetischen Eigenschaft die nähste in der Datenbank befindliche gefunden wurde, wird diese als aktuelle Position betrachtet.
  Die Auswahl dieses nächstgelegenen Nachbarn geschieht anhand des *k-nearest neighbor-Algorithmus*, der die euklidischen Distanzen zwischen den aktuell ermittelten und den in der Trainingsphase ermittelten Eigenschaften berechnet.

  Wird dieses Verfahren beispielsweise mit Access Points durchgeführt, so erfolgt dies nach einem definierten Schema: Im Vorfeld dient die so genannte Online-Phase der Einrichtung der Infrastrukturen. Es werden letztendlich aber nur Schätzungen über die Distanzen untereinander wiedergegeben, welche entsprechend von der Anzahl der örtlichen Access-Points abhängt. Zum Kalibrieren der Access-Points werden Laserstrahl-Entfernungsmessgeräte verwendet.

- *Sensor-fusion*

  Bei dieser Methode werden verschiedene von Sensoren erfasste Radiosignale zur Messung herangezogen, um eine größtmögliche Verfügbarkeit an Signalen, die über den Tag verteilt gesammelt werden, zu erhalten. Unterteilt sind die unterschiedlichen Sensoren in eine horizontale Klasse (gleichartige Sensoren) und vertikale Klasse (verschiedenartige Sensoren). Eine größtmögliche Verfügbarkeit (bis zu 94%) lässt sich beispielsweise bei Signalquellen, wie sie bei Zellfunk-Systemen (GSM, W-LAN usw.) verwendet werden, erhalten. Dies entspricht der Funktionsweise der Verfahren *Place-Lab* und *Quality of Information* auf die in Kapitel 2.3 gesondert eingegangen wird. Bei *Quality of Information* geht es darum, ausschließlich diejenigen Messpunkte zur Wahrscheinlichkeitsermittlung heran zu ziehen, die eine exakte Übereinstimmung von gemessener (durch Benutzer ausgelöste Messung) und realer Position (gegeben durch Sensoren) haben. Daraus folgt: Je höher die Anzahl der verwendeten Sensoren, desto

[3] Zur Speicherung der Radio-Fingerabdrücke in der Online-Phase, gibt es zwei Ansätze. Ein deterministischer Ansatz, bei dem der Durchschnitt aus kurzfristigen Abweichungen gespeichert wird und einem probabilistische Ansatz, bei dem die Abweichungen als Wahrscheinlichkeitsverteilung gespeichert wird. Demzufolge gibt es drei verschiedene Abfragealgorithmen, die aber nicht weiter erläutert werden.

mehr reale Positionen gibt es. Mit der Erhöhung der Anzahl der Sensoren kann also die Wahrscheinlichkeit erhöht werden, dass eine gemessene Position mit der realen Position übereinstimmt.

- *Visuelle Positionsbestimmung*

  Visuelle Positionsbestimmung findet beispielsweise bei dem Verfahren Visual Tags statt. Visual Tags sind Etiketten, die in einem Verfahren getragen werden müssen, um auf optischer Ebene mittels einer fest installierte Kamera erkannt zu werden. Dieses Etikett dient, dank seiner festen Größe, zur Bestimmung des Abstandes zu einer Kamera. Es kann aber ebenfalls die Ausrichtung eines Objektes festgestellt werden, da das durch die Kamera aufgenommene Bild der Etikette je nach Ausrichtung verzerrt erscheint.

- *Netzwerkgestützte Positionsbestimmung*

  Bei dieser Arte der Positionsbestimmung kann ein schon vorhandenes, drahtloses Netzwerk, etwa ein Mobilfunknetz, verwendet werden, um die eigene Position auf eine bestimmte *Zelle* einzuschränken. Befindet sich eine Person zusätzlich in einem *local area network*, beispielsweise einem W-LAN, so kann die eigene Position abermals präzisiert werden. Dazu werden die oben besprochenen Verfahren COO, sowie AOA und TOA kombiniert, um so die Vorteile der einzelnen Techniken zu erhalten. Diese Vorgehensweise erspart vor allem Kosten, beispielsweise zum Aufbau der Infrastruktur zur Positionsbestimmung.

  – GSM

    GSM-Netze bieten den Vorteil, dass auf mehreren Servern Daten über den Aufenthaltsort eines Netzbenutzers in einer Zelle temporär verteilt und gespeichert sind. Es werden zweierlei Daten gespeichert:
    Zum einen die *Zelle* in der sich ein Netzbenutzer befindet und zum anderen die Position in einer Zelle.
    Der Server, auf dem der Aufenthaltsorte eines Netzbenutzers gespeichert werden, nennt sich *visitor location register* (VLR). Auf einem zugehörigen *home location register* (HLR), der seine Daten von einem VLR erhält, werden Daten über den Aufenthaltsort (in einer Zelle) eines Netzbenutzers, ebenfalls temporär, gespeichert. Über diesen HLR sind dann die Aufenthaltsdaten abrufbar. Angaben über die Entfernung in Meter von einem Sendemast werden wiederum auf einem anderen Server festgehalten. Dieser Server ist ein so genanntes *mobile positioning center* (MPC). Die Genauigkeit der Abstandsmessung bestimmt sich aus dem verwendeten Verfahren. Drei Verfahren sind von den Netzbetreibern realisiert:

- CGI, *Cell Global Identity*: Nutzt die Zellinformation, um die Position auf einem Kreis um die Antenne zu ermitteln. Sind die auf einem Sendemast befindlichen Antennen jeweils auf einen bestimmten Sektor des Kreises ausgerichtet, so ist es zusätzlich möglich, festzustellen, in welchem Sektor auf dem Kreis der Zelle sich ein Mobiltelefon befindet.
  Die Positionsgenauigkeit ist abhängig von der Zellgröße.

- TA, *Timing Advance*: Wird dazu benutzt, die Position festzustellen, wenn sich das Mobilfunkgerät auf den Sendemast zu oder von ihm weg bewegt. Das TA-Verfahren steuert das Versenden eines Signals zu einem bestimmten zeitlichen Burst. Je weiter das Mobilfunkgerät vom Sendemast entfernt ist, desto jünger ist der Burst. Mit dem Senden innerhalb des jüngeren Burst ist gewährleistet, dass die Zeit, die zwischen Senden und Empfangen eines Signals innerhalb eines vorgegebenen Zeitschlitzes, vergeht.
  Die Positionierungs Genauigkeit liegt bei ca. 550m.

- UL-TOA, *Uplink Time of Arrival*: Über eine Laufzeitmessung des Signals eines Mobilfunkgerätes zu den Basisstationen kann die Position auf 50-150m bestimmt werden [(16)]. Dieses Auswertungs-Verfahren ist ähnlich dem Verfahren, das GPS verwendet.

– WLAN/Bluetooth

  Bei W-LAN [(34)] und Bluetooth-Verfahren [(41)] werden in einer Trainingsphase Karten mit Radio-Fingerabrdücken angelegt. Voraussetzung zur Erstellung einer solchen Karte ist aber letztendlich, dass die W-LAN-Karten und Treiber die Signaleigenschaften der Access-Points ermitteln können. Diese Signaleigenschaften sind als Received Signal Strength Indication (RSSI) bzw. Signal-to-Noise Ration (SNR) angegeben.

  Wird im Verlauf einer späteren Navigation eine Signalstärke gemessen, so wird diese mit einer Signalstärken eines Referenzpunktes aus der Trainingsphase verglichen. Ist die verglichene Signalstärken ähnlich der des Referenzpunktes, so ist davon auszugehen, dass die Position auch der des Referenzpunktes entspricht. Zu bedenken ist, dass eine solche Zuordnung nichts anderes ist, als eine Schätzung nach der Maximum-Likelyhood-Methode[4].

  Als nachteilig ist hier die Trainingsphase zu nennen. Diese muss, sobald ein Positionswechsel einer Basisstation stattgefunden hat, erneut durchgeführt werden. Umgangen werden kann diese Trainingsphase anhand eines Programmes,

[4] Bei dieser Methode wird die Dichte von Messergebnissen in einer Funktion erfasst. Anschließend wird diese Funktion maximiert. Das heißt es wird ein Wert gesucht, bei der die Funktion eine maximal Dichte- oder Wahrscheinlichkeitsfunktion aufweißt.

welches eine Liste mit Signalstärken selbstständig erstellen kann. Die Kommunikationsreichweiten der Systeme W-LAN und Bluetooth variieren zwischen 10 bis 100m (Bluetooth) und 300m (W-LAN). Die Genauigkeit liegt bei W-LAN zwischen 2-3m (mit Trainingsphase) und bei 4m, falls die Trainingsphase durch ein Programm ersetzt wird.

W-LAN und Bluetooth führen einem hohen Nachrichtenaustausch mit ihren Sensoren durch, um die Messungen durchzuführen. Dieser Nachrichtenaustausch wird aber zu Ungunsten begrenzter Ressourcen wie Netzwerkbandbreite und Prozessorzeit ausgeführt. Daher sollte der Nachrichtenaustausch über eine festverdrahtete Infrastruktur geleitet werden, um die Stand-by-Zeiten kabelloser Geräte zu erhöhen. Zudem sollte der Nachrichtenaustausch nur geschehen, wenn eine Positionsanfrage gestellt wurde. Dies erhöht wiederum die Skalierbarkeit, da so nicht übermäßig viele Nachrichten verschickt werden. Durch die so erhaltene, günstigere Skalierbarkeit können nun im System eine größere Anzahl an Benutzern angemeldet werden [(39)].

2.3. Indoor-Positionsbestimmung

Bei der Positionsbestimmung innerhalb von Gebäuden sind, wegen des erschwerten Empfangs von GPS-Signalen mit heutigen mobilen Geräten, autonome, GPS-unabhängige Positionierungssysteme aufzubauen. Unterschiede im Aufbau der Positionierungssysteme sind kostenbedingt und spiegeln sich in der Genauigkeit wieder. Bekannte Positionierungssysteme sind beispielsweise Active Badge oder Cricket. Sie sind eigens für den Einsatz in Gebäuden entwickelt worden. Diese Systeme haben eine Gemeinsamkeit: Sie benötigen eine Infrastruktur ähnlich der von GPS (Satelliten (Sender), Kontrollstationen, Empfänger usw.). Eine solche Infrastruktur ist dann in der Lage entweder Tracking oder Positioning oder beides zusammen durchzuführen, wobei die Methoden der Positionsbestimmung verwendet werden.

Einschränkungen in der Reichweite entstehen durch verschiedenartige Abschirmungen, wie sie beispielsweise in einem Gebäude durch umherlaufende Personen auftreten. Durch das Umherlaufen werden die Radiowellen[5] reflektiert[6].

Neben W-LAN-/ Bluetooth-basierten Positionierungssystemen gibt es auch andere Positionierungssysteme. Diese werden im folgenden, unterteilt nach ihrem Wellenspektrum, vorgestellt.

[5] Sie liegen im Frequenzband zwischen 2.400Hz und 2.483,5Hz

[6] Die Reflektion erfolgt durch das Wasser im menschlichen Körper in Abhängigkeit der Radiowellen des oben genannten Frequenzbandes.

Zu den Positionierungssystemen gehören:

Infrarot Ein Infrarot-System ist vergleichsweise günstig und hat eine hohe Verfügbarkeit. Allerdings lässt sich auf diese Art keine genaue Position bestimmen. Daher bleibt nur eine Schätzung über die Nähe des Aufenthaltsortes eines Benutzers zu einem Sensor.

- Active-Badge (Tracking):

  Ein bekanntes, bereits realisiertes System ist das *active badge system* von Olivetti. Eine *active badge* ist ein kleiner portabler Infrarotsender, der Impulse in einem bestimmten Intervall verschickt [(4)]. In diesem Verfahren wird die Reflektion von Infrarotstrahlen ausgenutzt. Demnach wird akzeptiert, dass die Strahlen innerhalb eines Raumes direkt oder durch Reflektion an ihr Ziel gelangen. Die Vital-Funktionen einer Badge werden über Batterien erhalten. Kollisionen der Strahlen sind auf Grund des gewählten Intervalles statistisch sehr gering [(16)]. Ein Client-Server-Architektur sammelt hierbei die Daten der Badges und stellt diese zum Abruf bereit. Zur eindeutigen Identifizierung einer *active badge* dient das so genannte *challenge-response*-Verfahren. Dieses Verfahren dient letztendlich der Sicherheit, fordert aber den Ausbau der Infrastruktur (serverseitig) und macht eine Zwei-Wege-Kommunikation notwendig. Die IR-Sensoren, die fest im Raum fixiert sind, sind untereinander und mit dem Server verbunden, also vernetzt.

- WIPS (Positioning):

  WIPS bedeutet *wireless indoor positioning system* und ist das umgekehrte Verfahren zu *active badges*. Allerdings liegt, wie der Name des Verfahrens schon andeutet, keine Verkabelung vor. Es handelt sich um ein Verfahren, bei dem die Sender-Barke fest im Raum fixiert ist und die Empfänger-Barke sich an einem sich bewegenden Objekt befindet. WLAN wird dazu verwendet, die Ortsinformationen von der Empfänger-Barke an den Server zu leiten und aufbereitet wieder zurückzusenden. Durch die kabellose Vernetzung eignet sich dieses Verfahren zum Empfang potenter ortsbezogener Dienste.

Die Genauigkeit bei diesen beiden Verfahren hängt vom Abstand Sender/Empfänger ab.

Funk Ein System, dass mittels Funk-Sender/-Empfänger aufgebaut ist, lässt Rückschlüsse auf eine wahrscheinliche Position zu. Auch außerhalb des Raumes, in dem sich ein bewegendes Benutzerobjekt befindet, ist diese berechenbar. Die Position wird durch Messung der Singalstärke gewonnen. Dabei wird berücksichtigt, dass die Signalstärke

mit zunehmender Entfernung abnimmt und beim Durchdringen von Materialien schwächer wird. Ein Indikator für die Qualität der Messung ist aber auch die Potenz der Abnahme der Signalstärke, die bei Funk quadratisch bei zunehmender Streckeneinheit ist.

- SpotON (Tracking):

  Dieses System funktioniert prinzipiell wie *active badge*. Statt Infrarot sind Funksensoren im Gebäude angebracht und mit einem Server und untereinander vernetzt. Funksensoren messen die Signalstärke und geben diese an den Server weiter. Der Server muss dann anhand der ankommenden Signalstärken eine Position finden, die zu einer früher gemessenen Signalstärke passt [(16)]. Die Positionsgenauigkeit liegt bei 3m (deterministisches Verfahren).

- RADAR [Karten mit Radio-Fingerabdrücken] (Tracking)

  RADAR ist ein Projekt von Microsoft aus dem Jahre 2000 das auf Karten mit Radio-Fingerabdrücken basiert. Der Ablauf des Verfahrens ist folgender: Mobile Endgeräte senden in regelmäßigen Abständen ein Radiosignal, welches zur Positionsmessung ausgesendet und von einem in der Umgebung befindlichen Access-Point gemessen wird. Der Access-Point wendet zur Positionsbestimmung den *k-nearest-neigbor* Algorithmus an. Das Verfahren hat eine Genauigkeit von 4 - 5m, wobei eine optimale Infrastruktur aus einer Anzahl von 4 bis 5 Access Points besteht[7].

- Rice [Karten mit Radio-Fingerabdrücken] (Tracking)

  Rice wurde an der University of Rice entwickelt und greift während der Lokalisierung sich bewegender Objekte auf Fluren oder in Räumen auf eine so genannte Fingerabdruck Datenbank zu. Mit Hilfe eines *Bayeschen-Filters*, bei dem eine bestimmte Signalstärke mit einer bestimmten Wahrscheinlichkeit an einem bestimmten, in einer Datenbank persistierten, Messpunkt vorkommt, werden die Entfernungen geschätzt. Das Verfahren nennt sich Markov-Lokalisierung und hat zur Besonderheit, dass die Wahrscheinlichkeit der letzten Position als *a-priori* Wahrscheinlichkeit für die neue Position in die Berechnung mit eingeht.
  Brauchbare Ergebnisse lassen sich noch mit einer Bewegungsgeschwindigkeit von 4m/s errechnen.

- PlaceLab [sensor fusion (vertikales Verfahren)]

  PlaceLab ist ein Projekt von Intel Research. Es stützt sich auf so genannte War-Driving-Datenbanken. Diese Datenbanken enthalten kartografisch festgehaltene

[7] Nicht bekannt ist allerdings, ob damit auch die Skalierbarkeit auf eine größere Fläche berücksichtigt ist.

Access Points, Bluetooth-Sender, Funkmasten und GPS-Positionen. Die *Wardriven-Datenbank* kann dann zur Navigation im Freien genutzt werden. Das Verfahren verwendet zur Bestimmung der in der Nachbarschaft befindlichen Objekte den *k-nearest neigbor* Algorithmus, beschränkt sich aber auf die Messergebnisse von 4 Access-Points; benutzt den Algorithmus also in abgewandelter Art. Auch Bluetooth wurde im Zusammenhang mit dem W-LAN-basierten PlaceLab auf seine Verwendbarkeit hin untersucht. Bluetooth ist aber aufgrund seiner beschränkten Reichweite eine ungeeignete Variante. Trotz dessen ist Bluetooth im Sensor-Fusions-Algorithmus berücksichtigt. PlaceLab beschränkt sich nicht ausschließlich auf W-LAN und Bluetooth sondern erfasst zusätzlich noch GPS und Mobilfunkzellen (GSM). Ein Testlauf in Seattle (einer Großstadt mit vielen Datenbankeinträgen) ergab einen Positionierungsfehler von 18,5 m und einen Positionierungsfehler von 30m in er Kleinstadt mit weniger Datenbankeinträgen (Probabilistisches Verfahren). Es ist daher für die Positionierung in Gebäuden ungeeignet.

- BIPS

  Beim *Bluetooth Indoor Positioning System* wird Nachbarschaftserkennung betrieben und so die Position unter Verwendung einer Infrastruktur ermittelt. Bei einem Nachbarschaftsscan können bis zu 22sec vergehen, um Geräte in Reichweite zu finden. Dies liegt an der Zeitspanne, während der ein Bluetooth Gerät bei der Frequenzspreizung auf einer Frequenz verweilen muss. [(11)]. Bluetooth-Geräte haben eine Sende- / Empfangsweite von 10 bis 100m bei einer Sendeleistung von 1mW. Dies bleibt aber aufgrund der geringen Sendeleistung auf innerräumliche Nachbarschaftserkennung beschränkt.

- Mankato [Entfernungspeilung]

  In Mankato wird die *Bit Error Rate* (BER) der L2CAP-Schicht von Bluetooth analysiert um anhand der Ergebnisse auf die Entfernung zu schließen. Der BER verschlechtert sich mit zunehmender Entfernung und eignet sich durch diese Eigenschaft zur Entfernungsmessung.

Positioning:

- RFID (ohne Server ohne Verkabelung):

  RFID oder *Radiofrequenz-Identifikation* nutzt so genannte RFID-Transponder, die mit einem kleinen Prozessor, einem Speicher und einer Antenne ausgerüstet sind. Der Speicher wird zum Speichern oder Weiterleiten von Daten an den Empfänger benötigt. Eine Besonderheit ist, dass die Transponder ihre Arbeitsenergie aus den empfangenen Funkwellen speisen. Ein ausgesendetes Signal wird allerdings nach einem Meter wieder empfangen und verarbeitet, so dass der Einsatz

meist an Förderbändern, auf denen der Weg eindeutig ist, vorbehalten ist. Das Verfahren dient somit der Wegekontrolle eines Objektes.

- MagicMap [Entfernungspeilung]

  MagicMap [(33)] verwendet das Received Signal Strength Indicator (RSSI) Peilverfahren, um Distanzen zwischen einem Referenzpunkt und einem Endgerät zu ermitteln. Vorab muss in einer Trainingsphase die Infrastruktur kalibriert und eingemessen werden. Mobile Endgeräte dienen in der Betriebsphase zur Weitergabe ihrer Position. Unter Zuhilfenahme von Positionen anderer mobiler Endgeräte werden Effekte der Signalstreuung statistisch ausgeglichen, was einer Erhöhung der Genauigkeit zu Gute kommt. Die Genauigkeit von MagicMap beträgt 1-5m. Eine genauere Erläuterung zu diesem Verfahren erfolgt in Kapitel 3.3.

- Hannover [Entfernungspeilung]

  Das Hannoversche Verfahren benutzt wie MagicMap das Received Signal Strength Indicator (RSSI). Die Entfernung zwischen Sender und Empfänger wird anhand der Empfangsstärke, die in dBm gemessen wird, geschätzt. Ein Modell mit Bluetooth-Geräten erbringt eine Genauigkeit von 2m.

Tracking/Positioning:

- Luleå [Entfernungspeilung]

  Eine Forschergruppe entwickelte in Luleå [(30)] ein Infrastruktur- und Endgerätebasiertes Positionierungssystem. Das Berechungsverfahren kann, abhängig von der Leistungsfähigkeit, wahlweise das Endgerät oder die Infrastruktur vorgenommen werden. Mit einem eigenen Entfernungspeilungs-Algorithmus erreichten die Autoren eine Genauigkeit von 1,7 m.

Ultraschallverfahren: Ultraschall hat gegenüber der normalen Funkwelle den Vorteil, dass die von der Welle zurückgelegte Strecke genauer ermittelt werden kann. Die bessere Genauigkeit gegenüber der Funkwelle folgt aus dem Schluss, dass die zurückgelegte Strecke proportional zu der verwendeten Signalstärke ist. Die Signalstärke der Funkwelle hingegen nimmt pro Streckeneinheit quadratisch ab.

Tracking:

- Active Bat (mit Server mit Verkabelung der Sensoren)

  Ein Bat sendet ein Ultraschall-Signal. Ein Sensor empfängt dieses und leitet es per Kabel an einen Server weiter. Die Sensoren, die das Signal erhalten, sind in einem Raster angeordnet, das dem Server bekannt ist. Der Server übernimmt dann die Aufgabe, ein durch die Messung erhaltenes, nichtlineares Gleichungssystem, ähnlich dem der Berechnung von GPS-Signalen, zu lösen. Anhand des

gelösten Gleichungssystems kann auf die Distanz zwischen Objekt und Sensor geschlossen werden. Zusätzlich misst der Server die Zeit, die vergeht, bis eine Rückkopplung eintrifft. Das Verfahren erreicht eine Genauigkeit von 10cm!

Positioning

- Cricket (ohne Server ohne Verkabelung)

  Das Verfahren ist die Umkehrung von Active Bat. Hier trägt das sich bewegende Objekt einen Sensor. Fest installierte Barken senden einen Ultraschall-Impuls. Zu diesem kommt ein gleichzeitig abgesendetes Funksignal. Anhand des Funksignals wird die Zeit gemessen, die der Impuls bis zum Empfänger braucht. Die Zeitmessung ist nötig, da keine weitere Einheit in diesem System existiert, die, wie ein Server dies könnte, die Zeit zwischen Aussenden eines Impulses und dem Empfang einer Rückmeldung misst.

Funk/Ultraschallverfahren IMAPS wurde von [(28)] an der HAW-Hamburg entwickelt. Es baut auf dem Cricket-System auf und nutzt daher ebenfalls Beacons, die ein Ultraschallsignal und ein Funksignal aussenden. Ein Listener fängt diese Signale auf. Sobald das Funksignal am Listener detektiert wurde, erfolgt der Start zweier Timer. Ein Timer zählt, bis 32 msek vergangen sind. Der andere Timer zählt, bis das Ultraschallsignal vom Listener empfangen wurde. Über die vergangene Zeit, bis das Ultraschallsignal eintrifft, werden Rückschlüsse auf die Entfernung getroffen. Tritt kein Ultraschallsignal ein, so ist der Listener nicht in der Nähe des Beacon, von dem das Funksignal ausgesendet wurde.

Wie beim Cricket-System muss es möglich sein, dass die Aussendung der Signale durch die Beacon ständig erfolgt. Dabei dürfen die Signale aber nicht kollidieren. Gelöst wurde das Problem durch das Senden nach einer zufallsverteilten Wartezeit. Nach dem Eintreffen des Ultraschallsignals werden die Positonsinformationen vom Listener errechnet.

Bei der Entwicklung von IMAPS wurde darauf geachtet, dass das System stromsparend bleibt. Bei einer Taktfrequenz von 8Mhz braucht beispielsweise der Kontroller 3,3 Volt Versorgungsspannung. Im Ruhezustand werden gerade einmal 7mA verbraucht. Die Stromversorgung wird durch Batterien gespeist.

Die IMAPS-Listener wurden für statische Entfernungen getestet. Die hohe Taktzahl der Messung deutet aber darauf hin, dass auch dynamische Objekte mit IMAPS erfolgreich erfasst werden können [(28)]. Innerhalb der Untersuchungen wurden nicht nur die Entfernung betrachtet, sondern auch die Reflektion in unmittelbarer Nähe von

hochfesten Materialien. Es zeigte sich, dass es nicht ratsam ist, Beacons in der unmittelbaren Nähe (< 1m) von hochfesten Baumaterialien, beispielsweise Beton, zu installieren. Durch zu nahes Installieren an solchen Materialien kann es zu Interferenzen kommen, wodurch die Signale verworfen werden.

Die vom Autor empfohlene Reichweite, um die Ultraschallsignale sicher zu empfangen, beträgt weniger als 120cm. Dieser Abstand muss eingehalten werden. Ebenfalls zu beachten ist, dass ein Listener ab einem Winkel von über 33 Grad keine Signale mehr von der Beacon empfangen kann. Bei einer Installation von IMAPS, die symbolische Positionsdaten ermitteln soll, sollten die Beacons daher eine geringere Entfernung als 120cm zueinander haben [(28)].

Das Empfangsmodul ist zur Übertragung der Daten mit einer seriellen Schnittstelle ausgestattet. In zukünftigen Versionen wird es aber angestrebt auf USB-Anschlüsse umzustellen oder eine Übertragung per Bluetooth zu realisieren.

Der Einsatzbereich von IMAPS bei der Ermittlung von physikalischen Positionsdaten wurde vor allem durch die starke Winkelabhängigkeit der Ultraschallsignale begrenzt. Diese Begrenzung führte dazu, dass eine sehr große Anzahl an Beacons benötigt werden, damit ein Listener beispielsweise in einem Museum flächendeckend physikalische Positionsinformationen ermitteln kann [(28)].

Die Messgenauigkeit liegt bei 1,2m und liefert damit die zweitbeste Genauigkeit der vorgestellten Verfahren.

Positionsbestimmung innerhalb von Gebäuden								
Infrarot f > 120THz		Funk f: 300MHz-30GHz			Ultraschall f > 20kHz		Visuell f > 348THz	Funk/Ultraschall f: 300MHz-30GHz/ f > 20kHz
Track.	Pos.	Track.	Pos.	Track./Pos.	Track.	Pos.	Track.	Pos.
Active Badge Gen.: 0,09m	WIPS Gen.: 3-4m	Spot On Gen.: 3m	RFID Gen.: 1m	Luleå Gen.: 1,7m	Active Bat Gen.: 2-30m	Cricket Gen.: 2-30m	Visual Tags Gen.: <1m	IMAPS Gen.: 1,2m
		Radar Gen.: 3-5m	MagicMap Gen.: 3-4m					
		Rice Gen.: 3-4m	Hannover Gen.: 2m					
		Place Lab Gen.: 3-5m						
		BIPS Gen.: 2m						
		Mankato Gen.: ?m						

Tabelle 2.1.: *Auflistung von Methoden zur Indoor-Positionsbestimmung aufgeteilt nach dem verwendeten Wellenspektrum und ihrer Klassen-Zugehörigkeit.*

2.4. Laufzeitumgebungen für moblie Geräte

In diesem Kapitel finden Technologien Erläuterung, mit denen die Entwicklung von Anwendungen auf mobilen Geräten vorgenommen werden. Dazu sollen zwei der namhafteren Technologien erläutert werden. Mittels dieser beiden werden derzeit, nicht ausschließlich aber vielfach, Anwendungen auf mobilen Endgeräten entwickelt.

.NET Compact Framework

Das .NET Compact Framework (.NET CF) von Microsoft bietet einem Entwickler die Möglichkeit eine Anwendung für ein mobiles Gerät zu entwickeln, um diese anschließend auf das Gerät zu portieren. Unterstützung auf mobilen Geräten findet diese Framework ab dem Windows-Betriebssystem CE 3.0. Mit Hilfe von so genannten .NET-Sprachen, wie C++, Visual Basic, C#, JScript, COBOL, lassen sich schließlich Anwendungen entwickeln.

.NET CF besteht aus einer Teilmenge von Klassenbibliotheken des .NET Framework, enthält aber auch eigens für das .NET CF entwickelte Klassenbibliotheken. Die mit Hilfe der .NET-Sprachen entwickelten Programme sind dann unter Vewendung der *Common Language Runtime* (CLR) lauffähig.

Da das .NET CF mit dem Hinblick auf die Interoperabilität zwischen anderen Microsoft Betriebssystemen entworfen wurde, können auch betriebssystemeigene Komponenten in eine Anwendung integriert werden. Auf diesem Wege können Anwendungen auf einfache und schnelle Weise entstehen. Dennoch bleibt die erstellte Software plattformabhängig, was deren Verbreitung stark einschränkt, zumal sehr wenige Mobiltelefone das .NET CF unterstützen.

J2ME

Der Einsatzbereich, der mit Mobiltelefonen, PDAs oder Pager erschlossen worden ist, nimmt einen hohen Stellenwert im heutigen Berufs- und Privatleben ein. Veranschaulicht wird dies durch einen Trend, der den gesteigerten Bedarfs an Mobiltelefonen und dem Abschluss von Mobiltelefon-Verträgen in den letzten fünf Jahre beschreibt. Schätzungsweise sind demzufolge heute weltweit 1 Milliarde solcher Geräte im Einsatz, was nach Angaben von dpa[8], dass nur noch 30% der Deutschen ohne Handy [(15)] sind, realistisch erscheint. Zusätzlich zum Boom in der Mobil-Telefon-Branche möchten immer mehr Menschen, Firmen und Organisationen mit dem Internet verbunden sein. Dies hat viele verschiedene Gründe, so

[8] Deutsche Presse Agentur

zum einen, um an Informationen zu gelangen, zum anderen, um Web-Inhalte, Firmen- oder auch persönliche Daten bereitzustellen. Geeigneter Weise soll dies zusätzlich von einem unbestimmten Ort, zu jeder Zeit und von verschiedensten Geräten möglich sein.

J2ME basiert auf der Java 2 Standard Edition (J2SE), besitzt aber nicht deren vollen Funktionsumfang/Klassenumfang. Der beschränkte Umfang von J2ME resultiert aus Ressourcenbeschränkungen der Zielgeräte, deren Systemarchitektur, der Leistungsfähigkeit und deren Anwendungsfeld. Mit der Mächtigkeit, der weit verbreiteten Nutzung und der erweiterbaren Entwicklungs-Plattform Java wird die Entwicklung von Diensten merklich erleichtert [(54)]. Durch die Konnektivität zum Internet oder einem anderen Netzwerk können zusätzlich Dienste in den Gerätespeicher geladen und ausführbar gemacht werden. Somit müssen mobile Geräte nicht mehr fest verdrahtet programmiert werden, um die Funktionalität eines Dienstes bereitzustellen. Auf dieser Grundlage kann eine individuelle Anpassung für jedes einzelnen Gerät vorgenommen werden, ganz so wie es sein Besitzer haben möchte. Ebenso bietet sich die Möglichkeit Dienste und Services auf den Geräten vorzuinstallieren um bestimmte Benutzergruppen anzusprechen und so den Absatz der Geräte zu beeinflussen.

Grundlagen zu J2ME

Eine umfassende Referenzimplementierung zu J2ME ist nicht entstanden, wohl aber Referenzimplementierungen für einzelne Spezifikationen oder Gruppen von verwandten Spezifikationen [(48)]. Daher gibt es für J2ME nur Spezifikationen, die aufeinander aufbauen, sich überlappen und sich teilweise ausschließen. J2ME ist, genauer gesagt, eine Zusammenfassung von Technologien von Spezifikationen und teilt sich in die Konfigurationen (CLDC & CDC-Spezifikationen der virtual machine (VM) und der Basis API für Geräte mit RISC/CISC Microprozessor/Controller) und Profiles (MIDP 1.0/2.0, Foundation Profile) auf.
J2ME ist, seit Aufspaltung von Java 1999, für Ressourcen beschränkte Geräte vorgesehen. Zudem ist es gegenüber J2SE bezüglich des Java-Klassenumfangs eingedampft worden. Beispielsweise fehlen Programmierschnittstellen gegenüber den Klassenbibliotheken von J2SE. Desweiteren ist ein „just-in-time“ Compiler im Leistungsumfang der VM nicht mehr vorhanden. Hinzu kommt, dass auch nicht der volle Java-Sprachumfang durch die VM unterstützt wird.

Trotz des geringeren Java-Klassen-Umfanges bleibt J2ME aufwärtskompatibel. Der Grund ist, dass die in J2ME enthaltenen Java-Klassen immer als echte Teilmenge zu den J2SE-Klassen zu verstehen sind. Leider aber sind nicht alle Klassen auf diese Weise vererbt. Beispielsweise jene nicht, die innere Abhängigkeiten in sich tragen. Das betrifft die Klassen der Sicherheit, der Ein-/Ausgabe, der Benutzerschnittstelle, des Speicherzugriffs und für die Vernetzung. Sie wurden an die Belange der Konfigurationen angepasst.

J2ME besteht in ihrer High-Level-Architektur aus den drei Teilen:

1. Konfigurationen
2. Profile
3. optionale Pakete

Die J2ME dient der obigen Architektur als so genanntes Rahmenwerk. Zu dieser high-level Architektur spezifiziert J2ME ebenfalls zwei verschiedene, an die Konfigurationen angepasste VMs.

Konfiguration

Der Grund der Entwicklung von Konfigurationen und Profilen, Kapitel 2.4, bestand darin, die große Spanne an Geräten mit unterschiedlicher Hardware in bestimmten Punkten zu vereinen und somit dem wachsenden Markt dieser Geräte eine Schnittstelle im Sinne der Plattform-Unabhängigkeit zu bieten. Diverse, schon bestehende Applikationen und Features, sollten ihre Verwendungsmöglichkeit beibehalten, gleichfalls aber auch die Entwicklungen nützlicher Applikationen und Fähigkeiten für den zukünftigen Gebrauch voran getrieben werden.

Ein Konfiguration hat das Ziel, mobile Geräte anhand ihrer Hardware in eine *horizontale* Klassen einzuteilen. (Eine solches Ziel aus einer Vielfalt von Geräteherstellern zu erreichen, resultiert aus der Bindung der unterstützenden Hersteller, den vollen Umfang der Spezifikation zu implementieren [(48)].) Somit entscheidet auch der Funktionsumfang einer Konfiguration über die Leistungsfähigkeit eines Gerätes der horizontalen Klasse. Eine Konfiguration stellt Klassenbibliotheken (Core Libraries) und eine VM bereit [(48)]. Anders ausgedrückt, definiert eine Konfiguration die minimale Java Technologie eines Gerätes einer bestimmten Gerätegruppe mit ähnlichen Merkmalen an Speicher und Prozessorleistung. Beispielsweise trennt eine Konfiguration über die Merkmale *kabellos* bzw. *verkabelt*, dass kabellos verbundene Geräte der CLCD- und verkabelte Geräte der CDC-Konfiguration angehören.

Der Vorteil einer Konfigurationen ergibt sich durch die Art der Aufteilung und die dadurch erworbene Plattformunabhänigkeit, auch *Cross Platform*-Fähigkeit genannt. Beispielsweise bestimmt sich dynamischer Inhalt durch den plattformunabhängigen Austausch.

Die Konfigurationen und die Zuordnung von Geräten

J2ME teilt Geräte in zwei horizontale Klassen (Konfigurationen) ein. Diesen beiden Konfigurationen sind die Konfiguration Connected Device Configuration (CDC) und die Connected Limited Device Configuration (CLDC). In den Dokumentationen der Konfigurationen sind

Einzelheiten über die Java-Programmiersprache, die JVM und die Java Application Programming Interfaces (API) enthalten. Das Dokument für die CLDC trägt den Namen java specification request (JSR)-139 bzw. -30 und das für die CDC den Namen JSR-36. Zu finden sind diese unter [(55)].

Zur Einordnung der verschiedenen Geräte in eine der beiden Konfigurationen dienen folgende Merkmale:

- Hauptspeicher
- Prozessorleistung
- Energiebedarf

Im folgenden wird nun beschrieben, welche Hardware-Eigenschaften Geräte besitzen müssen, um einer der beiden Konfigurationen zuordenbar zu werden. Zunächst erfolgt dies für die CDC, anschließend für die Geräte der CLDC.

Geräte der CDC

Die Geräte der CDC sind leistungsfähige, nicht mobile und vernetzte Geräte wie es beispielsweise Set Top Boxen, Videotelefone oder auch Entertainment- und Navigationssysteme in Fahrzeugen sind. Im Gegensatz zu Geräten der CLDC können Geräte der CDC einen höheren Funktionsumfang bieten. Die Einschränkungen durch die CDC sind im Vergleich zu Desktop-PCs weniger drastisch, da Geräte der CDC über einen entsprechend ausgelegten RAM-Speicher (ab 2MB) verfügen und die Prozessorleistung entsprechend hoch ist. Hinzu kommt, dass Geräte der CDC komfortable Bedienoberflächen bieten, einen Festspeicher von 2 bis 16MB besitzen und über eine leistungsstarke TCP/IP Netzwerkverbindung verfügen.
Die einzige Einschränkung gegenüber Desktop-PCs betrifft die Java-Klassen-Bibliotheken und in ihnen vorhandene Schnittstellen.

Diese Einschränkung besteht im Beibehalten von grundsätzlich benötigten Klassen-Bibliotheken und einer entsprechenden Überarbeitung dieser. Ansonsten sind sie identisch zu den Klassen-Bibliotheken der J2SE.

Für die CDC-Konfiguration wurde die *CDC HotSpot Implementation* entwickelt, die eine voll leistungsfähige Java virtual machine (VM) ist und keiner Einschränkung bezüglich des zu unterstützenden Klassenumfanges unterliegt. Damit wurde die Voraussetzung geschaffen, auf Geräten mit 32-bit Prozessoren und mehr als 2 MB Speicher (PDAs, Bild-Telefone und Auto-Navigationssysteme) kompatibel zu J2SE-Anwendungen zu sein. Einschränkungen der VM sind aufgrund der Leistungsstärke der Geräte nicht nötig.

Geräte der CLDC

Geräte, die der CLDC [(38)] entsprechen, haben vereinfachte Benutzerschnittstellen, einen geringen Speicher und oftmals eine instabile Netzwerkverbindung. Aus der instabilen Netzwerkverbindung folgt, dass sie nicht auf dem TCP/IP-Standard basieren [(47)]. Die CLDC-Spezifikation schreibt eine untere Schranke für den Speicher und die Vernetzung des Gerätes vor. Die Begrenzung des Speichermediums darf das Minimum an verfügbarem, nicht flüchtigem Speicher

- CLDC 1.0: ROM 128 kB
- CLDC 1.1: ROM 160 kB

nicht unterschreiten. Dieses Minimum entspricht dem von der CLDC HotSpot (kiloByte virtual machine) und den CLDC Bibliotheken benötigtem Speicher. An flüchtigem Speicher sollte bei CLDC 1.0/ 1.1 nicht weniger als 32kB bereitstehen, damit die Laufzeitumgebung (kVM[9]) darauf zurückgreifen kann.

Neben der Ressource Speicher schreibt die CLDC eine Netzwerkverbindung als „erforderlich“ vor. Die Bandbreite dieser Netzwerkverbindung, die sich in der Baudrate wiederspiegelt, darf nicht unter 9600 Baud liegen. In [(48)] wird beschrieben, dass eine drahtlose, bidirektionale Kommunikation mit geringer Bandbreite und einer nicht permanenten Verbindung für die CLDC 1.0 und 1.1 vorgesehen ist.

Im Gegensatz zu Geräten der CDC verfügen CLDC-Geräte neben der Stromversorgung via Kabel zusätzlich über eine mobile Stromversorgung.

Für das Betriebssystem auf einem der mobilen Gerät der CLDC gelten minimal Anforderungen; gerade genug, um die Steuerung der Hardware zu gewährleisten. An ein Betriebssystem werden durch die CLDC weder eine Forderung auf getrennte Adressräume erhoben, noch wird eine Ablaufsteuerung für mehrere Prozesse gefordert. Lediglich der Prozess der kVM ist zu überwachen.

Die CLDC legt auch Aufgaben fest, die ein Gerät erfüllen muss:

- Festlegung der Java-Sprache und die Merkmale der VM
- definieren des Kerns der Java Bibliothek [java.lang.*, java.util.*]
- Ein- und Ausgabe auf dem Endgerät [java.io.*]
- Vernetzung
- Sicherheit

[9] kVM bedeutet kilo-Byte virtual machine und deutet auf den Speicherbedarf dieser VM hin. Sie unterstützt die Konfiguration CLDC und trägt daher den Namen *CLDC HotSpot Implementation*.

- Internationalisierung
- die nutzbaren Klassenbibliotheken

Die Funktionalitäten der Aufgaben sind in Java-Klassen-Bibliotheken hinterlegt. Die Schnittstellen der CLDC-spezifische Klassen befinden sich in *javax.microedition.io*. Dieses Package stellt eine Reihe von Verbindungsklassen zur Verfügung und wird unter dem Namen Generic Connection Framework (GFC) zusammen gefasst. Durch dieses Framework wurde der Zugriff auf Verbindungsschnittstellen gegenüber der *java.util.io* durch sieben Interfaces vereinheitlicht. Die Methode *javax.microediton.Connector.open* gibt beispielsweise eines der sieben Interfaces zurück, das mit der Angabe des Protokolles (file, http, https, socket, ssl, datagram, comm) in der Methode angefordert wurde.

Sicherheitskonzept der CLDC

Ein 2-Ebenen Konzept verhindert die angesprochenen Bedrohungen. Das Sicherheitsmodell der CLDC ist in zwei Ebenen geteilt, eine untere und eine obere Ebene. Die untere Ebene sorgt dafür, dass die VM richtig läuft/funktioniert; sprich, dass das Gerät nicht abstürzt und dass kein fremder Code dem Gerät schadet. In der unteren Ebene werden die Klassen einer Pre-verifikation unterzogen. In dieser Überprüfung wird nach verbotenen Operationen gesucht, und auf falsche Referenzen geprüft. Dieses pre-verifying findet nicht erst auf dem CLDC-Gerät statt sondern schon auf dem Desktop-Rechner des Entwicklers. Beim pre-verifying werden korrekte Dateien mit einem StackMap-Attribut versehen.

Zur unteren Ebene gehört auch eine zweite Prüfung, bei der zur Laufzeit auf dem CLDC-Gerät eine so genannte *runtime verifikation* durchgeführt wird. Diese Laufzeit-Überprüfung besteht in der Kontrolle der zuvor gesetzten StackMap-Attribute, verbraucht aber sehr wenig Systemressourcen.

Auf oberer Ebene, der Ebene der Anwendung, besteht der Sicherheitsaspekt in der Kontrolle der Aktionen, die eine Anwendung durchführt oder durchführen möchte. Zudem wird sichergestellt, dass von der Anwendung nur auf die freigegebenen Bibliotheken und Systemressourcen zugegriffen wird. Realisiert wird diese Kontrolle durch die, schon bei der Entwicklung von Java erarbeitete, so genannte *sandbox*.

Die *sandbox* spiegelt den folgenden Sachverhalt wieder: Java-Applikationen dürfen nur auf Klassenbibliotheken und Systemressourcen zurückgreifen, die vorher definiert wurden. Die Typsicherheit (keine beliebige Typkonvertierung) von Java garantiert, dass eine Applikation keinen Zugriff auf die Systemressourcen hat, und unterstützt somit das Sandbox-Konzept. Die Speicherverwaltung wird vom Interpreter vorgenommen.

Ein so genannter *ByteCode-Verifizierer* existierte zur Kontrolle der wichtigsten Regeln (Typsicherheit) und zur Kontrolle des Kontroll- und Datenflusses. Ohne diesen wäre es beispielsweise mit einem Bytecodeassemblierer (Jasmin) möglich, gefährlichen ByteCode zu erzeugen, etwa eine Endlosschleife, und diesen unbemerkt in ein schon vorhandenes Programm einzuschleusen. Das Programm würde laufen, da der Bytecode auf dem PDA selbst nicht geprüft würde. Um dem entgegen zu wirken und um die Vertrauenswürdigkeit zu gewährleisten, werden überprüfte Midlets mit einem Hashcode signiert. Diese signierten Midlets werden Schutzbereichen zugeordnet. Ein Schutzbereich ist eine Sammlung von Zugriffsrechten (trusted, untrusted). Die Signaturen der Midlets werden nach ihrer Zuweisung im Applikationsdeskriptor eingetragen. Der Applikationsdeskriptor ist eine Datei mit der Endung *.jad.

Eine Signatur setzt sich zusammen aus einer Prüfsumme über das zu signierende Dokument und deren Verschlüsselung mit einem privatem Schlüssel. Das Signieren übernimmt ein JadTool, das zum Lieferumfang von J2ME gehört.

Weitere Beispiele zeigen auf, in welchen Bereichen die Sicherheit ebenfalls erhöht wurde:

- In der CLDC wurde durch den Zusatz, die Laufzeitumgebung nicht auf die eigenen Bedürfnisse zu berichtigen, die Sicherheit erhöht.
- Beim dynamischen Download von Java-Applikationen wird darauf geachtet, dass keine Systemklassen überschrieben werden.
- Durch die so genannte Policy-based security kann zusätzlich durch den Entwickler festgelegt werden, auf welche Daten, Schnittstellen oder Anwendungen ein Benutzer zugreifen darf. Einmal angepasst, können diese nur noch vom Systemadministrator geändert werden.
- Zur Verschlüsselung von Daten bei der Übertragung auf ein anderes CLDC- oder kompatibles System dient die Java Cryptography Architecture (JCA).
- Das so genannte Certificate Management sorgt für eine Authentifizierung und erzeugt ein Zertifikat, um einem Benutzer einen entsprechenden Zugriff zu ermöglichen. Ein Zertifikat ist gleichbedeutend einem so genannten *öffentlichen Schlüssel* (public key) innerhalb einer *public key infrastructure* (PKI).

Fazit: Midlets besitzen nur eingeschränkte Funktionalität, sind signiert und haben nur ein minimales Zugriffsrecht auf den Speicher des Gerätes. Zudem unterstützten sie die verschlüsselte Kommunikation.

Limitierung des Klassenumfanges bei CLDC

Wie erwähnt, wird durch die CLDC nicht der volle Klassenumfang in J2ME bereitgestellt. Durch das Fehlen von Klassenbibliotheken folgen auch eine Reihe von Einschränkungen. Von den Einschränkungen sind ebenfalls die Fehler- und Ausnahmebehandlung und das class-Loading betroffen.

- Die Entfernung der finalize-Methode: Mit dem Entfernen der Methode zum sicheren Schließen einer offenen Verbindung (finalize-Methode) bleibt dem Prozessor ebenfalls eine erhöhte Rechenleistung erspart.
- Auch durch das Weglassen von asynchronen Exceptions und eingeschränkter Implementierung anderer Exceptions spart J2ME beim Sicherheitskonzept ebenfalls an Platz.
- Das Laden von Klassen nach dem CLASSPATH-Konzept wurde zu Gunsten einer höheren Sicherheit nicht realisiert.

Auch die Java-VM ist von den Einschränkungen betroffen. Dies äußert sich zunächst durch die Veränderung des Namens der VM in kVM. Die kVM dient aber weiterhin zum Ausführung von Java-Anwendungen bei Geräten der Konfiguration CLDC und ist deren Schlüsselmerkmal. Weiterhin stellt sie sicher, dass alle Anwendungen in korrekter Weise zur Ausführung gelangen. Die kVM benötigt zur Erledigung ihrer Aufgaben lediglich 40-80 kB Speicher. Der reduzierte Speicherbedarf, im Vergleich zur JVM, entstand im Groben durch den Wegfall des bytecode-Verifiers.

Profile

Neben den besprochenen Aufgaben gibt es auch Disziplinen, die nicht Bestandteil der CLDC sind. Diese Disziplinen betreffen:

- die Installation
- die Ausführung und
- das Entfernen einer Anwendungen

Ebenso finden in den Konfigurationen die Benutzerschnittstelle und Ereignisbehandlung keine Berücksichtigung. Sie sind ebenfalls Teilgebiete der Profile.

Allgemein sind Profile für Entwickler und Anwender in vielfältiger Weise von Bedeutung. Sie legen die Leistungsanforderungen eines CLDC Gerätes für ein so genanntes vertikales Marktsegment fest, beispielsweise das *Personal Information Management*. Ein Profil, wie es

in der J2ME vorkommt, stellt Klassenbibliotheken, in der Leistungsmerkmale implementiert sind [(48)], bereit.

Die Wahl des Profils, das ein CLDC-Gerät unterstützen soll, trifft der Hersteller bei der Entwicklung. Hat sich ein anderer Hersteller für das gleiche Profil entschieden, so ist es möglich, eine identische Anwendung auf beiden Geräten auszuführen. Dies stellt den Vorteil für den Verbraucher dar, den Profile zusammen mit Konfigurationen aufweisen.

J2ME stellt insgesamt folgende Profile zur Verfügung:

- für die CLDC:

  MIDP 2.0 und das IMP (Information Module Profile (machine-to-machine Kommunikation))

- für die CDC:

  Foundation Profile, Personal Basis, Personal

Profile und Konfigurationen ergänzen sich. Profile ergänzen Konfigurationen um Klassenbibliotheken, womit sich vollwertige Anwendungen für mobile Endgeräte der Gerätekonfigurationen CDC und CLDC implementieren lassen. Mit MIDP 2.0 lassen sich beispielsweise Implementierungen in den folgenden Bereichen durchführen:

- Steuerung des Applikationszyklus und Packaging (javax.microedition.midlet)
- Bedienoberflächen, UI mit high and low-level API (javax.microedition.lcdui)
- Medienverarbeitung
- Netzwerkfunktionalität (HTTP etc.) (javax.microedition.io.HTTPConnection)
- Verwaltung persistenter Daten, Datenspeicherung (RMS API) (javax.microedition.rms)

Anders ausgedrückt sind dies die Aufgaben, die Profile übernehmen. Hinzu kommen, wie schon erwähnt, die io- und util- Packages der Konfiguration CLDC (java.lang, java.io, java.util).

MIDP 2.0

Das MIDP 2.0 [(37)] ergänzt die Konfiguration von CLDC-Geräten um die oben angegebenen Bereiche. Das Ziel dieses Profiles ist, eine erweiterte Architektur mit samt den dazugehörigen APIs zu definieren. Die API soll es ermöglichen, von dritten eine Entwicklungsumgebung erstellen zu lassen, um damit Applikationen für mobile Informationsgeräte (MIDs) zu entwickeln .

Es stellt sich die Frage, welche Geräte der *vertikalen Klasse* dem Profil MIDP entsprechen. Da MIDP die CLDC um Klassenbibliotheken ergänzt, ist daraus ableitbar, dass es Geräte aus der *horizontalen*-Gerätegruppe CLDC sein müssen.

Da der verfügbare Speicher von Geräten der CLDC variieren kann, ist auch der maximale Speicherverbrauch variabel. Dies ist beispielsweise bei der Implementierung zu berücksichtigen, wenn es darum geht, Speicher zu alokieren. Hat ein CLD einen genügend großen Speicher, so lassen sich bequem Dateien in den Dimensionen von mehreren Mega-Byte auf das Gerät laden, beispielsweise Spiele oder Anwendungen zur Outdoor-Navigation. Letztendlich schreibt die CLDC keine Beschränkung in der maximalen Speicherausrüstung vor. Somit obliegt es dem Hersteller, das CLD potent oder weniger potent auszurüsten.

Neben der Notwendigkeit eines genügend großen Speichers ist auch die Netzwerkverbindung obligatorisch. Um eine Netzwerkverbindung aufzubauen, wird die Anforderung in folgenderweise erhoben: Eine 2-Wege Verbindung soll ohne die Verwendung eines Kabels aufgebaut werden können. Die Verbindung darf aber lückenhaft sein und eine limitierte Bandbreite besitzen. [(37, S.8)] Ein Provider hat die Vorschrift eine Installation eines Programmes mittels i-mode oder WAP über den Browser des CLD zu gewährleisten und darf zusätzlich auch Bluetooth oder IrDA anbieten. MIDP-Bibliotheken müssen ein Netzwerkzugriff über HTTP/1.1 gewährleisten. Falls aber ein CLD einen Browser besitzt, der einen WAP-Protokollstack benutzt und das WAP Transportprotokoll beinhaltet, so darf WSP anstelle von HTTP genutzt werden [(37, S.19)].

Midlet-Suite

So genannte Midlets/Midlet-Suites sind lauffähige Applikationen auf einem CLD. Die MIDP 2.0 übernimmt ebenso die Aufgaben der Verwaltung des *Lebenszyklus* einer Midlet-Suite. Eine Midlet-Suite hat folgende Lebenszyklen:

Mittels MIDP können sog. Midlets entwickelt werden. Diese Midlets verfügen über Methoden, die den Lebenszyklus bestimmen. In ihnen wird bestimmt, was während des Starts der Anwendung (*startApp()*) geschieht oder zum Abschluss des Midlets (*destroyApp()*) ausgeführt werden soll. Die Methode (*pauseApp()*) kann beispielsweise dazu genutzt werden, die Anwendung aufgrund eines eingehenden Telefonanrufes pausieren zu lassen bis das Telefonat beendet ist.

Beispielsweise sei gesagt, dass der Entwickler einer Midlet-Suite, das sich in der Installation befindet, dazu verpflichtet ist, die Installation jeder Zeit optional vom Benutzer stoppen zu lassen. Kommt es während der Installation zu einem Fehler, sind Error-codes vom CLD an den Server zu schicken. Error-codes, die vom Server kommen, sind auf dem CLD zu interpretieren.

Um die Midlet-Suites vor Unbefugten zu schützen, stellt MIDP 2.0 das Konzept von vertrauenswürdigen Applikationen vor. Vertrauenswürdige Applikationen dürfen demnach APIs benutzen, die als empfindlich und zugangsbeschränkt gelten [(37, S.23)]. Eine Midlet-Suite gilt als nicht vertrauenswürdig, falls die Herkunft und die Integrität von der CLD nicht bestätigt werden kann. Diese laufen dann in einer nicht vertrauenswürdigen Domain innerhalb eines abgegrenzten Speicherbereiches ab.

Das Konzept der Vertrauenswürdigkeit von Midlets basiert, wie beschrieben, auf geschützten Domains und zusätzlich aus den Einstellungen eines Domain-Besitzers. Die Domain definiert die Genehmigungen für eine Midlet-Suite. Der Domain-Besitzer selbst spezifiziert, wie ein CLD eine Midlet-Suite als vertrauenswürdig zu identifizieren hat und zu überprüfen ist (37, S.24). Überprüft wird anhand einer Signaturen, die mittels einer X.509 public-key-infrastructure erstellt wurde.

Optionale Pakete

Optionale Pakete erweitern die zuvor besprochenen Profile um weitere APIs oder Bibliotheken. Der optionale Charakter dieser Pakete bedeutet, dass sie bei Bedarf verwendet werden können, beispielsweise um einen Web-Service zu nutzen. In Bezug auf die Hardware bedeutet dies, dass sie das erwünschte optionale Paket schon besitzt, es also bereits zum Funktionsumfang gehört oder ob es nachträglich installiert werden muss. Dies sollte der Entwickler vor der Entwicklung einer J2ME-Anwendung berücksichtigen. Es gibt eine Vielzahl solcher APIs, die in einem solchen Zusammenhang genannt werden können. Hier soll es genügen, zunächst einige wichtige darzustellen, die mit der Indoor-Navigation in enger Verbindung stehen.

- Bluetooth API (JSR-82)

  Kommunikation über die Bluetooth Schnittstelle

- Web-Service API (JSR-172)

  Nutzung entfernter Dienste über das SOAPProtokoll

- File-Connection API (JSR-75)

  Umgang mit Dateien und Verzeichnissen, die auf einem beliebigen Speichermedium vorliegen.

- Scalable 2D Vector Graphics API (JSR 226)

  Darstellung von Vektorgrafiken, welche auch im SVG-Format vorliegen können.

Optionale Pakete bieten den Vorteil der Erweiterung der Profile und daher des Java-Sprachumfanges. Allerdings wird dieser Vorteil mit einer verschlechterten Portabilität eingekauft. Die Anwendung ist so nicht mehr universell auf allen Endgeräten nutzbar [(29)].

Analyse

Unter den angeführten Plattformen hat JavaME die größte Verbreitung und bietet zudem den umfangreichsten Funktionsumfang. Auch für Entwickler, oder die, die es werden wollen, bieten sich Foren an, bei denen sich Hilfe verschafft werden könnte. Sun bietet ebenfalls die Möglichkeit in einem Forum Fragen zu stellen. Ebenso hilf- und umfangsreich wird in Büchern das Wissen über J2ME bereitgestellt. Aus diesen genannten Gründen wird Java zur Entwicklung eines Indoor-Navigations-Systems herangezogen.

3. Vergleichbare Arbeiten

In diesem Kapitel werden bestehende Out- und Indoor-Navigationssysteme vorgestellt. Deren Vorstellung dient der Studie der Darstellungsweise von Karten auf mobilen Geräten. Anhand dieser Studie sollen wichtige Design-Elemente herausgefiltert werden, die zum Entwurf der Anzeige des entstehenden Indoor-Navigationssystems dienen sollen.

Nach dieser Studie erfolgt eine Darstellung von bereits existierenden Indoor-Navigationssystemen. Da sowohl digitale Outdoor- als auch Indoor-Navigationssysteme zur Navigationshilfe auf standortbezogene Dienste zurückgreifen, soll in diesem Zusammenhang zunächst der Begriff Location Based Services (LBS) erläutert werden.

3.1. Location Based Services

Location Based Services [(52)] (LBS) ist ein Teilgebiet von GIS[1]. Die Bezeichnung LBS beschreibt den Ortsbezug von Diensten, welche nach Ermittlung der aktuellen Position eines Gerätes, dass sich in der Nähe befindet, so genannte kontextbezogene Daten an das Gerät verschicken können.

Der Ortsbezug kann dazu verwendet werden, einen Benutzer zu einem Zielort zu führen, etwa auf einer Reise mit dem Auto. Dazu dient beispielsweise ein Navigationssystem, das auf Grundlage von digitalisierten Karten den Weg beschreibt. Ein solches Navigationssystem beschränkt sich nicht allein auf das Anzeigen der Route; es liefert vielmehr auch Informationen über die Umgebung. Neben Navigationssystemen zeigen auch Touristenführer Informationen über Umgebungen, in Form von historischen Daten über Gebäude, an. LBS sind speziell für den nicht kommerziellen Anwendungsbereich erdacht. Hierbei wären beispielsweise Notsituationen im Rahmen von Emergency Services zu nennen. Auch in Supermärkten werden anhand von LBS Angebote offeriert, beispielsweise sobald ein Kunde an einem Regal vorbeigeht. LBS ermöglichen es einem Benutzer auch E-Mails zu senden oder zu empfangen. Dazu ist es allerdings weiterhin notwendig, mit dem Internet verbunden zu sein. Schließlich

[1] Dies ist ein „rechnergestütztes Informationssystem, das aus Hardware, Software, Daten und den Anwendungen besteht. Mit ihm können raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und grafisch präsentiert werden.“ R.Bill, 1994

bleibt noch zu erwähnen, dass der Server-Dienst, der zu der Anwendung benötigt wird, ein solcher LBS ist, da er die Karten bereitstellen und die Route berechnen soll. Er beschränkt sich im Rahmen dieser Arbeit auf mobile Geräte und bietet seinem Benutzer über den Bezug, der zuvor ermittelten Position, ortsbasierte Dienste an. Daher soll er im Anschluss an dieses Kapitel beschrieben werden.

LBS sind nicht so weit verbreitet wie anfangs der neunziger Jahre erdacht, erschließen dennoch einen großen Markt. Dieser Markt funktioniert jedoch nur, wenn die angebotenen Services qualitativ hochwertig sind, dem Nutzer Informationen liefern, die eine ausgeprägte und aktuelle Datengrundlage haben, sowie ansprechend aufbereitet sind. Der Mehrwert eines solchen Service muss sich dem Nutzer auf den ersten Blick erschließen. Ein solcher Markt erschließt sich beispielsweise nach folgenden Eigenschaften ortsbezogener Dienste:

- Geschäftsbezogene Eigenschaften

  Von Interesse für den Dienstanbieter sind Geschäftsmodelle wie: Consumer-to-Consumer, Consumer-to-Business, Business-to-Bussiness, Business-to-Consumer (Information, Kommunikation, Unterhaltung, Transaktion uvm.), Consumer-to-Government, Business-to-Workforce.

- Weitergabe der Position

  Werden Positionen an Dritte weitergegeben, so muss die Vertraulichkeit gewahrt bleiben. Dies ist beispielsweise in zellbasierten GSM-Netzen der Fall.

- Technische Eigenschaften

  Zur Unterscheidung der technischer Eigenschaften von LBS, kann man diese Services als push- und pull-Dienste bezeichnen. Die Unterscheidung beschreibt neben dem technischen Aspekt auch die Art der Interaktion. So arbeiten pull-Dienste nur auf Anforderung und push-Dienste durch Ereignisse, wie es beispielsweise ein Bewegungsereignis ist.

Der Fokus dieser Arbeit liegt im Bereich der geschäftsbezogenen Eigenschaft von Business-to-Consumer, die Weitergabe der Position erfolgt über eine Funkverbindung. Weitere Eigenschaften von ortsbezogenen Diensten können bei (48) nachgelesen werden.

All diesen Eigenschaften ist gemein, dass sie einer definierten Genauigkeit entsprechen, das bedeutet aber auch, dass beispielsweise die Kategorie „ortsbezogene Informationsdienste“ eine geringere Genauigkeit als „Trackingdienste“ oder „Notfalldienste“ aufweisen.
Im Mobilfunk-Bereich erlaubte der Erfolg von GSM die kommerzielle Einführung von ortsbezogenen Diensten. Bei der Bestimmung der Position des mobilen Gerätes ergibt sich aber durch das Fehlen der Hardware die Schwierigkeit zur Positionsbestimmung. Die besprochenen Genauigkeiten in GSM-Zellen reichen nicht aus, um eine genaue Position zu bestimmen. GPS, dessen Signale mittels einer GPS-Mouse, die mittels Bluetooth mit dem mobilen Gerät

verbunden ist, empfangen werden können, reicht zur Positionsbestimmung ebenfalls nicht aus. Gerade aber in Gebäuden ist es nötig, eine genaue Position ($\pm$ 1m) berechnen zu können, denn ein Fehler von wenigen Metern kann schon zur falschen Wegentscheidung führen.

Zur beispielhaften Erläuterung eines LBS soll ein Projekt der HAW dienen, dass einen Offering-Service und einen Routenplaner bereitstellt.

Innenraum-Routenplaner

Dieser LBS wurde von (40) als Web-Service entwickelt. Der Service ist in der Lage anhand einer Position und einem Zielort eine Route auf einer Datenbasis eines Gebäudes zu berechnen. Er baut auf offenen Standards, wie z.B. SOAP[2], WSDL[3], UDDI[4] auf, um so den Kauf von Lizenzen zu ersparen. Da der Service auf Grundlage dieser Standards aufbaut, treten auch keine Probleme mit Firewalls auf, wie sie beispielsweise bei JavaRMI der Fall wären.

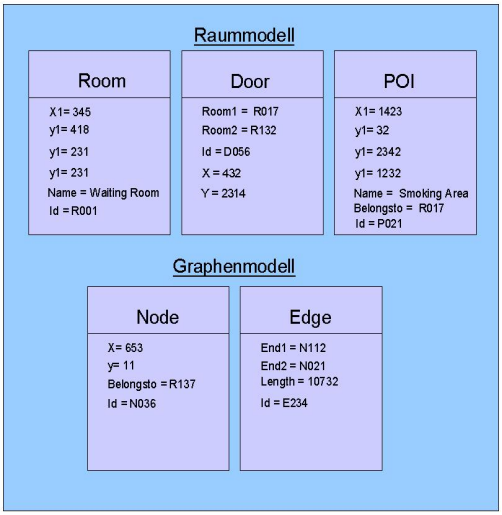


Abbildung 3.1.: *Kartendaten-Basis des Routenplanungs-Dienstes*

Der Web-Service bietet zwei Dienste an. Zum einen, besagten Routen-Planer und zum anderen, einen so genannten Offering-Service. Beide befinden sich in einem Web-Service-Container. Die Route, die der Routen-Planer berechnet, wird einerseits zu statistischen Zwecken auf dem Server gespeichert und andererseits dazu verwendet, dem Benutzer im Rahmen des Offering-Services während der Routenbegehung Angebote zukommen zu lassen.

[2] Simple Object Access Protocol (64)
[3] Web Service Definition Language (65)
[4] Universal Description, Discovery and Integration (61)

Der Service wurde in Java implementiert. Diese Entscheidung wurde getroffen da Java anerkannter Marktführer für die Netzwerkprogrammierung ist [(40)].

Der Server allein ist darauf ausgelegt, die ganze Business Logik des Systems zu übernehmen. Der Hauptgrund dafür ist die geringe Leistung mobiler Geräte, die schließlich den Service nutzen sollen. Daher werden die schwerwiegenden Aufgaben dem leistungsstarken Server auferlegt. Die Karten, die dem mobilen Geräte zugeschickt werden, befinden sich allerdings auf einem separaten Datenbank-Server (Database-Tier).

Das Konzept der Gebäudedaten sieht vor, dass die Karten in Form von XML-Dateien an den mobilen Client geschickt werden. Ein Auszug aus der Gebäudedaten-Basis ist in Abbildung 3.1 dargestellt. Zur Routenberechnung wurde der A*-Algorithmus verwendet, der eine Weiterentwicklung des Dijkstra Algorithmus darstellt und aus der Spiele-Industrie stammt. Zusätzlich zur Routendarstellung wird auch eine Routenbeschreibung generiert. Es ist noch festzuhalten, dass die Blickrichtung durch das System nicht berechnet werden kann. Die Orientierung basiert daher auf Türen einer Etage, die auf der Anzeige des mobilen Gerätes beschriftet angezeigt werden. Im Besonderen wird auch die Routenabweichung in einem Intervall bestimmt. Dazu dient der Datenaustausch der aktuellen Position mit dem Server, der aus diesen Daten die Abweichung zu Punkten der zuvor generierten Route berechnet.

Im folgenden Kapitel wird anhand von verwendeter Navigationssoftware dargelegt, welche Gemeinsamkeiten die Anzeige bei der Navigation aufweißt.

3.2. Outdoor-Navigationssoftware

Zur Planung eines Indoor-Navigationssystems soll auf schon vorhandene, bewährte Outdoor-Navigationssysteme zurückgegriffen werden. Dies geschieht um herauszufinden, welche grundlegenden Funktionen und Eigenschaften Navigationssysteme aufweisen, um diese für die Indoor-Navigation heranzuziehen. Im folgenden werden daher die verschiedenen Navigationssysteme auf Funktionalitäten untersucht, die nützlich für das eigene Indoor-Navigation sein könnten.

QueMap

QueMap von (22) setzt eine interaktive Anzeige des PDA voraus. Das bedeutet, dass es mit einem PDA-Pen möglich ist, Einfluss auf das Angezeigte zu nehmen. Die Software wurde von Garmin für das Betriebssystem Palm OS implementiert.

Die Anzeige der QueMap stellt die Umgebung des aktuellen Standortes detailgetreu dar. QueMap erlaubt das Umschalten der Sichten von der Aufsicht in die schräge Vogelperspektive. Die aktuelle Position wird durch eine schwarzfarbige Pfeilspitze angezeigt, wobei der Richtungssinn nicht dargestellt wird. Der Routenfortschritt ist ebenfalls kenntlich gemacht. Somit ist ein genügender Kontrast der einzelnen Elemente gewährleistet.

Die Farbe der Landmasse ist grün, die See wird blau dargestellt und die Autobahnen sind rot. Bundesstraßen haben die Farbe Schwarz und Siedlungsstraßen sind grün. Beschriftungen werden ebenfalls in schwarz angezeigt. Grenzen sind als schwarz-gestrichelte Linie auf weißem Hintergrund eingezeichnet.

Auf der Karte wird in der linken oberen Ecke symbolisch eine Kompassnadel angezeigt, die in Nordrichtung weist. Der zurückgelegte Weg kann anhand einer gestrichelten Linie angezeigt werden.

Abbildung 3.2.: *Die Que-Suite in 2D*

Abbildung 3.3.: *Die Que-Suite - Where am I?*

Zusammengefasst können also mit QueMap folgende Aktionen ausgeführt werden:

- Ermittlung des aktuellen Standortes
- Anzeige des zurückgelegten Weges
- Lokalisieren und Navigieren zu nahegelegenen Kartenelementen

- Verfolgung der Routenlinie ins Ziel
- Hervorheben und Anschauen von Informationen
- Skalierungsfaktor verändern (der Skalierungsfaktor reicht von 800feet/inch bis 800miles/inch)
- Anzeigeformat verändern
- Einen bestimmten Kartenausschnitt vergrößern/verkleinern
- Detaillierungsstand ändern
- Entfernungsberechnung auf einem Kartenausschnitt vornehmen (dazu wird ein Referenzpunkt auf der Karte festgelegt)
- Umschaltbar zwischen 2D- und 3D-Anzeige

Der normale Detaillierungsgrad umfasst Autobahnen, Straßen, Seen und Flüsse, Zwischen-Stationen-Marker und Siedlungsstraßen. Der Detaillierungsgrad kann durch den Benutzer bestimmt und nachgeladen werden. Dazu wird das vorhandene Kartenmaterial mit Kartenmaterial einer CD ergänzt.

Weitere Funktionen werden von anderen Software-Teilen übernommen, beispielsweise übernimmt QueFind das Suchen eines Ortes und die Routenberechnung zu diesem Ort.

Destinator

Die Software Destinator, von (14), setzt eine interaktive Anzeige voraus. Im folgenden wird die neuere Version, Destinator 6, in der 3D-Ansicht beschrieben. Die 2D-Ansicht der Abbildung 3.5 stammt aus einer älteren Version. Von Interesse bei der Handy-Version ist, dass sie einen Kompass auf der Anzeige darstellt sowie die Route, die Richtung und die Distanz ins Ziel.

Die Anzeige von Destinator 6 stellt Haupt- und Nebenstraßen, Straßen-, Brücken- und Namen von Plätzen, sowie Bahnstrecken rund um den aktuellen Standort dar. Der Routenfortschritt ist weniger gut zu erkennen und die Route selbst ist in weißer Farbe dargestellt. Die aktuelle Position ist als blaufarbige Pfeilspitze dargestellt. Die Pfeilspitze zeigt in Fahrtrichtung geradeaus. Die Farbe der Landmasse ist grün, Hauptstraßen sind leicht gelblich, die Nebenstraßen werden in einem Blauton gehalten. Desweiteren werden Flüsse in einem hellen Blauton; Beschriftungen in schwarzer Farbe angezeigt. Waldstücke werden mit symbolisch dargestellten Bäumen hervorgehoben. Der Kontrast ist weniger gut ausgeprägt.
Ein Maßstab ist ebenfalls vorhanden und wird unterhalb des Abzweigungshinweises in kleiner Schrift angegeben.

Bei starker Vergrößerung werden Ortschaften in einer dunkelgrünen Farbe angezeigt und der Detaillierungsgrad bleibt zunächst derselbe. Allerdings kann der Detaillierungsgrad vom Benutzer angepasst werden. Entsprechend werden bei solchen Vergrößerungsstufen ausschließlich Bundesstraßen und Autobahnen angezeigt.

Abbildung 3.4.: *Der Destinator in 3D*

Abbildung 3.5.: *Der Destinator in der Handy-Version*

POIs, wie Restaurants, Tankstellen, Hotels uvm. können nach Kategorien geordnet eingeblendet werden. Diese Funktionalität ist manuell einzustellen.

Ein Kompass wird dazu verwendet, die Fahrtrichtung anzuzeigen, d.h. nicht die Nordrichtung. Ein Abzweigungshinweis ist in der linken oberen Ecke zu finden, rechts daneben die „Wegbeschreibungsliste“. In der 2D-Ansicht wird ein Abzweigungshinweis direkt an Kreuzungen angezeigt. Vergrößerung und Verkleinerung des Maßstabes können über die Anzeige vorgenommen werden, wobei ein Pen benutzt werden sollte. Am unteren Rand ist die ausstehende Fahrtstrecke und die mögliche verbleibende Fahrtzeit in Minuten Angezeigt.

Das Navigationssystem verfügt unter anderem über den folgenden Funktionsumfang:

- Routenberechnung
- qualitativ hochwertige 2D/3D-Karten mit Anzeige der Straßennamen

- Schnellnavigation für die Navigation zu vordefinierten Adressen
- Adresseingabe über eine große, für die Fingereingabe optimierte Tastatur
- Speicherung beliebiger Positionen mit nur einem Mausklick
- Adressabruf aus den Points of Interest (POI)
- detaillierte Bildschirm- und Sprachanweisungen in 20 Sprachen
- Planung und Optimierung von Reisen mit Mehrfachstopps
- Autonavigation mit schnellster oder kürzester Route
- kurzes Lernprogramm für den Schnelleinstieg
- Grenzüberschreitende Navigation ohne Kartenwechsel

Navigator 5

Navigator setzt eine interaktive Anzeige voraus und ist von (59) entwickelt worden. Im folgenden wird Navigator in der Version 5 beschrieben, welche sowohl eine 2D- als auch eine 3D-Ansicht bietet.

Die Anzeige des Navigator 5 stellt in der 3D-Ansicht innerstädtische Haupt- und Nebenstraßen in weißer Farbe dar. Bahnstrecken werden blau-weiß-gestrichelt angezeigt. Die Landmasse ist in hell-grüner Farbe dargestellt. Der Routenfortschritt ist nicht sichtbar, wohl aber die gesamte Route, welche durchgehend in Rot angezeigt wird. Die aktuelle Position ist als blaufarbige Pfeilspitze dargestellt. Die Pfeilspitze zeigt in Fahrtrichtung geradeaus. Flüsse werden in einem hellen Blauton, Beschriftungen in schwarzer Farbe mit weißem Schatten angezeigt. Zudem werden Waldstücke gegenüber der übrigen Landmasse in einem dunkleren Ton hervorgehoben. Der Kontrast ist weniger stark ausgeprägt, dafür aber ist die Anzeige in folge des Fehlens von Details übersichtlich gehalten.

Die 3D-Anzeige stellt Vergrößerungs- und Verkleinerungssymbole in den oberen Ecken dar, allerdings nur für die PDA-Version. In der Handy-Version können mittels zweier Tasten Vergrößerungen oder Verkleinerungen vorgenommen werden. Ein Maßstab ist ebenfalls vorhanden und wird unterhalb des Abzweigungshinweises in kleiner Schrift dargestellt. Zum Komfort der Anwendung gehört auch, dass der Straßenname am unteren Rand zusätzlich dargestellt wird; gleich darüber befindet sich auch ein zusätzlicher Abzweigungshinweis mit Angabe der Distanz bis ins Ziel. Dieser wird ergänzend zu dem auf der Karte befindlichen angezeigt. Ebenfalls im unteren Drittel auf der rechten Seite befinden sich Angaben über die ausstehende Distanz und die verbleibende Zeit ins Ziel.

In der 2D-Anzeige werden auf der Karte ein Kompass und ein Maßstab angegeben. Diese beiden Elemente fehlen in der 3D-Anzeige. Die Ansicht zeigt Ortschaften in einem dunkleren Farbton im Vergleich zur üblichen Landmasse und den Wäldern an. Der Maßstab kann mittels eines Pen verändert werden.

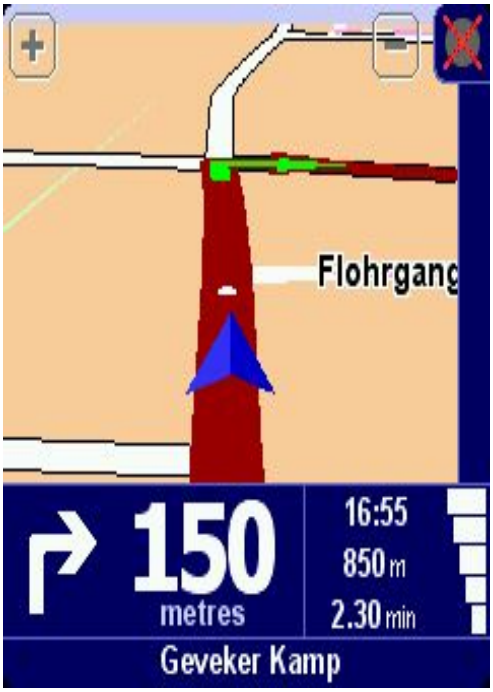


Abbildung 3.6.: *Der Navigator5 in der PDA-Version (3D)*

Abbildung 3.7.: *Der Navigator 5 in der Handy-Version (3D)*

Die 2D- und die 3D-Ansicht bieten die Möglichkeit zur Auswahl und Anzeige von POIs. Diese können aus einer Liste ausgewählt werden.

Navigon 7100

Navigon von (43) länzt mit einer fotorealistischen Ansicht in der 3D-Perspektive. Die Software wurde von der Navigon AG erstellt und ist PDA und Mobile Phone tauglich. Jegliche Art von Straßen ist in der 3D-Ansicht weiß dargestellt, dementsprechend wird die Route in kontrastreichem gelb angezeigt. Zur Darstellung des Standortes dient ein tellerförmiges 3D-Objekt mit gelbem Pfeil, das in Fahrtrichtung weißt.

Die Software setzt eine mindest Taktfrequenz von 500MHz voraus und benötigt eine Ar-

beitsspeicher von 128MB-RAM. Für mobile Geräte, beispielsweise ein Smartphone wird zusätzlich ein Betriebssystem Windows vorausgesetzt.

Das Stadtbild ist in Grautönen dargestellt. Die Landschaft, beispielsweise Felder, Wiesen und Wälder, sind in Grüntönen dargestellt. POIs werden wahlweise angezeigt, so ist beispielsweise schlechtes Wetter als Regen-Wolke zu wählen. Ein Spuranzeige zeigt vorausschauend die Straßenspur, die augenblicklich zu befahren ist, um damit dem nächsten Abzweigungshinweis nicht zu verpassen. Desweiteren wird der Name der Straße angezeigt, auf der sich der Benutzer befindet. Auf das Ziel wird mittels erwarteter Ankunftszeit, noch ausstehende Kilometer und Zeit bis ins Ziel hingewiesen. Der Kontrast wurde stark berücksichtigt und auch hier eine übersichtliche Karte geschaffen.

Die Software Mobile Navigator von Navigon kann von der Navigon-Homepage kostenlos auf Java-Handys geladen werden.

Abbildung 3.8.: *Das Navigon 7100 in 3D*

Abbildung 3.9.: *Das Navigon 7100 in der Handy-Version (3D)*

Funktionen	QueMap	Destinator	Navigatior	Navigon 7100
Route	ja	ja	ja	ja
Standort	Pfeil	Pfeil	Pfeil	Pfeil
Name des Standortes	ja	ja	ja	ja
Zurückgelegter Weg	nein	nein	nein	ja
Maßstab	ja	nein	nein	nein
Kompass	ja	ja	nein	nein
Distanz zum Ziel	nein	ja	ja	ja
Zeit zum Ziel	ja	ja	ja	ja
Ankunftszeit	ja	nein	ja	ja
Uhrzeit	ja	nein	nein	nein
Abzweigungshinweise	ja	ja	ja	ja
Entfernung zum Abzweig	ja	ja	ja	ja
POIs	ja	nein	nein	wahlweise
Zoom	ja	ja	ja	ja
Darstellungsqualität	befriedigend-gut	gut	gut	sehr-gut

Tabelle 3.1.: *Die Tabelle zeigt die Gemeinsamkeiten auf, die bei den vorgestellten Navigationssystemen der Fahrzeugnavigation vorkommen*

Analyse

Die folgende Analyse soll die wichtigsten Merkmale der Navigation mit dem Auto hervorheben. Wesentlich für den Erfolg einer Navigation mit dem Auto als auch für die Indoor-Navigation ist, keine Abzweigung innerhalb der vorgegebenen Route zu verpassen. Daher wird in diesen Systemen auch mit doppelten Angaben auf diese Abzweigungen verwiesen. Eines der vier Navigationssysteme ist zwar auf dem neusten Stand der Technik, diese spiegelt sich jedoch nur in der Darstellungsqualität wieder. Die Darstellungsqualität dieses Navigations-Systems ist abhängig von der verwendeten Hardware, wobei diese einen 400MHz Prozessor besitzt (Navigon 7100).

Anmerkung zu Que-Map: Der Pfeil, der den Standort bezeichnet, zeigt auf den Punkt, an dem sich der Benutzer befindet; nicht in die Richtung der Route.

Der Standort ist stets mit einer Pfeilspitze gekennzeichnet. Mit Ausnahme von QueMap zeigen diese Pfeilspitzen in Fahrtrichtung. Ebenso wichtig ist der aktuelle Standort; er wird anhand eines Straßennamen beschrieben und deutlich angezeigt. Ebenso bedeutungsvoll ist die Distanz ins Ziel, sowie Abzweigungshinweise, die Entfernung zur Abzweigung und die Zoomfunktion. Ein weiterer wichtiger Aspekt, der aber für ein Indoor-Navigationssystem vernachlässigt werden kann, ist die Zeit bis zum Erreichen des Ziels. Die Distanz und die Ankunftszeit sind als wichtige Attribute bei drei der vier Modelle berücksichtigt. Kompass

und POIs sind weniger oft berücksichtigt, wobei im neusten System die POIs auf Wunsch angezeigt werden können. Der zurückgelegte farblich verdeutlichte Weg, der Maßstab und die Uhrzeit sind Elemente, die nur vereinzelt in die Anzeige übernommen werden.

Zu berücksichtigen ist bei diesem Navigationstyp die Geschwindigkeit. Daher wird das Vorausschauen auf die nächste Verkehrssituation sehr bedacht. Die Verkehrssituation ist aber nicht vergleichbar mit der Schrittgeschwindigkeit von Personen. Daher fällt die Geschwindigkeit bei der Indoor-Navigation weniger ins Gewicht fällt und daher sind Elemente wie „Zeit zum Ziel“ und „Ankunftszeit“ zweitrangig.

Die wichtigen Elemente der Outdoor-Navigation werden auch in das Projekt der Indoor-Navigation miteinbezogen und sollen zusätzlich auf der Anzeige dargestellt. Diese Elemente werden angegeben:

- Route
- Standort
- Name des Raumes/des Flures in dem sich befindet
- Ausrichtung des Gebäudes gegen Nord
- Abzweigungshinweis
- Entfernung zur Abzweigung
- Zoomfunktion
- POIs

3.3. Indoor-Navigationssysteme

Für die Navigation in Gebäuden werden LBS herangezogen, um für einen Standort Karten und eine erstellte Route bereitzustellen. Die Karten weisen bestimmte kognitive Präsentationenformen der Indoor-Umgebung auf. Diese Präsentationsformen berücksichtigen, dass eine Entscheidungsfindung häufig auf dem in der Kindheit erlernten metrischen Abstraktionsvermögen des Raumes basiert und durch weitere Erfahrungen verbessert werden kann [P.Tomberge:2004] Die Entscheidungsfindung bei der Navigation beruht auf dem Zusammensetzen einzelner Faktoren im Sinne von:

- Selbstlokalisation
- Ziellokalisation
- Routenwahl

und der Wegfindung anhand

- Richtungswahl
- Wegwahl
- zielgerichteter Fortbewegung

Die Wegwahl hängt von den im Kapitel 2.1 angesprochenen Landmarken ab, also von der Sichtbarkeit entscheidungsunterstützender Objekte.
Bei der Navigation in Gebäuden kann die Sichtbarkeit dieser Landmarken stark eingeschränkt sein, da Wände, Türen, Säulen u.ä. die Sicht versperren können [Lynch:1960]. Eine entscheidende Rolle spielt die Dimension. Bei einer Routenplanung im Outdoor-Bereich ist es die Ansicht in mehreren Ebenen weniger wichtig. Viel wichtiger im Outdoor-Bereich ist die Bewegungsrichtung, anhand derer die aktuelle Straße identifiziert werden kann. Im Indoor-Bereich sollte, falls in mehreren Stockwerken navigiert werden soll, die dritte Dimension Berücksichtigung finden. Mit Hilfe dieser kann die aktuelle Etage angezeigt werden.

Zur Navigation in Gebäuden ist aus den oben genannten Gründen eine Geräte-Infrastruktur nötig, die es ermöglicht, quasi durch Wände hindurch und um Hindernisse herum zu schauen. Neben der Visualisierung ist auch die Position und die Positionsgenauigkeit von großer Bedeutung. Schwierigkeiten bei der Indoor-Navigation treten oftmals beim Lesen von Hinweisschildern und dem Identifizieren von Landmarken auf.

Da bis heute GPS zur Positionsbestimmung innerhalb von GPS-Signal abgeschirmten Gebäuden nicht genutzt werden konnte, werden eine Vielzahl von Sender-Empfänger-Systemen zur Positionierung/Ortung herangezogen, vergl. Kapitel 2.3. Erst mit dieser Technik ausgestattet, ist der Mensch in der Lage, durch ein Gebäude zu navigieren und an ein Ziel zu gelangen.

MOVING-Indoor Navigationssystem der Uni Münster

MOVING von (23) ist ein Projekt der Uni Münster. Sie entwickelte in Zusammenarbeit mit Herrn H.J. Müller, der ebenfalls der Uni Münster angehörig ist, das Navigationssystem MOVING. Der Name MOVING leitet sich von MObile Verbal Indoor NaviGation ab.

Das Navigationssystem arbeitet zur Routenbestimmung auf Basis eines zweidimensionalen digitalen Modells eines Gebäudes. Auf Grundlage dieses Models berechnet MOVING den kürzesten Weg zwischen einem Startpunkt und einem Zielpunkt. Die Eingabe von Start und Ziel erfolgt textuell; die Ausgabe der Route ist jedoch an ein *text-to-speech*-System gekoppelt. Aufgrund dieses *text-to-speech*-Systems erhält der Benutzer die Richtungshinweise in akustischer Form. Das System erwartet eine ständige Rückkopplung durch den Benutzers. Infolge dessen muss sich der Benutzer auf das Navigationssystem konzentrieren. Mit dieser

akustischen Präsentationsform hebt sich MOVING von den bekannten Methoden der visuellen Darstellungsform ab. Das Negative an dieser Verfahrensweise ist, dass der Benutzer hören und verstehen muss. Gerade dies kann ein Nachteil darstellen, falls einen großer Geräuschpegel herrscht und der Benutzer durch diesen einen akustischen Hinweis versäumt.

Das Prinzip der Navigation beruht auf Landmarken, an denen sich der Benutzer orientieren muss. Zu diesen Landmarken zählen beispielsweise durchquerte Türen, die Anzahl der passierten Türen, die Nutzungsart der passierten Räume, Flurenden, Pfeiler, durchquerter Raumtypen wie z.B. Raum, Flur und Treppenhaus.

Die Vorarbeit, die bei MOVING geleistet werden muss, besteht in der Erstellung digitaler Modelle. Ein Model des Gebäudes und ein weiteres von der Routen. Die Digitalisierung dieser erfolgt durch Scannen der Gebäudepläne und Transformieren in shape-files[5], die Attribute über die Gebäudedaten enthalten. Ein solches digitales Modell beinhaltet somit die besprochenen Landmarken einer Ebene. Mit dieser Technik ist es denkbar, auch in eine andere Ebene eines Gebäudes zu navigieren.

Das System basiert auf einer Applikation für Geoinformationen, ArcView GIS 9.1 von ESRI[6]. Die gesamte Applikation ist in den ArcGIS-Desktop eingebettet und läuft auf jedem handelsüblichen Desktop PC. Zukünftig soll es auch möglich sein, mit ArcGIS einen Service zu erstellen, um mit einem mobilen Gerät auf diesen zuzugreifen. Allerdings muss zu diesem Zweck ArcGIS auf dem mobilen Gerät installiert werden [(23)].

Die Sicherheit bezüglich Benutzerdaten ist unkritisch und daher gegeben. Dies wurde dadurch erreicht, dass der Benutzer keinerlei Daten über das eigene Gerät austauscht.

Im Fokus der Studie stand die Untersuchung von Landmarken und deren Angemessenheit bei einer Navigation. Die Autoren untersuchten die Wirkung von Landmarken und Abzweigungshinweise bezüglich des Vorankommens bei der Durchquerung eines Gebäudes. Ebenfalls wurde untersucht, zu welchem Zeitpunkt es Sinn macht, Türen zu zählen und zu welchem nicht. Festzuhalten bleibt, dass Landmarken dem Benutzer ausschließlich der Orientierung entlang einer Route dienen und Abzweigungshinweise zum Hinweis auf das Ändern der Laufrichtung. Im Gegensatz zu Abzweigungshinweisen sind Landmarken routenbegleitend zu finden.

Der Nutzen dieser Arbeit ergibt sich aus dem Ergebnis der Studie. Demnach haben Landmarken eine nicht zu vernachlässigenden Effekt auf die Navigationszeit und die Navigationsfehler. Zu dem gibt die Ausarbeitung einen Hinweis, Landmarken dann einzusetzen wenn sie nützlich sind. Der Einsatz ist unbestritten, doch die Verwendung von Landmarken kann durchaus überflüssig und ineffektiv sein.

[5] Das Dateiformat Shapefile ist ein von ESRI ursprünglich für ArcView entwickeltes Format für Geodaten.

[6] ESRI (Environmental Systems Research Institute (19)) ist einer der großen Softwarehersteller von Geoinformationssystemen (GIS)

Aufgrund der Ergebnisse von MOVING ist zu klären, welche Landmarken für die Ausarbeitung dieser Arbeit in Betracht gezogen werden. Sobald dies geschehen ist, erfolgt eine Einteilung der gefundenen Landmarken in eine unscharfe Bezeichnung, beispielsweise notwendig, weniger notwendig. Dies soll Bestandteil des Kapitels 4.3 sein.

CricketNav

CricketNav ist ein Projekt von (42), in dem er mit Hilfe von Cricket ein Indoor-Navigationssystem aufbaut. Der Prototyp der Anwendung ist in Java 2 und Java Swing geschrieben und daher nicht auf einem Gerät der CLDC lauffähig. CricketNav wurde in einer späteren Version auch für ein Handheld, die der CDC entsprechen, konzipiert und entwickelt. .

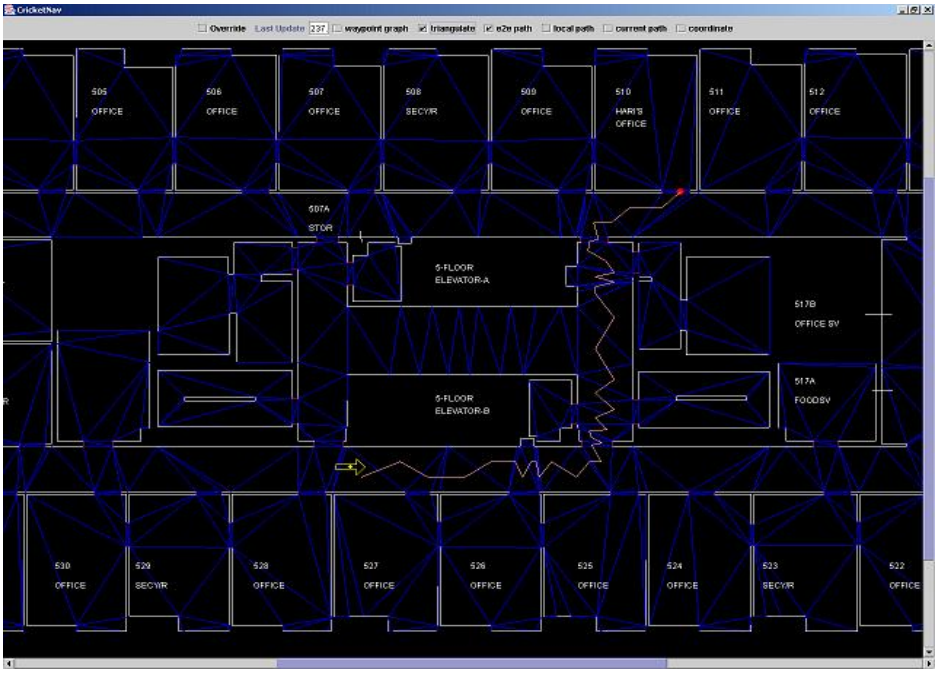


Abbildung 3.10.: *CricketNav auf einem Desktop-PC*

CricketNav setzt auf dem Cricket-Positioning-System des MIT auf. Aufgrund der verwendeten Positionsbestimmungsmethoden, (Funk in Kombination mit Ultraschall) ist es dem Positioning-System dieser Anwendung (IMAPS) sehr ähnlich und deshalb von Interesse. Die verwendete Gebäude-Karten basieren auf CAD-Zeichnung und werden von einem *Spatial Information Service-Server* (SIS-Server) bereitgestellt. Sie können durch das mobile Gerät nach dem Senden einer Message an den Server heruntergeladen werden.

Die Positionsaktualisierung des Benutzers ist aufgrund der Infrastruktur bewusst variabel gehalten. So ergeben sich Orte, an denen Aktualisierung öfter bzw. weniger oft pro Zeiteinheit stattfinden. Dies soll den Benutzer dazu animieren, Orte zu finden, an denen eine schnellere Aktualisierung stattfindet. Die Abzweigungen werden dem Benutzer anhand von Pfeilen im voraus, noch vor der Abzweigung angezeigt. Dieser Pfeil soll eine unzureichende Abzweigungshinweise unterbinden. Ein Pfeil weist immer auf einen Durchgang oder eine Tür hin. Zusätzlich, um Mehrdeutigkeiten zu vermeiden, wird in einem Durchgang oder einer Tür, auf den sich ein Pfeil bezieht, ein *Punkt* dargestellt. Sobald vom System festgestellt wird, dass der Benutzer in die falsche Richtung läuft, erscheint ein roter Pfeil, der dem Benutzer dies anzeigen soll. Die Autoren unterstellen dem Cricketsystem Fehlberechnungen in der Positionsbestimmung. Diese können sie nicht von vornherein ausschließen. Aus diesem Grund erlauben sie es dem Benutzer, eine selbständig Korrektur der eigenen Position vorzunehmen. Neben dieser Korrektur, kann der Benutzer auch eine Etage überfliegen, um diese so eigenständig zu erkunden. Zusätzlich dazu kann die Routeneingabe auch auf akustischem Wege erfolgen.

Neben dem User-Interface entwickelte der Autor die algorithmische Umsetzung des Aufenthaltsortes, der Pfad-Planung und der Richtungen der Abzweigungshinweise.

Magicmap

MagicMap [(33)] wurde am Lehrstuhl für Informatik an der Humboldt Universität in Berlin erarbeitet. Die Arbeit spiegelt die drahtlosen und allgegenwärtigen Vernetzung wieder, in der die Position und die Interaktion mit den umgebenden Räumlichkeiten eine starke Rolle spielt. MagicMap basiert auf der W-LAN-Funktechnik und ist daher ein Signalstärken-basiertes Ortungsverfahren. Die Anwendung ist in Java geschrieben und läuft auf den Betriebssystemen Windows (XP, Pocket PC) und Linux.

Außer den Access-Points sind zum Gebrauch von MagicMap weitere Geräte nicht erforderlich. Die Stärke des Systems liegt in der *ad-hoc*-Fähigkeit, was schließlich auch der Forschungsanlass der Autoren war. Zudem kommt die große Verbreitung von W-LAN in der Öffentlichkeit und die Anzahl an Anbieter von W-LAN-Infrastrukturen hinzu. Zu den Stärken dieses Ortungsverfahrens gehört das Entfallen der Einmessung der Bezugspunkte und der Wegfall der Synchronisation innerhalb der Infrastrukturen. Das bedeutet, dass es auch ohne Kenntnis über die umgebenden Access-Points möglich ist und ausschließlich über den Austausch von Messdaten, eine Positionsbestimmung durchzuführen.

Die Messung der Signalstärken erfolgt bei Windows mittels NetStumbler und unter Linux mit den Wireless Tools. Aufgrund dieser Programmabhängigkeit ist die Plattformunabhängigkeit nicht gegeben.

Infolge der Kalibrierung ergibt sich aber die Schwierigkeit, eine dynamische Veränderung, etwa durch Hindernisse, in der Umgebung zu erfassen. Nachteilig ist hierbei ein ständiges Updaten der Position, das durch diese Bewegung zustande kommt. Somit kann es zwangsläufig zu statistischen Mittelungsfehlern[7] kommen. Ein solcher Fehler tritt auf, wenn für die Anzahl der Referenzpunkte weniger als drei Access-Points verwendet werden. Durch den Fehler ziehen die Referenzmessungen den Peer in Richtung des Mittelpunktes. Den Positionierungsmehrdeutigkeiten soll ein manuelles Drag-and-Drop, das vom Benutzer vorgenommen werden kann, entgegenwirken. Zu dem soll der Benutzer befähigt werden, seine aktuelle Position oder die Position des Access-Points per Hand zu korrigieren. Somit ist sukzessive sichergestellt, dass eine Karte auf dem neuesten Stand bleibt.

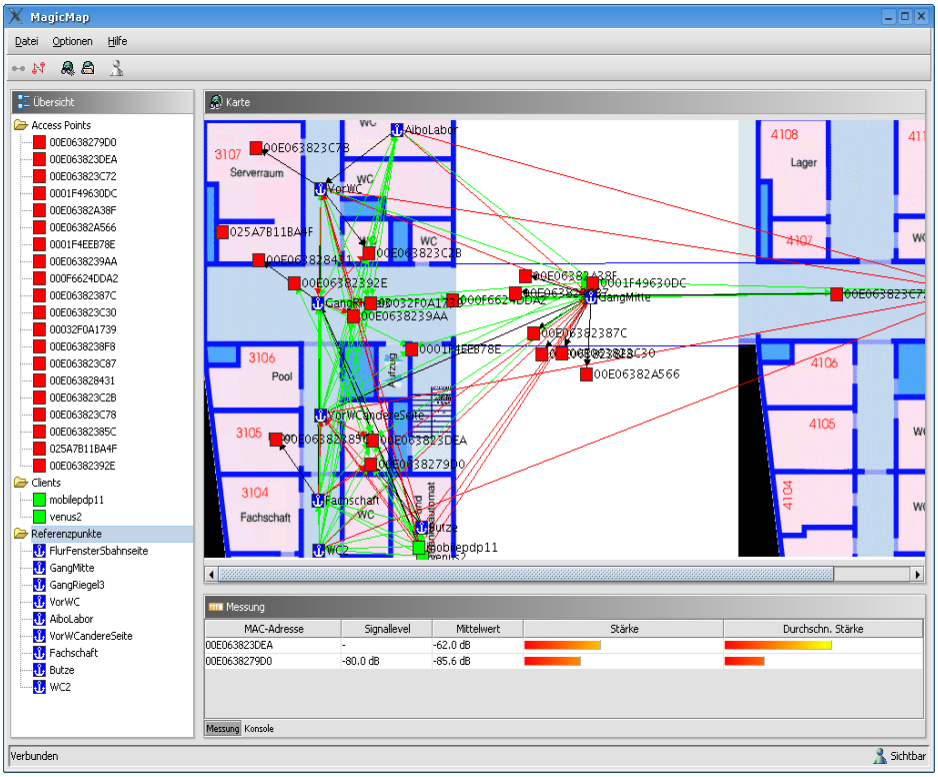


Abbildung 3.11.: *Umgebung auf einer MagicMap*

Da eine Karte einjustiert ist, kann jede sich ergebende Position in eine Geo-Koordinate umgerechnet werden. Durch die Umrechnung auf eine Geo-Koordinate ist es möglich, MagicMap auf Anwendungen laufen zu lassen, die sonst eine GPS-Satellitenortung voraussetzen.

MagicMap besteht aus sechs verschiedenen Modulen:

[7] Dieser Effekt entsteht allerdings bei allen gleichartigen Ortungsverfahren

- GUI

  Sie zeigt die Position der Knoten auf der Karte.

- Stumbler

  Für verschieden Sensoren gibt es verschiedene Stumbler, beispielsweise für W-LAN, für Bluetooth oder RFID. Sie messen die übermittelten Daten, beispielsweise die Signalstärke.

- Server

  Er verteilt die Daten zwischen den Knoten, wie die Signalstärke oder die Positionsberechnung.

- P2P-Kommunikation

  Dies ermöglicht den infrastrukturlosen Austausch von Positionsdaten.

- Positioning Engine

  Sie berechnet die Position, wobei die Berechnung auch verteilt durchgeführt werden kann, also ad-hoc.

- Tracker

  Dieser beobachtet die Positionsdaten auf voreingestellte Werte und triggert entsprechend der Voreinstellung.

Durch die Eigenschaft, dass die Positionsbestimmung nur am aktuellen Peer vorgenommen wird, bleibt auch der Privacy-Grundsatz[8] erhalten. Der Schutz der personenbezogenen Daten erfolgt mittels zweier Modi, *sichtbar* und *unsichtbar*. Im unsichtbaren Modus leitet ein Peer keinerlei Daten weiter, solange kein sonstiger W-LAN-Traffic auftritt. Ebenso erhalten andere Knoten keine Signale, solange in diesem Modus ist auch keine Ortung möglich. Im *sichtbaren* Modus ist der Peer nach persönlichen und situationsabhängigen sicherheitsrelevanten Einstellungen justierbar.

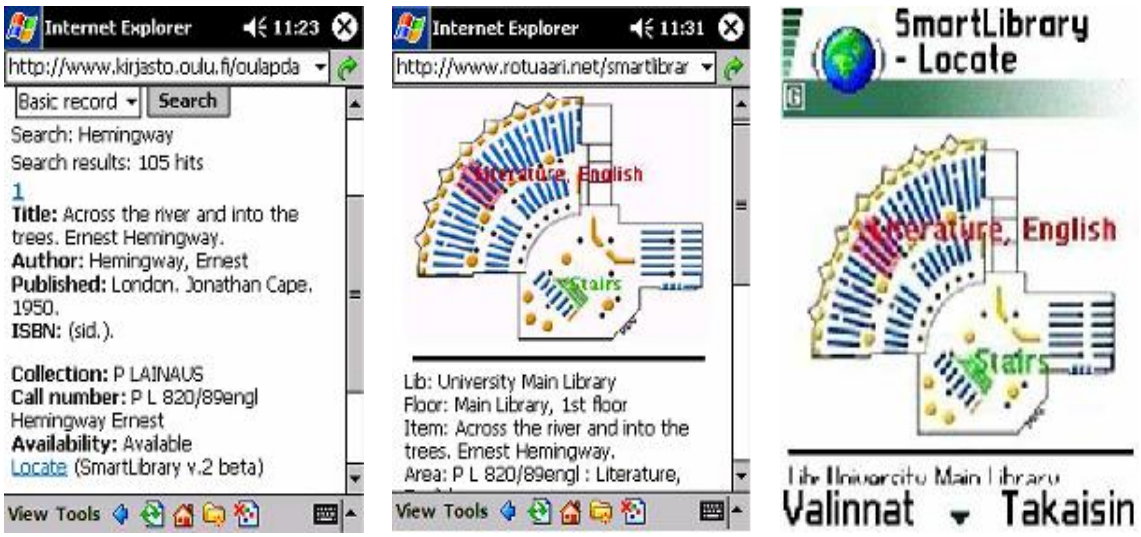
SmartLibrary

Das Projekt SmartLibrary [(3)] wurde an der Universität Oulu in Finnland entwickelt und ist dort auf der Hauptebene der Bibliothek installiert. Das Programm selbst ist ein Service, der auf zwei bereits bestehende Systeme der Universität aufgesetzt wurde. Diese sind das Datenbank Programm *Online Public Access Catalogue* OPAC, auf das mittels eines PDA (auf dem die Software OULA-pda installiert sein muss) zugegriffen werden kann. Zum Anderen ist

[8] persönlichen Daten sind zu schützten

das die Software zur Unterstützung eines kontextbezogenen Multimedia-Dienstes, der den Namen SmartWare trägt. SmartLibrary ist eine Weiterentwicklung von SmartWare, von dem bestimmte Programmteile weiterverwendet wurden. Mittels der Software OULA-pda kann in der Datenbank ein Buch gesucht werden. Ist das Buch gefunden und der Benutzer klickt es an, so wird er mit Hilfe von SmartLibrary und einem PDA zu diesem Buch geführt.

Während der Anwendung läuft SmartLibrary in einem XHTML-Browser des PDAs ab. Die Zielführung basiert auf der Grundlage statischer Karten. Zur Wegefindung werden Landmarken verwendet, die in grüner Farbe dargestellt werden. Das Zielareal ist rot. Die Benutzer-Position wird in Form eines „smileys“ angezeigt. Eine Rundreise durch die Bücherei ist dennoch nicht möglich, da das System nur ein einziges Objekt ansteuern kann.



(a) (b) (c)

Abbildung 3.12.: *a) Buchsuche mit OULA-pda, b) Routen-Ansicht auf einem PDA, c) Routen-Ansicht auf einem Handy*

Zur Positionsbestimmung läuft hintergründig die *Ekahau Positioning Engine*, die zusätzlich in das W-LAN Netz der Universität integriert werden muss. Das W-LAN Netz besteht aus 6 Access-Points, die für eine ausreichende Signal-Abdeckung der Bücherei sorgen. Aufgrund dessen, dass sich im Laufe der Zeit die Landmarken ändern und es weit über 100-Regale zu verwalten gilt, wurde eigens für diesen Sachverhalt ein Programm entwickelt. Das webbasierte Administrationsprogramm mit dem Namen Content Provider Interface (CPI), befähigt die Bibliotheksangestellten, die zur Navigation benötigen Hintergrundbilder auszutauschen oder auch den Grundriss der Etagenpläne zu ändern. Zur Änderung von Landmarken und Regalen werden diese mit einem Zeichentool als Polygone gezeichnet und verändert über die Grundrisskarte gelegt.

3.4. Informationsdarstellung auf mobilen Geräten

Das Design von Karten ist ein komplexe Aufgabe, die kognitive und psychologische Aspekte beinhaltet [(5)]. Trotz dieser Schwierigkeit ist die Informationspräsentation Hauptbestandteil eines Indoor-Navigationssystem. Da in solchen Systemen der mobile Charakter im Vordergrund steht und sich daher keine großen Displays eignen, werden vergleichsweise kleine Geräte zur Darstellung genutzt. Auf diesen ist es umso wichtiger, nur die nötigsten Informationen auf der Anzeige darzustellen.

Neben der übersichtlichen Gestaltung der Anzeige muss auch die Darstellungsqualität Beachtung finden. Die Darstellungsqualität ist aufgrund verschiedener Faktoren vergleichsweise schlechter. Ein Faktor ist die Programmiersprache Java, die mit J2ME nicht den vollen J2SE-Sprachumfang besitzt und damit zunächst keine 2D oder 3D Unterstützung bietet. Eine Unterstützung kann nur unter der Verwendung zusätzlicher Java-APIs erlangt werden.

Weiterhin ist bei mobilen Geräten die Darstellungsqualität auch infolge der Farbtiefe gehemmt. Der Trend liegt derzeit bei 24-bit. Dennoch sollten diese 24-bit bei der Planung von Anwendungen mit grafischen Elementen nicht angenommen werden. Viel eher sollten 16-bit in die Planung miteinfließen, da diese häufiger unterstützt werden. Die Farbtiefe ist ein Qualitätsmerkmal, soll aber zunächst nicht weiter betrachtet werden.

Von Interesse ist die Art der Darstellung, die Perspektive und die Dimension. Daher soll zunächst geklärt werden, welche visuelle Art der Informationsdarstellung sich am ehesten eignet. Zu diesen Darstellungsarten zählen:

- Die textuelle Darstellung

  Nicht alle mobilen Geräte der CLDC sind mit einem qualitativ hochwertigen Display ausgestattet, beispielsweise diejenigen nicht, die monochrome Anzeigen besitzen. Auf solchen Geräten können keine qualitativ hochwertigen Karten angezeigt werden, da Karten eine vergleichsweise Farbtiefe mehreren bit voraussetzen. Infolge der ungenügenden Anzeigequalität eignen sich diese mobilen Geräte bei der Navigation nur zu textuellen Beschreibungen.

  (57) stellt solch ein textbasiertes Navigationssystem für die Fußgängernavigation vor. Seine Informationsdarstellung beschränkt sich auf:

  - den aktuelle Straßennamen
  - den Straßennamen nach der Richtungsänderung
  - die Distanz zur nächsten Richtungsänderung
  - die Richtungspfeil mit darunterliegender Straßengeometrie

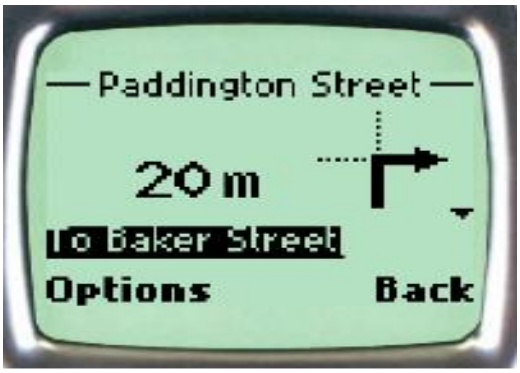


Abbildung 3.13.: *Tarkiainen - Textuelle Navigation*

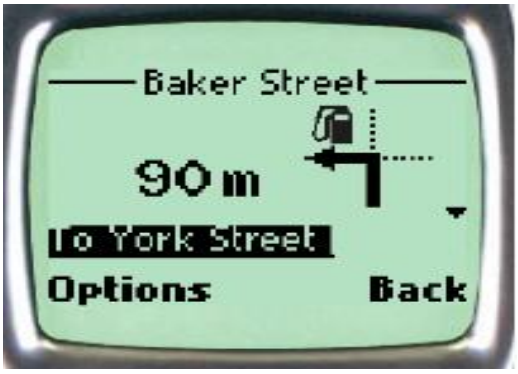


Abbildung 3.14.: *Tarkiainen - Textuelle Navigation mit Landmarke*

  - die Anzahl der kreuzenden Straßen bis zur Richtungsänderung
  - die Entfernung zum Ziel

  (Die beiden letzten Punkte der Aufzählung sind durch Scrollen der Anzeige sichtbar zu machen)

  Zusätzlich zu den textuellen Beschreibungen und Wegweisungen wurden Landmarken in Form von Piktogrammen zur Orientierung verwendet. Problematisch bei dieser Navigationsmethode war das Finden der Startrichtung. Dieses konnte nicht miteinbezogen werden, da das mobile Gerät keinen eingebauten Kompass besitzt, der die Blickrichtung hätte ausweisen können. Auch die Positionsmethode über das zelluläre Netzwerk des Telekommunikationsbetreibers konnte diesen Parameter nicht bereitstellen.

- die 2D-Darstellung

  Der Nachteil der rein textuellen Beschreibung der Navigation besteht darin, dass der Benutzer keinerlei Anhaltspunkte zum tatsächlichen Standort und umliegenden Orientierungspunkten hat. So muss sich der Benutzer beispielsweise stets ins Gedächtnis rufen, wieviele Straßen er bereits überquert hat.

  Die 2D-Darstellung ist im Gegensatz zur reinen textuellen Beschreibung wesentlich komfortabler. Mit dieser Darstellung muss sich ein Benutzer nicht erst ein Bild von der Umgebung machen, sondern erhält eine Aufsicht auf die Umgebung, in der er sich gerade befindet. Aufgrund der übersichtlichen Sichtweise kann der Benutzer stets in Erfahrung bringen, woher er kam, welche Objekte er bisher gesehen hat und wo er hin möchte.

Die 2D-Präsentation ist wohl auch die populärste Darstellungsart (siehe Landkarten und Skizzen). Sie ergibt sich automatisch durch die Verwendung einer zweidimensionalen Fotografie (Satellitenbild) oder einer Zeichnung. Diese Darstellung ist die älteste verwendete ihrer Art und geht zurück bis zu den ersten in den Sand gezeichneten, in den Fels gehauenen Zeichen.

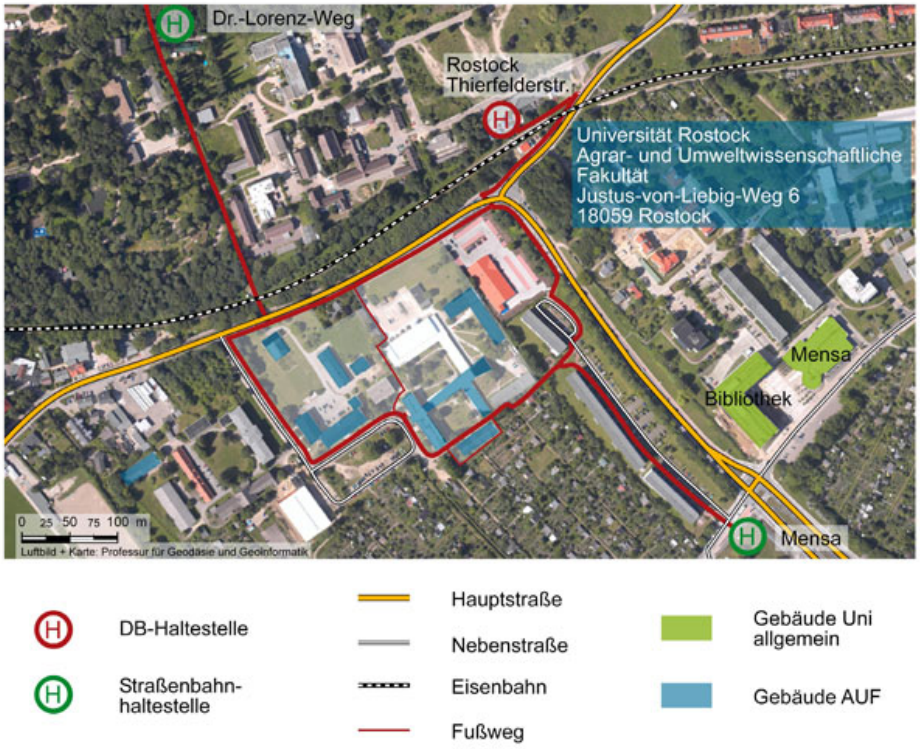


Abbildung 3.15.: *Ein 2D-Bild und darübergelegte Vektorgrafiken*

Routen wurden noch zu Beginn des vergangenen Jahrhunderts auf 2D-Karten aufgezeichnet. Diese Methode ist vergleichbar mit heutigen Methoden, jedoch mit dem Unterschied, dass das mobile Gerät dies erledigt.
Karten oder auch Kartenteile für Outdoor-Navigationssysteme bestehen meist aus Bildern. Diese haben entweder das JPG- oder das PNG-Format. Auch auf digitalisierten 2D Karten sollen Landmarken nicht fehlen. Sie fördern weiterhin die Navigation und die Orientierung des Benutzers. Sie werden meist in Form von 2D-Grafiken über die Karte gezeichnet und, je nach Relevanz oder Zoomstufe, in verschiedener Größe, Form oder Farbe dargestellt.

Neben Landmarken und Routengeometrien werden häufig auch POIs über Karten gezeichnet. Für POIs bietet sich vor allem *scalable vector graphics* [(67)] (SVG) an. SVG dient zur Beschreibung von zweidimensionalen Vectorgraphiken. Es basiert auf XML

und ist eine Empfehlung des World Wide Web Consortiums (W3C) [(63)]. [..] Eine Vektorgrafik wird durch analytisch vorliegende 2D-Primitive wie z. B. Punkte, Linien und Polygone, sowie durch mit ihnen assoziierte grafische Attribute wie Linienstärke oder Strichfarbe spezifiziert[..] [(D)].

Vorteile von SVG gegenüber anderen Grafikformaten lassen sich folgendermaßen darstellen:

- SVG-Dateien können auch mit einfachen Texteditoren bearbeitet werden
- SVG-Dateien sind kleiner und besser komprimierbar als beispielsweise JPEG oder PNG Bilder
- SVG sind skalierbar und zoombar, ohne Qualitätsverluste
- SVG kann nach Text durchsucht werden (Beschreibung eines Objektes)
- SVG arbeitet mit Java-Technologien
- SVG ist ein offener Standard und daher weit verbreitet
- SVG ist ein XML-Dialekt

SVG wurde nicht explizit für den Bereich der Indoor-Navigation entwickelt. Daher können mögliche Ansprüche so Beispielsweise Sach- und Geometriedaten, die bei der Navigation im Zusammenhang mit Landmarken genutzt werden könnten, nicht erfüllt werden. (7) stellten eine Lösung zu diesem Problem vor, sie nennt sich SVG-geo und erweitert den SVG-Sprachumfang. Die Lösung bietet eine bessere Unterstützung für Geodaten aber auch für die Visualisierung auf mobilen Geräten [(58)].

SVG wurde - wie z.B. HTML - von W3C entwickelt und hat die nahtlose Verbindung zwischen Text und Grafik zum Ziel.

- die 3D-Darstellung

  Die 3D-Darstellung optimiert die Wegebeschreibung in großen und unübersichtlichen Gebäuden [(12)]. Durch die realitätsnahe Abbildung können Umgebungen und Landmarken vom Benutzer besser erkannt und identifiziert werden. Ein formaler Aufbau einer 3D-Landschaft ist in Abbildung 3.4 zu sehen [(D)].

  3D-Szenen, wie sie in einem 3D-Gebäude vorzufinden sind, können mittels der *virtual reality modeling language* (VRML) realisiert werden. VRML ist ein ISO-Standart und besteht aus zwei Bänden. Band 1 definiert die Basisfunktionen und die Textcodierungen für VRML [(10)], Band 2 beinhaltet weitere Funktionalitäten und die Einbindung des VRML *External Authoring Interface* (EAI). Diese EAI ermöglicht es, von einem Java-Applet direkt auf eine VRML-Welt zuzugreifen.

Zur technischen Umsetzung virtueller Umgebungen werden Visualisierungs- und 3D-Graphiksysteme eingesetzt. 3D-Graphiksysteme wie z. B. OpenInventor [(53)] oder Java-3D [(51)] stellen im Allgemeinen Klassenbibliotheken bereit, mit der computergrafische Anwendungen objektorientiert programmiert werden.

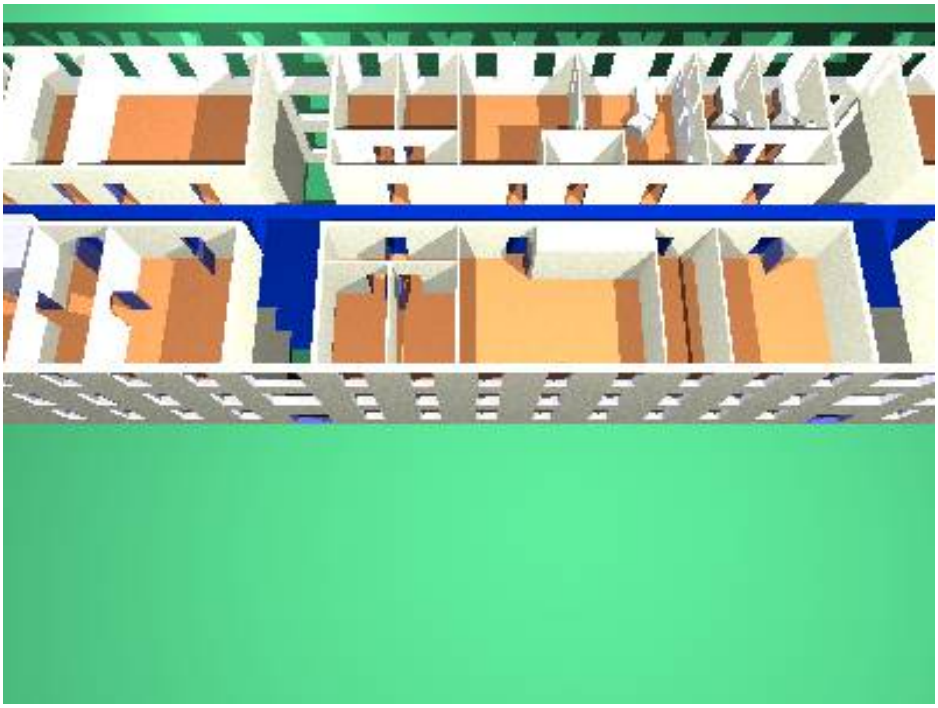


Abbildung 3.16.: *3D VRML-Bild einer Etage*

Im mobilen Bereich müssen bei der Übertragung mit VRML große Datenmengen übertragen werden. Es gibt jedoch auch einen Ansatz zur Reduzierung der Datenmenge. (13) entschied sich beispielsweise nur die Daten der unmittelbaren Nähe des Betrachters scharf darzustellen. Weiter entfernte Objekte werden entweder unschärfer oder gar nicht dargestellt. Er weist aber ausdrücklich darauf hin, dass weiterhin Objekte wie Start, Ziel und Landmarken sichtbar sein müssen. Neben der reduzierten Darstellung entwickelte er ein Komprimierungsverfahren, dass beispielsweise eine 3MB große VRML-Datei auf die Größe von 180kB verkleinern kann. Mit Hilfe dieser Komprimierung ist auch eine 3D-Darstellung auf mobilen Geräten möglich.

3.5. Design von Karten und Anzeigen für mobile Geräte

Im Rahmen dieser Arbeit wird auch die Gestaltung der Benutzeroberfläche vorgenommen. Daher ist es nötig, Konzepte zu deren Design, zur Vorgehensweise der Erschaffung und de-

ren Implementierung zu studieren. Im folgenden sollen daher zwei Artikel vorgestellt werden, die dieses Thema beleuchten. Der erste Artikel befasst sich mit dem Design und der Implementierung. Der darauf folgende mit der Vorgehensweise der Erschaffung von Karten zur Indoor-Navigation.

A Design and Implementation for Effectiv Computer-Generated Route Maps

Die Autoren (2) stellten auf der AAAI Symposium on Smart Graphics, March 2000, das Design und die Implementierung zu effektiv Computer-generierten Routen-Karten vor. Sie stellten fest, dass bei einer textuellen Beschreibung das Routeing zwar funktioniert, es aber für den Benutzer oft frustrierend ist, die dazugehörigen Karten zu lesen. Dies soll, so die Autoren, daran liegen, dass solche Routenkarten die Techniken und Prinzipien der Kartenhersteller unbeachtet lassen. Daher stellten sie folgende Designziele in den Vordergrund:

- Lesbarkeit
- Klarheit
- Vollständigkeit
- Angemessenheit

Weiterhin strebten die Autoren an, diese Designziele in einer Routenkarte zu vereinen, obwohl dies, so beschrieben es die Autoren, eine sehr schwierige Aufgabe darstellt.

Eine Routenbegehung erfolgt durch folgen einer Strecke $\overline{AB}$ bis zu einem kritischen Punkt [(2)], um dort die Orientierung zu ändern und einer anderen Strecke, $\overline{BC}$, zu folgen. Die wichtigste Information bei der Routengehung bleibt aber die Rückorientierung an besagtem kritischen Punkt. Die Erkenntnis, die sich aus dem Design der Autoren ergab, ist, dass die Länge, der Winkel und die Krümmung einer Route nicht zu berücksichtigen ist. Die Autoren verweisen dies bezüglich auf die Literatur von (60). Zur Erstellung einer effektiven Routenkarte, sollte zunächst ein geeignetes Kartenlayout gefunden werden. Karten sollten zudem bestimmt Eigenschaften verinnerlichen, d. h. die Designziele berücksichtigen und mittels dreier Faktoren die Karte gestalten. Diese Faktoren sind:

- Inhalt

  Der Inhalt einer Routenkarte bestimmt sich demnach aus dem Start, dem Ziel und den Rückorientierungspunkten. Weiterhin sind zwei Arten von kontextbezogenen Informationen zu berücksichtigen:

  - Local Context, eine Beschreibung in Form eines Namens oder einer Beschriftung. Hierzu gehören aber auch Landmarken, im Indoor-Bereich beispielsweise Türen, Schilder oder fest installierte Gegenstände.
  - Overview Context, der die Umgebung genau beschreibt, beispielsweise der Gebäudeplan mit den Räumen und Fluren, Aufzügen und Treppenhäusern.

- Präzision

  In Routenplänen gibt es drei verschiedene Arten von Verzerrungen:

  - Falsche Pfadlängen
  - Falsche Richtungsänderungen
  - Vereinfachte Straßenführungen

  Solche Verzerrungen können beim Routenerstellen aber hilfreich sein, da sie die Flexibilität eines Kartenerstellers beim Design und Layout der Karte erhöhen. Gleichfalls bringt es auch keine merkliche Verbesserung genauer darzustellen. Verzerrungen können somit sogar die Lesbarkeit und Einfachheit verbessern [(2)].

- Darstellungsform

  Verschiedene Darstellungsformen von Karten können die Lesbarkeit und die Klarheit beeinflussen. Augenfällige Eigenschaften können dazu genutzt werden, um die Aufmerksamkeit auf Objekt zu lenken. Hierzu gehört die farbliche Eigenschaft von Objekten sowie die Dicke eines Objektes.

A Hybrid Indoor Navigation System

Dieser Artikel wurde an der Universität des Saarlandes in Saarbrücken von (9) erstellt. Die Autoren erarbeiteten ein umfassendes Konzept zur Verwirklichung eines 3D-Indoor-Navigationssystems. Dieses Konzept wird durch zahlreiche weitere Artikel der Autoren, die sich mit der Indoor-Navigation beschäftigen, gestützt.

Diese Artikel-Sammlung beschreibt, welche Eigenschaften ein Indoor-Navigationssystem haben sollte und welche Vorüberlegungen zu erbringen sind, um ein solches System zu realisieren. Weiterhin wird insbesondere auf die Methodik der Routenbeschreibung eingegangen und erläutert, wie dem Benutzer anhand von Raumdaten in einem Gebäuden eine Blickrichtung gegeben werden kann. Kennt ein Benutzer seine Blickrichtung, so erleichtert diese die Orientierung erheblich.

Auch das Erscheinungsbild eines Indoor-Navigationssystems auf der Anzeige eines mobilen Gerätes wurde besprochen und in diesem Zusammenhang auf die Schwachstellen hingewiesen, die bei Geräten der Konfiguration CDC/CLDC bestehen. So beispielsweise die eingeschränkte Displaygröße oder deren Auflösung, durch die Platz zum Darstellen von Landmarken verloren geht. Gewarnt wird auch vor zu vielen, großformatigen und sich überlappenden Raumbezeichnern, die zu unleserlichen Beschreibungen von Landmarken führen können.

Weiterhin erläutern die Autoren, mit welchen Mitteln sie den Vorder- und Hintergrund ihrer Applikation erschließen. Im Vordergrund der Display-Ebene wird die Route angezeigt, die über die Wegplanungskomponente RANA erstellt wurde. Im Hintergrund ist dann der Gebäudegrundriss in Java3D zu sehen. Beides zusammen, die 2D Wegbeschreibung im Vordergrund und der 3D-Gebäudegrundriss, werden schließlich in das Vektorgrafikformat Xfig[9] exportiert.

Von Wichtigkeit ist den Autoren auch das Vorhandensein von Landmarken. Sie dienen dem Benutzer eines Navigations-Services als Entscheidungs- und Navigationspunkte. Darüber hinaus beschreiben Landmarken den Fortschritt einer Route und machen die Route dadurch einprägsamer. Dies kann beispielsweise durch Verwendung verschiedener Typen von Landmarken erreicht werden. Eine Landmarke kann als symbolisch dargestellte Abzweigungen, die zur Entscheidungsfindung des einzuschlagenden Weges dienen, angezeigt werden. Neben diesen sollte es auch Landmarken geben, die sich am Rande einer Route befinden und mit ihrer Anwesenheit der Orientierung dienen, sowie weitere welche die Position des Benutzers in Gebäuden aufzeigen. Eine ebenso nützliche Landmarke ist jene, die Rückschlüsse auf sich in der Umgebung befindlichen Landmarken liefert. Letztendlich aber ist neben diesen verschiedenen Landmarken diejenige am wichtigsten, die auf den genauen Aufenthaltsort einer Person hinweist, da diese das bedeutenste Entscheidungskriterium zur Reorientierung darstellt [(9)].

Im weiteren Verlauf des Artikels werden Einzelheiten zur grafischen Gestaltung einer Route besprochen und daraus auf den Aufbau einer Route geschlossen. Eine Route setzt sich aus Segmenten einzelner Teilabschnitte zusammen. Ein solches Segment teilt sich in vier Kategorien auf: Startpunkt, Rückorientierung, Pfad/Pfad-Fortschritt oder Endpunkt.
Die Knotenpunkte zwischen den Segmenten beschreiben entweder Abzweigungen oder Kreuzungen der realen Welt. Kanten sind Verbindungen zwischen zwei Punkten (Startpunkt zu Endpunkt/Startpunkt).

Auch die Dimension 2D oder 3D einer digitalen Karte wird von den Autoren aufgegriffen. Sie kommen zu dem Schluss, dass die Vogelperspektive, auch top-down-Perspektive genannt, die geeignetste ist.

[9] Xfig ist ein rein textbasiertes Format

Die grafische Präsentation des Navigationssystems der Autoren ist programmiertechnisch in einer Baumstruktur verwirklicht worden. Diese Struktur ist von den Autoren gewählt worden, um parallele, inkrementelle, sowie sequenzielle Abläufe gleichzeitig darzustellen und doch voneinander getrennt ablaufen zu lassen. Innerhalb der Baumhierarchie existieren *alternative*, *konditionale* oder *additionale* Wechselbeziehungen.

- Alternative Baumverzweigungen

  Sie zeigen an, dass die Entscheidung zur Präsentationszeit vorgenommen wird.

- Konditionale Verzweigung

  Dieser Verzweigungstyp stellt eine kontextbezogene Alternative dar, die zur Planungszeit aus einer Liste ausgewählt werden kann.

- Additionale Verzweigung

  Sie zeigen, dass zur Ausführungszeit einer laufenden grafischen Darstellung ein weiteres Element angezeigt werden kann, falls dies vom Benutzer erwünscht wird.

Die letzte Verzweigungsart beschreibt die additionale Wechselbeziehung. Sie kann beispielsweise während des Zoomens auftreten. Dies ist dann der Fall, wenn in einer Zoomstufe ein POI angezeigt wird, dass in der vorherigen noch nicht angezeigt wurde. Durch diese Form der Unterteilungen ergibt sich eine geringere zeitliche Verzögerung zwischen Prozessbeginn und Erscheinung der Grafik. Um einer solchen Verzögerung entgegenzuwirken, werden die grafischen Elemente, die weniger Ressourcen verbrauchen, zuerst gezeichnet. Während des Zeichenvorgangs werden die Elemente mit einem höheren Ressourcenverbrauch berechnet und später angezeigt.

Der wesentliche Grund zur Erfüllung eines solchen Konzeptes zur grafischen Realisierung zu entwickeln, waren die knappen Ressourcen, die Geräte der CD und CLD aufweisen.

Um ein Indoor-Navigationssystem zu entwerfen, sind nach Angabe der Autoren, folgende drei Schritte anzuwenden:

- Erster Schritt

  Die wenigen schriftlichen Erläuterungen und die skalierte Route sind auf dem Display erkennbar und deutlich darzustellen.

- Zweiter Schritt

  Im Hinblick auf verschiedene Routen sollte überlegt werden, wie eine Route durch ein Gebäude zu führen ist, um diese auf einfache Weise anzuzeigen.

- Dritter Schritt

  Der letzte Schritt betrifft die Entscheidung, ob die Beschreibung der Route textuell, grafisch oder durch beides geschehen sollte. In diesem Zusammenhang wird auch auf die unzureichende textuelle Beschreibung zur Orientierung Bezug genommen. Bestens geeignet sind Straßenkarten und ähnliche Grafiken mit liniengeführter Wegbeschreibung. Wichtig bleiben weiterhin Landmarken, die der Orientierung dienen. Rein verbale Wegbeschreibungen unterscheiden sich von der Beschreibung einer Karte. Offen bei einer Beschreibung bleibt jedoch immer, ob die Vorstellungskraft und die Aufmerksamkeit des Benutzers ausreichten, um auf diese Weise ans Ziel zu gelangen oder ob der Benutzer überfordert wird.

3.6. Zusammenfassung

An dieser Stelle sollen die schon herausgearbeiteten positiven Eigenschaften der vorgestellten verwandten Arbeiten zusammengefasst und dargestellt werden.

- Innenraum Routenplaner

  Der Innenraum Routenplaner hat Anstoß dazu geliefert, die Aufmerksamkeitsrichtung zu bedenken. Sie soll nach Möglichkeit bestimmt werden, ist das nicht möglich, so sollte der Benutzer in der Lage sein, das „Vorne“ im realen auch auf der Karten der Anzeige als „Vorne“ interpretieren können. Die Gebäudedatenbasis setzt auf XML-Dateien auf, die eine vergleichsweise einfache Basis repräsentiert. Von den Autoren kommt ebenfalls der Impuls, neben der verbildlichten Darstellung der Route eine textuelle Beschreibung der Route anzuzeigen. Der Innenraum Routenplaner basiert auf IMAPS und zeigt somit, dass es möglich ist, IMAPS als Grundlage der Positionsbestimmung zu verwenden.

- Outdoor-Navigationssoftware

  Die Anwendungen der Navigationssysteme in der Automobilindustrie boten eine sehr gute Grundlage zur Findung von Elementen, die auch der Navigation im Indoor-Bereich zu Gute kommen soll. In der abschließenden Analyse des Kapitels 3.2 wurden diese Elemente festgehalten.

- MOVING

  Das Projekt MOVING beschäftigt sich mit der Routenbestimmung anhand von zweidimensionalen Karten eines Gebäudes. Daher ähnelt es in diesem Punkt der vorliegenden Arbeit. Allerdings wird die Route nach ihrer Berechnung in Sprachform wiedergegeben. Dieser Punkt, sprich die akustische Navigation mittels der digitalisierten menschlichen Sprache bedeutete für die Erschaffer von MOVING die Findung von pregnanten und gut einprägsamen Objekten, den so genannten Landmarken. Diese werden im Kapitel 4 nochmals aufgegriffen, um sie im eigenen Projekt zu verwenden.

- CricketNav

  CricketNav arbeitet zur Bestimmung der Position auf der Grundlage von Cricket. Cricket diente (28) als Grundlage zur Entwicklung von IMAPS. Als Indiz für eine gute Eignung für IMAPS kann die erfolgreiche Positionsbestimmung innerhalb von Gebäuden herangezogen werden. (42) entwickelte CricketNav nicht auf der Grundlage von SVG (Kapitel 3.4), sondern auf der Grundlage von CAD-Gebäude-Daten. Dies gibt Anreiz dazu, eine ebenso einfache Technologie zu verwenden, die Gebäude-Karten zu erfassen und wiederzuverwenden. Der Aspekt der sicheren Wegbeschreibung wird durch die Abzweigungshinweise angesprochen. Aufgrund dessen wird auch für dieses Projekt mit besonderer Beachtung darauf verfahren.

- MagicMap

  Magic Map erläutert die Vor- und Nachteile der Nutzung einer W-LAN-Infrastruktur zur Indoor-Navigation. Von Vorteil hierbei ist, dass auf schon vorhandene Technik zurückgegriffen wird und daher wenig Zeit und Mühe in die Entwicklung einer Positionierungsmethode gesteckt wurde. Der Nachteil bei diesem Verfahren besteht sowohl in der Einrichtung einer Map, in der W-LAN Access-Points eingezeichnet sind, als auch in der großen Störanfälligkeit, die durch Rauschen (sich bewegende Objekte) oder durch Abschirmung in Folge einer Verdeckung entsteht, so dass keine Signalausbreitung stattfinden kann.

- Smart Library

  Smart Library ist ein umgesetztes und öffentlich genutztes Indoor-Navigationssystem. Es veranschaulicht wie diese Technik auf verschiedenen Sektoren ihren Nutzen darbieten kann. Weiterhin wird auch dargelegt, dass sich infolge der Technik ein Zeitgewinn bei der Suche erreichen lässt.

- Informationsdarstellung

  Das Kapitel 3.4 hat veranschaulicht, dass, wie auch schon (9) in Kapitel 3.5 festgestellt hatte, eine ausschließlich textbasierte Darstellung nich genügend Ausdrucksstark gegenüber der visuellen Darstellung ist. Dies mag möglicherweise damit zusammenhängen, dass der Mensch keinen visuellen Plan der Umgebung auf Basis eines Textes erstellen kann.

  Für die visuelle Darstellung ist die 2D-Darstellung unter Verwendung einer XML-Gebäudedatenbasis zunächst ausreichend. Auf dieser Basis lässt sich eine einfache und übersichtliche Darstellung verwirklichen.

- A Design and Implementation for Effectiv Computer-Generated Route Maps

  Die Arbeit von (2) weisen auf vier Designziele hin, die in dieser Arbeit verwirklicht werden sollten. Diese sind die Lesbarkeit, die Klarheit, die Vollständigkeit und die Angemessenheit. Weiterhin verweißt der Artikel auf drei Variablen hin, anhand derer das Design manipuliert werden kann, der Inhalt, die Präzision und die Darstellungsform.

- A hybird Indoor-Navigation-System

  Kapitel 3.5 zeigte an, auf welchen verschiedenen Gebieten Forschung betrieben werden muss um ein Projekt, das sich um Indoor-Navigation dreht, zu realisieren. Im Blickpunkt der Forschungen stehen die Landmarken aber auch die Berücksichtigung der Ressourcen von CD und CLD, speziell die Prozessorleistung. Die Empfehlung, die aus dem Artikel hervorgeht, betrifft den schonenden Umgang mit diesen Ressourcen. Aus diesem Grund empfehlen die Autoren ein Pattern zu Darstellung von Gebäude-Landschaften auf Geräten der Konfigurationen CD und CLD. Von Interesse ist ebenfalls das Konzept der Landmarken, die Aufteilung der Route in ihre Teile und die Planungen wann und in welcher Zoomstufe Landmarken anzuzeigen sind. Die vorliegende Arbeit wird ebenfalls der Empfehlung zur Einschränkung von textuellen Beschreibungen nachgehen und diese in geeigneter Weise realisieren.

Diese Arbeiten bieten durch die Erfahrungen aus der Forschung viele verschiedene Wege zur Umsetzung eines Indoor-Navigationssystems an. Ebenfalls beschreiben sie Vorgehensweisen, die unbedingt bei der Implementierung Berücksichtigung finden sollten, beispielsweise die Verwendung von Landmarken.

Dass eine digitale Revolution in der Kartographierung stattfindet, ist der Initiative namhafter Firmen wie Google oder MagicMap zu verdanken. Diese sorgen dafür, dass das Angebot für den Verbraucher gedeckt wird und weiterentwickelt wird.

Da nun die Technik, auf die das Projekt aufgebaut werden soll, hinreichend bekannt gemacht wurde, kann mit der Analyse der Funktionalität eine Indoor-Navigationssystems begonnen werden. Dies geschieht im Hinblick auf die gewählte IMAPS-Infrastruktur und dem aus der Motivation benannten PDA.

4. Analyse

Dieses Kapitel beschreibt das System aus der funktionalen Benutzersicht. Für das in Kapitel 1.1 erläuterte und motivierte Aufgabengebiet sollen zunächst die generellen Anforderung an das System abgegrenzt werden. Die funktionalen Anforderungen werden daraufhin im Kapitel 4.3 beschrieben. Dieses erläutert was die Anwendung leisten sollte, damit der Benutzer den Vorteil eines PDA als Navigationshilfe zu nutzen erkennt. Das Kapitel ist dazu in drei Abschnitte eingeteilt, um dadurch die schrittweise Herausarbeitung der funktionalen Anforderungen zu unterstützen. Im ersten Schritt wird der Arbeitsumfang des Systems beschrieben. Im zweiten Schritt erfolgt eine Abgrenzung des Systems. Nachdem die Abgrenzung abgeschlossen und der Arbeitsumfang festgelegt wurde, können die funktionalen Anforderungen konkretisiert werden. Verdeutlicht werden neben den Elementen der grafischen Interaktion (Pfeil, Route, Kartenteil usw.) auch Komponenten, die dem Komfort des Dienstes dienen (Suchfunktion, Zoomen usw.).

In Kapitel 4.3 werden die Anwendungsfälle (Use-Cases) des Systems aufgezeigt. Diese Use-Cases stellen außerdem die Möglichkeiten zur Interaktion mit dem System dar. Für die Fälle *Zielort wählen* und *Zoomen* werden Use-Case Diagramme gezeichnet, welche die Interaktionsmöglichkeiten grafisch veranschaulichen sollen. Der Use-Case *POI-Filtern* erfordert eine Klassifizierung sämtlicher POIs. Aus diesem Grund erfolgt nach der Beschreibung von *POI-Filtern* eine Auflistung und unscharfe Einstufung möglicher POIs.

Im Kapitel 4.4 werden die nicht-funktionalen Anforderungen der Anwendung erläutert. Diese Anforderungen berücksichtigen zusätzlich die Einschränkungen des Referenz-Modells, die durch die Konfiguration, das Profil und den technischen Möglichkeiten des PDA hervorgebracht wurden.

Als nächstes folgt nun ein Szenario das auf mögliche Geschäftsanwendungsfälle hinweisen soll.

4.1. Beispielszenario Frankfurter-Flughafen

Ein Szenario soll den Gebrauch eines PDA zur Indoor-Navigation verdeutlichen und dient der Abgrenzung des Aufgabengebietes. Es beschreibt, wie ein Benutzer das System kennenlernt und für die Indoor-Navigation nutzt.

Die Benutzer sollen während einer Zielsuche in den Hallen des Flughafen-FFM ausschließlich ihr PDA nutzen, um den Weg ins Ziel zu finden. Ziel der Anwendung soll es sein, das Routeing und die Zielsuche für den Benutzer zu organisieren, um so die Suche zu vereinfachen. Der Nutzen eines solchen Systems spiegelt sich letztendlich in zufriedenen Benutzern wider und könnte das Erfragen des Weges von einer anderen Person vollkommen ersetzen. Das Navigieren wäre somit auch mit einer erheblichen Zeitersparnis verbunden, die durch das Wegfallen von Erfragen, Suchen und Orientieren entstehen würde.

Abbildung 4.1.: *Einweisung nach der Installation*

Ankunft eines Gastes

Mit dem Betreten des Geländes des Frankfurter-Flughafen veranlasst ein Dienst die Installation der Navigations-Software auf dem PDA. Schon im Vorfeld, beim Ticketbuchen, wurde dem Benutzer mitgeteilt, dass er sein PDA zur Routenplanung auf dem Flughafen nutzen

könne. Dazu müsse er sich lediglich an der Information ein Empfangsmodul leihen und versichern, dieses beim Verlassen des Gebäudes wieder zurückzulassen. Der Besitzer eines PDAs kann sein eigenes Gerät benutzen. Sollte dies nicht möglich sein, da die Voraussetzungen fehlen oder wünscht der Benutzer nicht mit dem eigenen PDA zu navigieren, so könnte ein bereitgestelltes gleichwertiges Gerät zur Routenplanung genutzt werden. Ein solches wird ebenfalls an der Information bereitgestellt.

Der Benutzer möchte die Routenplanung mit dem eigenen Gerät durchführen. Sobald die Anwendung installiert wurde, erhält der Benutzer eine Einweisung in die Funktionalitäten des Routenplaners. Dies soll ihm helfen, die Suche zu verstehen, den Vorteil zu erkennen und sich auf das Gerät zu verlassen. Mit der Einweisung erhält der Benutzer auch Erklärungen, die Anwendung zur Orientierung und Informationsgewinnung zu nutzen. Darüber hinaus erfährt der Benutzer, wie in Fehlersituationen vorzugehen wäre. Dem Benutzer wird auch das Anlegen eines Profils offeriert, dabei sollte es ihm jedoch freigestellt werden, ob er dieses anlegen möchte oder nicht. Ermöglicht wird dies durch einen weiteren, unabhängigen Dienst.

Routenplanung

Nach der erfolgreichen Installation der Applikation auf dem PDA kann der Benutzer, solange er sich im Sendebereich des lokalen Netzwerkes des Flughafen-FFM befindet, Zielorte innerhalb beider Terminals wählen.

Zunächst wird das Terminal ausgewählt, in welchem Terminal sich der Zielort befindet. Anschließend erhält der Benutzer eine alphabetisch oder kategorisch geordnete Liste über die wählbaren Zielorte. Zu einzelnen Zielen können zusätzlich Informationen zu den Orten abgerufen werden. Ist der Benutzer sicher, dass der gewählte Zielort der richtige ist, so kann er dieses bestätigen.

Die Zielorte sind für den Benutzer gänzlich fremd und daher undurchsichtig. Daher kann der Benutzer in Folge seiner individuellen Situation zwischen zwei verschiedenen Listen-Typen wählen.

- Alphabetisch geordnete Liste

  Der Benutzer hat kein bestimmtes Ziel oder Bedürfnis und verschafft sich einen Überblick über die gesamten Zielorte. Zusätzlich möchte er Informationen über Orte erfahren oder diese möglicherweise als Zwischenstationen auswählen.

- Kategorisch geordnete Liste

Wenn ein Benutzer spontane Erkundungstouren zu bestimmten interessanten Orten innerhalb des Flughafens unternehmen möchte, benötigt er eine kategorische Auflistung der Zielorte. Somit wäre es ihm möglich, beispielsweise, zwischen Restaurants, Bars oder Einkaufsmöglichkeiten zu wählen.

Nach der Bestätigung des Zielortes erhält der Benutzer die Route vom aktuellen Standpunkt ins Zielgebiet. Er kann nun entscheiden, ob er der Route folgt oder einen anderen Zielort wählt. Der Benutzer erhält neben der Route auch eine Anzeige über die ausstehende Weglänge ins Ziel sowie die voraussichtlich benötigte Zeit dafür.

Abbildung 4.2.: *Zielortwahl anhand einer Liste*

Aufgrund eines dringenden Bedürfnisses entscheidet der Benutzer, den Zielort zu wechseln. Dazu wählt er mittels der Tastatur in einer Listen Kürzel „Toi“, woraufhin das PDA die nächst gelegene Toilette und die Route dorthin, vom aktuellen Standort aus, anzeigt.

Erkundung der Umgebung

Die Sichtweite, die sich für ein Benutzer im Flughafen ergibt, ist durch Wände, Türen und Stockwerke beschränkt. Solche örtlichen Gegebenheiten sind störend und engen den Benutzer ein. Daher sollte dem Benutzer zum Erkunden der Umgebung technische Hilfsmittel

an die Hand gegeben werden, die es erlauben, „hinter“ Wände und Türen und in die Stockwerke hinein zu schauen.

Beim Erkunden der Umgebung mittels des PDA verschiebt der Benutzer die Ansicht der Karte in jede Richtung, bis er am Rande der Kartenteile der Terminals angekommen ist; weiter geht es nicht. Er verschafft sich damit einen ersten Überblick über Terminal I und II. Nach dieser Erkundung drückt er einen Knopf, woraufhin der Standort an dem er sich befindet auf der Anzeige erscheint, an dem er sich befindet.

Auf der Karte der Anzeige findet der Benutzer ein Objekt, dass ihn interessiert. Daraufhin vergrößert er den Ausschnitt mittels Tastendruck. Zusätzlich zu den alten Objekten enthält die Karte weitere Objekte, die ihm zusätzliche Informationen liefern.
Vergrößern oder Verkleinern gibt dem Benutzer die Gelegenheit, die Umgebung, die auf der Anzeige des PDA zu sehen wäre, in unterschiedlicher Größe und Anzahl darzustellen. Verkleinerung hingegen bedeutet, dass weniger Objekte auf der Anzeige dargestellt werden und Räume eine kleinere Dimension erhalten. Diese Art der Übersicht bedeutet für den Benutzer, dass er das, was sich hinter den Wänden und Türen verbirgt, in verschiedenen Detailstufen einsehen kann. Diese Einsichten beschränken sich aber nur auf die von der Flughafen-AG bereitgestellten Daten.

Mit diesen zusätzlichen Informationen infolge der Vergrößerung und der Verkleinerung kann sich der Benutzer ein besseres, genaueres Bild über das Gebäude machen. Möglicherweise findet der Benutzer dadurch einen Ort, den er aufgrund des Zielortnamen als zunächst nicht interessant empfand. Größenveränderungen bieten in zweierlei Hinsichten Vorteile:

- Verringerung der Informationsflut auf einer beengten Anzeige, da der Ausschnitt vergrößert wird
- Umfangreichere Orientierung und Erkundung abseits der Route

Dem Benutzer fällt auf, dass Orte wie beispielsweise die Toiletten oder die Check-In-/ und Check-Out-Schalter und die Notausgänge stets auf der Anzeige zu finden sind. Dies ist von der Flughafen AG so gewollt, da dies einerseits den Komfort erhöht und andererseits die Sicherheit der Kunden berücksichtigt.

Orientierung

Der Benutzer findet die einfachste mögliche Perspektive zur Darstellung einer Etage vor. Die Perspektive beschränkt sich auf die Aufsicht (2D-Ansicht), sprich aus der *Vogelperspektive*, auf eine Etage. Während der Routenbegehung wird sich der Benutzer Orientierungspunkte suchen, sowohl auf der Anzeige des PDA als auch im Realen. Er wird Ähnlichkeiten zwischen

Realem und Virtuellem suchen. Daher sind die Objekte auf der Anzeige von ähnlicher Gestalt und haben zusätzlich kleine Bezeichnungen, damit sie auf der Anzeige wiedererkannt werden können.

Sollte der Benutzer ein Objekt durch Vergleichen der Form nicht identifizieren können, so sollte ihm die Möglichkeit bereitstehen, eines dieser Objekte zu markieren. Durch die Markierung sollte eine textliche Erläuterung angezeigt werden, der eine knappe Beschreibung des gewählten Objektes enthält. Sobald sich ein Benutzer für Objekte abseits der Route, für Objekte in Räumen oder hinter verschlossenen Türen interessiert, werden solche Anzeige-Objekte notwendig.

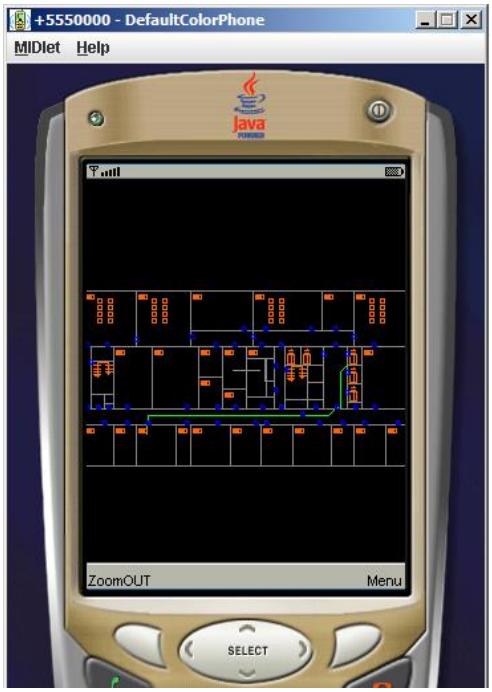


Abbildung 4.3.: *Aufsicht auf eine Etage*

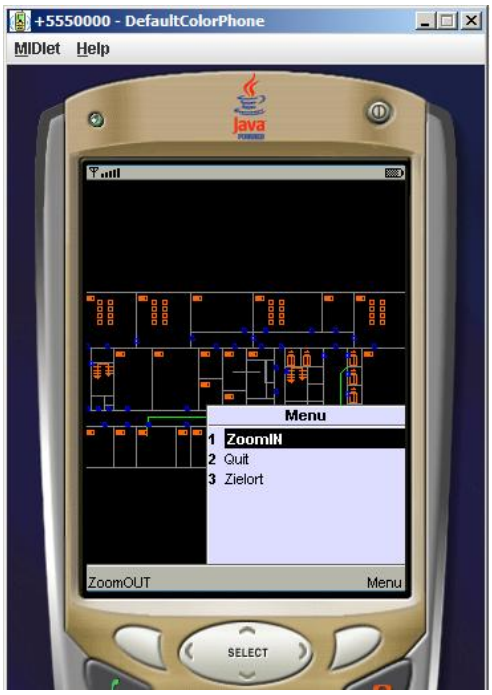


Abbildung 4.4.: *Das Auswahlmenü*

Damit der Benutzer auch jederzeit über den genauen Standort informiert wäre, könnte zum einen dieser angezeigt und zum andern der Aufenthaltsort textuell beschrieben werden. Neben dem Standort sollten dem Benutzer zusätzlich umliegenden Objekten, Landmarken, textuell und in 2D auf der Anzeige dargestellt werden. Zusammen mit einem Maßstab könnte der Benutzer somit auf die Entfernung vom Standort zu einem der umliegenden Objekte schließen. Zusätzlich zum Maßstab sollte ein virtueller Kompass die Ausrichtung des Gebäudes nach Norden anzeigen. Mit Hilfe dieser Einnordung sollte es dem Benutzer ermöglicht werden, die Orientierung gegenüber der Außenwelt aufrecht zu erhalten. Sind im Gebäude zusätzliche Schilder angebracht, auf denen die Himmelsrichtung steht (Ausgang West, Nord

usw.), so hätte der Benutzer anhand dieser Beschreibung eine grobe Orientierung. Die Richtung, die ein Benutzer beibehalten sollte, könnten über Komponenten, die auf der Anzeige dargestellt werden, mitgeteilt werden. Darüber hinaus sollte die Angabe der Entfernung zur nächsten Richtungsänderung die Routenverfolgung sichern. Sollte der Weg des Benutzers aus unbekannten Gründen von der Route abweichen, so sollte ihm dies auf dem Anzeige signalisiert und dargestellt werden.

4.2. Grundlegende Anforderungen an das System

Die zu entwickelnde Anwendung muss auf einem PDA lauffähig sein. Auf dessen Anzeige sollte es dann möglich sein, Gebäudekarten der Umgebung darzustellen. Mittels der Perspektive der Kartenteile und den Hilfsmitteln der Orientierung (Massstab, Landmarken usw.) hätte ein Benutzer die Möglichkeit mittels seines PDA durch ein Gebäude zu navigieren. Während des Gebrauchs der Anwendung sollte dem Benutzer zusätzlich die Möglichkeit geboten werden, verschiedene Interaktionen (Zoomen, Verschieben der Karte usw.) über das Bedienelement des PDA vorzunehmen. Für die Informationsbereitstellung, beispielsweise der Bereitstellung einer Zielort-Datei während einer Zielort-Suche, würde die Anwendung sorgen.

Zur Positionsbestimmung würde es sich empfehlen, eine vom PDA unabhängige Komponente zu verwenden. Anschließend sollte diese Position vom PDA ohne Verkabelung abfragbar sein. Desweiteren sollte zusammen mit dem Empfangsmodul auch ein unabhängiger Server-Dienst auf einem Karten-Server laufen.
Daraus ergibt sich der nötige technische Umfang des Systems, der im folgenden beschrieben werden soll. Zwei Arten von Komponenten sollten enthalten sein:

- *Karten-Server-Dienst*

  Standortkarte bereitstellen: Der Server muss ein angefordertes Kartenteil an die Anwendung des PDA übertragen. Er benötigt dazu den aktuellen Standort des Benutzers bzw. des PDA.

  Route berechnen: Eine Route muss vom Server berechnet werden können. Voraussetzung zur Berechnung sind Startpunkt und der Zielpunkt, die vom PDA übertragen werden müssen.

  Zielorte bereitstellen: Für eine Zielort-Suche auf dem PDA muss dieser vom Server die Zielort-Datei anfordern. Der Server muss diese daraufhin an den PDA übertragen.

Kartenelemente/Navigationselement bereitstellen: Neben den vorhanden Grundelementen einer Karte muss der Server auch Elemente bereitstellen, die der Navigation dienlich sind.

- *Infrastruktur-Komponenten*

  Empfangsmodul Ein Empfangsmodul sollte die Position eigenständig berechnen und bereitstellen können.

  Sender: Sie sollten ein Signal aussenden, dass angaben über die Position enthält.

Ohne diese Komponenten wäre die Anwendung nicht lauffähig und kann dem PDA somit auch nicht angeboten werden.

Mittels des Servers sollte der Dienst einer großen Anzahl von Benutzern bereitgestellt werden. Dabei sollte aber die ursprüngliche Funktionalität und Verwendung der Endgeräte in keiner Weise beschränkt werden. Um mit Hilfe des Entwurfes eine beständige Architektur zu erstellen, sollten grundlegende Anforderungen wie Modularität, Wiederverwendbarkeit, Wartbarkeit und Erweiterbarkeit qualitativ hochwertig erfüllt werden. Die zu entwickelnde Software soll außerdem ein hohes Maß an Portabilität besitzen.

4.3. Funktionale Anforderungen

Auf Erläuterung der grundlegenden Anforderungen an das System im letzten Kapitel erfolgt in diesem Kapitel die spezifischere Darstellung funktionaler Anforderungen.

Der mögliche Funktionsumfang der Anwendung ist im Kapitel 4.1 dargestellt. Anhand dessen wurde ein mögliches Arbeitsumfeld und die Einsetzbarkeit des Indoor-Navigationssystems besprochen. Der darauf folgende Schritt diente dem Herausarbeiten grundlegender Anforderungen an das System auf Basis des Szenarios. Dies geschah im Kapitel 4.2. Diese beiden Kapitel dienen nun zur Definition von System- und Geschäftsanwendungsfällen.

Systemanwendungsfälle

Für die Kommunikation zwischen den Infrastrukturkomponenten ergeben sich drei verschiedene Systemanwendungsfälle:

- Kommunikation zwischen Server und mobilem Gerät

  Kartenmaterial oder Zielorte müssen bereitgestellt werden; dies erfordert eine Verbindung zwischen Server und mobilem Gerät (PDA). Beim einem Request durch den PDA an den Server sollen nur Daten weitergegeben werden, welche die Anfragen an die Routen, die Karten, die Kartenelemente und die Zielorte betreffen. Beim einem Response durch den Server werden ausschließlich Kartenmaterial und die Routenbeschreibungen, sowie POIs, die sich in einer Etage befinden, übermittelt. Die Verbindung könnte bidirektional ausgelegt werde.

- Kommunikation zwischen Sensor und Empfangsmodul

  Zwischen Sender und Empfangsmodul sollen mittels Funk Daten übertragen werden. Neben diesem können auch noch weitere Signale anderer Wellenlänge Verwendung finden. Deren Kommunikation sollte unidirektional, von Sender zu Empfänger, ausgelegt werden.

- Kommunikation zwischen Empfangsmodul und dem mobilen Gerät

  Empfangsmodul und PDA kommunizieren per Funk. Die Kommunikation kann unidirektional ausgelegt werden.

Geschäftsanwendungsfälle

Dieses Kapitel beschreibt anhand des Szenarios Komponenten, die in der Anwendung vorhanden sein müssen, und stellt deren Beziehung zueinander her. Da im Szenario nicht alle Anwendungsfälle vorhanden waren werden diese ergänzt. Zur Beschreibung der Anwendungsfälle dient das Use-Case Diagramm aus Abbildung 4.5, in dem alle Geschäftsanwendungsfälle dargestellt sind.

Anmerkung: Die im Diagramm grau hinterlegten Ovale sind Komponenten des Server-Systems. Das Anzeigen dieser dient dem bildlichen Darstellen des Zugriffes durch die Komponenten des Client-Systems. Die Darstellung ist ein Vorgriff auf noch zu besprechende Komponenten und bezieht sich auf die besprochenen Abgrenzungen des Kapitels 4.2. Der Vorgriff dient sowohl der Sinngebung der eigentlichen Komponenten des Systems, als auch zur eindeutigen Veranschaulichung des Zusammenwirkens dieser.

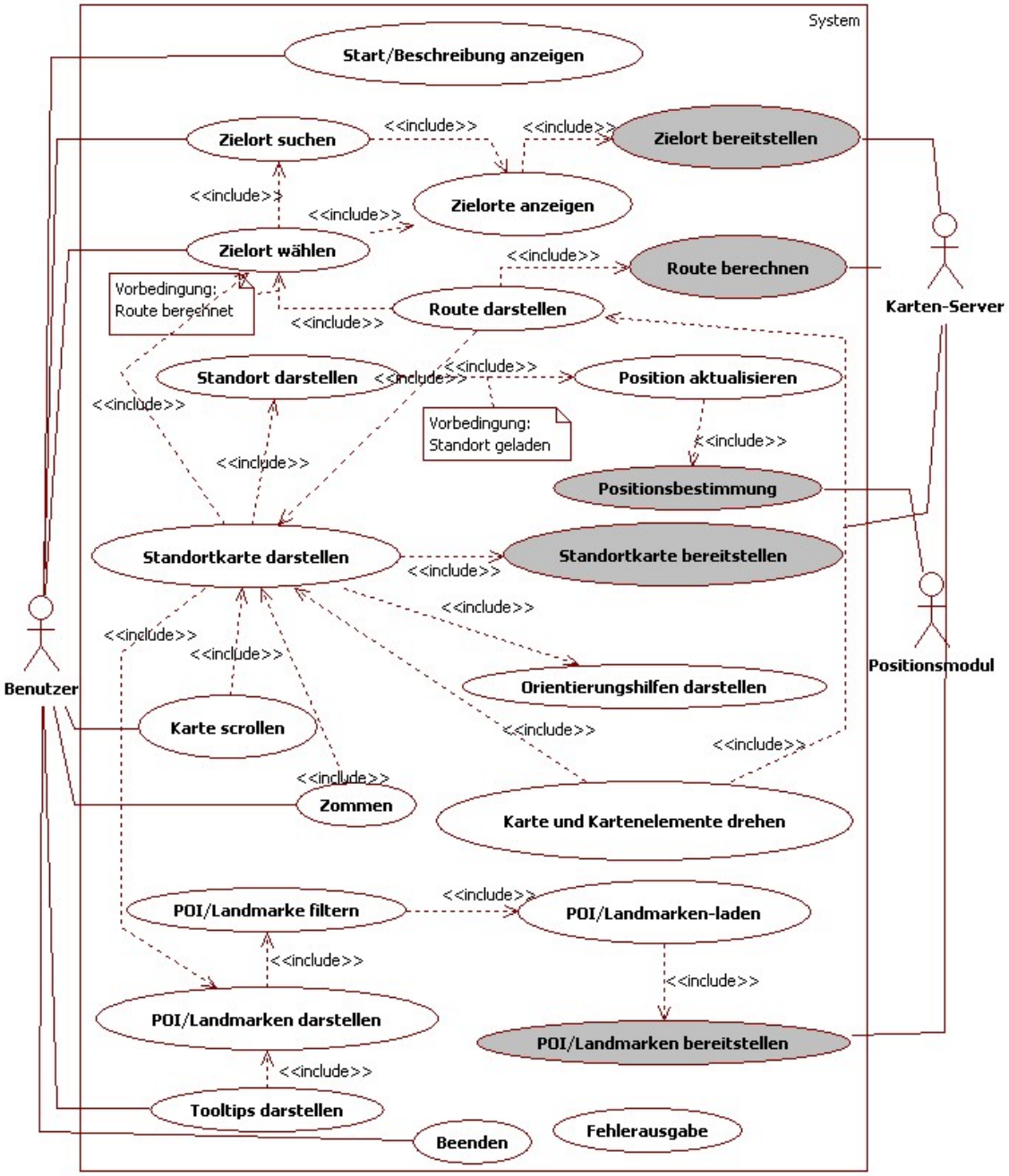


Abbildung 4.5.: *Anwendungsfälle aus Benutzersicht, (graue Use-Cases stellen Geschäftsanwendungsfälle des Karten-Servers oder des Empfangsmoduls dar).*

Zunächst werden die Aktoren des Diagramms 4.5 vorgestellt.

Aktoren aus Benutzersicht	
Benutzer	Jeder Benutzer der ein geeignetes mobiles Gerät, ein Empfangsmodul und die Indoor-Navigationssoftware installiert hat, kann die Anwendung nutzen. Der Nutzer kann eine Routenberechnung durchführen lassen und mittels dieser und einer angezeigten Karte eine Navigation durch ein Gebäude durchführen.
Empfangsmodul	Das Modul soll die Position errechnen und diese zur Abfrage bereitstellen.
Karten-Server	Der Server soll Zielort einer Etage bereitstellen, er soll eine Routenberechnung durchführen können und zu dem muss er eine Standortkarte mit Kartenelementen und Elementen, die der Navigation dienen, anbieten.

Im Folgenden werden ausgewählte Funktionalitäten des Systems systematisch, in einem Use-Case erfasst. Danach werden die Vorbedingungen beschrieben, die nötig sind, damit der Use-Case ausgeführt werden kann. Darauf wird der Use-Case Ablauf beschrieben. Da es während der Ausführung der Funktionalität eines Use-Case zu Ausnahmesituationen kommen kann, müssen individuelle Fehler erkannt und schriftlich fixiert werden. Dies erfolgt anhand der Formulierung von Ausnahmen. Abschließend werden Nachbedingungen benannt. Nachbedingungen sind Bedingungen, die nach Beendigung eines Use-Case gelten.

Nr.:**1** Start/ Beschreibung anzeigen	
Zusammenfassung	Der Dienst wird gestartet. Auf der Anzeige ist die Beschreibung des Dienstes zu lesen und die Funktionalität erklärt
Vorbedingungen	Das System war entweder noch nicht gestartet oder der Benutzer ist zur Beschreibung zurücknavigiert.
Beschreibung des Ablaufs	Sobald der Dienst gestartet ist, erscheint die Beschreibung. Erfolgt aus dem laufenden Betrieb eine Rückwärtsnavigation zur Beschreibung, so wird diese angezeigt.
Ausnahmen	E1: Sollte die Netzwerkverbindung unterbrochen sein, so ist dies dem Benutzer auf der Anzeige darzustellen.
Nachbedingung	N1: Die Beschreibung des Dienstes wird auf der Anzeige dargestellt.

Nr.:**2** Zielorte anzeigen	
Zusammenfassung	Sämtliche Zielorte werden auf der Anzeige dargestellt.
Vorbedingungen	Die Zielortliste muss übertragen worden sein.
Beschreibung des Ablaufs	Sobald die Zielortliste vorhanden ist, können diese angezeigt werden.
Ausnahmen	E1: Konnte die Übertragung der Zielort-Datei nicht stattfinden, so ist dies dem Benutzer mitzuteilen.
Nachbedingungen	N1: Alle Zielorte einer Etage werden angezeigt.

Nr.:**3** Zielorte suchen	
Zusammenfassung	Sämtliche Zielorte werden auf der Anzeige dargestellt.
Vorbedingungen	Die Zielortliste muss angezeigt werden.
Beschreibung des Ablaufs	Sobald die Zielortliste angezeigt wird, können Zielorte gewählt werden.
Ausnahmen	E1: Konnte die Übertragung der Zielort-Datei nicht stattfinden, so ist dies dem Benutzer mitzuteilen.
Nachbedingungen	N1: Der Zielort kann gewählt werden.

Nr.:**4** Zielort wählen	
Zusammenfassung	Dieses Use-Case beschreibt die Möglichkeit der Auswahl eines Zielortes aus einer Liste.
Vorbedingungen:	Die Zielorte müssen angezeigt werden und die Zielort suche muss abgeschlossen sein.
Beschreibung des Ablaufs	Ein Benutzer hat einen Zielort gefunden und wählt diesen per Tastendruck aus.
Ausnahmen	E1: Falls die Netzwerkverbindung unterbrochen wird, so ist dies dem Benutzer auf der Anzeige darzustellen. E2: Fällt das Modul zur Standortermittlung aus, so ist dies dem Benutzer auf der Anzeige darzustellen.
Nachbedingungen	N1: Der Zielort wurde gewählt und die Routenberechung wurde ausgelöst.

Nr.:**5** Route darstellen.	
Zusammenfassung	Eine Route wird berechnet und auf der Anzeige dargestellt.
Vorbedingungen	Ein Zielort muss gewählt worden sein.
Beschreibung des Ablaufs	Die Route wurde vom Server berechnet und an den PDA übertragen wurde.
Nachbedingungen	N1: Die Route wird auf der Anzeige dargestellt.

Nr.:**6** Standort darstellen	
Zusammenfassung	Das Display zeigt ein oder mehrere Kartenteile an, auf denen der aktuelle Standort des Benutzers zu sehen ist.
Vorbedingungen	Die Verbindung zwischen PDA und Empfangsmodul muss hergestellt sein und die aktuelle Position des Benutzers feststehen.
Beschreibung des Ablaufs	Das Empfangsmodul errechnet die Position, speichert diese in einem Koordinatenpaar. Das PDA pollt nach dem Koordinatenpaar der Position.
Ausnahmen	E1: Sollte die Verbindung unterbrochen sein, so ist dies dem Benutzer auf der Anzeige darzustellen. E2: Fällt das Modul, das den Standort ermittelt, aus, so ist dies dem Benutzer mitzuteilen.
Nachbedingungen	N1: Der Standort ist auf der Anzeige zu sehen.

Nr.:**7** Positionsaktualisierung.	
Zusammenfassung	Das System erfasst die aktuelle Position des Benutzers in regelmäßigen Abständen.
Vorbedingungen	Die Kartenteile müssen geladen und die Verbindung hergestellt sein.
Beschreibung des Ablaufs	Die Anwendung pollt in regelmäßigen zeitlichen Abständen nach einem Koordinatenpaar. Sobald sie eine neue Position gepollt hat, wird diese auf der Anzeige dargestellt
Ausnahmen	E1: Fällt das Modul zur Standortermittlung aus, so ist dies dem Benutzer auf der Anzeige darzustellen.
Nachbedingungen	N1: Die neue Position ist auf der Anzeige zu sehen. N2: Die alte Position ist überschrieben worden.

Nr.:**8** Standortkarte darstellen	
Zusammenfassung	Auf der Anzeige ist die Beschreibung des Dienstes zu lesen und die Funktionalität erklärt
Vorbedingungen	Ein Zielort muss gewählt worden sein. Der Standort muss feststehen. Die Standortkarte muss bereit übertragen worden sein. Die POIs und Landmarken müssen darstellbar sein.
Beschreibung des Ablaufs	Sobald die Route berechnet wurde, die Karte bereitsteht und die POIs und Landmarken dargestellt werden können, wird auch die Standortkarte bereitgestellt.
Ausnahmen	E1: Sollte die Netzwerkverbindung unterbrochen sein, so ist dies dem Benutzer auf der Anzeige darzustellen.
Nachbedingungen	N1: Die Standortkarte wird angezeigt.

Nr.:**9** Karte scrollen	
Zusammenfassung	Die Karte kann in vier verschiedene Richtungen verschoben werden.
Vorbedingungen	Die Standortkarte muss dargestellt sein.
Beschreibung des Ablaufs	Die Standortkarte kann mittels Tasten nach oben, unten, links und rechts verschoben werden.
Nachbedingungen	N1: Die Karte ist verschoben.

Nr.:**10** Zoomen	
Zusammenfassung	Die Karte kann vergrößert oder verkleinert werden.
Vorbedingungen	Die Karte muss dargestellt sein.
Beschreibung des Ablaufs	Die Karte kann mittels zweier Tasten vom Benutzer verkleinert oder vergrößert werden.
Nachbedingungen	N1: Die Karte ist vergrößert oder verkleinert und die POIs und Landmarken gefiltert.

Nr.:**11** Orientierungshilfen darstellen.	
Zusammenfassung	Die in der Analyse erläuterten Orientierungshilfen (Maßstab, Distanz ins Ziel, Nordausrichtung des Gebäudes und Abzweigungshinweise) werden auf der Anzeige dargestellt.
Vorbedingungen	Die Standortkarte muss dargestellt werden.
Beschreibung des Ablaufs	Sobald die Standortkarte angezeigt wird, können auch die Orientierungshilfen dargestellt werden.
Nachbedingung	N1: Die Orientierungshilfen werden auf der Anzeige dargestellt.

Nr.:**12** Karte und Kartenelemente drehen.	
Zusammenfassung	Alle auf der Karte befindlichen Elemente werden gedreht.
Vorbedingungen	Die Route ändert ihre Richtung.
Beschreibung des Ablaufs	Bei einer Richtungsänderung werden die POIs, die Landmarken gedreht. Die Orientierungshilfe, welche die Ausrichtung anzeigt, und die Route müssen ebenfalls aktualisiert werden.
Nachbedingung	N1: Alle POIs und Landmarken, die Orientierungshilfe und die Route werden entweder gedreht oder aktualisiert auf der Karte dargestellt.

Nr.:**13** POI/Landmarke laden.	
Zusammenfassung	Ein POI oder eine Landmarke wird geladen
Vorbedingungen	Die Zoomstufe wurde verändert, der Kartendienst gestartet oder die Route berechnet.
Beschreibung des Ablaufs	Sobald die Zoomstufe verändert wurde oder der Dienst gestartet wird erfolgt das Laden eines POIs oder einer Landmarke vom Server. Gleiches geschieht bei der Routenberechnung.
Ausnahmen	E1: Konnte kein POI oder keine Landmarke geladen werden, so ist dies dem Benutzer mitzuteilen.
Nachbedingungen	N1: Die Landmarken wurden geladen.

Nr.:**14** POI und Landmarken filtern.	
Zusammenfassung	Das Detailniveau hängt von der Zoomstufe ab. Die einem Detailniveau zugehörigen POIs/Landmarken werden vom System aus einer Datei geladen.
Vorbedingungen	Die Netzwerkverbindung muss hergestellt sein. Ein Kartenteil muss geladen sein. Zu Beginn der Anwendung legt das System die Zoomstufe und damit das Detailniveau für den ersten Kartenausschnitt fest.
Beschreibung des Ablaufs	Die Karte wurde vom Benutzer vergrößert oder verkleinert. Daraufhin erfolgt eine Filterung der POIs oder Landmarken.
Ausnahmen	E1: Sollte die Verbindung beim Laden eines Kartenteils unterbrochen worden sein, so ist dies dem Benutzer mitzuteilen und anschließend die Karte mit dem aktuellen Standort darzustellen.
Nachbedingungen	N1: POIs oder Landmarken einer Detailstufe sind gefiltert.

Die erwähnte Klassifizierung und die damit verbundene Einordnung der Landmarken in Detailstufen erfolgt nun an dieser Stelle. Es werden zunächst mögliche Landmarken, die Verwendung finden könnten, erörtert. Daraufhin folgt eine Bewertung der Relevanz einer Landmarke. Die Relevanz bewertet ob eine Landmarke vorhanden sein sollte oder nicht. In der Spalte „Klassifizierung“ wird für ein Landmarke der Nutzen bei der Orientierung des Benutzers festgelegt. Die in der rechten Spalte festgelegten Detailnamen sind Varianten für Tooltips. Daher sollten Tooltips diese Namen tragen.

Landmarke	Relevanz	Klassifizierung	Detailnamen (Zusätze)
Raum	ja	essenziell	Küche, Vorlesung, Bad, Anlage, Sekretariat, Arbeitsraum, Unterrichtsraum, Mensa, Büro
Flur	ja	essenziell	Nord, West, Süd, Ost, Mitte
Fahrstuhl	wenn vorhanden	essenziell	Nord, West, Süd, Ost, Mitte
Schilder	ja	essenziell	Wegbeschreibung
Raum-Beschreibungen	ja	sehr wichtig	Raumnummer, Name des Bediensteten
Tür	ja	sehr wichtig	Material, Aussehen
Maschinen	wenn vorhanden	wichtig	Anlagen, Ausstellungsstücke, Computer
Treppen	ja	essenziell	Nord, West, Süd, Ost, Mitte
Bilder	wenn wenig Ausstattung	weniger wichtig	Als Treffpunkte, Malername
Gebäudezugänge	ja	essenziell	Nord, West, Süd, Ost
Fussbodenbelag	wenn prägnant	nicht wichtig	Material
Kamera	wenn wenig Ausstattung	nicht wichtig	
Fenster	ja	weniger wichtig	Nord, West, Süd, Ost
Mobilar (statische)	ja	wichtig	Plastiken, Werkbänke, Tafel Schrank
Telefone	Wenn vorhanden	sehr wichtig	
Ersthelfer	ja	sehr wichtig	ERSTHELFER
Notausgänge	ja	sehr wichtig	Notausgang
Rolltreppen	wenn vorhanden	sehr wichtig	Rolltreppe
Brücken zw. Gebäuden	wenn vorhanden	wichtig	[Geb. A| Geb. B]

Aus der Tabelle ergeben sich somit vier Klassifizierungsstufen.

- **essenziell:** Raum, Flur, Fahrstuhl, Treppe, Gebäudezugang, Schilder

- **sehr wichtig:** Telefone, Ersthelfer, Notausgänge, Rolltreppen, Raumbeschreibungen, Tür
- **wichtig:** Maschinen, Mobilar (statisches), Brücken zw. Gebäuden
- **weniger wichtig:** Bilder, Fenster
- **unwichtig:** Fussbodenbelag, Heizkörper, Kamera, Spiegel

Nr.:15 POI/Landmarke darstellen.	
Zusammenfassung	Ein geladenes POI oder eine Landmarke wird auf der Anzeige dargestellt
Vorbedingungen	Eine Karte ist auf der Anzeige zu sehen. POIs und Landmarken sind gefiltert.
Beschreibung des Ablaufs	Ein neues POI oder eine neue Landmarke wurde geladen und gefiltert. Die POIs und Landmarken, die nicht der Detailstufe zugehören werden ausgeblendet.
Ausnahmen	E1: Konnte kein POI oder keine Landmarke geladen werden, so ist dies dem Benutzer mitzuteilen.
Nachbedingungen	N1: POIs und Landmarken einer Detailstufe werden auf der Karte angezeigt.

Nr.:16 Tooltips darstellen.	
Zusammenfassung	Zur eindeutigen Beschreibung eines POI, einer Landmarke werden Tooltips angezeigt.
Vorbedingungen	Die POIs/Landmarken müssen auf der Anzeige angezeigt werden.
Beschreibung des Ablaufs	Mittels eines gelben Punktes kann der Benutzer ein POI/Landmarke anvisieren. Sobald der gelbe Punkt über dem POI, der Landmarke platziert ist, erscheint ein Tooltip. Der Benutzer kann nun die Beschreibung des POI, der Landmarke lesen und dadurch dessen Bedeutung erkennen und verstehen. Sobald der gelbe Punkt nicht mehr über dem POI, der Landmarke platziert ist, wird der Tooltip ausgeblendet.
Ausnahmen	E1: Ist ein POI, eine Landmarke durch seine Form und sein Aussehen eindeutig und dadurch selbsterklärend, so ist die Anzeige eines Tooltips unnötig.
Nachbedingung	N1: Ein Tooltip wird auf der Anzeige dargestellt.

Nr.:**17** Beenden	
Zusammenfassung	Die Anwendung wird vom Benutzer oder auch vom System beendet
Vorbedingungen	Der Dienst muss laufen und das Beenden wurde initialisiert.
Beschreibung des Ablaufs	Erfolgt während der Anwendung eine vom System oder vom Benutzer ausgelöste Beendigung der Anwendung, so muss dies umgesetzt werden.
Nachbedingungen	Die Anwendung ist beendet.

Fehlerbehandlung

Da es nicht auszuschließen ist, dass Fehler auftreten sollten diese Berücksichtigung finden. Daher gilt es zunächst diese Fehlerquellen aufzudecken. Mögliche Fehlerquellen könnten auftreten bei der Kommunikation zwischen PDA und Server, zwischen PDA und Empfangsmodul und durch die Anwendung selbst. Es ist beispielsweise denkbar ist, dass es bei der Übertragung von Daten zu abrupten Verbindungsabbrüchen kommt, sollten geeignete Maßnahmen zur Kompensation dieser Fehler definiert werden.

Nr.:**18** Fehlerausgabe	
Zusammenfassung	Auf dem Display erscheint eine Fehlermeldung
Vorbedingungen	Die Installation der Anwendung sollte abgeschlossen sein.
Beschreibung des Ablaufs	Sobald während der Ausführung des Dienstes ein Fehler auftritt, wird dieser angezeigt.
Nachbedingungen	N1: Eine Fehlermeldung wird auf der Anzeige Dargestellt. N2: Das System ist in einem konsistenten Zustand.

4.4. Nicht funktionale Anforderungen an die Anwendung

Nicht-Funktionale Anforderungen drücken aus, wie ein System aufgrund seiner Software-Funktionalität und seiner Hardware-Ressourcen reagieren kann. Nicht-funktionale Anforderungen sind Anforderungen an die Umstände, unter denen die geforderte Funktionalität zu erbringen ist [(25)]. Demzufolge sind sie die globalen Zwänge einer Software [(26)].

Die ISO/IEC 9126 stellt ein Modell zur Sicherstellung der Softwarequalität zur Verfügung. Folgende Qualitätsmerkmale wurden diesem Modell entnommen und dienen der Beschreibung der nicht-funktionalen Anforderungen der Software dieser Arbeit.

Funktionalität

Das Qualitätsmerkmal Funktionalität zu erläutern erfordert zunächst die Auflistung der zu implementierenden Funktionen der Software, welche zur Erfüllung der Funktionalität eines Indoor-Navigationssystems nötig sind. An dieser Stelle soll es genügen einen Verweis auf Kapitel 4.2 zu geben.

Aus der Zusammensetzung der funktionalen Anforderungen in Bezug auf die jeweiligen Anwendungsfälle, kann und soll die *Angemessenheit* abgeleitet werden. Das bedeutet beispielsweise, dass die Funktionalität des Zoomen angemessen ist, wenn alle Attribute Räume, Türen, POIs, Standort, Route auf der Anzeige vergrößert/verkleinert dargestellt werden. Würden die Räumen nicht der gleichen Vergrößerung/Verkleinerung entsprechen, wäre keine Angemessenheit an die Funktionalität gegeben. Daher ist für jeden Anwendungsfall die Angemessenheit zu erfüllen. Auf die *Richtigkeit* der Positionierung der Elemente Räume, Türen, POIs, Standort und der Route ist ebenfalls zu achten. Das bedeutet, dass die Positionierung dieser Elemente auf der Anzeige so exakt wie möglich sein muss, da sonst keine wirklichkeitsgetreue Abbildung der Umgebung möglich ist.

Ebenfalls ist dafür zu sorgen, dass die Befehle, die über das Bedienelement abgegeben werden, die Funktionalität erfüllen. Beispielsweise muss der Befehl für das Verkleinern auf verschiedenen PDAs gleich funktionieren und nicht etwa das Gegenteil oder eine andere Funktionalität zur Folge haben.

Die *Interoperabilität*, also die Fähigkeit mit anderen Systemen zusammenzuwirken ist nicht zu berücksichtigen. Vorerst ist das System in sich geschlossen und greift nicht auf andere Systeme zu. Es wird daher auch nicht von anderen Systemen beansprucht.

Für die *Sicherheit* des Systems ist zu sorgen. Der unberechtigte Zugriff auf den Service und die Daten des Karten-Servers ist zu sichern. Mittels der Teil-Anwendung darf es also nicht möglich sein neben den erwünschten Funktionalitäten, wie denen der Anwendungsfälle, zusätzliche und unerwünschte Funktionalität zu implementieren.

Darüber hinaus sind die übertragenen Dateien zu sichern. Ohne Absicherung könnte im Falle der Veränderung und Wiedereinspielung von Daten-Material unvorhersehbare Folgen haben. Beispielsweise könnten Räume, Türen, POIs, der Standort oder die Route falsch oder gar nicht angezeigt werden.

Zuverlässigkeit

Die Zuverlässigkeit beschreibt ein bestimmtes Leistungsniveau, unter dem die Anwendung eine gewisse Zeit und unter bestimmten Bedingungen aufrecht erhalten werden könnte. Somit besteht die Zuverlässigkeit der laufenden Client-Anwendung darin, eine geringe Versa-

genshäufigkeit durch Fehlerzustände zu ermöglichen. Die Fehlerzustände sind daher zu minimieren und, falls unvermeidbar, zu kompensieren. Beispielsweise könnte es vorkommen, dass ein Benutzer einen Zielort unter den gespeicherten und angezeigten Zielorten nicht finden kann. Damit wäre es unmöglich, das Indoor-Navigations-System zu nutzen. Es ist daher stets im Sinne der *Fehlertoleranz* einen konsistenten Zustand für das System zu finden, aus dem der Benutzer dennoch Nutzen ziehen kann. Beispielsweise könnte der Benutzer bei Nichterfüllung der Routenfindung den Service zum Durchsuchen der Etage mittels des gelben Punktes nutzen.

Für Fehler, für die sich keine konsistenten Zustände finden lassen, beispielsweise einem Verbindungsabbruch während dem Start der Anwendung, sollten auf der Anzeige Erklärungen in Form von eingespielten Texten erscheinen. Diese Erklärungen sollten über den aufgetretenen Fehler informieren und dem Benutzer Ratschläge zum weiteren Fortfahren geben.

Ebenso notwendig ist es, dass das System *robust* genug ist, um Fehlereingaben und unsachgemäße Benutzung der Teil-Anwendung zu gewährleisten. Das fortwährende Drücken einer Taste darf keinen Systemfehler verursachen. Ein fortwährendes Drücken einer Taste kann beispielsweise beim Visieren mit dem gelben Punkt, bzw. dem Verschieben der Anzeige, oder beim Durchsuchen der Zielortliste auftreten. Das An- und Ausschalten des PDA während der laufenden Anwendung darf ebenfalls keinen Nachteil für den PDA, bzw. für die Anwendung auf dem PDA ergeben (Speicherüberfüllung durch gespeicherte Daten). Eine etwaige offene Netzwerkverbindung muss nach deren Nutzung wieder geschlossen werden.

Im Falle eines Systemausfalls des PDA, nachdem die Anwendung erneut installiert und wiederhergestellt werden muss, damit der Service wieder genutzt werden kann, müssen die zuletzt genutzten Kartendaten auf dem Karten-Server abrufbar sein. Das bedeutet, dass die Kartendaten, die unmittelbar vor dem Systemausfall genutzt wurden, wieder bereitgestellt werden müssen.

Benutzbarkeit

Für die Beurteilung der *Benutzbarkeit* sollte die Anwendung von einer Benutzergruppe getestet worden sein. Dieser Test kann aufgrund dessen, dass das System nicht komplett implementiert ist, nicht stattfinden. Dennoch wird auf die Kriterien der Benutzbarkeit eingegangen, um darzulegen, dass sie in der Anwendung Berücksichtigung findet.

Das Kriterium der *Verständlichkeit* wird durch die Auswahl der Muttersprache durch den Benutzer gewährleistet. Ebenfalls sollen kurze, eindeutige textuelle Beschreibungen die Verständlichkeit von Erläuterungen dienen. Beschreibungen erfolgen bei der Vorstellung der Anwendung und dienen der Erklärung mittels Tooltips. Die Anwendung soll ohne zusätzliche Hilfsmittel, wie beispielsweise einem Handbuch, *einfach erlernbar* und übersichtlich sein. Zu

diesem Zweck sollte zu Beginn der Anwendung eine Beschreibung der Anwendungsfälle sowie die Aufzählung der dazugehörige Tasten stattfinden.

Bedienerfreundlich sollte auch die Benutzerführung in Form eines Dialoges sein. Eine textuelle Beschreibung sollte den Benutzer in kleinen Schritten durch die Anwendung führen. Dadurch wird kein spezielles Wissen über den Aufbau der Software vom Benutzer abverlangt.

Sofern die Anwendung nur die nötigen Anwendungsfälle implementiert, bleibt sie übersichtlich. Anwendungsfälle, wie beispielsweise eine Routenberechnung nach unterschiedlichen Kriterien, verlangt nach zusätzlicher Erläuterung. Zusätzliche Erklärungen können die Anwendung aber unübersichtlich erscheinen lassen, so beispielsweise, weil sich der Benutzer zunächst ein Überblick über die verschieden Kriterien verschaffen muss und diese anschließend ausprobiert und vergleicht. Dem Benutzer sollte daher soviel wie nötig an Entscheidung, beispielsweise infolge der Einflussnahme durch die Wahl von Kriterien, abgenommen werden.

Weiterhin darf die Verarbeitung der Daten nicht so umfassend sein, dass der Benutzer eine relativ lange Wartezeit aufbringen muss. Dies beeinträchtigt die Benutzerfreundlichkeit, etwa wenn der Benutzer befürchtet, dass die Anwendung nicht mehr funktioniert.

Die *Bedienbarkeit* ist durch das Bedienelement geprägt. Um verschiedene Anwendungsfälle auszulösen, sind je nach Anwendungsfall ein bis drei Tasten nötig. Auf Doppelbelegung von Tasten sollte verzichtet werden, damit die Erlernbarkeit einfach und übersichtlich bleibt. Zur Erhaltung der Benutzerfreundlichkeit sollten die Interaktionen mit der Anwendung durch den Benutzer gering gehalten werden. Aus diesem Grunde sollten der Dienst auf die Anzeige- und Eingabefunktionen beschränkt bleiben.

Attraktiv muss die Anwendung durch die einfache Benutzung und die übersichtliche und farbliche Darstellung einer Etage sein. Zudem sollen POIs mit den dazugehörigen Tooltips den Komfort und dadurch die Anziehungskraft erhöhen. POIs können die Neugierde eines Benutzers wecken, Unbedachtes zu entdecken und zu erkunden.

Effizienz

Die Effizienz spiegelt sich im *Zeitverhalten* und dem *Verbrauchsverhalten* der Anwendung wieder. Dabei spielt die Hardware ebenfalls eine Teilrolle, denn je schneller die CPU ist, desto schneller ist das Zeitverhalten der Anwendung.

Bei der Implementierung ist zu berücksichtigen, dass wenig CPU-Zeit zur Verfügung steht. Daher sind komplexe Algorithmen zu vermeiden, um die Verarbeitungszeit so niedrig wie

möglich zu halten. Anderenfalls hat dies negative Auswirkungen auf die Benutzerfreundlichkeit. Es ist daher auch anhand der Komplexität einer API zu entscheiden, ob diese genutzt wird oder ob eine ähnliche Funktionalität eigens implementiert werden kann.

Das *Verbrauchsverhalten* zum Verarbeiten der Gebäudedaten-Basis berücksichtigt die benötigte Peripherie, den Karten-Server, die CPU, den Speicher und die Anzeige des PDA. Zum Erstellen der Karte auf der Anzeige muss der Karten-Server kontaktiert werden. Es ist zu entscheiden, ob diese synchron oder asynchron angefordert werden. Die CPU des *PDA* soll ausgelastet werden, damit die Anwendung die volle Effizienz erlangt. Auf die Qualität der Ausgabe auf der Anzeige ist daher besonders zu achten. Die Qualität der Anzeige ist so zu gestalten, dass sie die Ressourcen schont aber gleichzeitig die Benutzerfreundlichkeit gewahrt bleibt.

Änderbarkeit

Die Anwendung sollte anhand eines Design-Pattern entworfen werden. Damit kann ein modularer Aufbau der Anwendung erschaffen werden. Ein solcher modularer Aufbau sorgt dafür, dass die einzelnen Module austauschbar sind, was die Änderbarkeit begünstigt. Das passende Design-Pattern ist daher zu bestimmen und anzuwenden.

Durch ein Design-Pattern erhöht sich gleichfalls die *Modifizierbarkeit*, da die Module einzeln auf Fehler hin untersucht werden können und dadurch leichter anpassbar bleiben.

Ebenfalls ist durch den modularen Aufbau der Anwendung dafür gesorgt, dass die Anwendung nach eventuellen Änderungen einzelner Module stabil bleibt. Jedes Modul hat seine Aufgaben, die es erfüllen muss, um diese abschließend über Schnittstellen bereitzustellen. Die jeweils anderen Module, die auf diese Schnittstelle zugreifen, bleiben durch die Änderung unangetastet.

Um den Aufwand zur Durchführung von *Änderungen* zu minimieren, sollten die Klassen und Methoden der Anwendung mittels java-doc kommentiert werden. Darin werden die Funktionsweise und die Übergabe-Parameter beschrieben. Durch das Kommentieren erhält sich der Überblick über die Klassen und verringert die Zeit für die Einarbeitung in die Anwendung.

Zur *Prüfbarkeit* der Anwendung ist die benötigte Hard- und Software, die nötig ist um die Anwendung zu prüfen, zu beschreiben. Die verwendete Hard- und Software zur Erstellung der Anwendung ist daher ebenfalls anzugeben und zu beschreiben.

Übertragbarkeit

Eine Software gilt als portierbar, wenn sie ohne technische und software-technische Anpassung auf einen anderen Client übertragbar ist. Zur Übertragbarkeit der Anwendungs-Software sollte daher diskutiert werden, welche Programmiersprache zur Verwirklichung eingesetzt wird. Die Auswahl einer geeigneten Programmiersprache muss berücksichtigen, dass verschiedene PDAs mit unterschiedlichen Betriebssystemen verschiedener Hersteller die Anwendung nutzen werden. Die *Installierbarkeit* sollte ebenfalls besprochen und dargestellt werden. Eine erneute Installation der Anwendungs-Software, könnte nach einem Systemabsturz notwendig werden.

4.5. Nicht funktionale Anforderungen an die Benutzerführung und an die Karte

Um den Indoor-Navigationsdienst zu nutzen, ist es notwendig die Gestaltung der Benutzerschnittstelle zu konzipieren. Es müssen Richtlinien festgelegt werden, anhand derer die Konzeption stattfinden kann. In der DIN EN ISO 9241 sind die Kriterien für die Gestaltung der Benutzerschnittstelle wie folgt definiert:

1. Aufgabenangemessenheit

   Die Benutzer werden in der Erledigung ihrer Arbeitsaufgabe effizient unterstützt. Sie erreichen ihre Ziele schnell, ohne durch die Eigenschaften des Dialogsystems unnötig belastet zu werden.

   Ein Anwendungsfall sollte daher mit wenigen Aktionen ausführbar sein.

2. Selbstbeschreibungsfähigkeit

   Jeder Dialogschritt ist unmittelbar verständlich. Die Benutzer können sich eine für das Verständnis und für die Erledigung der Arbeitsaufgabe zweckmäßige Vorstellung der Systemzusammenhänge machen.

   Der Benutzer sollte über die Funktionen der Anwendungsfälle Information erhalten. Es muss beschrieben sein, mit welchen Tasten dieser durchführbar ist und was den Benutzer daraufhin erwartet. Die Anwendung soll dem Benutzer verdeutlichen, wie er sein Ziel erreicht. Klare Navigation und verständliche Anweisungen bei jedem Schritt sind die Voraussetzungen.

3. Steuerbar

   Der Benutzer soll die Anwendung steuern, nicht umgekehrt. Das heißt, dass Animationen abgebrochen und erneut gestartet werden können, es immer einen Weg zurück gibt und bei Ton die Lautstärke sich regulieren lässt.

4. Erwartungskonformität

   Der Dialog entspricht den Erwartungen, die die Benutzer aus Erfahrungen mit bisherigen Arbeitsabläufen oder aus der Benutzerschulung mitbringen.

   Es muss daher Rücksicht auf schon bekannte und verwendete Anwendungen Bezug genommen werden. Somit hat der Benutzer auch bei dieser Anwendung die Erwartung an eine ähnlich funktionierende Anwendung.

5. Fehlertolerant

   Das System soll mit falschen Benutzereingaben umgehen können und bei Fehlern klare Rückmeldung geben. Der Korrekturaufwand für den Benutzer soll minimal sein.

   Die Qualität der Fehlermeldung muss daher angemessen sein. Die Fehlermeldung muss lesbar und der Fehler, wenn möglich, mit einfachen Aktionen rückgängig zu machen sein.

6. Lernförderlich

   Die Anwendung soll den Benutzer dabei unterstützen, den Umgang mit ihr schrittweise zu erlernen.

   Es sollte daher zu Beginn eine Erläuterung der Vorgehensweise der Anwendungsfällen stattfinden.

Des weiteren sollten nach Empfehlung von (2) folgende nicht funktionalen Anforderungen bei der Präsentation der Gebäudekarten auf der Anzeige der Benutzerschnittstelle realisiert werden:

1. Lesbarkeit

   Alle essentiellen Komponenten der Route, speziell Flure und Durchgänge sollten sichtbar und leicht erkennbar sein.

2. Klarheit

   Die Route sollte klar bezeichnet und sich auch bei einem schnellen Blick eindeutig zu vom Hintergrund abheben und zu erkennen sein. Die Karte sollte nur soviel Informationen enthalten wie nötig.

3. Vollständigkeit

   Die Karte muss alle nötigen Informationen für die Navigation bereitstellen.

4. Angemessenheit

   Eine Routenkarte wird während der Routenbegehung verwendet. Daher sollte sie leicht zu transportieren und zu manipulieren sein.

4.6. Zusammenfassung

Das betrachtete Beispiel-Szenario stellt exemplarisch die Indoor-Navigation mittels eines PDA vor. So werden im darauf folgenden Schritt die funktionalen Anforderungen aufgearbeitet. Aus diesem Szenario sind sowohl die Systemanwendungsfälle als auch die Geschäftsanwendungsfälle entwickelt worden. Die Systemanwendungsfälle betreffen:

- Die Kommunikation zwischen Server und mobilem Gerät
- Die Kommunikation zwischen den IMAPS-Beacon und -Empfangsmodul
- Die Kommunikation zwischen IMAPS-Empfangsmodul und dem mobilen Gerät

Die Funktionalität der Anwendung ergeben sich durch die Elemente:

- Einleitung und Handhabung des Dienstes anzeigen
- Standortkarte darstellen
- Route anzeigen
- POIs anzeigen
- POI-Details anzeigen
- Landmarken darstellen
- Landmarken-Details darstellen
- Standort darstellen
- Standort-Details darstellen
- Zielorte wählen
- Zielorte anzeigen
- Zoomfunktion bereitstellen

Zusätzliche Elmente

- Maßstab anzeigen
- Abzweigungshinweise anzeigen
- Ausrichtung gegen Nord darstellen
- Distanz bis zum Ziel anzeigen

Für die Übersicht aller Use-Cases wurde ein Use-Case-Diagramm erstellt. Das Use-Case POI/Landmarken-darstellen erforderte eine Aufstellung von Landmarken, wie sie in einem Gebäude vorzufinden sind. Anhand dieser Liste wurde schließlich auch die Relevanz einzelner POIs festgelegt. Diese Einteilung legt fest, welche POIs wichtig sind und daher ständig anzuzeigen sind und solche, die in verschiedenen Zoomstufen angezeigt werden müssen.

Auf die Geschäftsanwendungsfälle folgte die Definition der nicht-funktionalen Anforderungen. Diese wurden sowohl für die Indoor-Navigationssoftware diskutiert, als auch für die Benutzerschnittstelle des mobilen Gerätes.

Im nächsten Kapitel 5 erfolgt der Grobentwurf des Systems auf der Basis der funktionalen und nicht funktionalen Anforderungen.

5. Grobentwurf

Dieses und das darauf folgende Kapitel 6 dienen der Ausarbeitung eines Lösungskonzeptes zur Erstellung der Anwendung, mit deren Hilfe eine Indoor-Navigation mittels eines PDA vorgenommen werden kann. Zu deren Entwicklung werden die in Kapitel 4 Analyse erarbeiteten funktionalen und nicht funktionalen Anforderungen umgesetzt.

Das Lösungskonzept hält die Struktur und die Abläufe des Systems fest. Die statische Struktur des Systems wurde mittels eines Schichtenmodels und die Geschäftsanwendungsfälle durch Use-Case-Diagrammen veranschaulicht. Dynamischen Vorgänge innerhalb des Systems werden mittels Sequenzdiagrammen dargestellt. Die drei Diagrammtypen halten sich an die standardisierte Beschreibungssprache *Unified Modeling Language 2.0*.

In Kapitel 5.1 soll zunächst, nach der Empfehlung von IEEE 1016, eine Nachvollziehbarkeitsmatrix aufgestellt werden. In Kapitel 5.3 wird dann auch die gesamte Anwendungsarchitektur des Systems spezifiziert, ohne die kein Design entwickelt werden könnte [(45)]. Kapitel 5.3 erläutert den Nutzen der Verwendung des *Model-View-Controller*-Pattern. Neben der Erläuterung des Pattern werden seine Komponenten (Model, View und Controller) übernommen, um daraus Komponenten für die eigene Anwendung zu erarbeiten, die diese Funktionalität verkörpern. Da die Kommunikation zwischen den Komponenten eine große Rolle spielt, werden die Kommunikationswege und deren Schnittstellen besprochen. Veranschaulicht wird die Kommunikation zwischen den Komponenten im Kapitel 6.5.

5.1. Nachvollziehbarkeitsmatix

Die Nachvollziehbarkeitsmatrix stellt Use-Cases und Komponenten der Anwendungs-Software dar. In der linken Spalte sind die Use-Cases zu sehen, die aus Kapitel 4 stammen. Die obere Zeile beinhaltet die Namen der Komponenten. Ein „•“ in der Matrix stellt die Verbindung zwischen Use-Case und Komponente dar. Das Use-Case nutzt zur Abarbeitung des Anwendungsfalles die Komponenten.

Die Komponente Raumsuche könnte aus einer Datei Java-Objekte vom Typ „Raum“ erzeugen, die von einem Karten-Server bereitgestellt wird. Die Komponente Türensuche würde auf die gleiche Weise verfahren, aber Türobjekte erzeugen. Die Komponente Route stellt eine vom Server erhaltene Datei mit den Angaben über Streckenabschnitte einer Route dar.

Use-Case	Raum-. suche	Tür-. suche	Route	Ziel- ort	Stand- ort	Karten- elemente	Navigations- elemente	Text
Beschreibung anzeigen								●
Zielort anzeigen				●				
Zielortsuche				●				
Zielortwahl				●				
Route darstellen	●	●	●		●	●	●	
Standort darstellen	●	●			●	●	●	
Position aktualisieren					●			
Standortkarte darstellen	●	●	●		●	●	●	
Karte scrollen			●		●	●	●	
Zoomen						●	●	
POI/Landmarken laden						●		
POI/Landmarken filtern						●		
POI/Landmarken darstellen		●			●	●	●	
Tooltips darstellen						●	●	
Orientierungshilfen darstellen			●			●	●	
Karte und Karten- elemente drehen			●			●	●	
Fehlerausgabe								●

Tabelle 5.1.: *Gegenüberstellung der Use-Cases aus Kapitel 4.3 zu möglichen Komponenten der Anwendung.*

Die Strecken führen dann stets von einem Raum, über Türen ins Ziel, der ebenfalls einem Raum ist. Die Komponente Zielort würde eine Datei mit Zielorten parsen, die sie ebenfalls vom Server erhalten würde. Die Komponente Standort sollte das Koordinatenpaar enthalten, dass die Position auf der Anzeige des PDA darstellt. Die Komponente Kartenelemente sollte die Landmarken und POIs beinhalten. Navigationselemente sollten neben der „Route“ Abzweigungshinweise, Maßstab, Gebäudeausrichtung und Distanz ins Ziel sein und sollten ebenfalls vom Server bereitgestellt werden. Die Komponente Text könnte zur Ausgabe eines Textes auf der Anzeige genutzt werden.

5.2. Gestaltung der Client-Server-Architektur

Die technisch restriktive Hardware des mobilen Endgerätes macht es notwendig, eine Kompetenz- und Funktionsverteilung für die Anwendungsarchitektur des Client-Server-Systems vorzunehmen. Die Fachliteratur von (56) beschreibt fünf nützliche Ansätze der Kompetenzverteilung für Client-Server-Systeme:

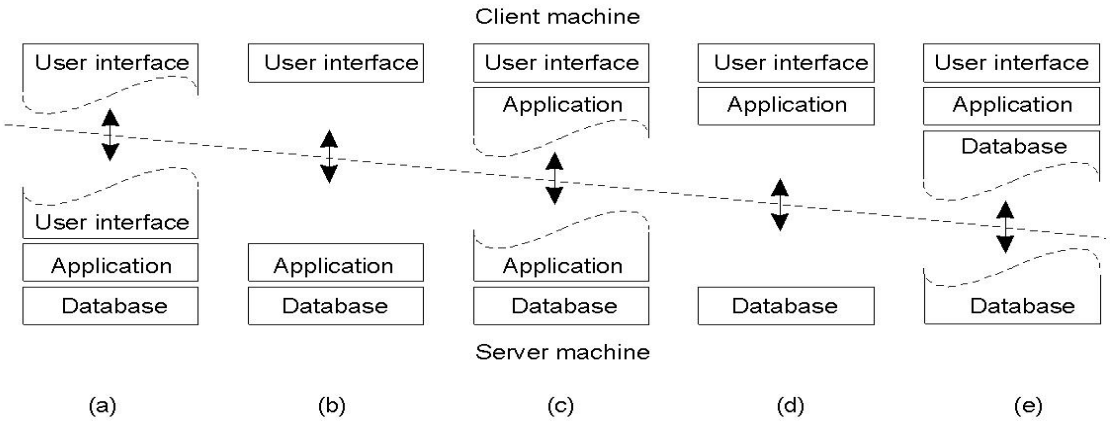


Abbildung 5.1.: *Eine Kompetenzverteilung nach Tanenbaum [(56)]*

Thin Client (a): Der Client dient ausschließlich zur Anzeige von Informationen. Die Client-Maschine ist nur ein Terminal, womit eine entfernt Kontrolle und Datendarstellung der Daten einer Applikation erreicht wird.

Thin Client (b): Der Client erhält zusätzlich zu (a) die Software über die Benutzeroberfläche. Die verteilte Applikation besteht also aus grafischem Frontend (auf dem Client) und kommuniziert mittels eines Protokolls mit der restlichen Applikation (auf dem Server).

Rich Client (c): Der Client erhält zusätzlich zu (b) einen Teil der Applikation. Beispielsweise wird neben der Karten und der Routenanzeige der aktuelle Standort ermittelt. Ebenso kann das Frontend die Konsistenz der Karten prüfen und, falls erforderlich, eine Fehlermeldung auswerfen.

Rich Client (d): Der Client erhält die gesamte Software der Applikation. Meist sind dies Geräte wie Desktop-PCs oder Workstations, die einen Netzwerkverkehr mit einem angeschlossenen Datenbank-Server führen.

Fat Client (e): Dieser Client besitzt die die Applikation und hat zusätzlich Datenbankeinträge gespeichert, auf die er im Verlauf der Anwendung zurückgreifen kann. Dies kann beispielsweise ein PC sein, mit dem der Benutzer im Internet surft und der einen Cache über zuletzt besuchte Webseiten angelegt hat.

Zur Realisierung der Anwendung sollte der Client, das mobile Gerät, als Rich-Client ausgelegt werden und zwar so wie er in (c) beschrieben wurde. Er übernimmt somit neben der Ein- und Ausgabe zusätzlich die Verarbeitung der Geschäftsanwendungsfälle und der programmtechnischen Umsetzung in graphische Elemente zur Darstellung auf der Anzeige.

5.3. Gestaltung der Anwendungsarchitektur

Die Anwendungsarchitektur definiert die prinzipiellen Möglichkeiten und Einschränkungen der Problemlösung und begrenzt dadurch den Lösungsraum [(45)]. Eine Anwendungsarchitektur gliedert sich allgemein in:

- technische Architektur und
- fachliche Architektur

Die technische Architektur stellt die fertige Implementierung der Anwendung dar. Sie repräsentiert unter anderem die technischen Komponenten. Ordnet man diese Komponenten einer Client-Server-Architektur zu, ist eine erste Unterteilung des Systems in unterschiedliche Schichten getan. Hierdurch ergibt sich eine Arbeitsteilung, die das System vereinfacht.

Die fachliche Architektur beschreibt die Struktur und die Partitionierung aus Sicht des Anwendungsgebietes und der Fachlichkeit [(45)]. Somit basiert die Beschreibung der fachlichen Architektur auf den in Kapitel 4.3 erarbeiteten funktionale Anforderungen und dem Erfahrungswissen über das Anwendungsgebiet.

Die Abbildung 5.3 zeigt das Client-Server-Modell und damit sollte erste Unterteilung des Indoor-Navigations-Systems erfolgen. Das Client-Server-Model ist ein so genanntes Schichtenmodel und vermittelt die angesprochene technische Architektur des Systems. In der Abbildung 5.3 werden zudem die Zugriffs- und Abhängigkeitsrichtung dargestellt.

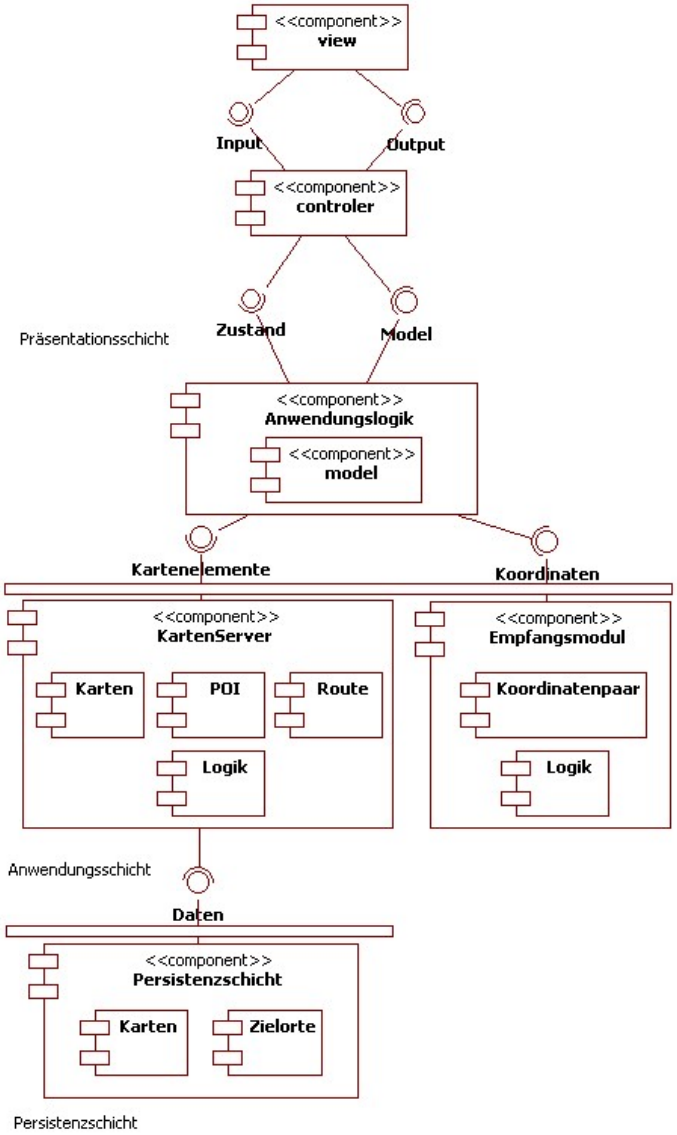


Abbildung 5.2.: *Die Drei-Schichten-Client-Server-Architektur*

Da durch die Schichtung das Indoor-Navigationssystem in verschiedene Teil-Anwendungen zerfällt, sollte erläutert werden, welche Aufgaben die Schichten übernehmen würden. Der Abbildung (5.3) entsprechend sollte das gesamte Projekt durch eine dreischichtige Client-Server-Architektur aufgebaut werden. Die vorliegende Schichten-Architektur ist in der Informatik unter dem Namen Three-Tier-Client-Server-Architektur bekannt:

Präsentationsschicht: Auf der Präsentationsschicht sollte die Präsentation des Indoor-Navigationssystems verwirklicht werden.

Karten-Server (Anwendungsschicht): Er sollte die Anwendung zur Bereitstellung von Route, Karten und Zielorten enthalten.

Empfangsmodul (Anwendungsschicht): Die Anwendung zur Ermittlung von Positionsdaten könnte durch das Imaps-Modul realisiert werden.

Persistenzschicht: Weitere Kartendaten und Zielortdaten sollten auf einem zusätzlichen Server gehalten werden.

Die Präsentationsschicht soll aus Hardwaresicht durch ein PDA verwirklicht werden, so wie in 5.2 beschrieben.

Die Schnittstelle *Kartenelemente* könnte durch einen Web-Service realisiert werden. Sie verbindet so die Präsentationsschicht mit dem Karten-Server der Anwendungsschicht. Die Kommunikation zwischen beiden erfolgt dann über das *SOAP* [(64)] welches in *HTML* eingebettet ist.

Der Karten-Server der Anwendungsschicht könnte die Routenberechnung übernehmen. Er könnte weiterhin die Kartenteilung übernehmen und die Zielort-Dateien bereitstellen. Die Route, die Karten und die Zielorte können dann in separaten Dateien über die Schnittstelle übertragen werden. Bevor die Übertragung stattfinden könnte, sollte auf der Präsentationsschicht mittels eines Client-Stub und einer URL[1] eine Instanz des Server-Dienstes erzeugt werden. Mittels des Client-Stub könnte so auf Methoden des Service zugegriffen werden.

Die Schnittstelle *Koordinaten* sollte zur Übertragung des Koordinaten-Paares dienen. Sie würde die Präsentationsschicht mit dem Empfangsmodul verbinden.

Die Komponente des Empfangsmoduls ist ebenfalls auf der Anwendungsschicht zu sehen. Sie beinhaltet ihre eigene Anwendung und sollte der Berechnung eines Positions-Koordinatenpaars dienen. So könnte die Präsentationsschicht auf dieses Koordinatenpaar zugreifen, um es zu nutzen.

Die Persistenzschicht sollte Gebäudekarten sowie sämtliche Zielorte beinhalten, die sich in einem Gebäude befinden. Diese Komponente sollte dann ausschließlich von den Komponenten der Anwendungsschicht genutzt werden.

[1] Uniform Rescource Locator

Somit ergibt sich aus oben genanntem, dass eine Schicht ihre Teil-Anwendung von einer anderen Schichten kapselt und die Kommunikation zwischen diesen Schichten über Schnittstellen abgewickelt kann.

Jede der vorgestellten Teil-Anwendungen würde zur Verwirklichung technische Komponenten verwenden, wie etwa Klassen oder Interfaces. Diese Klassen bzw. Interfaces sollten verschiedene Funktionen erfüllen. Diese Funktionen müssen festgelegt werden, bevor eine Einteilung technischer Komponenten in die Schichten der Architektur stattfinden kann. Daher müssen zunächst technische Komponenten gefunden werden, welche die Teil-Anwendung realisieren.

Technische Komponenten können durch Übertragung der Funktionalität der Komponenten eines Design-Pattern gebildet werden. Durch eine solche Zuordnung der Funktionalität ergäbe sich auch eine Struktur zwischen den Komponenten, wobei diese Struktur durch Abhängigkeiten zwischen ihnen ausgeprägt wäre. Daher könnte sowohl mit Hilfe eines Design-Pattern die Struktur der Präsentationsschicht, Kapitel 6.5, vorgegeben, als auch den Strukturelementen der Schicht (den Komponenten) Funktionalität übertragen werden.

Funktionalität kann auch durch Übertragung von funktionalen bzw. nicht funktionalen Anforderungen an Komponenten weitergegeben werden. Vor der Übertragung von Funktionalität eines Design-Pattern auf Komponenten muss ein geeignetes Pattern ausgewählt werden.

Ein Design-Pattern für die Anwendung

Ein Design-Pattern, das in der Softwareentwicklung oft verwendet wird, ist das Model-View-Controller-Pattern[2](MVC-Pattern). Das MVC-Pattern ist vielfach gut dokumentiert und dank dessen ist es einfach, Projekte mit ähnlichen Funktionalitäten in dieses Pattern zu überführen.

Die Funktionalität der Komponenten des MVC-Pattern lässt sich in folgender Weise beschreiben:

View: Das *View*-Objekt sollte die visuelle Darstellung übernehmen, die Bildschirmrepräsentation des *Model*-Objekts.

Controller: Das *Controller*-Objekt sollte die Möglichkeiten bestimmen, mit denen die Benuztungsschnittstelle auf Benutzereingabe reagieren kann.

Model: Das *Model*-Objekt stellt das Anwendungsobjekt dar. [(21)]

[2]Architekturmuster zur Aufteilung von Softwaresystemen in drei Einheiten.

Durch diese Funktionalitäten und deren Kapselung voneinander ist das MVC-Pattern dazu geeignet, das Problem der Anwendung zu vereinfachen. Das Problem besteht darin, das Display des PDA auf dem aktuellen Stand des Programmflusses zu halten, auch wenn sich Daten durch eine Benutzereingabe oder durch die Bewegung des Benutzers ändern.

Nun kann auch, im Vorgriff auf Kapitel 6.5, die Übertragung der Funktionalität des MVC-Pattern zu Komponenten der Teil-Anwendung stattfinden. Zur Implementierung der Funktionalitäten der Komponenten sollte auch die dazu verwendeten APIs angesprochen werden. Dies dient zum einen der Erläuterung der Komponenten und zum anderen zur Erläuterung der Funktionsweise aus Sicht des Entwicklers.

5.4. Gestaltung der Benutzerführung

Applikationen, die auf einem PDA laufen, unterliegen gewissen Einschränkungen. Diese Einschränkungen sind bei der Gestaltung zu berücksichtigen, zumal sie von Gerät zu Gerät unterschiedlich sind:

- die Bildschirmanzeige und die Auflösung
- einige Geräte haben eine monochrom-Anzeige, andere eine vielfarbige Anzeige
- die Geräte können eine unterschiedliche Anzahl an Tasten haben
- Tastenbelegungen können unterschiedlich sein

Softwaretechnisch finden diese Einschränkungen Berücksichtigung in der API *javax.microedition.lcdui*. Erläuterungen zur Verwendung werden in Kapitel 6.4 im Rahmen der Realisierung besprochen.

Die Programmführung sollte sich in drei verschiedene Phasen aufteilen. In die Startphase, die Phase der Routenbegehung und die Endphase. Für die Phasen sollte eine jeweilige Displayanzeigen beschrieben und eingeordnet werden. Beispielsweise wäre in der Startphase die Anzeige der Dienstbeschreibung und Erläuterung der Geschäftsanwendungsfälle sowie deren Auslösung durch Tastendruck zu beschreiben. In der Vorphase der Routenbegehung sollte der Benutzer den Zielort wählen können. In der Phase der Routenbegehung kann der Benutzer dann mittels der Karte navigieren und verschiedene Interaktionen, siehe Kapitel 5.5 vornehmen. Ebenfalls sollte der Benutzer die Möglichkeit haben die Endphase mittels Tastendruck zu starten.

In [(27)] werden allgemeine Richtlinien zum Design einer Benutzerschnittstelle vorgestellt. Aus diesen sind folgende von Interesse:

- *Konsistente Gestaltung der Bildschirmseite*

  Die Seiten sollten konsistent bezüglich Terminologie (z.B. Tastenbezeichnung), Organisation (z.B. Anordnung von Optionen in einer Auswahlliste) und Verwendung von UI-Elementen (z.B. Verwendung von Wizards und Forms) aufgebaut sein.

- *Vermeidung von Eingabefehlern*

  Eingabefehler, die ein Benutzer vorgenommen hat, sollten möglichst antizipiert werden oder gänzlich unterbunden werden. Auf Texteingabe sollte daher, wenn möglich, verzichtet werden. Lassen sich Fehler nicht abfangen, so sollte dennoch eine aussagekräftige Fehlermeldung auf den Fehler hinweisen.

- *Definition eines Hauptmenüs*

  Ein Hauptmenü sollte stets als Startpunkt der Anwendung gewählt werden.

- *Konsistente Navigation in der Applikation*

  Der Benutzer sollte Klarheit darüber haben, dass er in eine andere Bildschirmseite navigiert ist. Es empfiehlt sich, aus dem Hauptmenü eine Navigation zu allen anderen Teilen der Applikation zu ermöglichen und von allen, zumindest den meisten Bildschirmseiten, eine Navigation zum Hauptmenü anzubieten.

- *Hauptmenüs*

  Anordnung von Elementen nach der Häufigkeit ihrer Verwendung. Häufig benutzte Elemente sollten vor weniger benutzten Elementen positionieren werden. Navigationsmenüs sollten mindestens zwei Elemente beinhalten.

- *Integration eines „Back“-Buttons auf jeder Bildschirmseite*

  Ein „Back“-Button sollte nur dann verwendet werden, wenn genau eine Bildschirmseite zurücknavigiert werden kann.

- *Fehlermeldungen*

  Fehlermeldungen sollten dem Benutzer nach Möglichkeit sofort, klar und deutlich angezeigt werden.

Neben diesen Einschränkungen sollte sich die Gestaltung der Benutzerschnittstelle nach den nicht-funktionalen Anforderungen an die Benutzerschnittstelle des Kapitels 4.5 richten, in denen schon beispielhaft Anforderungen definiert wurden.

Überblick über die Informationsdarstellung

Die Anwendung sollte die Darstellung von großen Listen realisieren können. Große Listen können auf einem Display eines PDA nur unvollständig dargestellt werden, da der Platz nicht ausreichend vorhanden ist. Abhilfe können wiederum Pattern schaffen, mit deren Hilfe die Informationsvisualisierung übersichtlich gestaltet werden kann. Erläuterungen finden sich auf [(69)]. Hier sollen lediglich zwei davon herausgegriffen werden:

- *Search*

  Bei großen Informationsmengen, bei denen ein Scrollen zu unübersichtlich wäre, bietet sich eine Stichwortsuche an. Der Benutzer gibt Stichworte ein und erhält eine reduzierte Ergebnisliste.

- *Alphabetical List*

  Dieses Pattern eignet sich für große alphabetisch geordnete Listen, um diese auf eine ausreichende große Auswahl zu reduzieren.

  Kleinere Listen bis zu drei Seiten können belassen werden, da der Benutzer im Durchschnitt lediglich zwei Seiten besucht, bis er die gewünschte Information erlangt.

  Bei größeren Listen sollte die alphabetische Liste mit dem Search-Pattern kombiniert werden. Der Suchaufwand reduziert sich dann auf eine Seite und drei bis vier Tasten-Drücke.

  Um die Übersichtlichkeit zu erhöhen, wird ebenfalls empfohlen, Gruppierungen der Elemente vorzunehmen. Etwa Gruppierungen der Art „abc“, „def“ usw.

  Der Navigationsgrad sollte nicht größer als zwei sein, um die Übersichtlichkeit zu sichern.

5.5. Gestaltung der Karte

Zur Gestaltung der Karte dienen die funktionalen Anforderungen der Anwendung. Der Entwurf der Karte sollte dabei die Richtlinien des Kapitels 4.5 berücksichtigen.
Zu Beginn der Anwendung sollte darauf geachtet werden, dass nicht zu viel Kartenmaterial heruntergeladen wird. Es kann dadurch zu verlängerten Ladezeiten und Wartezeiten kommen.

Orientierung, Sicht auf die Karte und Ausrichtung der Karte

Punkt zwei (2.) des Kapitels 4.5 berücksichtigt die Selbstbeschreibungsfähigkeit. Um diese dem Benutzer näher zu bringen, sollte er darüber aufgeklärt werden, welche Sicht auf die Etage herrscht. Ebenfalls müsste die Ausrichtung der Karte zu beschreiben werden. Anhand dieser Beschreibung sollte es dem Benutzer stets möglich gemacht werden, sich beispielsweise nach einer Ablenkung wieder zurecht zu finden.

Kartenteilung

Die Kartenteilung sollte, um die Ressourcenbeanspruchung von Speicher und Prozessor des PDA zu minimieren, auf dem Karten-Server stattfinden. Dennoch sollten sie überlappend vorgehalten werden. Zu diesem Schluss kommt man durch Analyse von permanenten Richtungswechseln bei angrenzend dargestellten Kartenteilen. In ungünstigen Fällen könnte es vorkommen, dass bei jeder Bewegung ein Kartenteil nachgeladen werden müsste. Dies würde in der Frequenz erfolgen, in der der Benutzer hin und her wandern würde. Falls das Nachladen zu schnell geschehen würde, könnte das System überlastet werden und es sich nicht mehr performant genug verhalten. Kartenteile sollten daher in einem bestimmten Bereich am Rand der Anzeige überlappend angezeigt werden.

Kartenelemente

Die Anforderung verlangt die Realität so exakt wie möglich auf dem PDA abzubilden und dem Benutzer die bestmögliche Orientierung zu bieten. So sollte es dem Benutzer möglich sein, die Umgebung jederzeit identifizieren zu können. Da aber unter Umständen sehr steril wirkende Umgebungen vorgefunden werden können, beispielsweise lange Flure ohne markante Orientierungsmerkmale, sollte der Service in der Lage sein, auch diese Orientierungslücke zu beseitigen.

Der erste Punkt (1.) des Kapitels 4.5 betrifft die Aufgabenangemessenheit, sie sollte über die Elemente selbst und auch über die Anzahl der Elemente der Karte Berücksichtigung finden. Zur Reduzierung der Elemente, speziell der Landmarken, sollte eine Filterung dieser Elemente erfolgen. Die Filterung stünde dann in Zusammenhang mit der eingestellten Zoomstufe. In jeder Zoomstufe sollten nur bestimmte Landmarken angezeigt werden. Daher sollten die in Kapitel 4.3 des Use-Cases Nr.6 vorgestellten fünf Zoomstufen verwendet werden.

Die Kartenelemente die gleichzeitig der Orientierung dienen sollen werden im folgenden aufgelistet und besprochen.

Raum: Räume sollten die Angabe über ihre Maße und die Position im Gebäude in einer Etage besitzen.

Türen: Türen sollten mit den Informationen über die Position und den angrenzenden Räumen ausgestattet sein.

Route: Die Route sollte den Anfangs und den Endpunkt deutlich darstellen. Ebenso sollte sie sich farbliche vom Hintergrund hervorheben.

Richtungspfeile, Abzweigungshinweise: Die Richtungspfeile sollten vom Benutzer einfach zu interpretieren sein. Sie sollen die Vorwärts-Richtung und das Abzweigen beschreiben. Damit sich der Richtungspfeil eindeutig von anderen richtungsbeschreibenden Komponenten (Linie) abhebt, sollte er in einer Signalfarbe darzustellen. Gleiches sollte für den Abzweigungshinweis gelten. Zudem wird dieser auch im Straßenverkehr und gleichfalls in Outdoor-Navigations-Systemen genutzt, in dieser Hinsicht ist er somit ein häufig gebrauchtes und populäres Hilfsmittel.

POIs: POIs sollen dem Benutzer dazu dienen, beispielsweise eine nächstgelegene Toilette zu finden. Sie sollten durch kleine 2D-Grafiken realisiert werden. Die Klassifizierung der POIs als Strategie zur Anzeige von Informationen sollte ebenfalls berücksichtigt werden. Zum Identifizieren eines POI kann der Benutzer dieses beispielsweise mit einem gelben Punkt anvisieren. Liegt der gelbe Punkt über dem POI oder der Landmarke, so sollte ein Tooltip erscheinen.

Landmarken: Landmarken sind wichtige Hilfsmittel zur Wiedererkennung der Umgebung. Dies ist durch ihre Ähnlichkeit mit Objekten in der Realität begründet. Reale und angezeigte Objekte können somit vom Benutzer durch Vergleichen identifiziert werden. Ist die Anzeige auf dem Display identifiziert, so kann der Benutzer seine Position und Orientierung bestimmen. Hat der Benutzer seine Position in der wirklichen Umgebung festgestellt, so ist er in der Lage, die nächste Richtungsänderung nachzuvollziehen.

Tooltips: Tooltips sind kurze Erläuterungen zu den POIs. Eine kurze Erläuterung besteht minimal aus einem Wort und maximal aus drei Worten. Damit sollen eventuelle Mehrdeutigkeiten ausgeschlossen werden. Eine deutliche Erklärung wäre dann beispielsweise durch eine Beschreibung an einer Tür in der Art *Labor für 'Technik'* oder *Aufzug 'A'* analog *Restaurant, Italienisch* oder *Restaurant, Japanisch* gegeben. Ein Tooltip sollte also ein Name, ein Kurzhinweis, eine Hintergrundinformation oder eine Beschreibung sein. Tooltips sollten in unmittelbarer Nähe des POIs angezeigt werden, beispielsweise über ihnen in geringem Bildpunktabstand.

Geplante Route und zurückgelegter Weg: Der Startpunkt, der Zielpunkt und die Route sollten farblich voneinander unterscheidbar sein. Aus Sicht des Benutzers sollte die Route als Polygonzug erscheinen und sollte mit einem Pfeil, der die Richtung anzeigt,

annotiert werden. Damit eine Unterscheidung zwischen dem bevorstehenden und zurückgelegten Weg möglich ist, sollte dies in verschiedenen Farben dargestellt werden. Der bevorstehende Weg sollte durch eine gelbe Linie angezeigt werden, während der zurückgelegte Weg als rote Linie erscheinen könnte.

Ausrichtung des Gebäudes zur Nordrichtung: Die Ausrichtung des Gebäudes zur Nordrichtung müsste auf der Karte angezeigt werden. Dies sollte in Form eines symbolisch dargestellten Kompasses erfolgen, der zudem in der linken oberen Ecke zu sehen sein sollte.

Maßstab: Ein Maßstab sollte dem Abschätzen der Entfernung dienen. Er könnte in einer 2D-Grafik am unteren Rand der Karte angezeigt werden.

Distanz ins Ziel: Damit der Benutzer den Überblick über die Distanz zum Ziel behält, sollte, ebenfalls am unteren linken Rand der Anzeige, die Distanz in Metern bis zum Ziel angezeigt werden.

Die grafischen Elemente der Anwendung sind benannt und besprochen worden. Da der Benutzer diese teilweise nur in einer bestimmten Orientierung identifizieren kann, sollte diese noch festgelegt werden.

Orientierung der Karte beim Ändern der Bewegungsrichtung

Es ist nicht ausreichend, dass ein Objekt oder eine Karte angezeigt wird, vielmehr spielt das „Wie“ eine tragende Rolle.
Die Orientierung der Karte auf der Anzeige könnte zur schnelleren Orientierung des Benutzers genutzt werden. Da ein Benutzer ständig damit beschäftigt sein würde, das Angezeigte mit der realen Umgebung abzugleichen, könnte es hilfreich sein, die angezeigte Karte in die Blickrichtung zu drehen. Dadurch bliebe das auf der Anzeige vorne Dargestellte auch im Realen vorne.
Würde die Karte starr eingenordet bleiben, wäre „vorn“ plötzlich „rechts“ oder „links“ oder „hinten“. Dies ist abhängig von der eingeschlagenen Richtung. So würde der Benutzer nach dem Rechtsabbiegen die Route auf der Anzeige nach rechts gerichtet sehen. Dies könnte zu Verwirrungen führen, die es zu vermeiden gilt. Der Benutzer könnte sich zwar durch Drehen des PDA Klarheit verschaffen, dies erscheint aber eher unkomfortabel und unzumutbar.

Zur Drehung der angezeigten Karte und der darauf befindlichen Objekte sollten Bezugspunkte und Winkel herangezogen werden. Der erste Bezugspunkt könnte entweder der Startpunkt der Route oder ein Knickpunkt in der Route sein. Ein Knickpunkt stellt ein Punkt dar, an dem die Route eine Richtungsänderung erfährt. Den benötigte Winkel könnte man unter Zuhilfenahme eines Bezugs-Koordinatensystems und der Verwendung eines weiteren Bezugspunktes ermitteln. Dieser zweite Bezugspunkt sollte der nächste Knick einer Route sein. Dieser

zweite Knickpunkt wäre vergleichbar mit einem Punkt, den der Benutzer in der virtuellen Welt anvisiert, wenn er sich entlang einer Route bewegen würde.

Das Bezugs-Koordinatensystem hat seinen Ursprung in der linken oberen Ecke, d. h. die Ordinate verläuft von links oben nach rechts unten; die Abszisse verläuft entsprechend senkrecht dazu, von links oben nach rechts oben. Möchte man nun erreichen, dass die Anzeige die Route immer vom unteren Rand zum oberen Rand hin anzeigt, so dass der Benutzer „oben“ auf der Anzeige mit dem „Geradeaus“ in der realen Welt interpretieren kann, sollte man zunächst den Winkel im Schnittpunkt zwischen parallel verschobener Ordinate und dem Vektor $\vec{AB}$ liegen, siehe Abbildung 5.3. Um diesen Winkel und den Bezugspunkt A werden die Objekte der Anzeige schließlich gedreht. Durch die Drehung soll erreicht werden, dass der Vektor $\vec{AB}$ parallel zur Ordinate steht. Die Drehung eines Punktes wie sie bei der Umsetzung benötigt werden könnte entspricht folgender Formel (je nach Drehsinn):

$$\boldsymbol{Dr}(\alpha)B = \boldsymbol{Dr}(\alpha)\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} xcos\alpha - ysin\alpha \\ xsin\alpha + ycos\alpha \end{pmatrix}$$

$$\boldsymbol{Dr}(\alpha)B = \boldsymbol{Dr}(\alpha)\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} xcos\alpha + ysin\alpha \\ xsin\alpha - ycos\alpha \end{pmatrix}$$

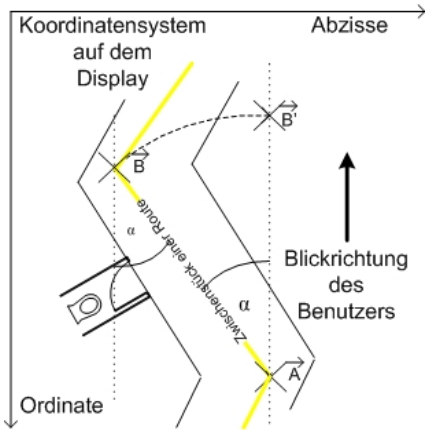


Abbildung 5.3.: *Drehung der Karte beispielhaft erklärt.*

Farbe verschiedener Kartenelemente

Unterschiedliche Kartenelemente sollten in unterschiedlicher Farbe angezeigt werden. Bei der Wahl der Farben ist auch auf Personen mit einer Farben-Seh-Schwäche zu achten. Es werden daher auch nur sehr konträre Farben verwendet:

- Der Hintergrund sollte in weißer Farbe dargestellt werden.
- Räume könnten in grauer Farbe darzustellen werden, in Anlehnung an die Farbe des Baumaterials Beton bzw. Stein.
- Der zurückgelegte Weg sollte in blauer, der ausstehende Weg in roter Farbe dargestellt werden.
- Türen können ebenfalls in schwarzer Farbe dargestellt werden.
- Landmarken können in orangener Farbe verwirklicht werden.
- Tooltips sollten in schwarzer Farbe dargestellt werden, da diese sich vom Hintergrund sehr stark abhebt.
- Es sollte eine farbliche Unterscheidung zwischen Räumen und Durchgängen, sowie Treppenhäusern zu erkennen sein.

Interaktionen mittels der Tastatur

Erforderlich sind Interaktionen beispielsweise beim Verschieben der Karte, beim Zoomen der Anzeige oder der Eingabe des Zielortes. Da sich neben diesen auch andersartige Interaktionen ereignen, sollte vor deren Erläuterung eine Trennung stattfinden.
Während der Anwendung finden zwei verschiedene Interaktionsarten statt. Die grafische Interaktion und die Interaktion mittels Komponenten:

Grafische Interaktion

- Scrollen

  Mittels der vier Cursor-Tasten (links, rechts, oben, unten) sollte es dem Benutzer freistehen, die Karte eine dieser Richtungen zu verschieben. Der erwähnte gelbe Punkt bleibt dabei stets in der Mitte der Anzeige. Würde nach dem Scrollen wieder die Begehung der Route erfolgen, so sollte auch die Position wieder aktualisiert werden. Dies hätte zur Folge, dass die Karte wieder den Ausgangspunkt, dem vor dem verschieben, anzeigen würde.

  Mittels Scrollen sollte ebenfalls das Anvisieren von POIs und Landmarken mit dem gelben Punkt ermöglicht werden. Dadurch könne die oben erwähnten Tooltips aktiviert werden. Der gelbe Punkt sollte auch vor dem Ändern der Zoomstufe genutzt werden. Mit diesem könnte der neue Mittelpunkt der Anzeige festgelegt werden. Die Karte der Anzeige könnte dann um dieses (neue) Zentrum herum vergrößert oder verkleinert werden.

- Zoomen

  Es soll einen Kartenausschnitt vergrößern oder auch verkleinern. Damit sollte das Zoomen einen zusätzlichen Anteil zur Erkundung der Umgebung auf der Anzeige beitragen. Da das Zoomen auf jedes Kartenteil angewendet werden sollte, hätte der Benutzer somit Gelegenheit auch abseits der Route gelegene POIs zu untersuchen.

  Das Zoomen im Zusammenspiel mit dem gelben Punkt würde es dem Benutzer ermöglichen Informationen über seine Umgebung zu beschaffen bzw. abzufragen.

Interaktion mittels Komponenten

- Listengesteuerte Auswahl der Zielorte

Neben den beiden grafischen Interaktionsmöglichkeiten zur Manipulation der Anzeige sollte es auch die Möglichkeit der Interaktion über Komponenten geben. Diese Interaktionsart soll eine weitere Nutzungsmöglichkeit des Dienstes, die listengesteuerte Auswahl, darstellen. Sie sollte dem Benutzer helfen, die auf der Etage liegenden Zielorte auszuwählen. Diese Auswahl müsste über die Tastatur getätigt werden. Die listengesteuerte Auswahl würde dazu dienen, langwierige Tastatureingabe zu verkürzen und eventuelle Eingabefehler zu vermeiden. Die listengesteuerte Auswahl hätte nur Erfolg, wenn der Zielort eindeutig bekannt wäre oder dieser in der Liste erscheint. Neben einer komfortablen logischen Programmführung würde der Benutzer somit auch über Interaktionsmöglichkeit verfügen.

Platzierung verschiedener Elemente auf der Anzeige

Zusätzliche Elemente[3], wie die Nordrichtung oder der Maßstab, sollten, falls möglich und sinnvoll, an den Rändern (oben, unten, links oder rechts) der Anzeige dargestellt werden. Ein zusätzliches Element, das nicht am Rand, sondern unmittelbar in wenigen Pixeln Abstand am Objekt platziert werden sollte, wäre ein Tooltip. Bei einer Platzierung in der Ecke, entstünde der Nachteil, dass der Tooltip evtl. nicht zuordenbar wahrzunehmen wäre. Von Vorteil wäre allerdings, dass die Sicht auf andere Elemente nicht verdeckt würde. Damit ein Tooltip direkt einem Element zugeordnet werden kann, sollte er in geringem Abstand über diesem Element angezeigt werden.

[3] Als zusätzliche Elemente sollten diejenigen dienen, die in der Zusammenfassung des Kapitels 4 festgelegt wurden.

Tasten-Belegung

Damit der Benutzer Interaktionen vornehmen kann, um Anwendungsfälle auszulösen, sollten diese aktivierbar sein. Dies sollte infolge eines Tastendruckes auf der Benutzerschnittstelle erreicht werden. Zur eindeutigen Verwendung sollten daher am unteren Bildschirmrand, links und rechts über den Tasten, Beschriftungen angezeigt werden, die in Verbindung mit der Funktionalität der unmittelbar darunterliegenden Taste gebracht werden können. Folgende Tasten-Belegungen sind möglich:

- *Tasten-Belegung während des Hauptmenüs*

  Der Benutzer sollte das Hauptmenü durchscrollen können. Dazu sollten zwei Tasten zum Auf- und Abwärtsscrollen dienen. Ebenfalls sollte es eine Taste ermöglichen zur Zielortwahl zu navigieren.

- *Tasten-Belegung während der Zielortwahl*

  Zu Beginn der Anwendung sollte die Auswahl des Zielortes aus einer Liste erfolgen. Um in dieser Liste nach einem Zielort zu suchen, sollten zwei Tasten verwendet werden. Eine Taste, mit der in der Liste einen Zielort oberhalb des aktuell markierten auswählen kann sowie eine Taste, mit der man den Zielort unterhalb des aktuellen Zielortes markieren kann.

  Zur Bestätigung eines Zielortes, sollte eine Taste mit „Berechnen“ benannt werden. Diese löst gleichzeitig die Routenberechnung aus. Die Routenberechnung wird in Gang gesetzt, ohne dass der Benutzer eine erneute Eingabe zu tätigen hat.

  Ebenfalls sollte der Benutzer wieder die Möglichkeit haben zum Hauptmenü zurückzukehren.

- *Tasten-Belegung während des Navigierens*

  Während der Navigation sollten dem Benutzer ebenfalls die Möglichkeit zur Wahl eines Anwendungsfalles gegeben werden. Daher sollten wiederum entsprechende Tasten definieren werden, die einen solchen Anwendungsfall auslösen sollen. Im Nachfolgenden sind die Beschriftungen aufgelistet, die auf der Anzeige zu sehen sein sollten:

  – *ZoomIn/-Out*

    Das Zoomen sollte anhand zweier Tasten ermöglicht werden. Eine Taste, die dem Verkleinern der Anzeige dient, und eine andere Taste zum Vergrößern der Anzeige.

- *Neues Ziel*

  Damit der Benutzer einen anderen Zielort wählen kann, sollte eine Taste vorhanden sein, die dies ermöglicht.

- *Beenden*

  Während sich der Benutzer anhand der Anzeige orientiert und sich auf dem Weg zum Zielort befindet, sollte er stets die Möglichkeit haben den Dienst zu beenden. Daher ist eine Taste mit dieser Funktionalität zu belegen. Die Funktionalität sollte mit „Beenden“ benannt werden.

- Scrollen

  Zum verschieben der Karte sollten die vier Cursor-Tasten verwendet werden.

Zusammenfassung

Das Kapitel Grobentwurf beschäftigte sich mit der Gestaltung zu den folgenden Themengebieten:

- Client-Server-Architektur

  Die Client-Server-Architektur baut auf einem von fünf vorgestellten Konzept von (56) auf. In diesem ist ein Rich-Client beschrieben, der ein Teil der Anwendung übernimmt, Fehlerausgaben vornimmt und auf Benutzereingaben reagiert. Ein solcher Client sollte auf der Präsentationsschicht realisiert werden. Auf Anwendungsebene sollten ein Karten-Server und ein Empfangsmodul angeschlossen werden, um einerseits die Route bereitzustellen und andererseits die Position ermitteln zu lassen. Auf der Persistenzschicht sollte ein weiterer Server zum bereithalten sämtlicher Karten und Kartenelementen bereitstehen.

- Gestaltung der Anwendungsarchitektur

  Die Anwendungsarchitektur sollte sich auf dem MVC-Pattern nach (21) aufbauen. Zu diesem Zweck wurde das Pattern beschrieben.

- Gestaltung der Benutzerführung

  Da die Benutzerführung dafür sorgen muss, dass die Anwendung übersichtlich erscheint, wurden in diesem Abschnitt verschiedene Techniken und Pattern besprochen, welche die Anwendung komfortabler erscheinen lassen sollen.

- Gestaltung der Karte

  Da der Schwerpunkt der Anwendung auf dem erstellen der Karte liegt, wurde diese Thema aufgegriffen und ausführlich besprochen.

Im nächsten Kapitel sollen noch einzelne bisher nicht berücksichtigte Entwurfsentscheidungen getroffen werden und darauf folgend mit den Ausführungen zur Realisierung begonnen werden.

6. Feinentwurf und Realisierung

In diesem Kapitel soll mit den im Kapitel 5 „Entwurf“ beschriebenen Gestaltungstechniken und mittels geeigneten Hard- und Software die Anwendung umgesetzt werden. Zu diesem Zweck erfolgt zu Anfang die vertiefte Betrachtung der zu verwendenden Hardware, auf der die Anwendung laufen soll. Nachdem die Software, die zur Erstellung des Codes dient, festgelegt ist, kann begonnen werden, die Anforderungen des Kapitels 4 umzusetzen und damit den Entwurf zu realisieren. Die Auswahl der Programmiersprache ist Gegenstand des Kapitels 6.4. Der realisierte Entwurf soll mit Abschluss des vorliegenden Kapitels einen Prototypen hervorbringen, mit dem es möglich ist, eine simulierte Indoor-Navigation durchzuführen. Zur Erfüllung der geforderten Funktionalitäten wurde insbesondere auf die in den Kapiteln 4.4 und 4.5 aufgestellten und geforderten nicht-funktionalen Anforderungen implizit Bezug genommen. Im letzten Abschnitt der Realisierung erfolgt die Besprechung einzelner Anwendungsfälle anhand von Objekt-orientierten-Modellen, verschieden Sequenzdiagramme der Geschäftsanwendungsfälle aus Kapitel 4.3.

6.1. Hardware

Die zur Erstellung der Anwendung benötigte Hardware, setzt sich aus drei verschiedenen Komponenten zusammen: Zum einen die Hardware, um das Indoor-Navigationssystem anzuzeigen, zum anderen die Hardware zur Erstellung des Codes der Anwendung. Die Infrastruktur zur Ermittlung und Weiterleitung der Positionsdaten ist die dritte Komponente. Die Hardware zur Darstellung wird im folgenden Kapitel erläutert. Zur Erstellung des Codes diente ein handelsüblicher Laptop.

Die in der Motivation angesprochene Hardware, ein PDA, dient der Anzeige von Karten. Das verwendete Referenz-Modell ist das Smartphone Nokia E70-1 [(44)] und ist in Abbildung 6.1 zu sehen. Das Nokia E70-1 unterstützt zu dem auch die erforderlichen APIs JSR-75, -82, -172 [(36)], die bereits in Kapitel 2.4 besprochen wurden.

Bei den verwendeten Geräten handelt es sich nicht automatisch um die bestmöglich gewählten Ressourcen. Vielmehr dient die Zusammenarbeit aller Geräte zum Zweck der Erschaffung einer innovativen Technik unter Verwendung handelsüblicher Mittel.

Einschränkungen durch die Verwendung des PDA als Anzeigekomponente

Für die sich aus der Analyse ergebenden Anforderungen an die Anwendung folgen durch die Verwendung eines PDA Einschränkungen aufgrund dessen technischer Eigenschaften. Die Einschränkungen sind Folgen der Anpassung der Größe und Aussehen eines PDA an menschliche Bedürfnisse. Daher müssen vor der Realisierung die funktionalen, nicht-funktionalen Anforderungen und die Einschränkungen auf den kleinsten gemeinsamen Nenner gebracht werden. Funktionale und nicht-funktionale Anforderungen sind nur über die technischen Eigenschaften des PDA zu realisieren.

Einschränkungen bestehen gegenüber PCs oder Laptops. Diese werden zwar schon in den Profilen bzw. den Konfigurationen berücksichtigt, dennoch ist auch bei der Entwicklung des Codes darauf Rücksicht zu nehmen. Mehr zu diesem Thema in Kapitel 6.4 Software.

Abbildung 6.1.: *Das Referenz-Model: Nokia E70-1*

Ressourcen dienen der Sicherung der Funktionalität. Zu den Ressourcen des PDA gehören dessen Interaktionsmöglichkeiten, der Akkumulator, die Netzwerkschnittstelle, der Speicher, die Darstellung und die Prozessorleistung. Einschränkend sollte gesagt werden, dass auf die Prozessorleistung und Akkumulator nicht direkt Bezug genommen wird, da diese in den Erläuterungen der Ressourcen Speicher, Netzwerkschnittstelle und Darstellung bereits Berücksichtigung finden.

Die Interaktionsmöglichkeit zwischen Anwendung und PDA ist allgemein durch die Ressource Tastatur eingeschränkt. Die Tastatur ist relativ klein und begrenzt auf 20 Tasten. Daher

ist die Zeichen-Menge des Alphabetes nur durch eine Mehrfachbelegung der Tasten umsetzbar. Infolge dessen erschwert sich die Eingabe von Informationen. Daher wird auf eine Texteingabe verzichtet, um die Anforderung der Bedienerfreundlichkeit zu erhalten.
Die geforderte Mehrfachbelegung lässt sich jedoch nicht vermeiden. Ausgangspunkt für diese Entscheidung ist, dass es keine Möglichkeit gibt, Tasten, die nicht unmittelbar am Anzeigenfeld (Tasten des übrigen Bedienfeldes) liegen, für den Benutzer kenntlich zu machen. Wenn diese Tasten berücksichtigt werden würden, müssten sie in der Einführung erklärt werden. Dann bestünde aber die Möglichkeit, dass der Benutzer dies später vergäße und zurück zur Erklärung navigieren müsste. Daher wurden ausschließlich die Kommandotasten, die sich direkt unter der Anzeige befinden, mit mehreren Kommandos belegt. Die Kommandos sind in einer vertikalen Liste enthalten und mittels Auf- und Abwärts-Taste zu wählen.

Eine weitere Interaktionsmöglichkeit, die der Verbindung eines PDA mit einem weiteren Gerät dient, ergibt sich durch die Netzwerk-Verbindung. Diese Netzwerk-Verbindungen können unterschiedliche Bandbreiten besitzen. Sinnvoll für diese Anwendung ist Bluetooth, da sie sowohl, von der Hardwareseite als auch von der Softwareseite unterstützt wird. Bluetooth soll zur Kommunikation von PDA und IMAPS-Empfangsmodul genutzt werden.

Die Ressource Speicher wird in einem separaten Kapitel berücksichtigt, da sie unter anderem eine strategische Rolle übernimmt.

Speicher für Caching und Kartenteilung

Infolge der Ressourcenbeschränkung welche die Bandbreite betrifft, sollte eine Strategie entwickelt werden, diese zu schonen. Diese Strategie sollte darüber hinaus auch der angesprochenen Effizienz, Kapitel 4.4, der Anwendung dienen. Mittels lokalem Caching sollte es daher ermöglicht werden, die an den aktuellen Kartenausschnitt angrenzende Kartenteile zu speichern, um diese bei Bedarf aus dem Speicher zu entnehmen und erneut darzustellen.

Der Speicher soll daher in drei verschiedenen Situationen zum Cachen von Daten gebraucht werden:

Caching von Karten im XML-Format

Das Ablegen von Roh-Daten im File-System des PDA ist notwendig, um das Vorhalten von Kartenteilen in der ursprünglichen Form, dem XML-Format, zu ermöglichen. Aus diesem Format werden die Kartenteile nur bei Bedarf durch die Anwendung erstellt. Dadurch kann die CPU geschont und der Energieverbrauch, infolge der Schonung der Netzwerkschnittstelle, vermindert werden. Die Notwendigkeit einer solchen Caching-Strategie ergibt sich beispielsweise durch Richtungsänderungen des Benutzers, bei der der Benutzer zwischen zwei sich überlappenden Kartenteilen hin und her laufen

könnte. Zur Speicherung der Karten im File-System des PDA dient die File-Connection API. Diese wurde bereits in Kapitel 2.4 besprochen.

Caching von Karten im konvertierten Format

Das Caching von Daten dient unter anderem zu deren Speicherung während der Ausführung der Anwendung. Caching könnte ebenfalls bei einem Systemausfall der Anwendung oder Netzwerk-Unterbrechungen notwendig sein. Dadurch könnte bei einem Neustart der Anwendung der vorherige Zustand aus den gespeicherten Karten-Daten, Routen-Daten und Standort-Daten rekonstruiert werden. Das Caching könnte somit dafür sorgen, dass die Anwendung in einen konsistenten Zustand, dem vor der Unterbrechung, geführt wird.

Ebenfalls sollte beim Speicher von Instanzen die Strategie des Caching verfolgt werden. Diesmal sollte das lokale Caching über Instanzen von java.util.Vector ermöglicht werden. In diesen sollten Instanzen von Room.java, Door.java, Route.java und POI.java gehalten und zur späteren Verwendung bereitgestellt werden. Die Instanzen dienen dem Aufbau der Karte.

6.2. IMAPS und Innenraum Routenplaner

Zur Positionsermittlung dient das IMAPS-Modul [(28)], welches Positionsdaten selbstständig errechnen kann. Das Konzept der Kommunikation und der Positionsbestimmung wurde in Kapitel 2.3 vorgestellt. Der Aufbau der Beacon ist dezentral, d.h. es besteht keinerlei Verbindung der Beacon untereinander. Dieser Aufbau dient dem Schutz der Privatsphere, der zur Konzeption des IMAPS-Systems gehört. Zur Verbindung des IMAPS-Empfangs-Moduls, siehe Abbildung 6.3 mit dem PDA soll Bluetooth dienen. Über diese Verbindung kann das errechnete Koordinaten-Paar unidirektional übertragen werden. Dazu muss das IMAPS-Modul auf Bluetooth umgerüstet werden. Das dies möglich ist, wurde bereits in Kapitel 2.3 festgehalten. IMAPS verfügt über eine Java-Schnittstelle. Die Schnittstelle gliedert sich, wie in Abbildung 6.2 zu sehen ist, in 3 Bereiche. Der erste Bereich der Schnittstelle ist der DatenWrapper - Modul. Dieser besteht aus einem Handler für die über die Bluetooth-Schnittstelle ankommenden Daten. Der DatenWrapper verwaltet die komplette Kommunikation des Gerätes auf dem die Schnittstelle läuft und dem Listener. Sobald ein Positionsdaten-Paket empfangen wurde, wird dieses an das PackageHandling weiter gereicht. Dieser Bereich der Schnittstelle extrahiert alle notwendigen Daten aus dem seriellen Paket und aktualisiert den internen Datensatz mit allen bisher empfangenen Daten.

Der dritte Bereich ist das Position-Modul. Dieses Modul wertet die intern gespeicherten Datensätze aus und errechnet die symbolischen oder physikalischen Positionsdaten. Für die

Berechnung der physikalischen Positionsdaten verwendet dieses Modul das Jama Package [(31)].

Bluetooth API

Diese API dient der drahtlosen Datenübertragung über kurze Distanzen mittels Bluetooth [(41)], vergl. Kapitel 2.4. Durch den Umstand, dass das IMAPS-Modul auf dem PDA angebracht wird, folgt ein kurzer Signalübertragungsweg (< 5cm). Die API eignet sich also für den Empfang und die Umsetzung der Signale.

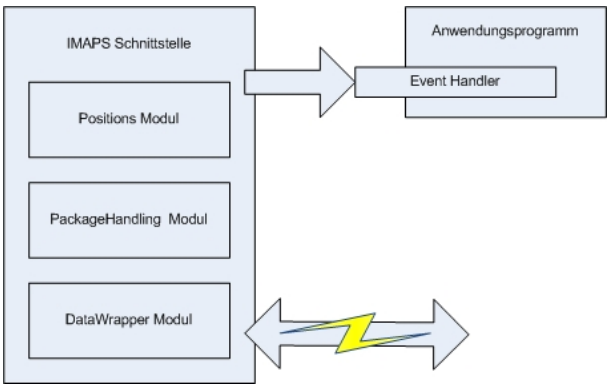


Abbildung 6.2.: *Aufbau der Java-Schnittstelle*

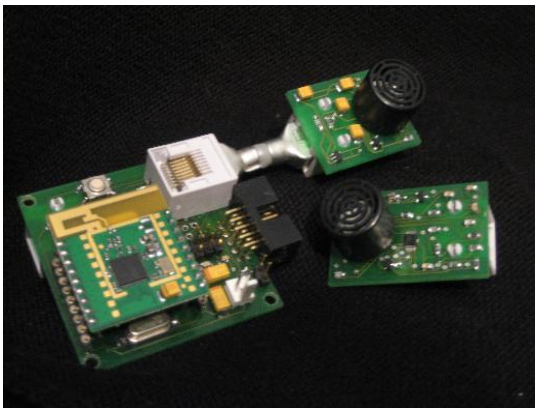


Abbildung 6.3.: *Hauptplatine mit verbundenem Ultraschallsendemodul und Ultraschallempfangsmodul*

Innenraum Routenplaner

Der LBS dieser Anwendung ist der in Kapitel 3.1 vorgestellte Innenraum-Routenplaner von (40). Um Zugriff auf die Methoden zu erlangen, sollen auf dem PDA Instanzen eines sog. Client-Stubs des Services erzeugt werden.

Um nun auf den Web-Service vom PDA aus zuzugreifen zu können, muss dieser das Web-Service API implementieren. Diese API und seine Funktion wurde bereits in Kapitel 2.4 vorgestellt.

Web-Services

Allgemein ist ein Web-Service [(66)] eine Software-Anwendung, die mit einem Uniform Ressource Identifier (URI) eindeutig in einem Netzwerk identifizierbar ist und deren Schnittstellen als XML-Artefakte definiert, beschrieben und gefunden werden können [(8)].Ein Web Service macht eine Anwendung plattformunabhängig und interoperabel. Dazu nutzt er für die Datenübertragung SOAP-Nachrichten [(64)] die in das http-Protokoll eingebunden sind. Die Beschreibung des Web-Service erfolgt über die Web Service Definition Language (WSDL). In dieser sind sowohl Schnittstellen und Datentypen definiert, als auch das Aussehen der auszutauschenden Nachrichten, welches standardisiert ist. Clients können aufgrund der unabhängige Beschreibung der Schnittstellen in verschiedenen Programmiersprachen entwickelt werden. Dadurch entsteht der Vorteil für heterogene Systeme verwendbar zu sein.

6.3. Karten und Darstellung

Die Karten werden vom Karten-Server, Innenraum Routenplaner, bezogen; Kapitel 6.2. Die Darstellung der Karte nach dem Start der Anwendung sollte dem Benutzer genügend Anhaltspunkte geben, so dass er sich einfach orientieren kann. Daher ist eine geeignete Größe dieser Karte festzulegen. Für die Festlegung der Größe eignet sich zunächst die Annahme, dass ein Benutzer alles in einem Umkreis von 10m gut erkennen kann. Daraus ergibt sich ein Anzeigenareal von ca. 100qm. Darüber hinaus wird die Karte in einer geeignete Darstellung der im Design Kapitel, Kapitel 5.5, vorgestellten Elemente realisiert.

Der gelbe Punkt, der zur Erkundung der virtuellen Etagen-Landschaft dienen soll, könnte die die Ressource Rechenleistung sehr stark beanspruchen, da bei jedem Versetzen ein Neuzeichnen der Elemente der Karte veranlasst wird. Gleiches ereignet sich auch bei der Bewegung des Benutzers. Daher sollte für den Versatz des gelben Punktes die Schrittweite pro Zoomstufe um drei Pixel zunehmen. Bei einer geringen Schrittweite würde die Anzeige zu oft neu gezeichnet. Die Schrittweite sollte aber auch nicht zu groß gewählt werden, sonst können kleinere Objekte mit dem gelben Punkt nicht anvisiert werden.

Drehung der Karte und der dargestellten Objekte

Aufgrund dessen, dass in einem Gebäude eine Navigationslandschaft, die aus rechtwinkligen Räumen und Fluren besteht, vorzufinden ist, ist eine Drehung im Sinne von wenigen Grad-Einheiten nicht sinnvoll. Wohl aber ist es sinnvoll Karten eines Gebäudes in $90°$-Schritten zu drehen. Dies entspricht wiederum der Rechtwinkligkeit der Räume und Flure und erscheint auch im Sinne der Drehung des Benutzers während der Richtungsänderung

am geeignetsten. Durch diese Vorgehen erkennt der Benutzer auch die Richtungsänderung in Bezug auf die aktuelle und die vorherige Sicht vor der Drehung. Diese Richtungsänderung würde er beim Verharren der Karte im Ausgangszustand nicht erkennen können.

Entsprechend der Drehung der Räume und Flure würde sich auch die Anzeige des Kompass und die Ausrichtung der Landmarken ändern. Auf diese Weise könnte dem Benutzer der Bezug zur Ausrichtung des Gebäudes und seiner Objekt erhalten bleiben. POIs bleiben von der Ausrichtung verschont.

6.4. Software

Die Software, die verwendet wird, besteht aus der Entwicklungs-Software zur Erstellung der Anwendung und eine Softwareumgebung, auf der die Anwendung ablaufen kann. Da die Anwendung letztendlich nicht nur auf dem Entwicklungsrechner laufen soll, ist eine Softwareumgebung nötig, welche die Entwicklung von der Nutzung trennt. Diese Voraussetzung wird durch die Forderung, dass die Anwendung auf verschiedenen mobilen Endgeräten unterschiedlicher Hersteller laufen sollte, verstärkt.

Software zur Entwicklung der Anwendung

Um der geforderten Übertragbarkeit aus Kapitel 4.4 zu entsprechen, fällt die Entscheidung auf J2ME, die unabhängig von der Betriebssystem-Plattform eingesetzt werden kann. J2ME ist an keine Gerätekonfiguration gebunden und erfüllt somit das Kriterium der Übertragbarkeit. Ein weiterer Grund zur Wahl von J2ME besteht in der großen Verbreitung der Laufzeitumgebung auf mobilen Geräten.

Die Client-Anwendung dieser Arbeit stützt sich auf die CLDC, Kapitel 2.4. Zur Entwicklung der Client-Anwendung wird das *Mobile Information Device Profile* (MIDP) genutzt. Der Vollständigkeit wegen soll noch erwähnt werden, dass auch das optionale Paket, JSR-82 der Bluetooth API, verwendet wird, Kapitel 6.2. Die CLDC und die MIDP stellen gemeinsam folgende Klassenbibliotheken zur Entwicklung bereit. Diejenigen, die hinter dem Semikolon stehen werden in der Anwendung verwende:

java.lang: Fundamentale Klassen;
Object, Thread, String, Boolean, Byte, Integer, Character.

java.util: Kollektionen, Datum und Uhrzeit, Zufallsgenerator;
Vector, Hashtable.

java.io: Stream- und Reader-Klassen für Ein- und Ausgabe;
ByteArrayInputStream.

javax.microedition.io: Generic Connection Framework;
InputConnection, Connector.

javax.microedition.midlet: Grundfunktionalität für Applikationen, Steuerung des Lebenszyklus;
Midlet

javax.microedition.lcdui: Dient der Implementierung von Bedienoberflächen;
Display, Displayable, CommandListener, Command, Canvas

Es sind noch fünf weitere Klassenbibliotheken des MIDP vorhanden. Diese wurden aber zur Entwicklung der Anwendung nicht genutzt und finden daher keine Berücksichtigung.

Anhand der verschiedenen Bibliotheken wird die Funktionalität technischer Komponenten und Klassen implementiert und somit die Anforderungen aus dem Kapitel 4 und die Kartenelemente aus Kapitel 5.5 programmiertechnisch umgesetzt.

Verwendete Klassenbibliotheken zur Gestaltung der Karte und der Benutzerschnittstelle

Die Benutzerschnittstelle wird mittels der API *javax.microedition.lcdui* entwickelt. Die API teilt sich in zwei Kategorien zur User-Interface Programmierung:

- High-Level APIs
- Low-Level APIs

Die High-Level API stellt vorgefertigte Elemente zur Gestaltung der Benutzerschnittstelle bereit, darunter beispielsweise Listen, Forms und Textfelder. Die Nutzung dieser Elemente bietet die beste Möglichkeit, die geforderte Übertragbarkeit der Applikation von Gerät zu Gerät zu unterstützen, ohne diese zu modifizieren. Aus diesem Grund ist die High-level API die beste Möglichkeit eine konsistente Benutzerschnittstelle zu kreieren.

Die Low-Level API kann eine Übertragbarkeit zwischen Geräten nicht garantieren. Dennoch hat diese API positive Eigenschaften, beispielsweise durch das An-und Ausschalten einzelner Pixel. Somit ist es möglich, viele verschiedene grafische Elemente auf der Anzeige darzustellen, ohne sich an die Vorgaben der High-Level-API zu halten. Aber genau aus diesem Grund muss sich der Entwickler schließlich auch um die Übertragbarkeit kümmern.

6.5. Indoor-Navigations-Client

Bevor der Indoor-Navigations-Client realisiert werden kann, wird aufgezeigt, mit welchen Techniken dies zusätzlich geschieht.

Entwurfsmuster

Zur Entwicklung des Codes der Anwendung werden folgende Design Patterns eingesetzt:

- *Singleton*

  Das Singleton-Pattern begrenzt durch Privatisierung des Klassen-Konstruktors die Anzahl an Instanzen, die erzeugt werden können. Somit müssen zur Erzeugung von Instanzen ausgezeichnete Methoden herangezogen werden. Eine Methode liefert dann entweder ein existierendes Exemplar zurück oder erzeugt ein neues. Durch das Singleton-Pattern wird vermieden, dass mehrere Exemplare des Model erzeugt werden.

- *Model-View-Controller*

  Das MVC-Pattern dient der funktionalen Unterteilung der Anwendungsarchitektur und wurde in Kapitel 5.3 vorgestellt. Mittels dieses Pattern wurde eine erste Unterteilung der Anwendung in Komponenten vorgenommen.

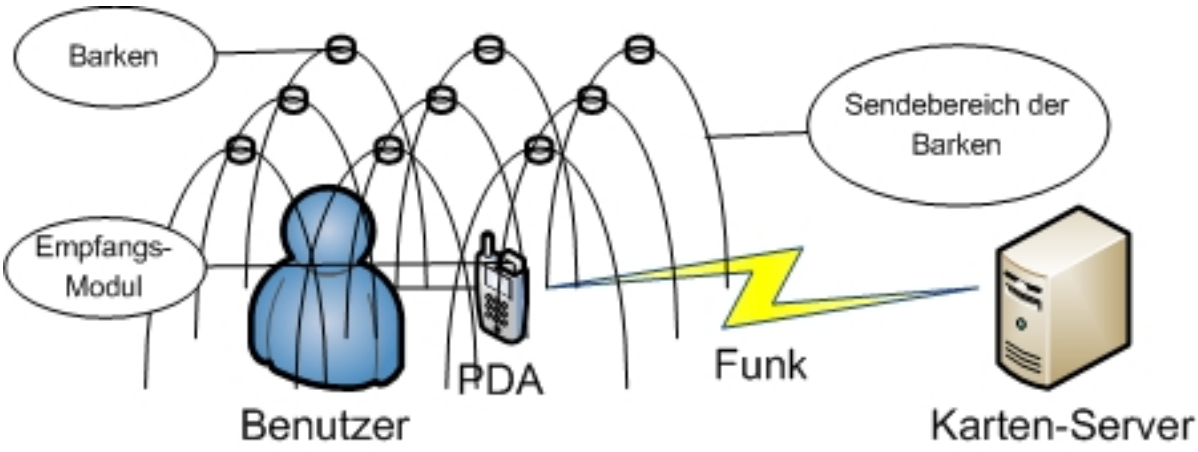


Abbildung 6.4.: *Das Gesamtsystem, welches das Indoor-Navigationssystem darstellt.*

Die Abbildung 5.3 zeigt die durch Übertragung der Funktionalität des MVC-Pattern zustande gekommenen technischen Komponenten *view* und *controller*. Dies resultiert daraus, dass der PDA sowohl ein Bedienelement, als auch ein Anzeigeelement zur Interaktion mit dem

Benutzer zur Verfügung stellt. Das *view* steht für die Anzeige, der *controller* für das Bedienelement. Die Komponente *view* und *controller*. Dies resultiert daraus, dass der PDA sowohl ein Bedienelement, als auch ein Anzeigenelement zur Interaktion mit dem Benutzer zur Verfügung stellt. Die Komponenten *view* und *controller* sind letztenendes wieder in einer Instanz der Klasse *javax.microedition.midlet.Midlet* vereint, siehe Kapitel 6.5.

Das verwendete MVC-Pattern unterstützt zu dem die geforderte Änderbarkeit aus Kapitel 4.5, auf die auch im Rahmen einer späteren Erweiterung oder Verbesserung Rücksicht genommen werden muss.

Die *view* wird durch das API *javax.microedition.midlet.MIDlet* implementiert. Das API hat die Aufgabe neben der Darstellung des *model* auf der Anzeige, den Lebenszyklus der Anwendung zu steuern. Um die Benutzereingaben abzufangen, implementiert das Midlet das Interface *javax.microedition.lcdui.CommandListener*. Somit ist sichergestellt, dass, falls ein Benutzer einen *javax.microedition.lcdui.Command* in Form eines Tastendrucks abgibt, der *controller* diesen abfängt und weiterverarbeiten kann.

XML-Parsing

Die vom Innenraum Routenplaner, Kapitel 6.2, zu übertragenen Karten werden in Form von XML-Dateien bereitgestellt. Sie werden anhand des in Anhang A.2 vorgestellten Verfahrens (pseudo-code) geparsed, um daraus Objekte zu erzeugen, die der während die der Anwendung dienen. Mit dem Ansatz, die XML-Dateien durch einen eigenen Algorithmus zu parsen, wurde versucht, die Ressourcen des Clienten weiter zu schonen. Daher wurde auch auf die Verwendung eines SAX- oder DOM-Parser verzichtet, die bei der Verwendung zusätzliche Objekte erzeugen würde.

Parallelverarbeitung

Die Parallelverarbeitung wird in J2ME mittels eines Threads ermöglicht. Anhand dessen ist es möglich, einen quasi-parallelen Ablauf von einzelnen Prozessen im Hintergrund zu erreichen. Vorteilig ist dabei, dass diese Threads den Ablauf der Applikation nicht unterbrechen.

Da die Aktualisierung der Position des Aufenthaltsortes es verlangt, diese vom Empfangsmodul zu beziehen, kann die Anwendung in dieser Zeit nicht gestoppt werden. Diese Situation würde die geforderte Benutzerfreundlichkeit herabsetzen. Durch Anwendung des Thread-Models ist dies auch nicht nötig.

Aufenthaltsort

Der Aufenthaltsort wird mittels der Thread-Funktionalität von Model.java, Kapitel 6.5, realisiert. Dazu pollt[1] der Thread nach einem Koordinatenpaar, dass vom IMAPS-Modul bereitgestellt wird. Das Polling erfolgt in periodischen Zeitabständen.

Hat der Thread ein Koordinaten-Paar erhalten, so werden die alten Positionsdaten mit den neuen überschrieben und anschließend die *paint()*-Methode mittels *repaint*() aufgerufen. Der aktuelle Standort wird mittels g.fillArc(Position,Radien,Winkel) gezeichnet.

Realisierung der Client-Anwendung

Zur Sicherung der Funktionalität der Anwendung, die in Kapitel 4.5 gefordert wird, sollen nun die Client-Anwendung beschrieben werden, welche die Funktionen der Software realisiert.

Die Indoor-Navigation findet auf der Präsentationsschicht der Anwendungsarchitektur des Kapitel 5.3 Platz. Die Komponenten *view* und *controller* bilden zusammen die GUI-Komponente. Die Komponente *model* ist eine Komponente der Anwendungslogik. Die Anwendungslogik ist über zwei Schnittstellen mit der GUI verbunden, über die *model* einerseits die Zustände erhält und andererseits das erstellte Model zurückliefert. Außerdem ist die Komponente *Anwendungslogik* mit dem Karten-Server über eine Schnittstelle verbunden. Eine weitere Schnittstelle verbindet sie mit der Komponente *Empfangsmodul*.

View-Controller

Die Funktionalität der Komponenten *view* und *controller* werden in der Anwendung durch die Klasse *View-Controller.java*, welche die Klassen *javax.microedition.midlet.Midlet* erweitert, verwirklicht. Dies ist infolge der Programmiersprache Java möglich, da *View-Controller.java* zusätzlich das Interface *javax.microedition.lcdui.CommandListener* implementiert. Die Abgrenzung von *view* und *controller*, wie es der MVC-Pattern vorsieht, bleibt daher nicht erhalten.

Enthält die Klasse *View-Controller.java* nur von javax.microedition.lcdui.Form beinhalten, so würde lediglich die Komponente view implementiert werden. Als mögliche Konsequenz sollte dann ein separater *javax.microedition.lcdui.CommandListener* implementiert werden, um dann mittels zweier Java-Klassen die Funktionalität *view* und *controller* zu verwirklichen. Auf dieses Vorgehen wird verzichtet. Denn einerseits wird der Code der Klasse *View-Controller.java* durch das Verschmelzen nicht weniger lesbar und man erspart sich so die Erzeugung einer weiteren Instanz.

[1] Polling, aktives erfragen von Informationen von einem asynchron laufenden Prozesses

Damit die *view* des MVC-Pattern in Java-Code umgesetzt werden kann, müssen die Objekte, die auf der Anzeige dargestellt werden sollen, „displayable“ sein. Folgend werden die Klassen aufgelistet, die dies erfüllen und genutzt wurden (sie sind, mit Ausnahme von javax.microedition.midlet, alle im package *javax.microedition.lcdui* enthalten):

- javax.microedition.midlet

  Im Midlet *View-Controller.java* ist der Lebenszyklus der Anwendung enthalten. Hier wird vereinbart, was beim Start, in Pausen und beim Beenden der Anwendung passieren soll. Es ist zugleich die Main-Klasse der Anwendung. In ihr werden die nachfolgenden Klassen, die Objekte für die Ausgabe auf der Anzeige sind, untergebracht.

- Canvas

  Auf dieser „Leinwand“ kann Pixel-genau (ein Pixel kann mittels g.drawLine(x,y,x,y) gezeichnet werden) unter Verwendung eine Graphics-Objektes gezeichnet werden. Das Graphics-Objekt wird in einer Methode, *.paint()*, verwendet, um verschiedene geometrische Figuren zu zeichnen.

- Display

  Diese Klasse stellt, wie es der Name schon erläutert, die Objekte, die ihr über die Methode .setCurrent() übergeben werden, auf der Anzeige dar.

- Form

  Diese Klasse wird verwendet, um einen so genannten StringItem auf der Anzeige darzustellen.

- Command

  Mittels dieser Klasse lassen sich Tasten mit einer Bezeichnung (*java.util.String*) versehen und über diesen mit Funktionen belegen.

- List

  Mittels dieser Klasse werden die Zielorte auf der Anzeige anhand einer Liste abgebildet. Man kann mittels zweier Tasten Auf- und Abwärts navigieren. Mit der *select*-Taste kann ein Ziel ausgewählt werden.

- CommandListener

  Der CommandListener empfängt einen Interrupt, der durch eine Taste, die einem *Command* zugeordnet wurde, ausgelöst wird. Sobald der CommandListener einen Interrupt empfängt, wird die Methode .actionCommand(Command, Displayable) aufgerufen. In ihr werden dann die Commands ausgewertet und nachfolgende Befehle ausgeführt.

Model

Die Komponente *model* beinhaltet die acht Komponenten, die in Kapitel 5.1 dargestellt wurden. Mit deren Hilfe erstellte *model* das Anwendungsobjekt, dass über eine Schnittstelle an die *view* zur Ausgabe weitergereicht wird. Die Erstellung des Anwendungsobjektes und Zusammenarbeit der acht Komponenten aus der Tabelle 2.3, wird in diesem Abschnitt genauer erläutert. Die Funktionalitäten ihrer Komponenten erfüllen diese Klassen mittels der besprochenen Klassenbibliotheken aus Kapitel 6.4, welche in MIDP 2.0 und CLDC 1.1 enthalten sind.

Die Komponente *model* beinhaltet die Klasse Model.java, das Anwendungsobjekt des MVC-Pattern. Model.java erweitert zu diesem Zweck die Klasse *javax.microedition.lcdui.Canvas* (Canvas) und implementiert das Interface *java.lang.Runnable* (Runnable).

Mittels der Methode .paint(Graphics g) ist es mit Model.java möglich, Elemente wie Text, Linien und einfache Figuren pixel-genau zeichnen zu lassen. Aus diesen Elementen besteht das *model* der Anwendung. Zur Generierung solcher Elemente werden Methoden der Klasse *javax.microedition.lcdui.Graphics* genutzt. Die Methoden, die dazu notwendig sind, sollen nun vorgestellt werden.

- *g.drawString(Bezeichnung, x, y, Ausrichtung)*[2]

  Mittels dieser Methode werden Beschriftungen auf der Anzeige dargestellt. Die *Bezeichnung* wird per *java.lang.String* übergeben und an die *Stelle (x, y)* gesetzt und kann zusätzlich über das Attribut *Ausrichtung* eine bestimmte Stellung einnehmen. Die Beschriftungen werden von den Komponenten „Text“ und „Tooltips“ benötigt.

  Die Komponente „Navigationselemente“ soll auch die Distanz ins Ziel anzeigen. Diese Angabe soll in der Mitte des oberen Randes platziert werden. Text wird auf der Anzeige in weißer Farbe dargestellt.

- *g.drawRect(x-Anfang, y-Anfang, x-Ausbreitung, y-Ausbreitung)*

  Diese Methode dient dazu, ein Rechteck an einer vereinbarten Stelle auf der Anzeige darzustellen. Das Rechteck beginnt im Punkt (x-Anfang, y-Anfang) und endet im Punkt (x-Ausbreitung, y-Ausbreitung). Durch diese Methode werden Räume dargestellt. Das bedeutet, das Räume mit Rechtecken assoziiert werden. Jeder dieser Räume implementiert das Interface Room.java und enthält Anfangspunkt und Ausbreitung; entsprechend obiger Notation der Methode. Ein Raum wird auf der Anzeige grau dargestellt. Diese Farbe entspricht der Farbe des Mauerwerks.

[2] Mit klein „g“ ist eine Instanz von Graphics bezeichnet.

- *g.drawLine(int x-Anfang, int y-Anfang, int x-Ausbreitung, int y-Ausbreitung)*

  Mittels „drawLine“ kann eine Linie auf der Anzeige dargestellt werden. Die Linie beginnt im Punkt (x-Anfang, y-Anfang) und endet im Punkt (x-Ausbreitung, y-Ausbreitung). Türen, Navigationselemente und Kartenelemente (Landmarken) werden in der Anwendung als Linien oder Figuren aus Linien dargestellt. Jede Tür ist eine Instanz der Klasse Doors.java und enthält Anfangs-Punkt und Ausbreitung, entsprechend der oben genannten Anfangs- und Endpunkte. Navigationselemente wie der Maßstab oder die Ausrichtung des Gebäudes gegen Nord sind aus Linien zusammengesetzt. Der Maßstab wird beim Vergrößern oder Verkleinern entsprechend angepasst, d. h. vergrößert oder verkleinert. Die Landmarken werden vom Server bezogen. Sie werden aus einer Datei geparsed und Stellen eine Figur dar, beispielsweise die Figur einer Treppe oder eines Aufzuges. Diese Figur wiederum besteht aus Linien. Ebenso ist jede Route eine Aneinanderreihung von Linien. Daraus entsteht ein so genannter Polygonzug, der die Route des Anwendungsobjektes darstellt.

- *g.fillArc(int x,int y,int xRad,int yRad,Winkel.anfang,Winkel.ende);*

  Diese Methode füllt einen Bogen mit den Radien xRad, yRad an der Stelle x,y zwischen den Winkeln Winkel.anfang und Winkel.ende aus. Für die Anwendung wird anhand dieser Methode der Standort als ausgefüllter Kreis dargestellt. Die Farbe des Kreises ist gelb und er ist stets in der Bildschirmmitte zu sehen. Der gelbe Kreis dient gleichzeitig dazu, die virtuelle Umgebung zu erkunden, zu diesem Zweck kann man ihn steuern. Zum Steuern kann der Benutzer vier Cursor-Tasten (oben, unten, links und rechts) des PDA benutzen.

Zur zusätzlichen Unterscheidung werden die Kartenelemente in unterschiedlicher Farbe dargestellt. Die Einteilung findet anhand der in Kapitel 5.5 definierten farblichen Zuordnung statt. Um die Kartenelemente in dieser Farbe darzustellen, wird jeweils vor der Benutzung die Farbe festgelegt. Dazu dient die Methode:

- *g.setColor(Farbe)*

  In der Anwendung wurde anstelle einer Integer-Zahl eine hexadezimale Farben-Bezeichnung gewählt. Diese Bezeichnungen sind in Farbtafeln [(32)] gebräuchlich und daher immer wieder zu finden. Die Farbe lässt sich anhand von 256 Rot-Tönen und der gleichen Anzahl an Gelb- und Blau-Tönen zusammensetzten. Eine „00“ steht in der hexadezimal-Schreibweise für den hellsten Ton der Farbe und ein „FF“ für den dunkelsten Ton. Die Farbwahl der Anwendung entfällt nur auf die dunkelsten Töne der Farben, da diese auch für Menschen mit einer Farb-Seh-Schwäche unterscheidbar bleiben.

Die zuvor erläuterten Methoden zur Zeichnung der Kartenelemente benötigen Positions- und Ausbreitungsdaten. Diese werden für jede Objekt-Gruppe aus XML-Datei gewonnen,

die der angesprochene Karten-Server bereitstellt. Wie dies geschieht, wann die Klassen der Kartenelemente instanziert werden und welche Daten diese dann beinhalten, soll nun beleuchtet werden.

In den Komponenten Raumsuche, Türsuche, Route und Kartenelemente aus der Tabelle 2.3 sind Parser-Klassen enthalten. In diesen Parser-Klassen werden mittels Daten aus XML-Dateien Instanzen von Room.java, Door.java, Route.java, POI.java und Landmarke.java erschafft. Diese Instanzen werden mittels eines java.util.Vector im Speicher des PDA persistent gehalten.

Zur Erledigung der beiden Aufgaben, Instanzierung und Persistierung, werden zum einen die erwähnten Parser-Klassen verwendet, als auch die erwähnten Daten-Klassen.

Die Parser-Klassen werden von *View-Controller.java* instanziert und erhalten über ihren Konstruktor eine URL des Web-Service, der sich auf dem Karten-Server befindet. Mittels dieser URL und einem Client-Stub des Web-Service erzeugt die Parser-Klasse eine Instanz des Web-Service auf dem PDA. Über die Instanz des Web-Service erfolgt dann der Zugriff auf die Methoden des Web-Service, welche die XML-Dateien bereitstellen.

Vor dem Zugriff auf die Methoden, muss der aktuelle Standort zur Verfügung stehen und beim Methoden-Aufruf übergeben werden. Nach dem Absetzen des Methodenaufrufes wird vom Web-Service die XML-Datei übersendet. Diese XML-Datei enthält Einträge einer unbestimmten Anzahl an geforderten Elementen einer Elementgruppe (Raum, Tür, Route, Kartenelement und Standort). Beispielsweise enthält die XML-Datei rooms.XML eine unbestimmte Anzahl an room-Elementen.

Es gibt fünf verschiedene Klassen, die auf diese Weise verfahren:

- RoomSniffer.java
- DoorSniffer.java
- RouteSniffer.java
- POISniffer.java
- Landmarkensniffer.java

Der Algorithmus, siehe Anhang A.2, dieser Parser-Klassen beginnt mit dem Anlegen eines Objektes der Klasse java.lang.Class. Das Class-Objekt bietet eine Methode .getRescourceAsStream(URI), die es erlaubt, eine Ressource anhand einer URI byte-weise einzulesen. Zu diesem Zweck liefert die Methode einen InputStream zurück. Der InputStream wird aus Performanz-Gründen zunächst gänzlich in ein Byte-Array fester Größe eingelesen. Danach werden anhand eines Algorithmus die Daten für die folgenden zu erzeugenden Objekte eingelesen. Ebenfalls aus Performanz-Gründen sind *continue*-Anweisungen Eingebaut. Diese sollen es erzwingen, nach erfolgreicher Detektion eines Attributes eines Kartenelementes

nicht mit dem nächsten if-clause weiterverfahren wird, sondern das nächste Byte eingelesen werden soll. Im letzten Attribut eines Kartenelementes, d.h. am Ende des Algorithmus werden Daten-Objekte angelegt, welche die Attribute der Kartenelemente der XML-Dateien persistieren (siehe lokales caching):

- Room.java

  Eingelesen werden die ID zur eindeutigen Bezeichnung, der Name der in der Zielortliste erscheinen wird, der Startpunkt (x,y) und die Ausbreitung in x- und y-Richtung. Die zugehörige XML-Datei heißt rooms.XML. Ein Eintrag über eine Raum in der *rooms.XML* hat folgende Syntax:

  <room
  id ="11.01A"
  name ="Hotel Suite"
  x1 ="0"
  x2 ="833"
  y1 ="0"
  y2 ="841"
  />

  Ein Raum wird mittels g.drawRect(Position, Ausmaße) in der paint(Graphics-Objekt)-Methode gezeichnet

  Das Interface *Room.java* berücksichtigt durch seine Attribute die grundlegenden Eigenschaften eines Raums:

  - zwei Koordinaten, x und y, die den Ursprung des Raumes festlegen
  - zwei Attribute über die Ausbreitung in x- und y-Richtung.

  Mittels dieser Angaben kann eine Klasse, die dieses Interface implementiert, als Raum bezeichnet werden.

  Da Räume unterschiedliche Geometrien besitzen können, aber zugleich bestimmte markante Eigenschaften eine Raumes haben, müssen gleichzeitig die Eigenschaften des Interface Room.java implementiert sein. Die Klasse *Rectangle.java* implementiert das Interface Room.java. Das bedeutet, das Rectangle alle Anforderungen an einen Raum beinhaltet. Aus geometrischer Sicht stehen bei einem Rectangle-Objekt die vier Seiten senkrecht zueinander, was ebenfalls der Anforderung an ein Raum genügt.

- Door.java

  Eine Tür hat eine ID zur eindeutigen Bezeichnung, kennt die beiden Räume room1 und room2, die sie verbindet, und erhält zusätzlich ihren Standort in x- und y-Koordinaten.

Ein Eintrag einer Tür in der *doors.XML* hat folgende Syntax:

<door
id ="D001"
room1 ="11.31"
room2 ="11.01A"
x ="62"
y ="844"
/> Ein Raum wird mittels g.drawLine(Position, Ausmaße)und g.drawArc(Position,Radien, Winkel) in der paint(Graphikobjekt) gezeichnet.

- Route.java

  Die Route hat einen Startpunkt, dieser ist immer ein Raum. Der Startpunkt der Route besteht aus der ID des Raumes. Weitere Elemente sind Türen, ebenfalls bekannt durch ihre ID. Die Türen wurden auch als Routenelemente gewählt weil sie auf der Route liegen und vom Benutzer durchquert werden müssen. Gleichfalls ist dies vom Karten-Server schon berücksichtigt, da die Routenführung auf dem Durchqueren der Türen beruht. Am Ende einer Route ist der Zielort, dem Startpunkt entsprechend ein Raum. Eine Route stellt dabei immer den kürzesten Weg vom Startpunkt zum Zielpunkt dar.

Folgende Kartenelemente werden vom Kartenserver noch nicht bereit gestellt aber dennoch exemplarisch auf der Anzeige zu sehen sein:

- Landmarken
- POIs

Darüber hinaus sollte der Kartenserver zusätzlich Informationen für die Navigationselemente-Komponente, wie beispielsweise über die Ausrichtung des Gebäudes bereitstellen und einen Massstab definieren. Beispielhaft soll hier erklärt werden, wie ein POI oder eine Landmarke in Java implementiert werden könnte:

POI.java/Landmarke.java

Ein/-e POI/Landmarke besitzt ein Aussehen. Diese wird anhand einer ID identifiziert. Die ID ist einzulesen, ebenso wie die Position und die Farbe des/der POI/Landmarke. Eine Instanz der Klasse *POI*/*Landmarke* wird ihrer Zoomstufe entsprechend in einem *java.util.Vector* gespeichert. Das *model* nutzt das Koordinaten-Paar der Instanz der *POI*/*Landmarke* und stellt eine geometrische Figur an entsprechender Stelle auf der Anzeige dar, die das Koordinaten-Paar beschreibt.

In verschiedenen *java.util.Vector* werden POIs/Landmarken verschiedener Zoomstufen gespeichert, um diese nach dem Filtern abzulegen. Für das Filtern ist die Instanz *POI-Filter.java*

zuständig. Diese wird von der Instanz *Model.java* erzeugt. Ein bestimmter *java.util.Vector* steht für eine bestimmte Zoomstufe.

Ist die Zuordnung erfolgt, so wird in jeder Zoomstufe auf die Instanzen von POI.java zugegriffen und für jede dieser Instanzen mittels verschiedener *javax.microedition.lcdui.Graphics*-Methoden Figuren auf der Anzeige erzeugt und an entsprechender Stelle angezeigt.

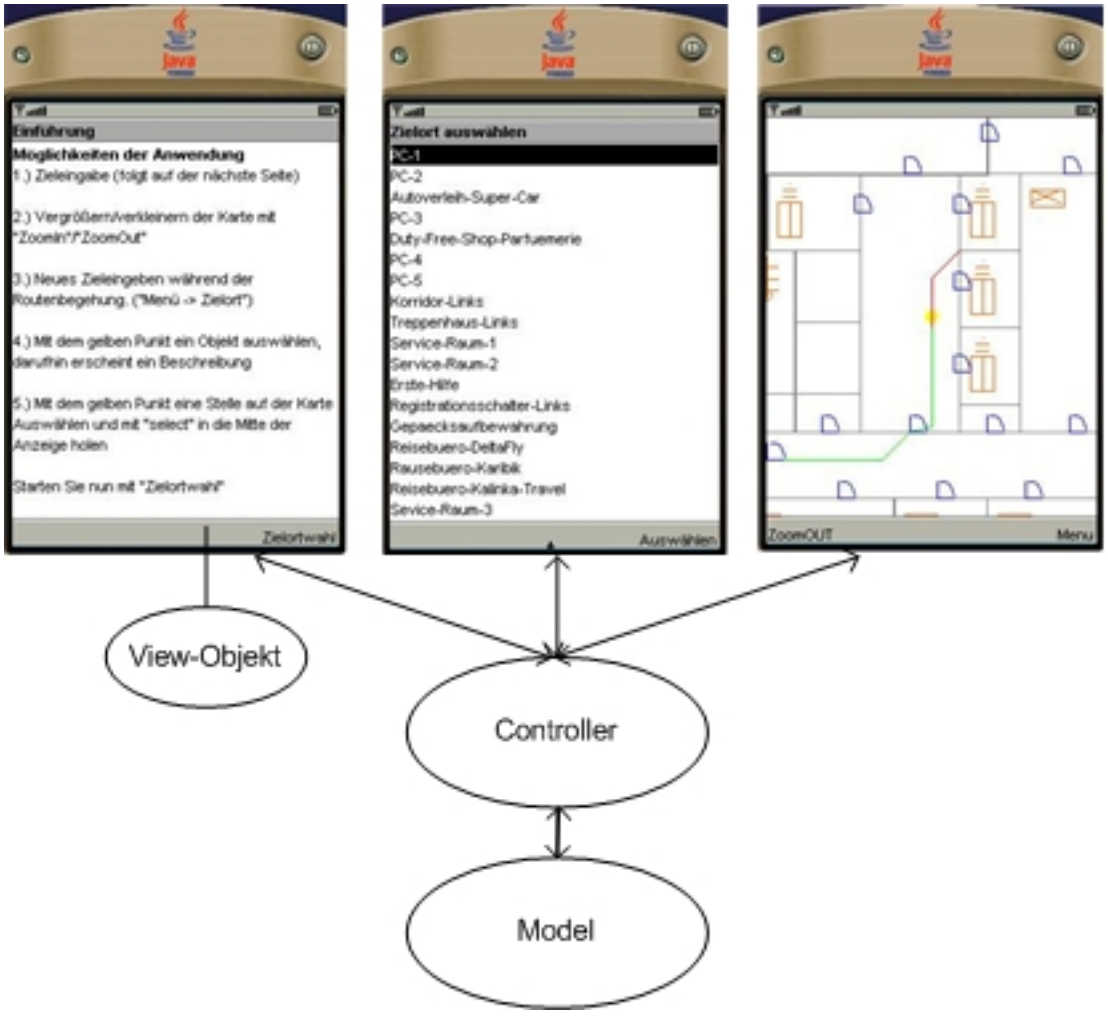


Abbildung 6.5.: *Umsetzung des MVC-Pattern auf die Anwendungsschicht. Die linke View zeigt den Startbildschirm, die mittlere die Zielortauswahl und die rechte den Navigationsbildschirm*

Schnittstellen

In diesem Kapitel sollen die Schnittstellen zum Model, zum Karten-Server und zum Empfangsmodul beschrieben werden.

Schnittstellen zum Model

Der Abbildung 5.3 ist zu entnehmen, dass zwischen der Komponente *model* und *controller* zwei Schnittstellen existieren.

- *Zustand*

  Über diese Schnittstelle gelangen von der Instanz *View-Controller.java* die Anwendungsfälle in Form von Methoden-Aufrufen und Parameter-Übergaben an die Instanz Model.java.

- *view*

  Die Instanz von *Model.java* erstellt mittels der zuvor besprochenen Daten-Klassen in ihrer Canvas.paint(Graphics g)-Methode sowie den besprochenen Graphics-Methoden ein Model für die Darstellung auf der Anzeige. Dieses wird in der Abbildung 5.3 symbolisch durch die Schnittstelle *model* für *View-Controller.java* bereitgestellt.

Schnittstelle zum Karten-Server

Die Schnittstelle zum Karten-Server ist in der Abbildung 5.3 als Kartenelemente beschrieben. Sie ist mittels eines Web-Service implementiert. Dies bedeutet, dass zunächst auf dem PDA ein Client-Stub des Web-Service instanziert werden muss. Über diese Instanz kann dann auf die Methoden des Web-Service zugegriffen werden. Der Request und der Response werden über SOAP-Nachrichten, die in http-Request /-Response eingebettet sind, an den Empfänger (erhält http-Request) und dem Sender (erhält http-Response) übermittelt. Die SOAP-Nachricht vom Request-Sender enthält die Methode, die aufgerufen werden soll. Die SOAP-Nachricht an den Response-Empfänger enthält dementsprechend den Rückgabewert der Methode.

Schnittstelle zum Empfangsmodul

Die Schnittstelle zwischen der Instanz von Model.java und dem Empfangsmodul ist noch nicht realisiert. Der Grund ist, dass MIDP 2.0 keine Verbindung über die POP-Schnittstelle des NOKIA E70-1 mit der RS232-Schnittstelle des Empfangsmoduls erstellt.

Die POP-Schnittstelle ist mittels USB-Verbindung an die RS232-Schnittstelle zu koppeln. Diese Steckverbindung gehört zum jetzigen Zeitpunkt zwar nicht zur Ausstattung des Moduls, ließe sich aber realisieren.

Eine andere Lösung würde sich wegen der kurzen zu überbrückenden Distanz anbieten: Bluetooth, siehe Kapitel 6.2.

Die Klasse Standort.java implementiert daher die Bluetooth-API für den späteren Einsatz.

Sequenzdiagramme der Anwendungsfälle

Anhand von Sequenzdiagrammen lässt sich die Kommunikation der Komponenten bzw. Instanzen in einem zeitlichen Zusammenhang darstellen. Dieser Zusammenhang ergibt sich aus der Abarbeitung der Anwendungsfälle.

Die im Kapitel 4 erarbeiteten Anwendungsfälle sollen anhand dieser Diagramme veranschaulicht werden. Die Anwendungsfälle werden durch Benutzereingaben angestoßen und daraufhin durch sequenzielle Methodenaufrufe abgearbeitet. Die Methodenaufrufe und die folgende Übergabe von Rückgabewerten spiegelt den Austausch von Nachrichten wieder und beschreibt so die Kommunikation zwischen den beteiligten Instanzen.

Folgende ausgewählte Anwendungsfälle, die beim Benutzen der Anwendung entstehen, sollen erklärt werden:

- Start der Anwendung
- Zoomen
- Routenberechnung
- Zielort Wahl
- Visieren mit gelbem Punkt

Anmerkung: Das verwendete UML-Tool[3] kann die UML-Konvention, der Erzeugung eines Objektes auf zeichnerischer Ebene nicht erfüllen. Das Tool kann dies aber mittels «create»-Attribut über dem Erzeugungs-Pfeil anzeigen. Im folgenden wird daher auf diese Weise verfahren.

Situation - Start der Anwendung

Zur Darstellung der Zielorte einer Etage zu Beginn der Navigation dient eine Liste, die mittels zweier Tasten durchsucht werden kann. Zur Auswahl eines Zielortes aus der Liste mit anschließender Routenberechnung dient eine weitere Taste.

1. *View-Controller* erzeugt eine Instanz von *DoorSniffer* und *Durchsucher*.
2. Die Instanzen von *DoorSniffer* und *Durchsucher* erzeugen jeweils eine Instanz des Web-Service mittels einem URL und dem Client-Stub.
3. *Durchsucher* fordert *rooms.xml* über die Instanz des Web-Service an.
4. *DoorSniffer* fordert *door.xml* über die Instanz des Web-Service an.
5. Die Instanz des Web-Service schickt *rooms.xml* und *door.xml*.
6. Instanz von Durchsucher parsed *rooms.xml* und erzeugt für jeden Eintrag eine Instanz der Klasse *Room*.
7. Instanz von Durchsucher legt die Instanz der Klasse *Room* in einem *java.util.Vector* ab.
8. *DoorSniffer* parsed *doors.xml* und erzeugt für jeden Eintrag eine Instanz der Klasse *Doors*.
9. DoorSniffer legt die Instanz der Klasse *Doors* in einem *java.util.Vector* ab.
10. *view-controller* erzeugt einen Text, der eine Einweisung in die Funktionalität der Anwendung gibt.
11. Im Hintergrund der Anwendung, während der Anzeige des Textes, wird die Zielortliste erstellt. Diese ergibt sich aus den Namen sämtlicher Räume einer Etage.
12. Nachdem der Benutzer eine Bestätigungstaste gedrückt hat, wird die Zielortliste angezeigt.

[3] Das verwendete UML-Tool ist ein Open-Source-Projekt und nennt sich Star-UML

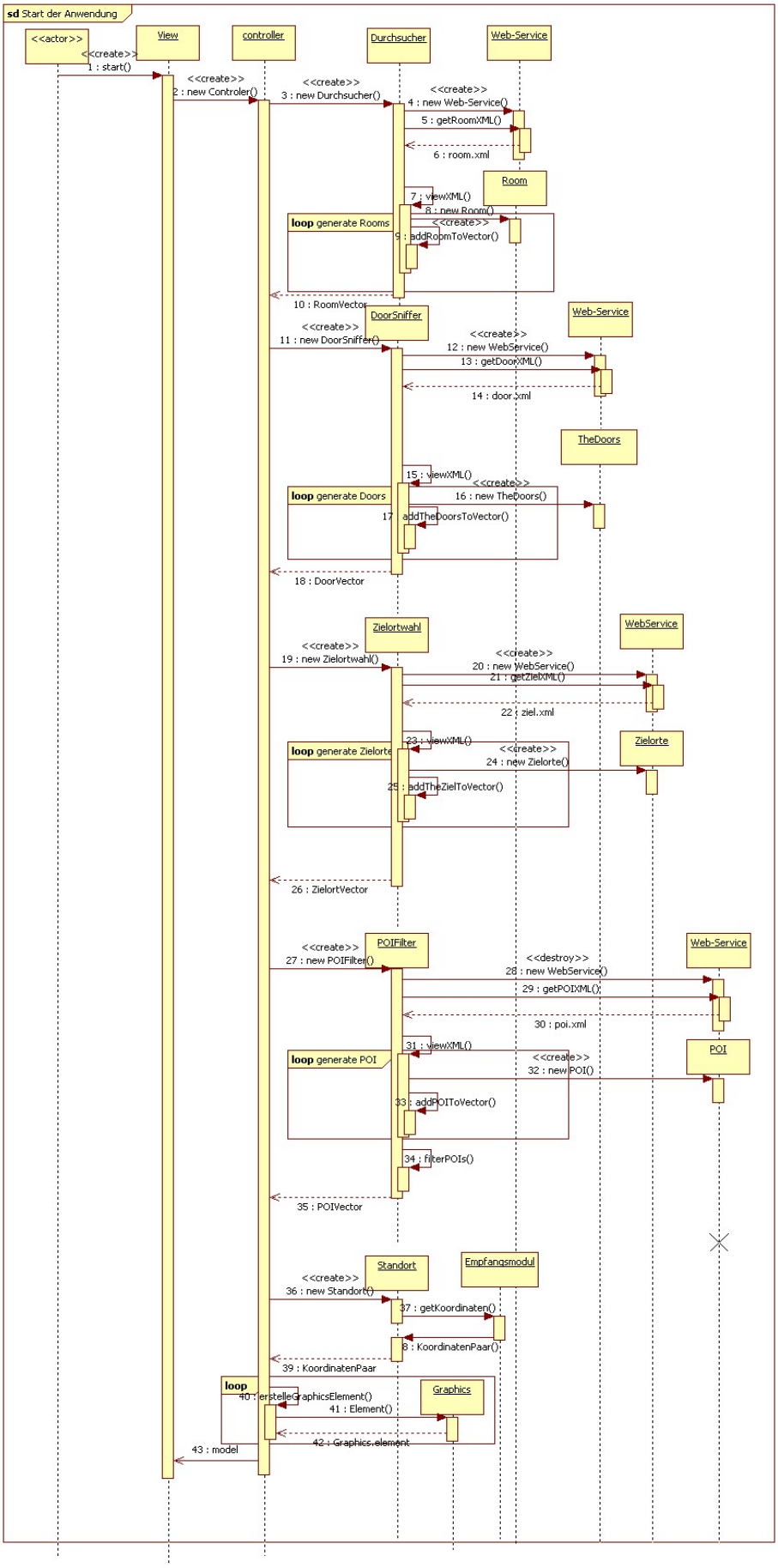


Abbildung 6.6.: *Sequenzdiagramm des Starts der Anwendung*

13. Sobald der Benutzer sich für ein Zielort entschieden hat und abermals die Bestätigungstaste gedrückt hat, wird eine Instanz des Web-Service mittels einem URL und einem Client-Stub erzeugt. Danach wird auf die Methode zur Routenberechnung des Web-Services zugegriffen und Standort und Zielort mit übergeben.
14. Die Instanz von *Zielortwahl* erzeugt eine Instanz von Web-Service mittels einem URL und dem Client-Stub.
15. Der Web-Service liefert die Route in Form einer XML-Datei zurück.
16. Die Route wird von RouteSniffer.java geparsed und elementweise ein Route-Objekt angelegt.
17. *model* erzeugt Instanz von *POIFilter*.
18. Die Instanz von *POIFilter* erzeugt eine Instanz von Web-Service mittels einem URL und dem Client-Stub.
19. *POIFilter* fordert *poi.xml* von Web-Service an.
20. *POIFilter* parsed *poi.xml* und erzeugt für jeden Eintrag eine Instanz der Klasse *POI*.
21. *POIFilter* legt die Instanz der Klasse *POI* in einem *java.util.Vector* ab.
22. *Model* erhält das geforderte Koordinaten-Paar.
23. In der .paint()-Methode von *model* werden für die Objekte Room, Doors, POIs, Route und für das Koordinaten-Paar geometrische Figuren angelegt.
24. Die *view-controller* zeigt mit display(model) das *model* an.

Situation - Zoomen

Durch das Zoomen kann eine Verkleinerung oder Vergrößerung der Elemente auf der Anzeige vorgenommen werden. Zur Verkleinerung dient die *ZoomIn-Taste*, für das Vergrößern die *ZoomOut* Taste. Im folgenden wird exemplarisch nur das vergrößernde Zoomen beschrieben und veranschaulicht. Das Szenario spielt sich während der Navigation ab.

1. Der Anwender drückt die Taste *ZoomIn* auf dem Bedienelement des PDA. Der *view-controller* nimmt den Befehl entgegen und erhöht den Parameter der Zoomstufe.
2. Anschließend werden die Positions- und Ausbreitungsdaten (Breite und Höhe) der zu skalierenden Objekte in der .paint()-Methode von *model* mit dem Skalierungsfaktor multipliziert.
3. Die Komponente *model* übergibt das Model über die Schnittstelle *Anzeige* an die Komponente *view-contoller*.

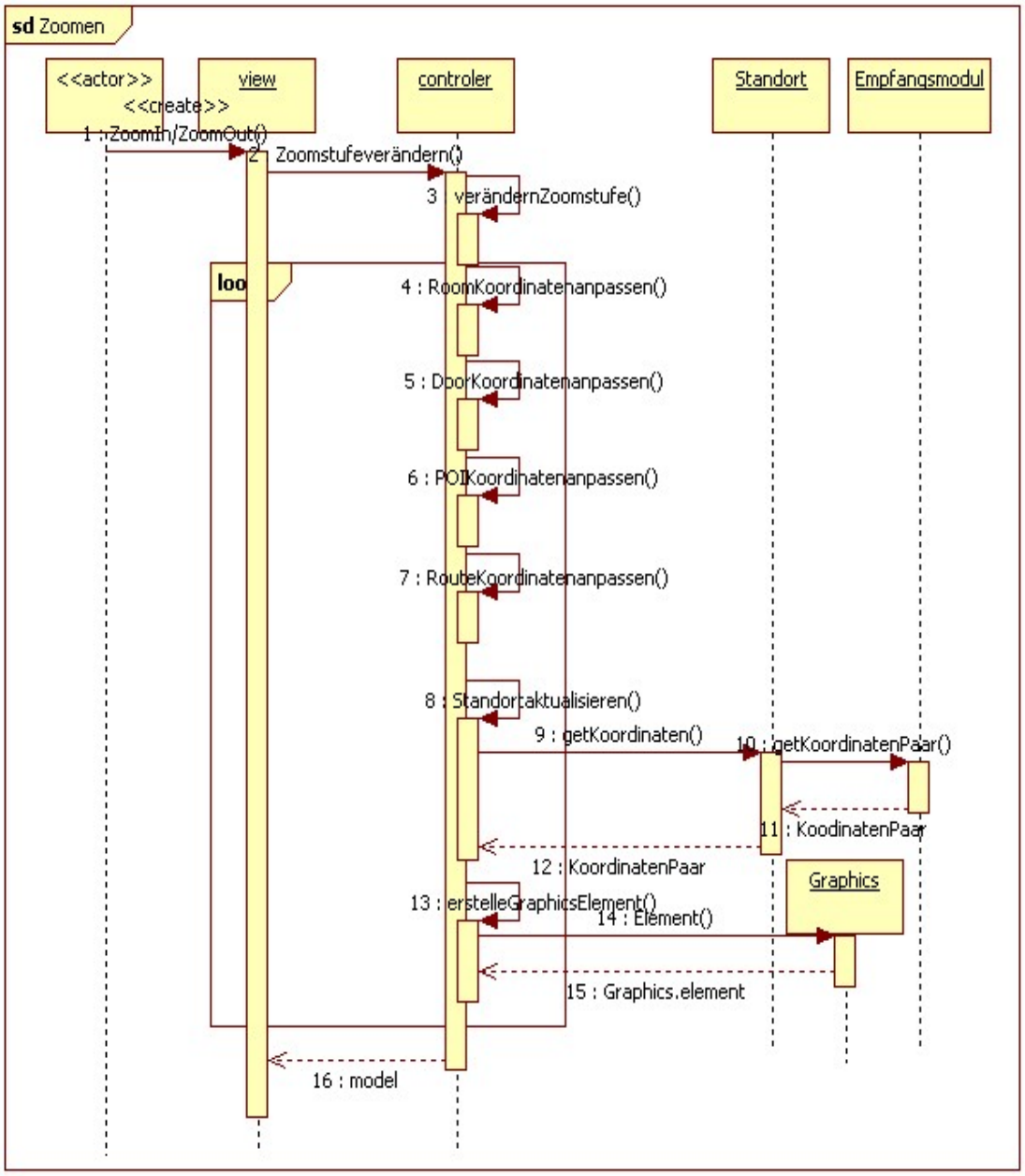


Abbildung 6.7.: *Sequenzdiagramm, welches das Zoomen darstellt.*

4. Die Komponente *view-controller* gibt das Graphics-Objekt mit display(model) auf der Anzeige aus.

Situation - Routenberechnung

Hiermit soll die Routenberechnung durch den Benutzer in Gang gesetzt werden.

1. Benutzer drückt die Taste *Zielort*, um die Routenberechnung zu bestätigen.
2. *View-controller* löscht eventuell das vorherige *model*.
3. Die Instanz von *view-controller* erzeugt eine Instanz des Web-Service mittels einem URL und dem Client-Stub.
4. *View-Controller* übergibt die Methode zur Routenberechnung den aktuellen Standort und den gewählten Zielort.

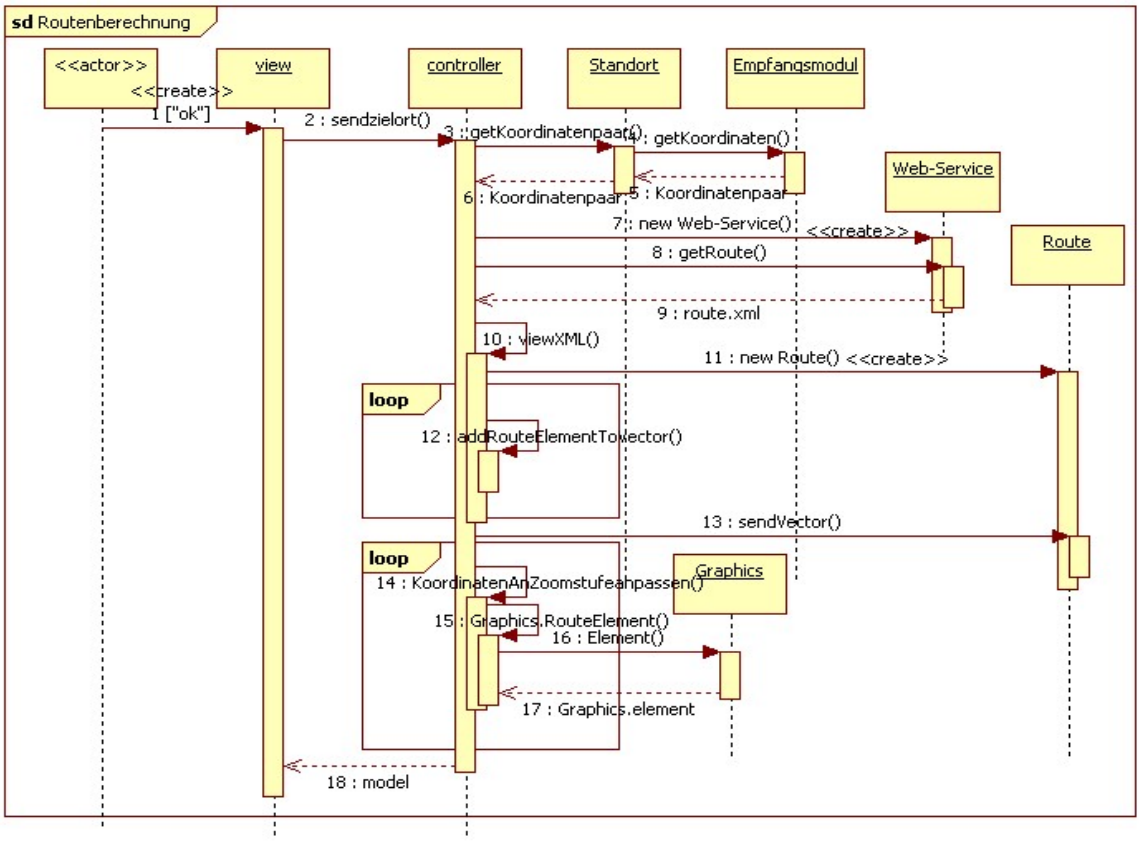


Abbildung 6.8.: *Sequenzdiagramm, welches die Routenberechnung darstellt.*

5. Der Web-Service liefert die Route in Form einer XML-Datei zurück.
6. Die Route wird von *RouteSniffer.java* geparsed und anschließend ein Route-Objekt angelegt.
7. *Model* erzeugt eine Instanz von *POIFilter*.
8. Die Instanz von *POIFilter* erzeugt eine Instanz von Web-Service mittels einem URL und dem Client-Stub.
9. *POIFilter* fordert *poi.xml* von Web-Service an.
10. *POIFilter* parsed *poi.xml* und erzeugt für jeden Eintrag eine Instanz der Klasse *POI*.
11. *POIFil*ter legt die Instanz der Klasse *POI* in einem *java.util.Vector* ab.
12. *Model* erhält das geforderte Koordinaten-Paar.
13. In der .paint()-Methode von *model* werden für die Objekte Room, Doors, POIs, Route und für das Koordinaten-Paar geometrische Figuren angelegt.
14. Die *view-controller* zeigt mit display(model) das *model* an.

Situation - Zielort Wahl (Etage)

Zielorte sind in einer Liste in horizontaler Weise platziert. Um einen Listeneintrag zu markieren kann ein Benutzer die Tasten *up* und *down* verwenden. Mit der Taste *up* kann der Benutzer einen Zielort markieren, der oberhalb des aktuell markierten Zielortes platziert ist. Mit der Taste down kann der Benutzer einen Zielort markieren, der unterhalb des aktuell markierten Zielortes platziert ist. Zu Beginn ist der erste und zugleich der oberste Eintrag der Liste markiert. Drückt der Benutzer in dieser Stellung die Taste up, so wird der letzte Eintrag der Liste markiert.

1. Die *view-controller* stellt eine Liste mit Zielorten durch Display(Liste) auf der Anzeige dar.
2. Der Benutzer drückt die Taste *up* des Bedienelementes, um in der Liste einen oberhalb des aktuell markierten Listeneintrags platzierten Listeneintrag zu markieren.
3. Die *view-controller* nimmt den Befehl entgegen und markiert diesen Eintrag und legt eine Variable mit dem Namen des Eintrages an.
4. Der Benutzer drückt die Taste *ok*, um den aktuell markierten Listeneintrag als Zielort für die Route auszuwählen.

5. Die *view-controller* nimmt diesen Befehl entgegen und erzeugt zwei Instanzen, eine von *DoorSniffer.java* und eine von *Durchsucher.java*.
6. Beide Instanzen erzeugen daraufhin eine Instanz des Web-Services und fordern ihre benötigten XML-Dateien an. DoorSniffer fordert *doors.xml*, Durchsucher fordert *rooms.xml*.
7. Aus dem von *Durchsucher.java* angelegten *java.util.Vector* liest *view-controller* den Eintrag heraus, dessen Namen identisch mit dem des zuvor gespeicherten Namen ist.
8. Ist der Name gefunden, wird eine neue Instanz von *model* erstellt und über deren Konstruktor der Standort und der Zielort übergeben.

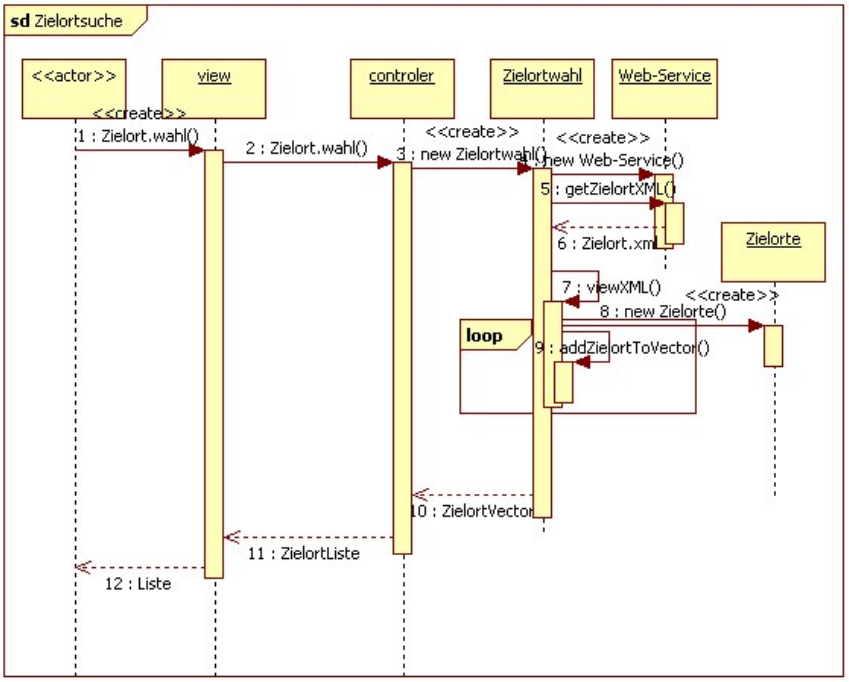


Abbildung 6.9.: *Sequenzdiagramm das die Zielortwahl darstellt.*

Situation - Visieren mit gelbem Punkt

Der Benutzer hat sich entschieden, einen bestimmten Raum in der Mitte der Anzeige zu platzieren. Mittels des gelben Punktes kann bestimmt werden, welcher Raum der Anzeige das sein sollte. Sobald sich der Benutzer für eine Stelle, die er Zoomen möchte, entschieden, so vollzieht sich folgende Sequenz.

1. Der Benutzer drückt die Taste *Ok*, um den Raum in der Mitte zu platzieren.
2. Die *view-controller* nimmt den Befehl entgegen und leitet ihn weiter an das *model*. Danach wird ebenfalls repaint().model aufgerufen.
3. In der Zwischenzeit ersetzt *model* die Variablen StandortX und StandortY mit den neuen Koordinaten, dem Mittelpunkt.
4. Der *view-controller* stellt das *model* auf der Anzeige dar.

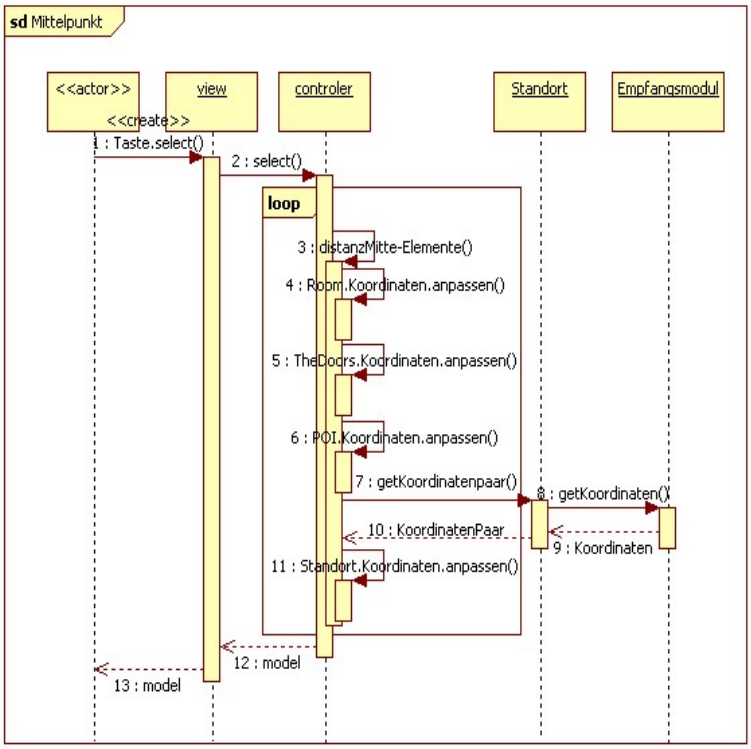


Abbildung 6.10.: *Sequenzdiagramm, welches das Festlegen eines neuen Mittelpunktes darstellt*

6.6. Fazit

Mittels Java und dem implementierten Algorithmus aus Anhang A.2, war es möglich eine Anwendung zu erschaffen, die im Zeitverhalten auf den Geräten ein gutes Ergebnis lieferte. Die Anwendung läuft ruckfrei und ohne festfrieren der Anzeige ab. Die geforderten nichtfunktionalen Anforderungen aus Kapitel 4 konnten durch den Prototypen der Anwendung erfüllt werden.

- Die Funktionalität sah vor, alle funktionalen Anforderungen zu erfüllen. Dies ist verwirklicht worden.
- Die Zuverlässigkeit wurde durch eine fehlerfrei Anwendung bewiesen.
- Die Anwendung ist effizient genug, um auf mobilen Geräten zu laufen; dazu wurde auf optimierten, schlanken Code hingearbeitet.
- Die Anwendung bleibt auch für zukünftige Versionen änderbar.
- Der Übertragbarkeit wurde durch die Verwendung von J2ME nachgekommen.
- Die Steuerbarkeit der Anwendung ist gegeben, da der Benutzer vorwärts und rückwärts navigieren und die Anwendung stets beenden kann.
- Zudem ist die Anwendung selbstbeschreibungsfähig, da sie in einem Hauptmenü beschrieben wird und alle weiteren Geschäftsanwendungsfälle per Tastendruck ausgelöst werden können. Die Funktionalität der Tasten ist zu diesem Zweck auf der Anzeige dargestellt.
- Die Karte kommt der geforderten Lesbarkeit nach, d. h. die Elemente sind voneinander unterscheidbar sowohl farblich als auch von der Form.
- Ebenso wurde der Klarheit nachgekommen, die es vorsieht, dass die Elemente sich klar vom Hintergrund abheben sollen.
- Die Karte erfüllt ebenso die Vollständigkeit, da sie alle zur Navigation nötigen Element anzeigt.

Damit wurde gezeigt, dass die wichtigsten nicht-funktionalen Anforderungen berücksichtigt wurden.

Der Prototyp der Anwendung ist in der Lage, die Umgebung maßstabsgetreu in 2D darzustellen. Die Darstellung berücksichtigt die Route von einem Startpunkt zu einem Zielpunkt, POIs. Zusätzlich zu diesem werden einen Massstab, die Nordausrichtung des Gebäudes und der Weg in Metern bis zum Ziel angezeigt.
Der entwickelte Client bietet ein Hauptmenü zur Erläuterung des Dienstes, eine Zielortliste zur Wahl mittels Tasten, das Zoomen und das Verschieben der Karte an. Für das Zoomen verschiedener Detailstufen ist in Kapitel 4.3 eine stufenweise Einteilung von Landmarken bezüglich ihrer Relevanz vorgenommen worden. Anhand dieser wurden Landmarken einer Zoomstufe zugeordnet. Dies wurde implementiert.

Die Anwendung basiert zur Positionsbestimmung auf dem IMAPS-Modul, dass via Bluetooth mit dem PDA verbunden sein soll. Aus der Tabelle 2.3 des Kapitel 2.3 ist zu entnehmen, dass IMAPS die geforderte Genauigkeit von 2m einhalten kann. Dennoch stößt die Entwicklung

dieses Systems an ihre Grenzen, da die Navigation mittels der Beacon noch nicht ausgereift genug ist, um einen Testlauf zu gewähren. Der Grund ist das Fehlen der Bluetooth-Verbindung, um das Koordinatenpaar vom IMAPS-Modul zum PDA zu übermitteln. Während einer Unterhaltung mit S. Gregor [(28)] versicherte er, dass es möglich sei, eine Bluetooth-Verbindung anstelle der RS232-Schnittstelle anzubringen. Die RS232-Schnittstelle wird nicht weiter von Herstellern für die Geräten der CLDC vorgesehen, womit diese Schnittstelle vor dem Aussterben steht.

Das Problem des unscharfen Zeichnens von schrägen Linien konnte theoretisch gelöst werden, praktisch aber nicht. Es trat bei der Routenbegehung auf, sobald es eine Richtungsänderung erzwang, die Kartenansicht zu drehen. Die Schwierigkeit bei der Drehung infolge der Umsetzung der Formeln aus Kapitel 5.5. Aufgrund von Rundungsfehlern ist es nicht möglich, eine exakte Drehung um 90° der Objekt auf der Anzeigt vorzunehmen. Diese Rundungsfehler entstehen in Java durch den *typecast nach int* obwohl *double*-Werte benötigt werden würden. Auf *int* muss gecastet werden, da man an die Methode *g.drarLine(..)* keine anderen Typen außer *int* übergeben kann. Zur Demonstration dieser ungenauen Ergebnisse habe ich versucht das Gebäude auf der Anzeige um 90°-Schritte zu drehen. Das Ergebnis ist in Abbildung f.) 6.12 zu sehen. Eine Drehung ist zwar möglich, jedoch sehr ungenau, wie auf dem Bild zu sehen ist.

Dennoch konnte in dieser Arbeit die Visualisierung von Karten-Material, das auf XML-Dateien basierte, umgesetzt werden. Auf der Karte sind zusätzliche Elemente zur Navigation enthalten. Diese wurden durch den Vergleich mit heutiger Navigationssoftware ermittelt, Kapitel 3.2 Tabelle 3.1.

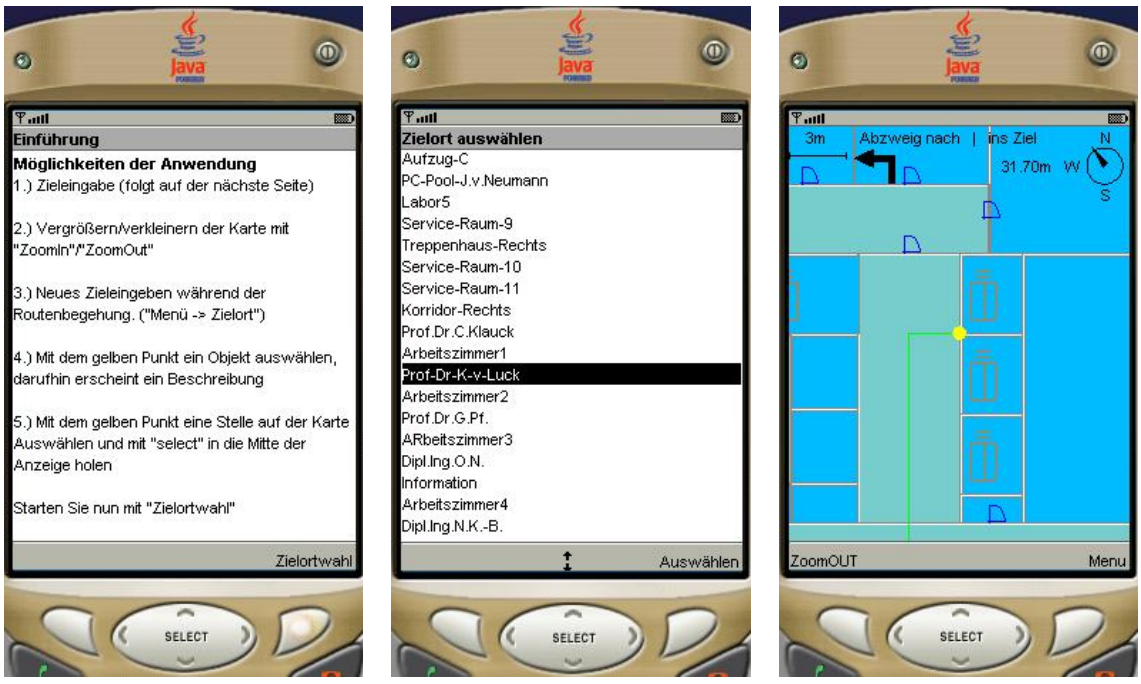
Weiterhin ergab sich eine Schwachstelle beim Refactoring, die wohl mit der Entwicklungsumgebung Eclipse und EclipseME zusammenhängt. In der Manifest-Datei der Anwendung steht ein alter Klassen-Bezeichner. Dieser sorgt dafür, dass die Anwendung auf dem Client zunächst nicht ausführbar. Das Problem wurde auf umständliche Art und Weise behoben. Schwierigkeiten hat ebenfalls der Web-Service bereitet. Dieser ließ sich nicht starten. Der Grund sind wohl fehlende .jar-Dateien, die ich im Netz nicht finden konnte. Daher war eine Instanzierung des Client-Stub und ein Zugriff auf den Web-Service nicht möglich.

Der Innenraum Routenplaner bietet über die Web-Service-Schnittstelle seinen Routenservice an und könnte so für die Übermittlung der Karten- und Kartenelementen sorgen. Aufgrund dessen, dass auf den Web-Service nicht zugegriffen werden konnte, war auch keine Übermittlung der Distanzen möglich. Die Angabe der Distanz auf den Bildern sind dementsprechend reine Schätzungen. Die XML-Dateien wurden kopiert und für die Zwecke der Anwendung genutzt. Die Positionsermittlung mit dem IMAPS-Modul fand bisher noch nicht statt, da derzeit keine Bluetooth-Verbindung vorhanden ist. Die Ermittlung des Standortes ist daher ebenfalls nur angedeutet.

Generell müssen die Geräte, auf denen die Applikation installiert werden soll, Java-fähig sein und wenn sie das sind, sollten sie zusätzlich über bestimmte optionale Pakete verfügen. Das verwendete Smartphone verfügt beispielsweise (herstellerbedingt) über APIs wie die der FileConnection (JSR75). Für beliebige Smartphones oder PDAs ist dies nicht vorauszusetzen, da das Einbinden der APIs dem Hersteller des Gerätes freigestellt ist. Ältere Handys, die der CLDC angehören, verfügen meist nicht über die Kommunikationsverbindung mittels Bluetooth und haben für die Anzeige der Karten monochrome Displays. Somit ist die Anwendung nicht geeignet für ältere Geräte. Diesem API-Chaos und dem daraus resultierenden Schwierigkeiten bei der Portierung von Anwendungen mobile Geräte wird mit der *Mobile Service Architektur*[4] Einhalt geboten. Die API definiert Pakete mit standardisierten APIs und soll bereits Anfang 2006 von einem Großteil von Herstellern und Providern beigefügt werden.

Auf den nun folgenden Seiten soll nun die Anwendung anhand von Bilder dargestellt werden. Sie dokumentieren den Programmablauf, der bei einer Benutzung entstehen könnte.

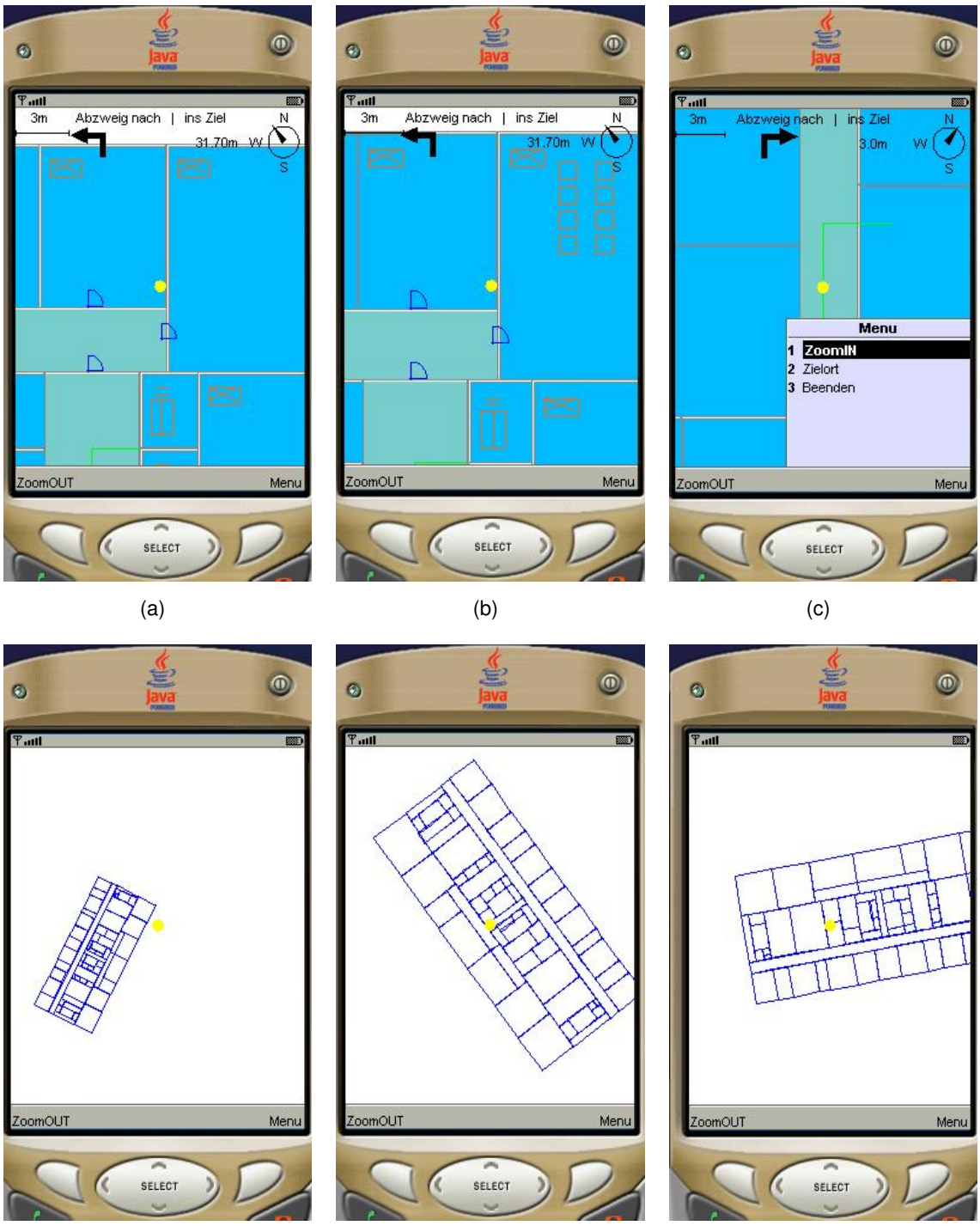
[4] JSR248, MSA



(a) (b) (c)

(d) (e) (f)

Abbildung 6.11.: *a) Das Hauptmenü, b) Wahl eines Zielortes aus einer Liste, c) nach der Routenberechnung Ansicht auf ca. 100qm des Gebäudes (Ausgangszustand),d) Karte während der Routenbegehung, e) ein Tooltip, f) noch ein Tooltip*



(a) (b) (c)

(d) (e) (f)

Abbildung 6.12.: *a) Skalierungsstufe in der nur Mobilar zu sehen ist, b) PCs sind nun dazu gekommen, c) Beispiel einer Karte, die sich mit der Route dreht, d) Karte von der Software um* $90°$*, e) um* $180°$ *und f) um* $270°$ *gedreht.*

7. Zusammenfassung

Das in dieser Arbeit entwickelte Indoor-Navigationssystem bietet einen konzeptuellen und softwaretechnischen Rahmen zur Darstellung von digitalisierten Gebäudedaten in zweidimensionaler Ausbreitungsrichtung aus der Aufsicht.

Bei der Betrachtung anderer akademischer Entwicklungen auf dem Gebiet der Indoor-Navigation ergibt sich als wichtigste Erkenntnis, dass die detail- und maßstabsgetreue Darstellung sowie eine Positionsgenauigkeit der Messungen ausschlaggebend für die erfolgreiche Verwendung in unübersichtlichen Gebäuden ist. Das hierzu verwendete Modell der Visualisierung (PDA) und Positionsbestimmung (IMAPS) stellt dafür eine gut geeignete Grundlage dar.

Als wichtige Anforderungen an die Anwendung haben sich die Aspekte Übertragbarkeit, Zuverlässigkeit, Effizienz sowie Sicherheit erwiesen. Der Entwurf und die Implementierung wird, mit Ausnahme der Sicherheit und der Zuverlässigkeit, ihnen voll gerecht. Obwohl eines der wesentlichen Merkmale, die Sicherheit, aufgrund fehlender Bearbeitungszeit nicht zu realisieren war, ist der Entwurf infolge der Zurückhaltung der personenbeziehbaren Daten darauf ausgelegt. Würde in Zukunft erhöhte Anforderungen an die Sicherheit entstehen, so sollte die Erweiterung keinen großen Aufwand bedeuten. Infolge dessen, dass eine prototypische Anwendung geschaffen wurde, kann auch die Zuverlässigkeit nicht gewährleistet werden. Die Portierbarkeit bleibt durch die Sprachwahl, J2ME, erhalten und das Zeitverhalten ist durch Verwendung eines eigenen Parsing-Algorithmus weiter eingdämmt worden. Zudem wurde gleichfalls auf die unnötige Erzeugung von Objekten geachtet und der Code der Anwendung performant verfasst (keine Methoden-Aufrufe in Schleifen etc.). Die Anforderungen an die Benutzerschnittstelle wie die Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Lernförderlichkeit wurde mit Ausnahme der Fehlertoleranz berücksichtigt. Die Fehlertoleranz konnte aufgrund des Fehlens der Positionsbestimmung, infolge der nicht vorhandenen Bluetooth-Verbindung, nicht ausreichend ausgearbeitet werden.

Das prototypisch realisierte Indoor-Navigationssystem hat die Verwendbarkeit sowohl des Konzeptes als auch der Implementierung deutlich gezeigt. Zudem verspricht die verwendete Infrastruktur eine angemessene Genauigkeit, ist wartungsfreundlich, relativ kostengünstig und kann für hybride Ansätze genutzt werden. Zusammen mit dem zugrundeliegenden

Analyse-, Entwurfs- und Implementierungsprozess entspricht die Lösung den Anforderungen evolutionärer Systementwicklung.

Meiner Meinung nach, zeigen die Ergebnisse weiterhin folgendes: Obwohl ein Indoor-Navigationssystem konzipiert wurde, das in erster Linie in der Lage versetzt werden soll, eine Routenplanung vorzunehmen, ist das entstandene System insbesondere durch die Eigenschaft der Autonomie nicht ausschließlich auf die Navigation beschränkt. Vielmehr kann es dazu genutzt werden, einen virtuellen Rundgang durch die Etagen eines Gebäudes durchzuführen[1]. Die Synergie beider Konzepte, Navigationssystem und Informationssystem, erweist sich hier als nützlich. Über die Anbindung an die Infrastruktur von IMAPS lassen sich schließlich die Konzepte der Navigation realisieren. Zukünftig wird es sicherlich als gutes Mittel für die Darstellung und Beschreibung von Etagen einerseits und für die Navigation für das Umfeld unübersichtlicher Gebäuden andererseits verwendet werden können. Die vorhandene Medien- und Aktorenunabhängigkeit des Modells, die durch die Einhaltung der nicht funktionalen Anforderungen gewährleistet wurde, stellt die für eine breite Nutzung erforderliche Voraussetzung dar.

Auf der Ebene des Entwurfs und der Implementierung lässt sich rückblickend auf die weiteren in Kapitel 1.2 genannten Ziele folgendes feststellen: Die zum Entwurf verwendeten Methoden haben sich bewährt. Insbesondere die Verwendung von Entwurfsmustern fördert das Verständnis von objektorientiertem Entwurf und ermöglichten sehr klare, gut erweiterbare Strukturen. Die verwendete Sprache J2ME ist sehr gut geeignet, um auf hohem Niveau entworfene Systeme zu realisieren. Tatsächlich kann durch die Mächtigkeit der vorhandenen Sprachmittel fast immer auf eine Unterscheidung zwischen Entwurfsmodellen und Implementierungsmodellen verzichtet werden.

Die Java Micro Edition erwies sich als gute Programmiersprache bei der Umsetzung des Indoor-Navigationssystems. Hervorragende Dienste hat auch das optionalen Packages JSR-179 geleistet. Mittels diesen war es möglich, auf gespeicherte Dateien im Dateisystem zuzugreifen. Ebenfalls hätte mittels des JSR-82 auf die Java-Schnittstelle des IMAPS-Modul zurückgegriffen werden können. Zu dem hätten anhand von JSR-184 mobile 3D-Grafiken erstellt werden können.

Erweiterungen und Ergänzungen

Das entwickelte Indoor-Navigationssystem lässt sich weiter ausbauen. Daher soll an dieser Stelle eine Anregung zu Erweiterungen und Ergänzungen des Prototyps erfolgen. Der Nut-

[1] In diesem Zusammenhang möchte ich auch auf Handy-Games hinweisen, die als Grundlage einer ähnlichen Gebäudedatenbasis aufbauen, beispielsweise einem Adventure, anhand dessen ein Benutzer das Gebäude „spielend“ erlernen könnte.

zen aus diesen Anregungen ergibt sich schließlich durch ein komplett hauseigenes Indoor-Navigationssystem.

- Der wichtigste offene Punkt ist sicherlich die fehlende Infrastruktur zur Positionsbestimmung. Die Indoor-Navigation stellt allerdings auch hohe Anforderungen an die Genauigkeit, die mit Hilfe der Infrastruktur erreicht werden sollte. Sie liegt im Bereich von 0,5 bis 2m. Dementsprechend präzise sollte auch die maßstäbliche Darstellung auf der Anzeige sein. Dies sind zwei Gebiete, die noch viel Raum für weitere Untersuchungen bieten.
- Weiterhin von Interesse ist der Aufbau eines flächendeckenden Sensor-Systems. Zu diesem Zweck müssten allerdings entsprechende Empfangstest durchgeführt werden, da, wie in Kapitel 2.3 besprochen, an bestimmten Stellen eine Reflektion der Signale zu erwarten ist.
- Das Empfangs-Modul, das von (28) entwickelt wurde, sollte ebenfalls um eine kabellose Übertragungstechnik ergänzt werden. Eine Anregung wäre beispielsweise das Koordinatenpaar per Bluetooth an den PDA zu übertragen. Als besonders interessant könnte sich dabei die Nutzung von Bluetooth herausstellen. J2ME unterstützt diese Übertragungstechnik, somit könnte die Anwendung dieser Arbeit das Koordinatenpaar in Empfang nehmen.
- Ein einheitlich aufgebautes System von Karten-Daten zur Abbildung von Etagen eines Gebäudes fehlt ebenfalls noch. Die technische Umsetzung ist einfach, würde aber eine gewisse Zeit zur Umsetzung in Anspruch nehmen. Sinnvoll ist solche Arbeit sicherlich erst, wenn eine sichere Positionsbestimmung zu erwarten ist. In diesem Zusammenhang sollten Landmarken erfasst werden. Zu dem sollte über den Karten-Server die Nordausrichtung des Gebäudes bereitgestellt werden, um die Navigationselemente und Kartenelemente zu komplettieren..
- Zur Drehung der Gebäudekarte auf der Anzeige liegt es nahe, eine XML-Datei serverseitig bereitzustellen, die das Gebäude in 90grad-Schritten gedreht enthält. Von so einer Drehung sind schließlich alle Elemente der Karte betroffen. Daher sind diese Dateien, beispielsweise die für Landmarken oder POIs, ebenfalls zu berücksichtigen.
- Ein Szenario bei dem die so genannte Hough-Transformation ihren Einsatz finden könnte wäre die Bildverarbeitung, d.h. beim einlesen von Blaupausen, aus denen die Gebäudedaten errechnete werden müssen, wenn keine CAD-Daten oder ähnliches vorhanden sein sollt. Im Rahmen einer Hough-Transformation wird mit Hilfe von Geradenscharen versucht, auf den Kreuzungspunkt zweier Geraden zu schließen, um damit die genaue Ausrichtung einer Gerade zu erfassen und um diese später um einen Winkel zu drehen. Die Hough-Transformation eignet sich zur Geradenerkennung, wenn keine festen Bezugspunkt die Gerade beschreiben.

- Die Dimension der Anzeige, 3D oder aus der schrägen 2D Aufsicht, ist ebenfalls ein Ansatzpunkt für weitere Ausarbeitungen. So könnten mittels VRML 3D-Karten erstellt und auf einem PDA zur Anzeige gebracht werden. Dabei müssen allerdings große Datenmengen, die infolge von Kartendaten entstehen, übertragen werden.
- Der in dieser Arbeit unberücksichtigte Aspekt der Sicherheit eines Indoor-Navigationssystems stellt ein weiteres Feld für Untersuchungen dar. Hiermit ist insbesondere das Fehlen von grundlegenden Mechanismen für eine eventuelle Authentisierung oder Autorisierung innerhalb des Systems gemeint.
- Der Server-Dienst könnte, gleichfalls um eine Suchfunktion erweitert werden, der für denn Fall, dass ein Benutzer sein Ziel nicht genau kennt, berücksichtigt werden sollte. Ebenfalls sollte der Server-Dienst Element vorhalten, die der Navigation anhand der Karte dienen. Ein definierter Massstab und die Nordausrichtung mit einem Kompass sollten übermittelt werden.

Auch die Ergebnisse weiterer indirekt oder direkt mit demselben Thema befassten Arbeiten können interessant sein. Hier sind beispielsweise Infrastruktur-Modelle mittels W-LAN von besonderem Interesse, um auf Basis von schon vorhandener Infrastruktur ein System aufzubauen. Eine weitere Möglichkeit bietet auch die Erweiterung des Empfangsmoduls um die Auswertung von Signalstärken von Access-Points, um so eine hybride Positionsbestimmung zu nutzen.

Mit dieser Ausarbeitung ist ein weiterer elektronischer Helfer in Form eines Indoor-Navigationssystems entstanden. Dieser ermöglicht eine komfortable Navigation eines Benutzers, der mit seinem PDA in einem Gebäude aufhält. Es lässt sich schnell und effizient eine gewünschte Route, angepasst an individuelle Benutzer-Anforderungen, berechnen. Im Vergleich zur alternativen Navigation stellt die erhebliche Zeitersparnis durch einen elektronischen Helfer einen weiteren Vorteil für den Benutzer dar.

Literaturverzeichnis

[1] : *Map24 - Routenplaner*. 2007. – URL http://www.map24.de

[2] AGRAWALA, M. ; STOLTE, C.: *A Design and Implementation for Effective Computer-Generated Route Maps*. 2000

[3] AITTOLA, M. ; PARHI, P. ; VIERUAHO, M. ; OJALA, T.: *SmartLibrary*. 2004. – URL http://www.kirjasto.oulu.fi/english/palvelut/mobileservices/paikannuspalvelu.html

[4] AT&T: *The Active Badge System*. 2002. – URL http://www.cl.cam.ac.uk/research/dtg/attarchive/ab.html

[5] BARKOWSKY, T. ; FREKSA, C.: *Cognitive requirements on making and interpreting maps.* 1997. – URL http://www.cosy.informatik.uni-bremen.de/spp/SPP_onlines/ProjektB/Cosit97.pdf

[6] BLASCHKE, T.: *Überblick im Ernstfall*. 2005. – URL http://www.dradio.de/dlf/sendungen/forschak/394528/

[7] BRINKHOFF, T. ; WEITKÄMPER, J.: *Visualisierung und interaktive Bearbeitung von Geodaten mit SVG+-geo*. 2004

[8] BUCHHOLZ, C.: *Entwicklung einer standortabhängigen, GPS gestützten Mitfahrbörse auf Smartphones für eine mobile Peer-to-Peer-Community*. 2006

[9] BUTZ, A. ; BAUS, J. ; KRÜGER, A. ; LOHSE, M.: *A hybrid indoor navigation system*. 2001. – URL http://w5.cs.uni-sb.de/~butz/publications/papers/hybrid-iui.pdf#search=%22Indoor-Navigation%22

[10] CAREY, R. ; BELL, G. ; MARRIN, C.: *The Virtual Reality Modeling Language - Part 1*. 1997

[11] CHEVERST, K. ; DAVIES, N. ; MITCHELL, K. ; FRIDAY, Ch.: *Developing a Context-aware Electronic Tourist Guide*. 2000. – URL http://www.comp.lancs.ac.uk/~adrian/Papers/cheverst-guideexperiences-chi00.pdf

[12] CHEWAR, C. ; MCCRICKARD, D.: *Dynamic route descriptions: tradeoffs by usage goals and user characteristics*. – URL citeseer.ist.psu.edu/683677.html

[13] CORRS, V.: *Dreidimensionale Karten für Location Based Services*. 2002

[14] DESTINATOR-TECHNOLOGIES: *Destinator*. – URL http://www.destinatortechnologies.com
lner:2003

[] DÖLLNER, J. ; BAUMANN, K. ; KERSTING, O.: *LandExplorer, Ein System für interaktive 3D-Karten*. – URL http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_Doellner/publication/2003_KS_doellner/DoellnerBaumannKersting_LandExplorer_GeoVis2003.pdf

[15] DPA: *Rechnerisch besitzen 90 Prozent ein Handy*. 2005. – URL www.verivox.de/News/ArticleDetails.asp%3Faid%3D10737+statistik+handy-boom&hl=de&ct=clnk&cd=1&gl=de

[16] DR. ROTH, J.: *Mobile Computin*. dpunkt.verlag, 2006. – ISBN 3-89864-366-2

[17] EISFELLER, B.: Untersuchungen zum GPS-Satellitenempfang in Gebäuden. In: *Forschungsbericht* (2004)

[18] ELIAS, B.: *Extraktion von Landmarken für die Navigation*, DEUTSCHE GEODÄTISCHE KOMMISSION bei der Bayerischen Akademie der Wissenschaften, Dissertation, 2006

[19] ESRI: *ESRI*. 2007. – URL http://de.wikipedia.org/wiki/Esri

[20] FIDURA: *FH Osnabrück und WEBfactory präsentieren gemeinsame Indoor-Navigation-Lösung*. 2007. – URL http://openpr.de/pdf/129174/FIDURA-Fonds-empfehlen-Besuch-der-Hannover-Messe-2007.pdf

[21] GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J.: *Entwurfsmuster*. Addison-Wesley, 2004. – ISBN 3-8273-2199-9

[22] GARMIN: *QueMap*. – URL http://www.garmin.com

[23] GERHARZ, H.J. ; MÜLLER, L.: *Usability of user adapted Indoor Walking Descriptions*. 2006. – URL http://ifgi.uni-muenster.de/~muellerj/lbs06/proceedings/8-IndoorNavigation.pdf

[24] GLEINHOSS: *Ortung*. 2005. – URL http://de.wikipedia.org/wiki/Ortung

[25] GLINZ, M.: *Nicht-funktionale Anforderungen*. 2006. – URL http://www.ifi.unizh.ch/req/ftp/ses/kapitel_12.pdf

[26] GRAUE, N.: *Requirements Engineering Seminarvortrag*. 2004. – URL http://www.fb9dv.uni-duisburg.de/se/de/education/ws0405/RE/NilsGraue(2004)SemRE-NFRAusarbeitung.pdf

[27] GRAUE, N.: *Requirements Engineering Seminarvortrag*. 2004. – URL http://www.fb9dv.uni-duisburg.de/se/de/education/ws0405/RE/NilsGraue(2004)SemRE-NFRAusarbeitung.pdf

[28] GREGOR, S.: *Entwicklung einer Hardwareplattform für die Ermittlung von Positionsdaten innerhalb von Gebäuden*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2006

[29] GROSSKOPF, P.: *JavaME Profile und optionale Pakete*. 2006

[30] HALLBERG, J.: *Bluetooth Positioning*. 2001. – URL http://media.csee.ltu.se/publications/2002/hallberg02bluetooth.pdf

[31] HICKLIN, J. ; BOISVERT, R. F.: *A Java Matrix Package*. 2005. – URL http://math.nist.gov/javanumerics/jama/#Background

[32] HÄUSLER, K.: *Farben Tabelle*. 2007. – URL http://www.farb-tabelle.de/de/farbtabelle.htm

[33] IBACH, P.: *MagicMap*. 2004. – URL http://www2.informatik.hu-berlin.de/rok/MagicMap

[34] IEEE: *W-LAN*. 2007. – URL http://de.wikipedia.org/wiki/IEEE_802.11

[35] ISGPS200: *GPS-Module für Indoor-Navigation*. 2007. – URL http://de.wikipedia.org/wiki/GlobalPositioning_System

[36] JCP: *JSR*. 2007. – URL http://jcp.org/aboutJava/communityprocess/mrel/

[37] JSR-118: *MIDP 2.0*. 2004. – URL http://www.jcp.org/en/jsr/all/

[38] JSR-139: *Connected Limited Device Configuration 1.1*. 2004. – URL http://www.jcp.org/en/jsr/all/

[39] KING, Th. ; HAENSELMANN, Th.: Positionierung mit Wireless-LAN und Bluetooth. In: *K.G. Saur Verlag* (2006), März, S. 9–17

[40] MAIER, W ; KOGAN, B.: *Entwicklung eines mobilen Innenraum Routenplaners als Web Service mit Nutzung von Adhoc auftretenden Diensten*, Hochschule f Diplomarbeit

[41] MICROSYSTEMS, Sun: *Bluetooth-API, JSR 82, Dokumentation*. 2006. – URL http://jcp.org/aboutJava/communityprocess/mrel/jsr082

[42] MUI, A.: *Design and Implementation of an Indoor Mobile Navigation System*, University of California at Berkeley, Diplomarbeit, 2002

[43] NAVIGON: *Navigon 7100*. – URL http://www.navigon.com/

[44] NOKIA: *Technische Informationen über die Test-Hardware NOKIA E70*. 2006. – URL http://www.forum.nokia.com/devices/E70

[45] OESTEREICH, B.: *Analyse und Design mit UML 2.0*. Oldenbourg, 2005. – ISBN 3-486-57654-2

[46] RAUBAL, M.: *Menschlich denkende Navigationssysteme*. 2005. – URL http://ifgi.uni-muenster.de/~raubal/Publications/NonRefJournals/

[47] REISS, F.: *CLDC vs. CDC*. 2004. – URL http://medien.informatik.uni-ulm.de/lehre/courses/ss04/prosem_mobilejava/CLDCvsCDC.pdf

[48] SCHMATZ, K.-D.: *Java 2 Micro Edition*. dpunkt-verlag, 2004. – ISBN 3-89864-271-2

[49] SCHULER, H. ; HELL, B.: *Eignungsdiagnostische Auswahl von Studierenden*. 2006. – URL http://www.uni-heidelberg.de/studium/bologna/beispiel_projektbeschreibung_hohenheim.pdf

[50] SENGPIEL, E.: *Lokalisation und Ortung, gibt es einen Unterschied?* 2002. – URL http://www.sengpielaudio.com/LokalisationUndOrtung.pdf

[51] SOWIZRAL, H. ; RUSHFORTH, K. ; DEERING, M.: *The Java 3D API Specification. 2nd Edition*. Addison-Wesley, 2000. – ISBN 978-0201710410

[52] STEINIGER, St. ; NEUN, M.: *Location Based Services*. 2007. – URL http://www.geo.unizh.ch/publications/cartouche/lbs_lecturenotes_steinigeretal2006.pdf

[53] STRAUSS, P. ; CAREY, R.: *An Object-Oriented 3D Graphics Toolkit*. – URL http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_Doellner/publication/2001_WebMap_doellner/doellner_virtuelleKartenmodelle.pdf

[54] SUN: *J2ME Building Blocks for Mobile Devices*. 2000. – URL http://java.sun.com/products/cldc/wp/KVMwp.pdf

[55] SUN: *Java APIs*. 2007. – URL http://www.jcp.org/en/jsr/all/

[56] TANENBAUM, A. ; STEEN, M. van: *Verteilte Systeme*. Pearson Studium, 2003. – ISBN 3-8273-7057-4

[57] TARKIAINEN, M. ; KAUVO, K. ; KAIKKONEN, J. ; HEINE, H.: *Simple turn-by-turn route guidance for pedestrians*. 2001

[58] TOMBERGE, P.: *Navigation mittels RFID*, Westfälischen Wilhelms-Universität Münster, Diplomarbeit, 2004

[59] TOMTOM: *Navigator 5*. – URL http://www.TomTom.com

[60] TVERSKY, B.: *Distorts in Cognitive Maps*. 1993

[61] UDDI: *Universal Description, Discovery and Integration*. 2006. – URL www.uddi.org

[62] VANETTI, E. J. ; ALLEN, G. L.: *Communicating environmental knowledge : The impact of verbal and spatial abilities on the production and comprehension of route directions.* 1988. – URL http://citeseer.ist.psu.edu/context/2737847/0

[63] W3C: *World Wide Web Consortium*. 2004. – URL www.w3c.org

[64] W3C: *SOAP Spezification*. 2006. – URL http://www.w3.org/TR/soap

[65] W3C: *Web Service Definition Language*. 2006. – URL www.w3.org/TR/wsdl

[66] W3C: *Web Services Activity*. 2006. – URL http://www.w3.org/2002/ws

[67] W3C: *SVG-Scalable Vector Graphics*. 2007. – URL http://de.wikipedia.org/wiki/Scalable_Vector_Graphics

[68] WEISER, M.: *UbiComp*. 1991. – URL http://www.informatik.uni-rostock.de/mmis/courses/ws0607/23167/06-weiser-parc.pdf

[69] WELIE, A. van: *Patterns in interaction design*. 2007. – URL http://www.welie.com/patterns/index.php

A. Anhang

A.1. Anhang A

Folgendes Code-Beispiel zeigt eine einfache Grafik:

```
<svg
    width="600"
    height="800"
    </defs>
    <rect style=ßtroke:none;
        fill:#7f7f7f;
        fill-opacity:100%;
        fill-rule:evenodd;
        stroke-opacity:100%;
        stroke-width:1px;
        stroke-linejoin:miter;
        stroke-linecap:butt; "
    id="rect4"
    x="100"
    y="150"
    width="300"
    height="150»
</rect>
</svg>
```

Code-Beispiel: Grundgerüst einer SVG-Grafik

Dieses einfache Beispiel zeichnet ein ausgefülltes Rechteck.

A.2. Anhang B

Pseudo-Code des XML-Parsers Durchsucher.java:

Die Methode viewXML parsed die XML Datei

```
public viewXML(EingabeStrom)
    SET x1 to zero,x2 to zero,y1 to zero,y2 to zero
    SET id to null,name to null
    SET vectorFürRaum to null
    SET array = fügeEingabeStromInByteArrayEin(EingabeStrom)
    WHILE array not end of file
        SET bz to next array element of bar
        SET b to char array size of 30
        IF next arras element of bar = 'i' THEN
            IF next array element of bar = 'd' THEN
                IF next array element of bar = ' ' THEN
                    WHILE next array element of bar != "
                    ENDWHILE
                    WHILE ins = next array element of bar != "
                        b at ic = char value ins
                        INCREMENT ic
                        id = string value of b
                    ENDWHILE
                    continue
                ENDIF
            ENDIF
        ENDIF
        IF next arras element of bar = 'n' THEN
            IF next array element of bar = 'a' THEN
                WHILE next array element of bar != "
                ENDWHILE
                WHILE ins = next array element of bar != "
                    b at ic = char value ins
                    INCREMENT ic
                ENDWHILE
                SET i1 to zero
                WHILE next b element != ' '
                ENDWHILE
                SET b1 to char array size of i1
```

```
                FOR each b1
                    SET b to b1
                ENDFOR
                SET name to string value of b1
                continue
        ENDIF
ENDIF
IF next arras element of bar = 'x' THEN
        IF next array element of bar = '1' THEN
                WHILE next array element of bar != ";
                ENDWHILE
                INCREMENT z
                WHILE ins = next array element of bar != ";
                    SET c to c plus char value of ins
                ENDWHILE
                SET x1 to integer value of c
                continue
        ENDIF
        IF next array element of bar = '2' THEN
                WHILE next array element of bar != ";
                ENDWHILE
                INCREMENT z
                WHILE ins = next array element of bar != ";
                    SET c to c plus char value of ins
                ENDWHILE
                SET x2 to integer value of c devided by 20
        continue
        ENDIF
ENDIF
IF next arras element of bar = 'y' THEN
        IF next array element of bar = '1' THEN
                WHILE next array element of bar != ";
                ENDWHILE
                INCREMENT z
                WHILE ins = next array element of bar != ";
                    SET c to c plus char value of ins
                ENDWHILE
                SET y1 to integer value of c devided by 20
                continue
        ENDIF
```

```
                IF next array element of bar = '2' THEN
                    WHILE next array element of bar != ";
                    ENDWHILE
                    INCREMENT z
                    WHILE ins = next array element of bar != ";
                        SET c to c plus char value of ins
                    ENDWHILE
                    SET y2 to integer value of c devided by 20
                    SET room to rectangle-room with values x1, x2, y1, y2, id, name
                    SET room add to rooms
                    continue
                ENDIF
            ENDIF
        ENDWHILE
        SET bar to null
ENDPROCEDURE
```

Auf die gleiche Art und Weise verfahren auch DoorSniffer.java und RootSniffer.java.

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 21. September 2007	
Ort, Datum	Unterschrift