Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Mark Thomé

Ortsbezogene Dienste im Paradigma des Web 2.0

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Mark Thomé

Ortsbezogene Dienste im Paradigma des Web 2.0

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. Olaf Zukunft
Zweitgutachter : Prof. Dr. Kai von Luck

Abgegeben am 8. Mai 2007

Mark Thomé

Ortsbezogene Dienste im Paradigma des Web 2.0

Stichworte

Ortsbezogene Dienste, Web 2.0, soziale Software, Ubiquitous-Computing, Smart-Client

Kurzzusammenfassung

Mobile ortsbezogene Dienste haben sich trotz ihres offensichtlichen Mehrwerts bisweilen nicht etabliert. Aktuelle technische Fortschritte im Bereich mobiler Informationssysteme, sowie das Verständnis von Diensten und das Verhalten von Nutzern im Internet der zweiten Generation, dem so genannten Web 2.0, bergen nach Meinung des Autors enormes Potential für ortsbezogene Dienste. In dieser Arbeit wird untersucht, ob die aus diesen aktuellen Veränderungen resultierenden Synergieeffekte derartigen Diensten zum Durchbruch bei mobilen Anwendungen verhelfen können. Dazu werden Berührungspunkte diskutiert und in Anforderungen für ein exemplarisches mobiles Informationssystem überführt. Durch den Entwurf, die prototypische Implementierung und eine erste Evaluierung der mobilen Smart-Client- und der stationären Server-Anwendung wird diese These gefestigt.

Mark Thomé

Location-based services in paradigm of Web 2.0

Keywords

location-Based services, Web 2.0, social software, ubiquitous computing, smart client

Abstract

Mobile location-based services have not been a success yet, although they provide added value. Current proceedings in the area of mobile software as well as the understanding of services and the behavior of users within the second generation of the Internet, dubbed Web 2.0, contain enormous potential for location-based services. Within this thesis the author discusses the objective whether the resulting synergy effects due to these changes can turn mobile location-based services into success. Therefore certain topics are discussed and transformed into requirements for an exemplary mobile information system. By the design, the prototypical implementation and the first evaluation of the mobile smart client and stationary server software this assumption is proofed.

Für Mama und Papa

Danksagung

Diese Arbeit stellt den Abschluss meines Informatikstudiums dar. Dem Studium habe ich immerhin mehr als ein Fünftel meiner bisherigen Lebenszeit gewidmet und ich muss gestehen, es war auch nicht immer leicht, die Begeisterung für das Studium beizubehalten. Doch Kommilitonen - die zu Freunden wurden - haben mir geholfen meinen Weg zu gehen. An dieser Stelle möchte ich insbesondere Tobias Krause und Thies Rubarth dafür danken.

Bei Prof. Dr. Olaf Zukunft und Prof. Dr. Kai von Luck möchte ich mich für die hervorragende Zusammenarbeit während dieser Arbeit und während des gesamten Studiums bedanken. Weiterhin möchte ich Matthias Neugebauer für die zahlreichen konstruktiven Rückmeldungen zu dieser Arbeit danken.

Mein besonderer Dank gilt Marlene Dütting, die mir mit ausführlichen Korrekturarbeiten zu dieser Arbeit geholfen und mich auch immer wieder aufgebaut hat, wann immer es notwendig war.

Mark Thomé, im Mai 2007

Inhaltsverzeichnis

Abbildungsverzeichnis

Listings

1 Einleitung

Mobile Informationssysteme spielen in der heutigen Informationsgesellschaft eine wichtige Rolle. Nahezu jede Person besitzt ein mobiles Gerät, wie z. B. einen Personal Digital Assistant (PDA) oder ein Mobilfunktelefon, auf dem anspruchsvolle Anwendungen lauffähig sind. Diese Applikationen steigern ihren Nutzen durch den Austausch von Daten mit anderen Anwendungen oder einem zentralen Dienst unter Nutzung des Internets.

1.1 Motivation

Da mobile Nutzer ihre Anwendungen an verschiedenen Orten nutzen und technische Voraussetzungen existieren, um diese Nutzer bzw. die Anwendungen und Geräte geografisch zu lokalisieren, können ortsbezogene Dienste (engl.: Location-Based Services) angeboten werden. Diese können bisherigen Anwendungen einen Mehrwert liefern oder sogar komplett neue Anwendungen entstehen lassen.

Durch Ortungstechnologien und ortsbezogene Dienste können z. B. bestehende Notfallsysteme so erweitert werden, dass bei einem telefonischen Notruf mit einem Mobilfunktelefon die Notfallzentrale das Gerät ortet und so eventuell eine präzisiere Ortsangabe erhält, als sie der Nutzer selber liefern könnte. Auch Nachrichten, wie z. B. der Wetterbericht, nur bezogen auf den aktuellen Aufenthaltsort des Nutzers, können einen Mehrwert bringen.

Eine durch ortsbezogene Dienste mögliche Anwendung ist die heute weit verbreitete Kraftfahrzeugnavigation mittels GPS-Technik. Auch die Lokalisierung von Personen oder Objekten schafft gänzlich neue Anwendungsgebiete. So können die Positionen der Fahrzeuge eines Taxiunternehmens permanent an die Taxizentrale übermittelt werden, so dass der Einsatz der Fahrzeuge effizient geplant werden kann. Mit einer anderen Anwendung könnte ein Nutzer in einer fremden Stadt Anfragen nach Orten stellen, die ihn interessieren (engl.: Points of Interests).

Obwohl diese neuartigen bzw. mehrwertigen Anwendungen auf breites Interesse stoßen, haben sie sich bisher nicht ausreichend durchgesetzt. Als Gründe werden häufig die nicht zufriedenstellende Technik, sowie die Komplexität ortsbezogener Dienste angeführt. So sind

an der Realisierung und Bereitstellung eines Dienstes in der Regel mehrere Unternehmen beteiligt (Martens u. a. (2006)).

Die technischen Voraussetzungen für ortsbezogene Dienste entwickeln sich permanent weiter. Durch das neue Mobilfunknetz Universal Mobile Telecommunications System (UMTS) werden höhere Datenübertragungen ermöglicht, das europäische Satellitennavigationssystem Gallileo verspricht eine genauere Positionsbestimmung und die neueste Generation mobiler Geräte kann ortsbezogene Dienste mit hochauflösenden Umgebungskarten darstellen. Darüber hinaus werden diese Karten kostenlos von Firmen wie Google, Microsoft und Yahoo in Produkten für lokale Suchen zur Verfügung gestellt.

Neben der Technik ändert sich auch das Verständnis von Diensten und Dienstleistungen im Internet. Das Internet befindet sich aktuell im Aufbruch in die zweite Generation, dem so genannten Web 2.0. Eine Besonderheit im Paradigma[1] des Web 2.0 ist die Tatsache, dass Daten von Diensten und Web-Seiten aus dem Internet nicht mehr von einigen wenigen, z. B. Firmen, bereitgestellt und verwaltet werden. Vielmehr sind es die Nutzer selber, die Inhalte generieren und pflegen (O'Reilly (2005)).

In einer aktuellen Studie des US-amerikanischen Marktforschungsunternehmens Gartner (2006a) wird auf die Bedeutung des Web 2.0 für die Zukunft der Informationstechnologie eingegangen. Abb. 1.1 zeigt den Gartner-Hype-Cycle[2] aktueller Technologien. Deutlich wird, dass das Web 2.0 auf dem „Gipfel hoher Erwartungen" angekommen ist. Nach Meinung der Autoren der Studie werden nicht alle Erwartungen erfüllt, die zu diesem Zeitpunkt an eine Technologie gestellt werden. Dennoch messen die Autoren dem Web 2.0 die Bedeutung zu, die Informationstechnologie grundlegend zu ändern und in weniger als zwei Jahren deutlich an Relevanz zu gewinnen. In der Abbildung ist ebenfalls erkennbar, dass Ortungstechnologien und Ortungsanwendungen das „Tal der Desillisionierung" durchschritten haben und nun zu einsetzbaren Technologien und Anwendungen reifen. Gleiches gilt für Tablet PCs und Smartphones; diese können mobile Anwendungen für ortsbezogene Dienste anspruchsvoll bereitstellen.

Der Autor hat im Rahmen des Technologiewettbewerbs Imagine Cup 2006 (Microsoft (2006b)) der Firma Microsoft zusammen mit seinen drei Kommilitonen Sven Stegelmeier, Martin Stein und Piotr Wendt die Projektidee und die Client-Server-Software eines mobilen Navigationssystems für Rollstuhlfahrer entwickelt. Die Trailblazers genannte Software bietet den Nutzern eine barrierefreie Routenplanung. Die Grundlage für die Routen bilden via Rollstuhl zugängliche Wege sowie Orte von Interesse, die von den Nutzern selbst implizit und

[1] Im Allgemeinen ist ein Paradigma ein Denkmuster oder Musterbeispiel, also ein Beispiel, das typisch für eine Klasse von Beispielen ist. Nach Claus und Schwill (2003) bezeichnet ein Paradigma in den Methoden der Informatik ein übergeordnetes Prinzip, das für eine ganze Teildisziplin typisch ist, sich aber nicht klar ausformulieren lässt, sondern nur an Beispielen manifestiert werden kann.

[2] Der Hype-Cycle dient dazu, die Entwicklung aufkommender Technologien aufzuzeigen und deren Relevanz und Rolle für die Märkte einzuordnen.

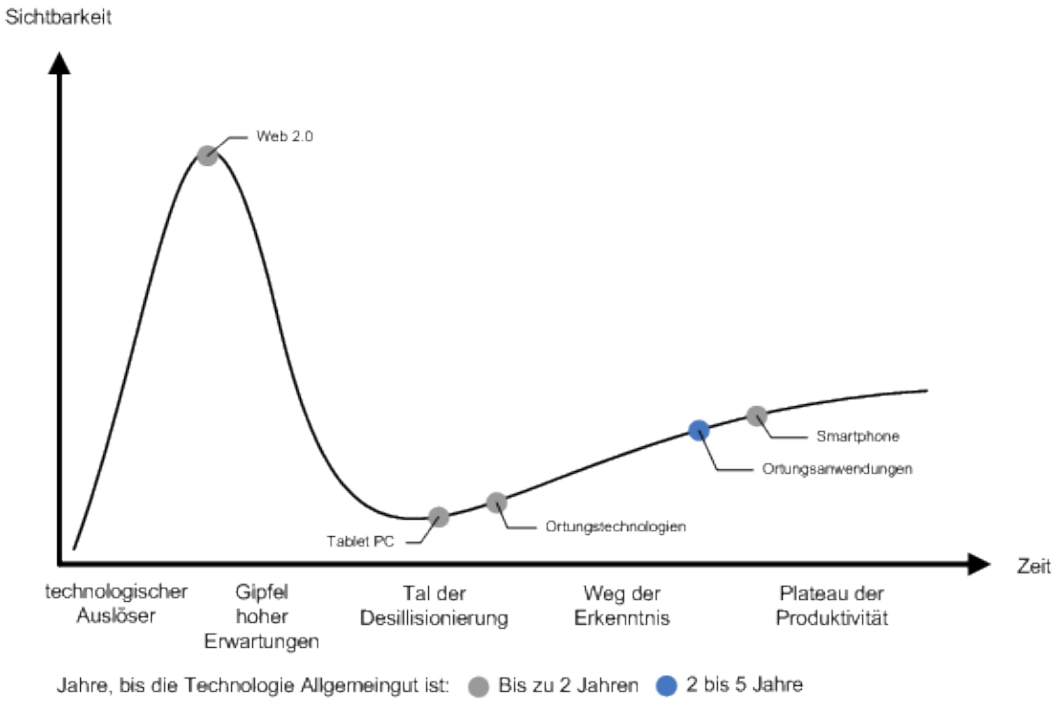

Abbildung 1.1: Gartners Hype-Cycle 2006 für aufkommende Technologien (Gartner (2006a))

explizit erfasst werden. Somit verbindet Trailblazers mobile, ortsbezogene Dienste mit dem Internet der 2. Generation.

1.2 Ziele der Arbeit

In dieser Arbeit werden ortsbezogenen Dienste und deren Einsatz im Paradigma des Web 2.0 untersucht. Dabei soll gezeigt werden, dass sich ortsbezogenen Diensten in der neuen Generation des Internet eine Reihe interessanter Möglichkeiten bietet, um zum Schlüsselfaktor für mobile Anwendungen zu werden. Ziel der Arbeit ist dazu der Software-Entwurf und die zugehörige, prototypische Implementierung einer neuen, erweiterten Trailblazers-Software um diese Hypothese zu festigen. Dabei gilt es, den aktuellen Stand von Forschung und Entwicklung zu reflektieren, jedoch ausschließlich verfügbare Technologien einzusetzen. Dazu werden bestimmte Anwendungsfälle skizziert, die der Veranschaulichung und Diskussionsgrundlage für diese aktuellen Fragestellungen im Zusammenhang mit ortsbezogenen Diensten dienen sollen.

Die Arbeit ist dem Fachgebiet des Software-Engineering zuzuordnen. Die Entwicklung einer mobilen Anwendung auf Basis ortsbezogener Dienste ist eine relativ neue Disziplin in diesem

Bereich, die sich zudem rasant weiterentwickelt. Aktuelle Literatur geht häufig von mobilen Informationssystemen auf Basis des Wireless Application Protocols (WAP) aus. Diese nicht mehr zeitgemäße Betrachtung soll mithilfe des gewählten Szenarios auf den aktuellen Stand der Technik aktualisiert werden.

1.3 Inhaltlicher Aufbau der Arbeit

Die Vorgehensweise in dieser Arbeit orientiert sich am Unified-Process (Jacobson u. a. (1999)), was durch die einzelnen Arbeitsschritte Analyse, Entwurf und Implementierung verdeutlicht wird. Diese Arbeitsschritte werden in einzelnen Kapiteln gesondert betrachtet; insgesamt gliedert sich die Arbeit in sechs Kapitel.

Im Anschluss an diese Einleitung werden in Kap. 2 alle erforderlichen Grundlagen für die Entwicklung des mobilen Informationssystem aufgezeigt. Dazu werden zunächst in Abschn. 2.1 die besonderen Eigenschaften aktueller, mobiler Informationssysteme skizziert. Daran anschließend werden in Abschn. 2.2 ortsbezogene Dienste beschrieben; dabei werden insbesondere Eigenschaften und Anforderungen an diese Dienste herausgestellt. Die für ortsbezogene Dienste erforderlichen Techniken zur Positionsbestimmung werden in Abschn. 2.3 aufgezeigt. Geografische Informationssysteme bilden weitere wichtige technische Grundlagen ortsbezogener Dienste und werden in Abschn. 2.4 umrissen. Das Kapitel Grundlagen wird in Abschn. 2.5 mit einem Überblick über die Merkmale des Web 2.0 abgeschlossen. Dabei werden neue Anwendungen des Web 2.0, sowie eine Abgrenzung zum bisherigem Web aufgezeigt.

Zu Beginn von Kap. 3 wird in Abschn. 3.1 ein vorhandener ortsbezogener Dienst als Ausgangsszenario für diese Arbeit vorgestellt. Dieser Dienst wurde im Kontext der Arbeit erweitert. Dazu wird in Abschn. 3.2 diskutiert, wie ortsbezogene Dienste und Dienste des Web 2.0 gewinnbringend vereint werden können. Dabei werden Akteure, Dienste, Daten, Geschäftsmodelle und die Privatsphäre der Nutzer bei derartigen Diensten gesondert betrachtet. Die in dieser Diskussion gewonnenen Erkenntnisse werden genutzt, um die funktionalen (Abschn. 3.3) und nichtfunktionalen Anforderungen (Abschn. 3.4) für das in dieser Arbeit entwickelte mobile Informationssystem zu definieren.

Ausgehend von der Analyse wird in Kap. 4 der Software-Entwurf des entwickelten Informationssystems diskutiert. Dazu werden in Abschn. 4.1 mögliche Architekturalternativen voneinander abgegrenzt und bewertet. Danach wird in Abschn. 4.2 die gewählte Anwendungsarchitektur skizziert. Der software-technische Entwurf wird in den Abschnitten 4.3 bzw. 4.4 für die stationäre Server-Anwendung bzw. für die mobile Client-Anwendung detailliert vorgestellt.

Die praktische Implementierung des Software-Entwurfs wird in Kap. 5 beschrieben. Dazu wird in Abschn. 5.1 eine Übersicht über den Implementierungsstatus gegeben. Danach werden die verwendete Hard- (Abschn. 5.2) und Software (Abschn. 5.3), sowie die genutzten externen Software-Komponenten (Abschn. 5.4) aufgelistet. Des weiteren werden die gewählten Maßnahmen zur Quaalitätssicherung bei der Implementierung erläutert (Abschn. 5.5) und abschließend ausgewählte Implementierungsdetails ausführlich besprochen (Abschn. 5.6).

Eine erste Evaluierung des entwickelten Informationssystems wird in Kap. 6 beschrieben. Dazu werden die Versuchsziele (Abschn. 6.1), der Versuchsaufbau (Abschn. 6.2) und die Versuchsergebnisse (Abschn. 6.3) beschrieben.

Zum Abschluss werden in Kap. 7 die Erkenntnisse dieser Arbeit zusammengefasst (Abschn. 7.1) und bewertet (Abschn. 7.2). Ferner wird ein Ausblick auf mögliche Verbesserungen und Weiterentwicklungen gegeben (Abschn. 7.3).

1.4 Verwandte Arbeiten

Die Entwicklung der Trailblazers-Software im Rahmen des Imagine Cup ist abgeschlossen. Die Autoren der Trailblazers-Software haben ihre bisherigen Ergebnisse unter Stegelmeier u. a. (2006) veröffentlicht. Der Autor wird seine Komponenten aus dem Trailblazers-Projekt des Imagine Cups in dieser Arbeit weiterentwickeln und vertiefen. Martin Stein und Piotr Wendt werden ihre Masterarbeiten ebenfalls auf Grundlage der Trailblazers-Software aufbauen, allerdings mit anderen Ausrichtungen und Zielen, so dass eine Abgrenzung zwischen den drei Masterarbeiten gegeben ist.

Darüber hinaus hat sich der Autor in einer Seminararbeit (Thomé (2005b)) mit Fragestellungen zu Informationssystemen für ortsbezogene Dienste beschäftigt. Diese konzeptionellen Betrachtungen dienten dem Autor als Grundlage für eine Projektarbeit (Thomé (2006)), in der er ein mobiles Informationssystem für ortsbezogene Dienste prototypisch entwickelt hat. Diese Arbeiten werden indirekt mit dieser Masterarbeit fortgeführt.

Nach Kenntnis des Autors gibt es zur Zeit nur wenige Arbeiten und Projekte, die mobile ortsbezogene Dienste mit den Möglichkeiten des Web 2.0 verbinden. Dennoch gibt es viele angrenzende Projekte im Bereich der ortsbezogenen Dienste. Unter anderem werden in den Veröffentlichungen von Retscher (2004), Akasaka und Onisawa (2004) und Helal u. a. (2004) Navigationssysteme für Fußgänger vorgestellt. Letztere Arbeit stellt, ähnlich wie die vorliegende Arbeit, körperlich eingeschränkte Nutzer in den Mittelpunkt. Das Microsoft Research Projekt SLAM (Microsoft-Research (2006)) verbindet ortsbezogene Dienste mit virtuellen, sozialen Netzwerken von Nutzern, ebenfalls ein Merkmal des Web 2.0.

2 Grundlagen

Dieses Kapitel zeigt relevante, zum weiteren Verständnis notwendige Grundlagen auf. Zunächst werden Eigenschaften, Anforderungen und Laufzeitsysteme für mobile Informationssysteme skizziert. Danach werden ortsbezogene Dienste charakterisiert und deren verschiedenen Eigenschaften und Anforderungen voneinander abgegrenzt. Die für diese Dienste erforderlichen Techniken zur Positionsbestimmung werden aufgezeigt. Geografische Informationssysteme beinhalten weitere technische Voraussetzungen für ortsbezogene Dienste, die herausgearbeitet werden. Abschließend werden die Eigenschaften des Web 2.0 skizziert und eine Abgrenzung zum bisherigen Web herausgestellt.

2.1 Mobile Informationssysteme

Mit dem Jahr 2000 begann die Ära des Ubiquitous Computing (Weiser (1991)). Seitdem besitzt nahezu jede Person ein oder mehrere mobile Geräte, auf denen anspruchsvolle Anwendungen lauffähig sind. Die Leistungsfähigkeit mobiler Hardware nimmt weiter zu; im Gegenzug werden die Anforderungen an mobile Informationssysteme immer komplexer. Verschiedene Anwendungsmöglichkeiten finden sich für jegliche Art von Geräten. Die mobilen Rechner dienen z. B. der Kommunikation, der Verwaltung persönlicher Informationen, der Beschaffung von Informationen via Internet, oder auch der Unterhaltung.

2.1.1 Eigenschaften

Für Informationssysteme existieren eine Reihe von Definitionen. In dieser Arbeit ist im Wesentlichen gemeint, dass ein solches System, bestehend aus Mensch und Maschine, Informationen gewinnen, speichern und nutzen kann (Hansen und Neumann (2005)).

Mobile Anwendungen besitzen die gleichen Eigenschaften wie stationäre Anwendungen, haben darüber hinaus allerdings weitere Eigenschaften, die durch die Dimension der Mobilität entstehen (B'Far (2005)).

Netzwerkanbindung Mobile Anwendungen kommunizieren in der Regel über eine drahtlose Schnittstelle mit anderen Anwendungen.

Ortsunabhängigkeit Dadurch, dass Nutzer Anwendungen auf mobilen Geräten an beliebigen Orten aufrufen und auf lokale und entfernte Daten und Dienste transparent zugreifen können, wird Ortsunabhängigkeit erreicht.

Ortung Da mobile Anwendungen auf Endgeräten ausgeführt werden, die physikalisch mobil sind, ergibt sich die Möglichkeit der Ortung. Diese ist entstanden aus der technischen Notwendigkeit, eine Verbindung zwischen zwei Funkzellen eines Mobilfunknetzwerkes aufrechtzuerhalten (Roth (2005)). Die Möglichkeit der Ortung bildet die Grundlage ortsbezogener Dienste.

2.1.2 Anforderungen

Es existiert eine Reihe von Anforderungen an mobile Informationssysteme, die im Wesentlichen auch für stationäre Informationssysteme gelten, aufgrund der oben genannten Eigenschaften aber besonders wichtig sind.

Netzwerkanbindung Da mobile Informationssysteme über drahtlose Schnittstellen kommunizieren, ist der Zugriff auf entfernte Dienste im Netzwerk nicht immer gewährleistet. So müssen entfernte Dienste bzw. mobile Anwendungen anderer Nutzer adäquaten Ersatz finden, wenn ein mobiles Gerät nicht erreicht werden kann. Andersherum muss die mobile Anwendung auch lokal einsatzfähig sein. Dazu ist es notwendig, dass mobile Anwendungen Daten auf dem mobilen Gerät zwischenspeichern, die dann in dem vorliegenden *Online-Offline-Szenario* bei bestehender Netzwerkverbindung synchronisiert werden müssen (B'Far (2005)). Darüber hinaus muss bei bestehender Verbindung die zur Verfügung stehende Übertragungsbandbreite berücksichtigt, sowie ein Netzwerkwechsel bzw. ein Wechsel der IP-Adresse erkannt werden. Einige Dienste erfordern zudem, dass mobile Anwendungen, durch einen so genannten *Push* vom Dienst benachrichtigt werden können.

Beschränkte Ressourcen Mobile Endgeräte haben im Vergleich zu stationären Geräten meistens beschränkte Ressourcen, wie z. B. Prozessorleistung, Speicherkapazität und Stromversorgung (Roth (2005)). Mobile Anwendungen müssen dies berücksichtigen.

Heterogene Hardware Mobile Endgeräte als Plattform für mobile Informationssysteme sind sehr heterogen. Große Unterschiede sind in den vorhandenen Ressourcen und in der Benutzerschnittstelle auszumachen (B'Far (2005)). Mobile Informationssysteme sollten so konzipiert und implementiert werden, das diese möglichst unabhängig von bestimmten Plattformen sind. Wichtig ist dabei, dass eine Anwendung gut auf eine Plattform zugeschnitten ist.

Sicherheit Eine angemessene Sicherheit[1] und Identifizierbarkeit zu gewährleisten, ist eine der größten Herausforderungen bei mobilen Informationssystemen. Ein Sicherheitsmodell muss die vier Teilbereiche Geräteebene, Übertragungsebene, Netzebene und Anwendungsebene berücksichtigen (Mutschler und Specht (2004)).

2.1.3 Anwendungsarchitekturen

Bei Architekturen verteilter Client-Server-Anwendungen wird in aller Regel zwischen leichtgewichtigen und schwergewichtigen Clients unterschieden. Diese basieren auf einer konzeptionellen Drei-Schichten-Software-Architektur, in der die Schichten Präsentations-, Anwendungs- und Datenhaltungsschicht die grundsätzlichen Aufgaben von Software-Systemen widerspiegeln. Sind logische Schichten zudem physikalisch verteilt, werden sie oftmals als *Tier* bezeichnet (vgl. Tanenbaum und van Steen (2003)). Leichtgewichtige Clients dienen lediglich der Präsentation von Daten und der Interaktion mit dem Benutzer; insbesondere befindet sich die Datenverarbeitung und -haltung auf dem Server. Typischer Vertreter sind Web-Browser basierte Anwendungen, bei schwergewichtigen Clients hingegen wird die Datenverarbeitung auf dem Client durchgeführt. Die Daten können zudem, in Teilen oder komplett, auf den Clients gehalten werden.

2.1.3.1 Leichtgewichtige Clients

Die Installation von Anwendungen für leichgewichtige Clients ist unkompliziert, da die Anwendungen zentral auf einem Server installiert werden, wie z. B. bei Web-Applikationen. Entsprechend ist eine einfache Aktualisierung der Anwendungen möglich. Die Hardware-Anforderungen an das Client-Gerät sind gering, da die gesamte Datenverarbeitung und -speicherung auf einem Server durchgeführt wird.

Die Benutzerschnittstelle dagegen ist eingeschränkt und in vielen Fällen der Eingabe mit einem Formular vergleichbar. Die Techniken zur Darstellung von Informationen auf leichtgewichtigen Clients entwickelt sich jedoch weiter. So ist es mit den in Abschn. 2.5.1 vorgestellten neuen Technologien möglich, eine interaktive und den schwergewichtigen Clients ähnliche Benutzerschnittstelle zu schaffen. Der leichtgewichtige Client muss während der Ausführung permanent mit einem Server verbunden sein, daher sind nur Online-Szenarien möglich.

[1] Sicherheit im Sinne des englischen Begriffs *security*.

2.1.3.2 Schwergewichtige Clients

Die Benutzerschnittstelle von Anwendungen für schwergewichtige Clients ist flexibel und kann vielen Anforderungen gerecht werden. Viele Anwendungen für schwergewichtige Clients müssen nicht permanent über ein Netzwerk mit dem Server verbunden sein. Insbesondere dann, wenn Daten auch dem Client zwischengespeichert werden, ergeben sich gute Antwortzeiten bei der Datenverarbeitung. Werden gleiche Daten auf verschiedenen Clients gleichzeitig geändert, wie z. B. bei E-Mail- oder Groupware-Anwendungen, müssen die Daten später auf dem Server wieder konsolidiert werden.

Anwendungen für schwergewichtige Clients müssen immer installiert werden. Dies kann zwar mit Hilfsprogrammen von zentraler Stelle aus geschehen, ist aber nicht so einfach durchführbar wie bei leichtgewichtigen Clients. Da die Verarbeitung und die Datenhaltung auch auf dem Client stattfinden, ist eine höhere Anforderung an die Hardware gegeben. Zudem ist der Implementierungsaufwand deutlich höher.

2.1.3.3 Smart-Clients

Neben leichtgewichtigen und schwergewichtigen Clients hat sich eine dritte Art von Anwendungsarchitekturen etabliert. Bei mobilen Anwendungen, die für komplexe Aufgaben konzipiert sind, wird immer häufiger eine Smart-Client-Anwendungsarchitektur (vgl. Microsoft (2006g)) eingesetzt. Darauf basierende Anwendungen erheben für sich den Anspruch, die Vorteile von leichtgewichtigen und schwergewichtigen Clients zu vereinen, ohne deren Nachteile aufzuweisen. Smart-Client-Anwendungen können auf externe Hardware, wie z. B. GPS-Empfänger, zugreifen. Sie bieten flexible Benutzerschnittstellen, die an die Leistungsfähigkeit schwergewichtiger Clients heran reichen. Sie sind sowohl in einem Online-, als auch in einem Offline-Szenario einsetzbar und Daten können lokal verarbeitet und gespeichert werden. Es besteht die Möglichkeit, bei bestehender Netzwerkverbindung auf entfernte Daten zuzugreifen. Die mobilen Anwendungen können einfach installiert werden, indem diese auf das mobile Gerät geladen werden. Smart-Client-Anwendungen werden dabei nicht direkt für eine Hardware-Plattform implementiert, vielmehr benötigen sie zur Ausführung eine Laufzeitumgebung.

2.1.4 Mobile Laufzeitumgebungen

Um der oben genannten Forderung nach Unterstützung heterogener Hardware zu entsprechen, werden mobile Anwendungen häufig nicht für eine bestimmte Hardware, sondern für eine Laufzeitumgebung entworfen. Diese Systeme sind meist für verschieden Hardware-Plattformen verfügbar und stellen somit selbst eine Plattform dar.

Generell bieten Laufzeitsysteme damit die oben genannte Plattformunabhängigkeit und Portabilität, sowie Sicherheitskonzepte und die einfache Installation von Anwendungen. In der Regel existieren für diese Systeme moderne Programmiersprachen, die produktiver und weniger fehleranfällig sind als z. B. die Sprachen C bzw. C++ (vgl. z. B. Schmatz (2004) und Neable (2002)).

Wie in Domer u. a. (2004) gezeigt wird, weisen Laufzeitumgebungen Defizite bei der Ausführungsgeschwindigkeit auf. Eine Laufzeitumgebung muss auch immer auf dem mobilen Gerät installiert sein. Gerade bei Laufzeitumgebungen für mobile Geräte wird darauf geachtet, dass die Größe des Laufzeitsystems nicht zu groß ist, was jedoch immer auf Kosten des Funktionsumfangs geht.

Die gebräuchlichsten Laufzeitumgebungen für stationäre Anwendungen sind Java Platform, Standard Edition (Java SE) der Firma Sun und .NET Framework der Firma Microsoft. Diese existieren auch in angepassten Versionen für mobile Anwendungen und Geräte als Java Plattform, Micro Edition (Java ME) bzw..NET Compact Framework (.NET CF). Der Autor hat sich in einer Seminararbeit (Thomé (2005a)) mit diesen beiden mobilen Laufzeitumgebungen für mobile Anwendungen beschäftigt. Im Folgenden werden die .NET-Plattformen exemplarisch vorgestellt.

2.1.4.1 .NET Framework und .NET Compact Framework

Aus Sicht der Firma Microsoft ist .NET sowohl Strategie als auch Programmierplattform (Westphal (2002)). Letztere besteht aus Produkten für die Software-Entwicklung, architektonischen Aspekten und der .NET Laufzeitumgebung nebst zugehöriger Klassenbibliotheken.

Die .NET Laufzeitumgebung und die zugehörigen Klassenbibliotheken, das .NET Framework[2], sind in der aktuellen Version 3.0 für die Betriebssysteme Windows XP und Vista erhältlich. Die Version 3.0 erweitert die Vorgängerversion 2.0 um zusätzliche Funktionalitäten wie z. B. auf Vektorgrafik basierende Benutzerschnittstellen. Das .NET Framework unterstützt eine Vielzahl von Programmiersprachen.

Das .NET CF ist eine reduzierte Version des .NET Frameworks, was durch die verkleinerte Klassenbibliothek deutlich wird. Insgesamt stellt das .NET CF nur 25 Prozent der Funktionalität[3] des .NET Frameworks bereit. Dennoch wurde es um Klassen erweitert, damit es gezielt in mobilen Geräten eingesetzt werden kann. So bietet es z. B. Klassen für den Zugriff

[2] Im weiteren Sinne werden die Laufzeitumgebung und die Klassenbibliotheken zusammen als Framework bezeichnet.

[3] Eine genaue Übersicht ist unter Dröge u. a. (2006) ersichtlich.

auf Infrarotschnittstellen und Telefoneinheiten. Das .NET CF unterstützt lediglich die Programmiersprachen Visual Basic und C-#. Es existiert nur eine Implementierung des .NET Compact Frameworks für Windows Mobile[4] Betriebssysteme. Es gibt eine Spezifikation, die in den Versionen 1.0 und 2.0 vorhanden ist, letztere wurde im Herbst 2005 veröffentlicht. .NET CF ist für alle mobilen Geräte entwickelt worden, die mindestens über 1,5 MB ROM, 128 KB bis 1 MB RAM und einen 32-Bit Prozessor verfügen. Das .NET CF ist auf einer Reihe von mobilen Geräten, hauptsächlich auf PDAs mit dem Betriebssystem Windows Mobile, vorinstalliert. Unter Microsoft (2006f) und Wigley und Wheelwright (2003) sind weiterführende Informationen zu finden.

Abb. 2.1 zeigt die Architektur der beiden .NET Laufzeitsysteme. Basierend auf dem Betriebssystem, bestehen beide aus der Common Language Runtime (CLR) und dem Framework aus Basisklassen für jegliche Anwendungen. Das .NET Framework der Version 3.0 bietet zusätzlich Klassenbibliotheken für die Implementierungen speziellen Windows-Anwendungen.

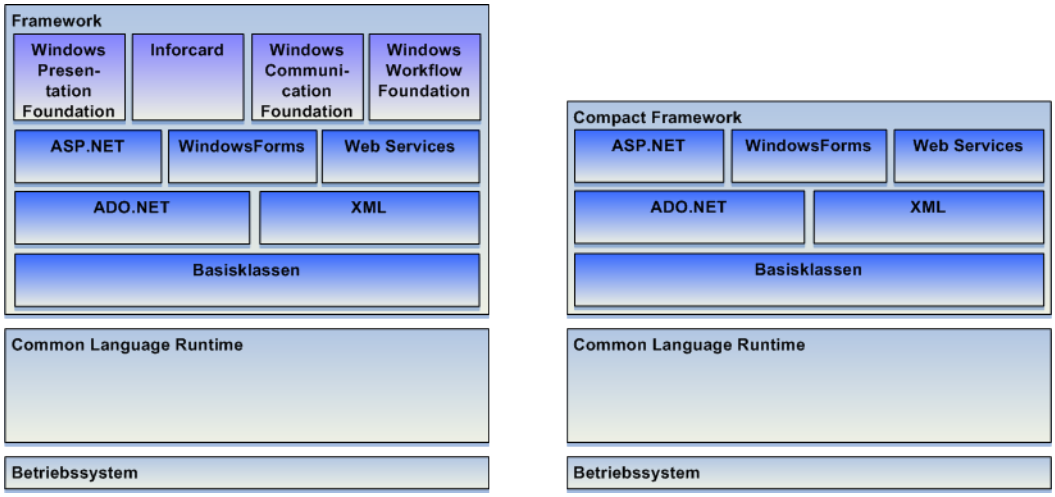

Abbildung 2.1: .NET-Architektur (nach Wigley und Wheelwright (2003) und Chappell (2006))

Microsoft hat .NET[5] und zugehörige Artefakte, wie z. B. die Programmiersprache C-#[6], von der European Association for Standardizing Information and Communication Systems (ECMA) und der International Organization for Standardization (ISO) standardisieren lassen.

[4] Windows Mobile in der aktuellen Version 5.0 sowie die Vorgängerversionen Pocket PC 2002, Pocket PC 2003 und Pocket PC 2003 SE. Dies sind konkrete Implementierungen der Spezifikation Windows CE, welche in diversen Versionen existiert.

[5] Standards ECMA-335, ISO/IEC 23271

[6] Standards ECMA-335, ISO/IEC 23270:2003

2.1.4.2 Bewertung

Laufzeitumgebungen bringen erhebliche Vorteile, wie z. B. Plattformunabhängigkeit, dennoch werden häufig mobile Anwendungen angeboten, die für bestimmte Typen von mobilen Geräten optimiert sind.

Prinzipiell leisten verfügbare Laufzeitumgebungen ähnliches. Für Anwendungen, die auf PDAs oder Smartphones mit dem Betriebssystem Windows Mobile laufen sollen, sollte .NET CF genutzt werden. Für Anwendungen zur Nutzung auf mobilen Standardcomputern mit dem Betriebssystem Windows, für die keine mobilen Laufzeitumgebungen verfügbar sind, sollte .NET herangezogen werden. Eine Entscheidungsempfehlung kann an dieser Stelle nicht gegeben werden, da mehrere Faktoren berücksichtigt werden müssen. Nach Meinung des Autors sollten sich Überlegungen zum Einsatz einer der beiden Technologien auch nicht nach der unterstützten Programmiersprache richten, da deren Konzepte und große Teile der Syntax ähnlich sind.

2.1.5 Mobile Geräte und Betriebssysteme

Mobile Geräte ermöglichen die Ausführung mobiler Anwendungen. Für viele Betriebssysteme dieser Geräte existieren speziell angepasste Laufzeitumgebungen. Roth (2005) identifiziert folgende Arten mobiler Geräte, deren Klassifizierung sich dabei nach ihrem Einsatzgebiet richtet:

Mobile Standardcomputer Diese stellen mobile Ausführungen stationärer Rechner dar, wobei sie ähnliche Leistungsmerkmale, jedoch eine deutlich kompaktere Bauweise aufweisen. Der Bildschirm und die Tastatur sind in den Rechner integriert, die Stromversorgung wird über einen internen Akku gewährleistet. Gängige Bezeichnungen sind Laptop und Notebook. Eine weitere Variante ist der Tablet-PC, der über einen berührungsempfindlichen Bildschirm bedient werden kann. Eine besonders kompakte Version des Tablet-PCs ist der Ultra-Mobile-PC (UMPC). Es werden häufig die gleichen Betriebssysteme wie für stationäre Rechner genutzt, z. B. Windows oder Linux. Dabei werden diese den speziellen Bedürfnissen der mobilen Rechner angepasst.

Bordcomputer Es handelt sich um hochspezialisierte Rechner, die in Fahrzeugen fest installiert sind. Zum Teil übernehmen sie Steuer- und Regelungsaufgaben ohne Nutzerinteraktion, wie z. B. die Steuerung des Antiblockiersystems (ABS) in Kraftfahrzeugen. Bestimmte Bordcomputer übernehmen spezielle Aufgaben mit Nutzerinteraktion, wie z. B. Navigationssysteme. Es kommen häufig speziell angepasste Betriebssysteme zum Einsatz. Für Bordcomputer mit Nutzerinteraktion wird unter anderem WindowsCE eingesetzt.

Handhelds Diese haben im Vergleich zu mobilen Standardcomputern eine sehr eingeschränkte Leistungsfähigkeit. Sie können durch ihre sehr kompakte Bauweise in einer Hand gehalten und dabei bedient werden. Typische Vertreter sind PDAs und Organizer. Besitzen sie zusätzlich Telefonfunktionalität, werden sie als Smartphones bezeichnet. Für Handhelds existieren eine Reihe verschiedener Betriebssysteme. Gängig sind z. B. Symbian OS, Palm OS, Windows Mobile und Linux Embedded.

Werables Diese kleine Rechner werden am Körper bzw. an der Kleidung getragen; die Hände sollen dabei für andere Tätigkeiten frei sein.

Chipkarten Für diese kleinsten mobilen Geräte wird immer ein Lesegerät benötigt. Sie werden häufig zur Speicherung kleiner Datenmengen genutzt. Typischer Vertreter ist die Geld-Karte.

Die Geräte dieser Kategorien können orthogonal auch in Spezial- und Universalgeräte unterteilt werden. Dies ist davon abhängig, ob sie einem bestimmten Einsatzgebiet zuzuordnen sind oder nicht. Insgesamt ist zu beachten, dass sich die Klassifikation über einen kurzen Zeitraum verändern kann, da die Innovationszyklen in diesem Bereich sehr kurz sind.

2.2 Ortsbezogene Dienste

Die Entwicklung ortsbezogener Dienste entstand aus den Bereichen des Ubiquitous Computing und des Telekommunikationssektors (Küpper (2005)). Zum einen haben es erst mobile Rechner möglich gemacht, dass Nutzer sich räumlich bewegen, und dass die aktuelle Position eines Nutzers die Ausführung einer laufenden Anwendung beeinflusst. Zum anderen hat der massive Ausbau zellenbasierter Mobilfunknetze den Netzbetreibern die Möglichkeit geschaffen, ortsbezogene Anwendungen mit Mehrwert für ihre bereits bestehenden Netze zu etablieren, um damit die hohen Kosten für den Ausbau der Netze zu kompensieren.

Ortsbezogene Dienste sind eine Untergruppe der kontextbewussten Dienste (engl.: context-aware services). Kontextbewusstsein (engl.: context awareness) beschreibt die Beeinflussung der Anwendungsausführung durch Informationen der äußerlichen Umgebung (Küpper (2005)).

2.2.1 Kontextbewusstsein und ortsbezogene Dienste

Informationen über die äußerliche Umgebungen des Nutzers sind vielfältig. In Küpper (2005) werden diese in primäre und sekundären Kontextinformationen unterteilt. Primäre Kontextinformationen lassen sich als Rohdaten von externen Sensoren, wie z. B. durch die im weiteren

Verlauf der Arbeit vorgestellten GPS-Empfänger, sammeln. Diese Daten können durch Kombination, Ableitungen oder Filtern zu sekundären, hochwertigen Kontextinformationen angereichert werden. Abb. 2.2 zeigt unter Berücksichtigung verschiedener Kontextinformationen die Einordnung ortsbezogener Dienste in kontextbewusste Dienste.

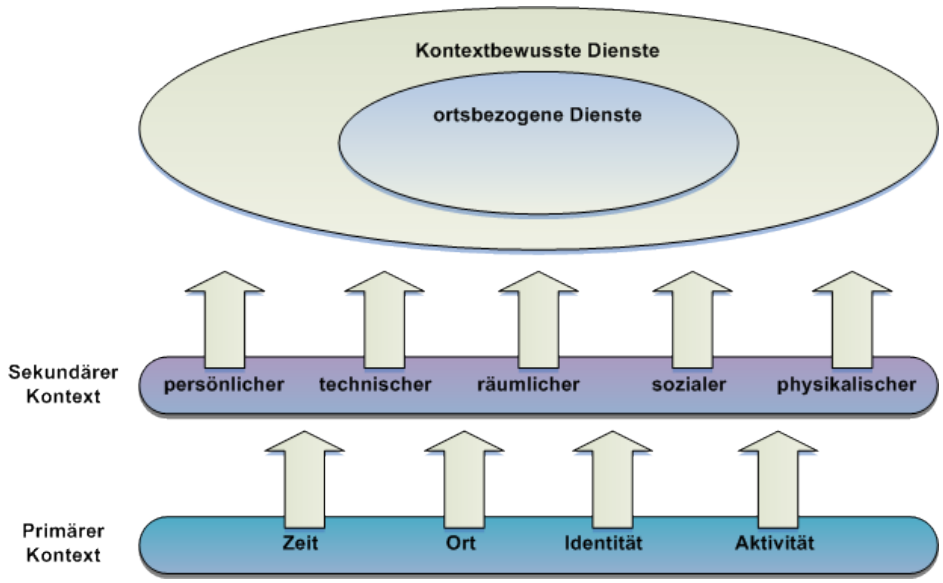

Abbildung 2.2: Abgrenzung kontextbewusster und ortsbezogener Dienste (Küpper (2005))

Viele kontextbewusste Informationen sind statisch und ändern sich selten. Nur einige wenige ändern sich häufiger, auch zur Laufzeit einer Anwendung. Dabei wurde gezeigt, dass diese Veränderungen direkt oder indirekt mit der räumlichen Position des Nutzers bzw. des mobilen Gerätes zusammenhängen. Demnach kommt der Position eine besondere Bedeutung im Rahmen des Kontextbewusstseins zu (vgl. Dix u. a. (2000)).

2.2.2 Eigenschaften

Wie zu Beginn dieses Abschnitts erwähnt, sind ortsbezogene Dienste aus verschiedenen Motivationen hervorgegangen. Nach Schiller und Voisard (2004) werden ortsbezogene Dienste hauptsächlich beim Militär, für Notfalldienste und im kommerziellen Sektor eingesetzt. Dem kommerziellen Sektor wird dabei eine besondere Bedeutung zugesprochen. In diesem haben sich insbesondere Navigationsanwendungen auf breiter Front in allen Bereichen des Transportwesens, wie der Schiff- und Luftfahrt, durchgesetzt. Das in dieser Arbeit entwickelte Informationssystem ist ebenfalls diesem Bereich zuzuordnen, und der Autor legt seinen Fokus auf ortsbezogene Diensten für den kommerziellen Einsatz.

Bisher existiert keine einheitliche Klassifizierung ortsbezogener Dienste. In Schiller und Voisard (2004) basiert die Klassifikation auf der Sicht der Ausrichtung und der Art der Interaktion auf konzeptioneller Ebene.

Personenorientierte Dienste Dabei werden die Dienste vom Nutzer kontrolliert. Ein Beispiel ist die GPS-Navigation im Kraftfahrzeug. Der Nutzer lässt seine eigene Position bestimmen und kann ausgewählte Routen berechnen lassen.

Geräteorientierte Dienste Bei dieser Kategorie von Diensten kontrolliert der Nutzer den Dienst nicht selbst. Dies ist z. B. beim Flottenmanagement der Fall. Das System übernimmt die Positionsbestimmung einzelner Objekte, wie z. B. von Autos. Die Fahrer haben eventuell keinen Zugriff auf ihre Positionsdaten.

Push-Dienste Die Nutzung des Dienstes erfolgt ohne aktive Anfrage seitens des Nutzers. Die Daten werden quasi aus dem Netz zum Nutzer „gedrückt“. Ein möglicher Anwendungsfall eines Push-Dienstes ist ortsbezogene Werbung.

Pull-Dienste Die Nutzung erfolgt mit einer aktiven Anfrage des Nutzers, wie z. B. durch eine Anfrage nach Orten von Interesse im Umkreis der aktuellen Position.

Die Art der Einteilung ortsbezogener Dienste ausschließlich in Push- bzw. Pull-Dienste scheint nicht sehr aussagekräftig, zumal Anwendungen existieren, die beide Kommunikationsarten bereitstellen. Daher existieren eine Reihe weiterer Kriterien zur Einordnung ortsbezogener Dienste. So werden in einem späteren Kapitel in Schiller und Voisard (2004) sowie in Küpper (2005) ortsbezogene Dienste nach ihrer Funktionalität und nach ihrem Applikationsszenario eingeordnet. Diese Einordnung scheint gut geeignet, um Anforderungen an den jeweiligen Dienst abzuleiten.

Informationsdienste Diese Dienste ermöglichen die ortsbezogene Informationsbeschaffung für den Nutzer, wozu dieser Anfragen an das System stellt. Es sind somit personenorientierte Pull-Dienste. Die Anfrage wird auf einem entfernten Server unter Berücksichtigung der Position des Nutzers evaluiert. Diese Dienste können mit personalisiertem Inhalt angereichert sein; typische Anwendungen sind Navigationssysteme, sowie Finder von Orten von Interesse und von Personen.

Tracking-Dienste Tracking-Dienste ermöglichen die geografische Positionsbestimmung sich bewegender Objekte. Diese Informationen werden an andere Nutzer übermittelt, die eine stationäre oder veriable Position haben. Dabei werden nicht nur die aktuelle, sondern auch mögliche zukünftige Positionen bestimmt. Eine erdenkliche Anwendung ist das Tracking von Personen. Bei Tracking-Diensten müssen in aller Regel eine hohe Anzahl an Entitäten simultan überwacht werden. Die Dienste sind sowohl Pull- als auch Push-basiert und stellen typische geräteorientierte Dienste dar. In Abschn. 2.4.4 wird näher auf die Besonderheiten von Datenbankanfragen an sich bewegende Objekte eingegangen.

Gemeinschaftsdienste Bei klassischen Gemeinschaftsdiensten (engl.: community services) interagieren gleichgesinnte Nutzern innerhalb des Internets über Client-Server-Anwendungen. Dabei kommunizieren sie in Echtzeit über Nachrichtendienste (engl.: chat, messaging service). Die Nutzer erstellen über sich Profile, anhand derer gleichgesinnte Nutzer gefunden werden können. Sehr verbreitet ist der *Instant-Messaging-Dienst*, der von verschiedenen Firmen wie Google, Microsoft und Yahoo angeboten wird. Nutzer können sich Listen von Freunden zusammenstellen, diesen Kurznachrichten senden und permanent sehen, ob die einzelnen Nutzer aktuell mit dem Dienst verbunden sind. Diese Dienste können um die mobile Dimension erweitert werden, indem sie auf mobilen Geräten zum Einsatz kommen. Eine mögliche Erweiterung in diesem Zusammenhang ist das Anzeigen der geografischen Position von Nutzern aus der Freundesliste und das automatisierte Benachrichtigen, wenn ein Nutzer eine nahe gelegene Position erreicht. Diese Dienste sind sowohl Push- als auch Pull-basiert, sowie personen- und geräteorientiert. Gemeinschaftsdienste sind Vertreter sozialer Software, die in Abschn. 2.5.3 vorgestellt wird.

Dienste zur selektiven Informationsweitergabe Bei diesen Diensten werden speziell ausgewählte Informationen vom System an bestimmte Nutzer übertragen. Der Dienst basiert auf der Push-Technik, der Nutzer hat in der Regel kaum Einflussmöglichkeiten. Informationen werden aus dem vorliegenden Nutzerprofil abgeleitet. Eine mögliche Anwendung ist nutzer- und ortsbezogene Werbung, bei der Nutzer, die sich an einem bestimmten Ort aufhalten, auf ihr Profil zugeschnittene Werbung bekommen. In diesem Zusammenhang wird in Abschn. 3.2.4 die Wirtschaftlichkeit für derartige Werbung diskutiert.

Notfalldienste Notfalldienste dienen der Anreicherung vorhandener um ortsbezogene Informationen bei Notrufen. So soll bei Notrufen mit Mobilfunktelefonen die geografische Position des Anrufers an die Notfalldienste mit übertragen werden. Auch soll der Anruf an die nächst gelegene Notfallzentrale geleitet werden, so dass erforderliche Notfallmaßnahmen effizienter ergriffen werden können. Diese Dienste haben in Nord-Amerika und innerhalb der Europäischen Union die Entwicklung und Bereitstellung ortsbezogener Dienste im Mobilfunkbereich maßgeblich beeinflusst. Dazu sei auf die entsprechenden Initiativen von FCC (2006) und EU (2003) verwiesen. In Deutschland wird seit Spetember 2006 ein entsprechender Dienst[7] angeboten.

Ortsbezogene Gebührenerfassung Diese Dienste ermöglichen eine dynamische Berechnung von Leistungen in Abhängigkeit der Position des Nutzers. So bieten viele Mobilfunkbetreiber[8] kostengünstige Telefongespräche an, wenn der Nutzer sich innerhalb

[7] Informationen sind unter http://www.ptv.de/download/mobility/Referenzen/ref_BjoernSteiger.pdf erhältlich; Zugriffsdatum: 04.05.2007

[8] Der Mobilfunkanbieter O2 (http://www.o2online.de/; Zugriffsdatum: 04.05.2007) bietet dies mit der so genannten Homezone, einer frei festzulegenden Mobilfunkzelle.

einer bestimmten Mobilfunkzelle aufhält. Auch die elektronische, dynamische Abrechnung von Mautgebühren[9] bei der Nutzung bestimmter Straßen ist Teil dieser Dienste.

Ortsbezogene Spiele Ortsbezogene Spiele dienen der Unterhaltung teilnehmender Nutzer. Die Spiele verbinden die virtuelle mit der realen Welt. Diese Art von Diensten läßt sich schwer kategorisieren, die vorher genannten Eigenschaften sind hier nicht gleich erkennbar. Diese Dienste werden immer beliebter und permanent weiterentwickelt. Beispiel eines solchen Dienstes ist das so genannte Geo-Caching.

Ortsbezogene Dienste können auch, wie in Roth (2005) beschrieben, nach geschäftsbezogenen Eigenschaften unterschieden werden, da dies für den Dienstanbieter von besonderer Bedeutung sei. In Abschn. 3.2.4 wird besonderes Augenmerk auf das mögliche Geschäftsmodell des im Rhamen dieser Arbeit entwickelten Informationssystem gelegt.

2.2.3 Anforderungen

Zusätzlich zu den in Abschn. 2.1.2 vorgestellten Anforderungen an mobile Informationssysteme, werden weitere, spezielle Anforderungen an ortsbezogene Dienste gestellt; diese sind je nach Funktion und Einsatzbereich des Dienstes unterschiedlich. Tsalgatidou u. a. (2003) skizziert Anforderungen an orstbezogene Dienste für den kommerziellen Einsatz mit mobilen Geräten. In Schiller und Voisard (2004) werden allgemeinere Anforderungen an eine Middleware für ortsbezogene Dienste skizziert. Im Folgenden werden die daraus resultierenden Anforderungen aufgezeigt. Eine konzeptionelle Middleware wird im folgenden Abschnitt vorgestellt.

Funktionalität Die Anforderungen der Nutzer müssen erfüllt werden, wie z. B. personalisierter Zugang oder manuelle Eingabe einer Adresse, wenn keine automatische Positionsbestimmung möglich ist.

Bedienbarkeit Die Benutzerschnittstelle sollte mobilen Geräten angepasst sein, Positionsinformationen sollten im Idealfall als Karte angezeigt werden. Dynamische Zusatzinformationen, wie Orte von Interesse oder Routen, sollten entsprechend erkennbar sein. Die Benutzerschnittstelle sollte auch Audiounterstützung für die Ein- und Ausgabe von Daten beinhalten.

Verlässlichkeit der Daten Die Qualität ortsbezogener Daten muss hoch sein, da der Nutzer aufgrund dieser Informationen seine Entscheidungen trifft. Zudem sollte der Betrieb auch möglich sein, wenn keine Netzverbindung zu entfernten Ressourcen besteht. Wie bei allen mobilen Informationssystemen ist hier die Unterstützung eines Online- und Offline-Szenarios wichtig.

[9] In Deutschland etwa durch das Mautsystem der Firma Toll Collect (http://www.toll-collect.de; Zugriffsdatum: 04.05.2007).

Verlässlichkeit der Positionsinformationen Die Positionsinformationen des Nutzers müssen, je nach eingesetztem System zur Positionsbestimmung, entweder vom Nutzer selbst, vom Netzwerk oder von einem dritten Dienst bestimmt werden. Bei ersterem werden die Informationen der Dienstanfrage hinzugefügt. Bei letzterem muss der Dienst diese Informationen erfragen. Durch verschiedene Genauigkeiten ergeben sich Einflüsse auf die Informationen, die der Dienst zur Verfügung stellen kann. Dies muss in jedem Fall berücksichtigt werden.

Privatsphäre Dem Datenschutz zur Privatsphäre (engl.: privacy) kommt bei ortsbezogenen Diensten eine wichtige Rolle zu. Speziell Informationen über den Aufenthaltsort einzelner Nutzer müssen vertraulich sein. Die Privatsphäre scheint ein entscheidender Faktor bie der Akzeptanz ortsbezogener Diensten durch die Nutzer zu sein.

Dienstoperabilität Ein dynamischer Wechsel zwischen verschiedenen Systemen zur Positionsbestimmung und Endgeräten sollte möglich sein. Dazu ist es erforderlich, die Nutzung verschiedener Koordinatenreferenzsysteme zu ermöglichen, um entsprechende Daten aus verschiedenen Geodateninformationssystem auslesen zu können.

Personalisierung Ortsbezogene Dienste müssen direkte oder indirekte Personalisierung unterstützen und Anfragen der Nutzer an den Dienst damit anreichern. Bei direkter Personalisierung wird das Benutzerprofil vom Nutzer manuell vorgegeben und gespeichert. Bei indirekter Personalisierung wird das Profil bei jeder Anfrage aus dem Nutzerverhalten abgeleitet.

Interaktionsszenarien Bei ortsbezogenen Diensten gibt es immer eine Entität, die eine Anfrage an einen Dienst stellt, und eine weitere, die diese verarbeitet. Beide Entitäten können dabei mobil oder stationär sein. Dadurch ergeben sich die vier Kommunikationsmuster mobil - stationär, mobil - mobil, stationär - mobil und stationär - stationär.

Zustand Ortsbezogene Dienste müssen in der Lage sein, den Zustand mobiler Nutzer über mehrere Anfragen hinweg zu verwalten. Dies ist besonders bei den im vorherigen Abschnitt vorgestellten Tracking-Anwendungen von Bedeutung.

Art der Informationsquellen Es müssen verschiedene Arten von Informationsquellen verwaltet werden können. Dabei gilt es, statische und dynamische Daten zu identifizieren. Statische Daten sind z. B. Kartendaten, hochgradig dynamisch dagegen sind vor allem die Positionsdaten der Nutzer. Weiterhin können auch Daten über Benutzerprofile oder Daten externer Datenlieferanten, wie z. B. Werbedaten, dynamisch sein.

2.2.4 Middleware für ortsbezogene Dienste

Ortsbezogene Dienste beinhalten immer eine technischen Komponente für die Positionsbestimmung der Nutzer bzw. deren mobiler Geräte, sowie einen Datenbestand, aus dem

Informationen mithilfe der zuvor bestimmten Position gewonnen werden. In einem solchen Datenbestand werden z. B. Orte von Interesse für bestimmte Positionsdaten gespeichert. Abb. 2.3 zeigt eine konzeptionelle Architektur der Middleware eines ortsbezogenen Dienstes. Techniken zur Positionsbestimmung werden in Abschn. 2.3, räumliche Datenbanken und geografische Informationssysteme werden in Abschn. 2.4 beschrieben.

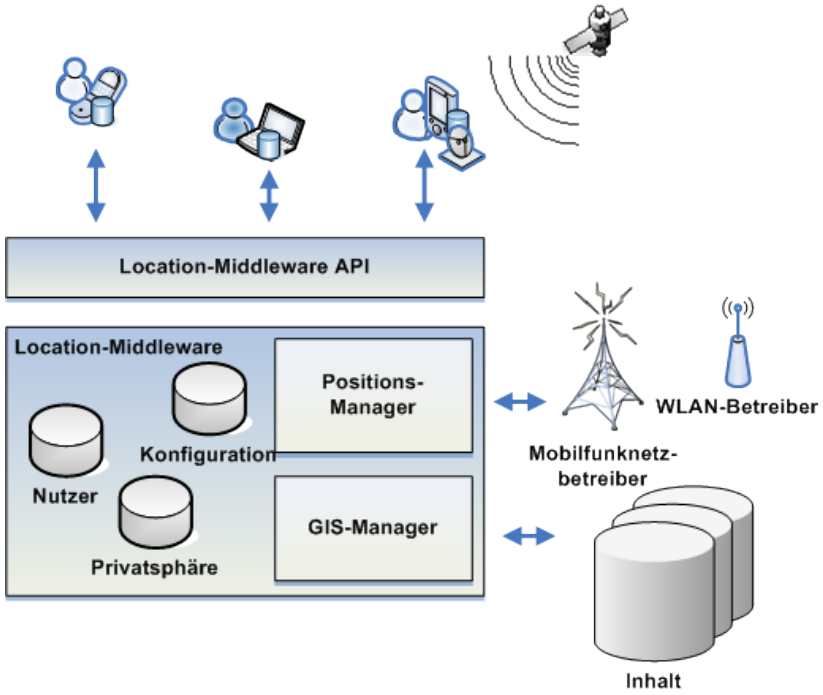

Abbildung 2.3: Middleware für ortsbezogene Dienste (nach Schiller und Voisard (2004))

Eine Middleware für ortsbezogene Dienste nutzt demnach Komponenten für Positionstechniken, den Positions-Manager, sowie Komponenten für die Beschaffung ortsbezogener Daten, den GIS-Manager. Schlussfolgernd hat die Middleware im Wesentlichen zwei verschiedene Aufgaben: Zum einen dient sie als Abstraktionsschicht zwischen den Anwendungen und Mechanismen zur Positionsbestimmung sowie den Daten für die Dienste. Dazu bietet die Middleware eine einheitliche Programmierschnittstelle für Anwendungsentwickler an. Zum anderen stellt sie den Integrationspunkt für verschiedene Positionsbestimmungs- und Dateninhaltsanbieter dar. So können Positionen transparent mittels verschiedener Technologien, z. B. per netzwerkbasierter Technologien im WLAN oder im GSM-Mobilfunknetz, bestimmt werden. Dateninhalte können von verschieden Dienstleistern bereitgestellt werden. Damit kommt die Middleware der wesentlichen, der in Abschn. 2.2.3 skizzierten, Anforderung nach Dienstoperabilität nach. Darüber hinaus übernimmt die Middleware noch weitere Aufgaben, wie die Verwaltung der Nutzer, die Konfiguration der Dienste und die Schutzmechanismen für die Privatsphäre.

Es scheint nicht sinnvoll zu sein, eine ortsbezogene Middleware komplett neu aufzubauen. Vielmehr sollte eine vorhandene Middleware für verteilte Systeme um die Funktionen für ortsbezogene Dienste erweitert werden. Es existieren einige Middleware-Produkte, die die oben beschrieben Funktionalitäten zum Teil oder vollständig besitzen. Einheitliche Standards, wie z. B. Open GIS Location Services (vgl. OGC (2007a)), haben sich bisher jedoch nicht durchgesetzt. Eine Middleware für ortsbezogene Dienste kann sowohl Client- als auch Server-seitig integriert werden. Beispiele von Middlewares für ortsbezogene Dienste sind MiddleWhere (Ranganathan u. a. (2004)), Nexus (Dürr u. a. (2004)) und Microsoft MapPoint Location Server (Microsoft (2006c)). Letztere basiert auf einem Anwendungs-Server des Herstellers und erweitert diesen um die oben beschriebenen Funktionalitäten.

2.2.5 Navigationssysteme

Navigationssysteme bilden eine Kernfunktion ortsbezogener Dienste und werden meist als mobiles Infomationssystem entworfen. Die Position eines Nutzers wird in Echtzeit bestimmt, und der Nutzer bekommt Anweisungen für die Wegfindung zu einem von ihm bestimmten Ziel. Zusätzlich erhält er Informationen über Orte von Interesse, die sich in seiner Umgebung befinden.

2.2.5.1 Geräte und Benutzerschnittstellen

Aktuelle mobile Geräte, wie die in Abschn. 2.1.5 vorgestellten, bieten eine geeignete Grundlage für Navigationssysteme.

Die Eingabe durch den Nutzer erfolgt je nach mobilem Gerät unterschiedlich; sie kann textbasiert oder per Sprachsteuerung erfolgen. Die Routenführung erfolgt grafisch mittels Karte und Piktogrammen für bestimmte Situationen, wie z. B. durch einen Pfeil als Hinweis zum Abbiegen. Erweitert werden diese Anzeigen durch sprachliche Fahranweisungen für den Nutzer. Diese können generisch aufgenommen sein oder als Sprachsynthetisierung direkt erzeugt werden. Durch letztere lassen sich präzisere Anweisungen erzeugen, die auch Orts- und Straßennamen beinhalten (Kray u. a. (2003)).

2.2.5.2 Arten von Navigationssystemen

Aus den verschiedenen Ausprägungen der Hardware und den dadurch beschränkten Ressourcen, sowie in Bezug auf die Geschäftsmodelle der Dienstanbieter, haben sich zwei verschiedene Arten von Navigationssystemen entwickelt: *Onboard-* und *Offboard-Navigation*.

Onboard-Navigation Bei der Onboard-Navigation sind die Anwendung und sämtliche Kartendaten auf dem mobilen Gerät installiert. Die Routenberechnung wird auf dem Gerät durchgeführt. Insbesondere neue Berechnungen der Route beim Verlassen derselbigen werden sofort ausgeführt. Dafür ist keine Kommunikation mit externen Datenquellen erforderlich; die Anwendung wird in einem Offline-Szenario ausgeführt. Dynamische Daten, wie Stau- oder Wetterinformationen, können je nach mobilem Gerät und je nach Anwendung bei der Routenberechnung via Traffic Message Channel (TMC) berücksichtigt werden. Kartendaten veralten zunehmend; da die Daten in aller Regel einmalig gekauft werden, müssen etwaige Aktualisierungen ebenfalls erworben werden. Ein Beispiel für ein Onboard-Navigationssystem ist TomTom NAVIGATOR[10] der Firma TomTom.

Offboard-Navigation Bei der Offboard-Navigation wird lediglich die Anwendung auf dem mobilen Gerät installiert, die Routenberechnung wird auf einem Server durchgeführt. Das Ergebnis wird als Korridor entlang der berechneten Route vom Server an die mobile Anwendung übertragen. Beim Verlassen der Route muss eine erneute Verbindung zur Server-Anwendung hergestellt werden. Die Anwendung wird in einem Online-Szenario ausgeführt. Dynamische Daten können einfach Server-seitig integrieren werden, Kartenaktualisierungen auf der Client-Seite entfallen. Je nach Dienstanbieter ist jede Routenberechnung separat vom Nutzer zu zahlen, zusätzlich müssen etwaige Kosten des Datentransports, etwa via GPRS, berücksichtigt werden. Die Hardware-Anforderungen sind jedoch geringer. Vertreter eines Offboard-Navigationssystems ist T-Navigate[11] der Firma T-Mobile.

Hybride-Navigation Hybride Ansätze versuchen, die Vorteile der beiden oben genannten Ansätze zu vereinen. Dabei können ausgewählte Kartendaten auf dem mobilen Gerät gespeichert werden. Nicht auf dem Gerät vorhandene Daten können, wie bei der Offboard-Navigation, von einem Server geladen werden. Die Berechnung wird auf dem mobilen Gerät durchgeführt. Die Firma Elektrobit Automotive bietet mit der Software Street Director[12] ein hybrides Navigationssystem an.

2.3 Positionsbestimmung

Die Positionsbestimmung[13] von Nutzern bzw. von mobilen Rechnern ist ein wichtiges technisches Element und bildet eine Grundlage für ortsbezogene Dienste. Wie in Abschn. 1.1

[10] http://www.tomtom.com/index.php; Zugriffsdatum: 04.05.2007
[11] http://www.t-mobile.de/navigate/; Zugriffsdatum: 08.02.2007
[12] https://www.street-director.de/catalog/index.php; Zugriffsdatum: 02.03.2007
[13] Der Begriff Positionsbestimmung wird von dem Autor mit dem Begriff Ortung synonym verwandt.

erläutert, haben Ortungstechnologien nach Gartner (2006a) eine hohe Relevanz für die Informationstechnologie. In diesem Abschnitt sollen existierende Konzepte und Systeme vorgestellt und bewertet werden. Für weiterführende Informationen sei auf Roth (2005) und Küpper (2005) verwiesen.

2.3.1 Positionsinformationen

Die im Folgenden vorgestellten Arten der Positionsbestimmung liefern numerische, räumliche Positionsinformationen. Global eindeutige numerische Werte sind die geografische Länge und Breite, sowie die Höhe des geografischen Koordinatensystems. Die geografische Länge beschreibt, wie weit westlich oder östlich vom Nullmeridian[14] eine Position ist. Wie weit südlich oder nördlich vom Äquator eine Position ist, wird durch die geografische Breite ausgedrückt. Bei beiden Angaben handelt es sich um Winkel, ihre Werte werden in Grad angeben. Der Wertebereich der geografische Länge reicht von Westen nach Osten von -180 bis 180, der der geografischen Breite reicht von Süden nach Norden von -90 bis 90. Neben dem geografischen Koordinatensystem existieren kartesische und ebene Koordinatensysteme, die Positionen über drei Achsen spezifizieren. Ebene Koordinatensysteme teilen die gewölbte Erdoberfläche in Abschnitte ein. Zur Projektion existieren viele mathematische Modelle. Das bekannteste zweidimesionale Koordinatensystem zur Projektion der Erdoberfläche auf Karten ist das *Universal Transverse Mercator* (UTM) Koordinatensystem.

Diese numerischen Werte müssen ggfs. in andere Formate konvertiert werden, insbesondere dann, wenn Positionsinformationen in Datenbanken als Zeichenketten mit bestimmter Semantik vorliegen. Diese semantischen oder beschreibenden Positionsinformationen dienen in erster Linie dem Nutzer zum Verständnis. Zur Ausführung geografischer Berechnungen müssen semantische Positionsdaten wieder in numerische, räumliche Positionsdaten überführt werden. Die zu bestimmende Position muss nicht immer weltweit eindeutig bestimmt werden. Viele Anwendungsfälle erfordern lediglich die relative Positionsangabe zu einer definierten Position.

2.3.2 Verfahren zur Positionsbestimmung

Grundsätzlich existieren zwei verschiedene Verfahren zur Positionsbestimmung: *Positioning* und *Tracking*.

Positioning Beim Positioning sendet ein System von Sendern Signale aus, die von geeigneten Geräten empfangen werden. Die Position wird somit beim Nutzer bestimmt und

[14] Der Nullmedian ist senkrecht zum Äquator stehende Referenzlinie vom Nord- zum Südpol und verläuft dabei durch Greenwich in England.

liegt in aller Regel nicht im System vor. Dadurch können aufwendige Verfahren zur Sicherung der Privatsphäre vermieden werden.

Tracking Wird die Position des Nutzers mithilfe eines Sensornetzwerkes erkannt, spricht man von Tracking. Der Nutzer bestimmt seine Position nicht aktiv. Somit liegen Informationen über die Position zuerst dem System vor und müssen mittels geeigneter Verfahren an den Nutzer übertragen werden. Dazu müssen ausreichende Sicherungsmechanismen zur Gewährleistung der Privatsphäre des Nutzers implementiert sein. Während die Anfrage der Position bei einem Tracking-Verfahren in aller Regel an eine Datenbank gestellt wird, wird beim Positioning die Position ad-hoc errechnet.

Die in Abb. 2.4 gezeigten und im Folgenden beschriebenen Basistechniken werden beim Positioning und Tracking einzeln oder in Kombination eingesetzt.

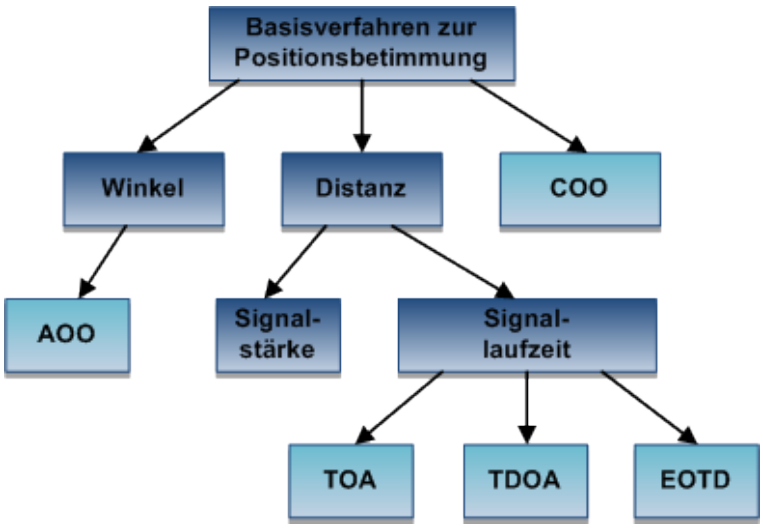

Abbildung 2.4: Basistechniken der Positionsbestimmung

Messung der Signalstärke Eine einfache Berechnung des Abstandes ist durch die Signalstärke beim Empfänger durchführbar. Dieses Verfahren ist jedoch nicht sehr genau und von starken Schwankungen durch Umwelteinflüsse unterlegen.

Cell Of Origin Durch Nutzung der Zellenstruktur können bei einem solchen Netzwerk durch die Zellenidentifikation Rückschlüsse auf die Position durchgeführt werden. Man nutzt dazu die Eigenschaft, dass Signale nur einen bestimmten Umkreis erreichen zu können. Dieses Verfahren wird als *Proximity Sensing bezeichnet*. Jedes mit der Zelle verbundene mobile Gerät bekommt als Position die des Zellenmittelpunktes.

Angle Of Arrival Durch den Einsatz gerichteter Antennen kann der Eingangswinkel empfangener Signale bestimmt werden. Mit diesem, auch als *Angulation* bezeichneten

Verfahren ist eine Verbesserung der COO-Technik möglich, da die Position des mobilen Gerätes nun auf bestimmte Bereiche der Zelle genau bestimmt werden kann.

Time Of Arrival Die Berechnung des Abstandes zum Zellenmittelpunkt ist durch die Ankunftszeit (Time Of Arrival (TOA)) bzw. den Laufzeitunterschied (Time Difference Of Arrival (TDOA)) einfach möglich. Sobald ein mobiles Gerät Signale von drei bzw. vier Zellen empfängt, kann durch einen Algorithmus, wie *Circular Lateration* oder *Hyberbolic Lateration*, die zwei- bzw. vierdimensionale Position aus der Schnittmenge der empfangenen Signale errechnet werden. TOA ist eine deutliche Verfeinerung der COO-Basistechnik. Bei GSM-Netzwerken wird dafür auch der Begriff Enhanced Observed Time Difference (EOTD) genutzt.

2.3.3 Systeme zur Positionsbestimmung

Es gibt kein universelles System zur Positionsbestimmung. In Forschung und Praxis haben sich bestimmte Systeme bei verschiedenen Anforderungen gebildet. So werden Systeme zur Positionsbestimmung in der Literatur (u.a. in Roth (2005) und Küpper (2005)) grob in satellitenbasierte Positionsbestimmung, Positionsbestimmung innerhalb von Gebäuden und netzwerkgestützte Positionsbestimmung unterteilt. Wesentlicher Grund für diese Unterteilung ist zum einen, dass unterschiedliche Anforderungen an die Positionsbestimmung im Freien und innerhalb von Gebäuden existieren. So können zwar Systeme wie das GPS eine hohe Genauigkeit liefern, jedoch funktionieren diese nicht innerhalb von Gebäuden. Zum anderen ist man bestrebt, die vorhandene Netzwerkinfrastruktur für die Positionsbestimmung zu nutzen. Allen Systemen ist gemein, dass die Positionsbestimmung mittels verschiedener der oben vorgestellten Basistechniken realisiert wird. Der satellitenbasierten Positionsbestimmung mit dem Global Positioning System (GPS) und der netzwerkbasierten Positionsbestimmung im Global System for Mobile Communication (GSM) Mobilfunknetzwerk bzw. im Wireless Area Local Network[15] (WLAN) werden in der Praxis die größte Bedeutung beigemessen. Diese Systeme werden in den nachfolgenden Unterabschnitten vorgestellt. Die speziellen Systeme zur Positionsbestimmung innerhalb von Gebäuden werden nicht weiter betrachtet. Für eine genaue Abgrenzung der einzelnen Systeme sei auf Roth (2005) verwiesen.

2.3.3.1 GPS

GPS nutzt ein System aus 24 Satelliten auf 6 festen Umlaufbahnen, um eine Position zu bestimmen. Als Basistechnik kommt TOA zum Einsatz. Aus den Laufzeitinformationen (TOA)

[15] In dieser Arbeit sind die WLAN-Standards des Institute of Electrical and Electronics Engineers (IEEE) 802.11 gemeint. Weitere Informationen sind unter http://standards.ieee.org/getieee802/802.11.html zu finden; Zugriffsdatum: 04.05.2007.

von drei Satelliten kann mittels Triangulation eine Position auf der Erde auf wenige Meter genau bestimmt werden. Die exakte Zeitmessung ist jedoch problematisch, da sich die Signale der Satelliten in Lichtgeschwindigkeit ausbreiten. Daher werden die Signale eines vierten Satelliten zur Korrektur genutzt.

Mittels Differential GPS (DGPS), einem System aus zusätzlichen, stationären Korrektursendern, wird eine höhere Genauigkeit erreicht. DGPS basiert auf dem Prinzip, dass Nutzer in einer bestimmten Umgebung ähnliche Fehler von GPS-Signalen wahrnehmen. Die Bodenstation vergleicht empfangene GSP-Signale mit ihrer eigenen, exakten Position. Die daraus errechnete Differenz wird an GPS-Empfänger übertragen, die mit dieser zusätzlichen Information etwaige Fehler in der Positionsbestimmung kompensieren können.

Mit dem europäischen Satellitensystem Gallileo ist eine neue Generation satellitenbasierter Positionsbestimmung in der Entwicklung. Diese ist zum GPS-System kompatibel und verspricht eine Steigerung der Verfügbarkeit und Genauigkeit.

2.3.3.2 WLAN

Immer mehr Gebäude oder Areale werden mit drahtlosen Netzwerken ausgestattet. Auch verfügen immer mehr mobile Geräte über eine drahtlose Netzwerkschnittstelle. WLANs sind zellenbasierte Netzwerke. Jede Zelle beinhaltet eine Basisstation, den *Access-Point*, der eine fest definierte Position hat. Zur Erkennung einer Position innerhalb des Netzwerkes wird als Basisverfahren die Messung der Signalstärke vom Access-Point herangezogen. Da im WLAN die Uhrzeit nicht exakt bestimmt werden kann, sind andere Basisverfahren, wie z. B. die Messung der Laufzeit eines Signals, nicht möglich. Aufgrund der Signalstärke kann die relative Position zum Access-Point mittels drei verschiedener Verfahren bestimmt werden:

Zellenbasiertes Verfahren Beim einfachsten, dem zellenbasierten Verfahren wird die Position des Access-Points mit der höchsten Signalstärke angenommen. Das Verfahren gleicht somit dem COO und ist sehr ungenau. Dieses Verfahren wird daher nur eingesetzt, wenn keine hohe Genauigkeit gefordert wird. Dazu werden Datenbanken mit Informationen über die Positionen von Access-Points gespeichert, wodurch ein einfaches System zur Positionsbestimmung für großflächige Areale, wie z. B. Städte, etabliert werden kann.

Laufzeitbasiertes Verfahren Beim laufzeitbasierten Verfahren wird versucht, die Entfernung zwischen mehreren Access-Points und dem mobilen Gerät zu bestimmen und mittels bekannter Lateration-Algorithmen die Position zu berechnen. Problematisch ist dabei die Bestimmung der Entfernung an sich, da eine einfache Laufzeitmessung nicht möglich ist. Daher wird die Ausbreitungsdämpfung (engl.: path loss) als grobe Abschätzung herangezogen, jedoch leidet gerade diese unter der Reflexion der Signale

durch Objekte innerhalb eines Gebäudes. Dies wird vor allem deutlich, wenn sich die Objekte bewegen und dadurch eine genaue Positionsbestimmung unmöglich machen.

Tabellenbasiertes Verfahren Einzig das tabellenbasierte Verfahren (engl.: fingerprinting) eignet sich zum praktischen Einsatz, wenn Genauigkeit gefordert ist. Bei diesem Verfahren werden die Signalstärken, die von mehreren verschieden Access-Points empfangen werden, mit Werten in einer Tabelle verglichen. Die Werte beschreiben die Signalstärken aller Access-Points an Referenzpositionen. Diese Werte werden in einer Trainingsphase aufgebaut und müssen aktualisiert werden, sobald sich die Netzwerktopologie ändert. Die Trainingsphase kann durch mathematische Modelle, die eine Simulation der Signalausbreitung durchführen, automatisiert werden.

Einige Städte[16] haben damit begonnen, ihre Gebiete flächendeckend mit WLAN zu versorgen. Auch entsteht mit Worldwide Interoperability for Microwave Access (WiMAX) bereits der nächste Standard für drahtlose Datenübertragung. Dieser ermöglicht, im Vergleich zum WLAN, den Aufbau regionaler Funknetze. So kann die Positionsbestimmung mittels vorhandener, drahtloser Netze auf immer größere Gebiete ausgeweitet werden.

2.3.3.3 GSM und UMTS

Jeder Sendemast einer Mobilfunkzelle hat eindeutige, global gültige Koordinaten. Bei der Positionsbestimmung kommt im Wesentlichen die Basistechnik COO zum Einsatz. Die Position innerhalb der Mobilfunkzelle wird dabei gleich der Position des Funkmastes gesetzt, die Genauigkeit ist somit abhängig von der Größe der Zelle. In urbanen Gebieten können Zellen mit einem Durchmesser von 100 m relativ klein ausfallen, in ländlichen Gegenden können diese jedoch bis zu 35 km betragen. Eine Verfeinerung der Positionsbestimmung wird durch zusätzliche Anwendung der AOA-Basistechnik erreicht. Ist ein Empfänger gar in der Lage, Signale von vier Sendemasten zu empfangen, kann mittels TOA und Triangulation aller empfangenen Signale eine wesentlich genauere Positionsbestimmung erreicht werden.

2.3.3.4 Software-Schnittstellen

Für die oben aufgeführten Systeme zur Positionsbestimmung existieren zahlreiche Software-Schnittstellen für mobile Anwendungen. Diese sind als Bibliotheken diverser Hersteller verfügbar und können in Anwendungen integriert werden. Grundsätzlich gibt es keinen einheitlichen Standard für .NET-basierte Anwendungen. Stellvertretend seien GPS.NET[17] der

[16] Das Projekt TechConnect der US-amerikanische Stadt San Francisco hat zum Ziel, das Stadtgebiet mit kostenlosen WLAN zu versorgen (http://www.sfgov.org/site/tech_connect_index.asp; Zugriffsdatum: 04.05.2007).

[17] http://www.geoframeworks.com/Products/GPS/; Zugriffsdatum: 04.05.2007

Firma GeoFrameworks für Positionsbestimmung via GPS, das Wi-Fi Positioning System[18] der Firma Skyhook Wireless und der MapPoint Location Server[19] der Firma Microsoft für Positionsbestimmung via GSM genannt. Mit Place Lab[20] von Intel Research existiert eine weit verbreitete Software-Schnittstelle für netzwerkbasierte Positionsbestimmung.

2.3.3.5 Bewertung

Es gibt kein System zur Positionsbestimmung, das für alle Anwendungsfälle geeignet ist. Es gibt große Unterschiede in der Genauigkeit und in den Kosten, die durch Bereitstellung und Nutzung entstehen. Sinnvoll erscheint eine Kombination verschiedener Systeme zur Positionsbestimmung, die dabei den Kontext des Nutzers berücksichtigt. Dazu gibt es mit *Assisted-GPS* einen viel versprechenden Ansatz, der Netzwerk- und GPS-basierte Positionsbestimmung vereint.

Genauigkeit Das GPS-System bietet eine hohe Genauigkeit bei der Positionsbestimmung, die auch global möglich ist. Allerdings haben Umweltbedingungen, wie z. B. schlechtes Wetter oder Reflexion der Signale durch Gebäude, Einfluss auf die Genauigkeit. Zudem können die GPS-Signale nur bei direktem Sichtkontakt des Empfängers mit den Satelliten empfangen werden. Eine Positionsbestimmung ist in der Regel nur im Freien möglich.

Die Positionsbestimmung mithilfe von WLAN und dem tabellenbasierten Verfahren kann auf wenige Meter genau sein. Diese Genauigkeit ist für viele Anwendungen ausreichend, da eine Aussage getroffen werden kann, auf welchem Stockwerk bzw. in welchem Raum sich ein Nutzer befindet. Die Bestimmung kann sowohl im Freien, als auch innerhalb von Gebäuden zum Einsatz kommen. Sollte sich die Netzwerktopologie nicht ändern, ist die Trainingsphase nur einmalig auszuführen. Die Messung der Signalstärke an sich kann dennoch zu Ungenauigkeiten führen; so können Messungen durch Reflexionen aufgrund sich bewegender und nur temporär vorhandene Objekte, wie z. B. Personen, die Werte verfälschen.

Die Genauigkeit der Positionsbestimmung mittels GSM ist nicht für alle Anwendungsfälle ausreichend. Zudem ist diese auch den beschriebenen Schwankungen, bedingt durch die verschiedenen Größen der Zellen, unterlegen. Dies muss von den Anwendungen entsprechend berücksichtigt werden.

[18] http://www.skyhookwireless.com/index.html; Zugriffsdatum: 04.05.2007
[19] http://www.microsoft.com/mappoint/products/locationserver/default.mspx; Zugriffsdatum: 04.05.2007
[20] http://placelab.org/; Zugriffsdatum: 04.05.2007

Kosten Nachteilig bei der GPS-Technologie sind insbesondere die hohen Kosten für die Installation und Instandhaltung der Satelliten, dennoch ist der GPS-Basisdienst für Nutzer kostenlos. Allerdings ist zusätzliche Hardware für mobile Endgeräte erforderlich, welche in der Lage ist, GPS-Signale zu empfangen.

Die Positionsbestimmung mittels WLAN basiert auf einer bereits vorhandenen Netzwerkinfrastruktur. Für die Positionsbestimmung ist keine zusätzliche Hardware erforderlich. Die Trainingsphase zur Erstellung der Positionstabelle ist aufwändig, sofern die Werte manuell erfasst werden.

Für die Positionsbestimmung mittels GSM kann auf eine bestehende, weit verbreitete Netzwerkinfrastruktur zurückgegriffen werden. Es ist keine zusätzliche Hardware bei einer Vielzahl mobiler Endgeräten, wie Mobilfunktelefonen, erforderlich. Weiterhin muss bei diesem Verfahren der Positionsbestimmung ein geeignetes Konzept zum Schutz der Privatsphäre der Nutzer integriert werden, da die Positionsinformationen nicht nur dem Nutzer, sondern auch dem System vorliegen. Zusätzlich können, je nach Dienstanbieter, für jede Positionsbestimmung Kosten entstehen.

2.4 Geografische Informations- und räumliche Datenbanksysteme

Geografische Informationssysteme (GIS) und räumliche Datenbanksysteme (engl.: spatial databases) (DBS) sind ein elementarer Bestandteil von ortsbezogenen Diensten. In diesen Systemen wird sämtliche Verarbeitung und Speicherung räumlicher Daten vorgenommen. Eine wie in Abschn. 2.2.4 vorgestellte Middleware für ortsbezogene Dienste bietet oftmals eine Programmierschnittstelle für die Funktionalitäten von GIS und räumlichen DBS.

Es existieren mehrere Definition von GIS. Im einfachsten Fall wird als GIS ein System zur Erzeugung grafischen Kartenmaterials angesehen. Ein GIS ist ein Informationssystem, das auf einem räumlichen DBS basiert und zusätzlich Funktionen zum Suchen, zur grafischen Ausgabe, zur Zusammenführung geografischer Daten und zur Darstellung spezifizierter Daten und Attribute usw. besitzt. Ein DBS ist ein System zur Beschreibung, Speicherung und Wiedergewinnung umfangreicher Datenmengen, die von mehreren Anwendungen genutzt werden (Claus und Schwill (2003)). Demnach übernimmt ein räumliches DBS diese Aufgaben speziell für räumlichen Daten. In diesem Abschnitt soll ein Überblick über GIS, räumliche DBS und räumliche Daten gegeben werden, wie er für das Verständnis der Arbeit erforderlich ist. Für weiterführende Informationen sei auf Rigaux u. a. (2001) und Shekhar und Chawla (2002) verwiesen.

GIS beinhalten zwei verschiedene Abstraktionsniveaus für räumliche Daten. Zum einen bieten sie eine konzeptionelle Sicht, das so genannte *geografische Datenmodell*. Daten dieser Abstraktionsebene bilden Elemente aus der realen Welt, wie z. B. Gebäude und Straßen, ab; diese Elemente werden *Abstract Data Types* (ADTs) genannt. Jedes der ADTs bietet eine definierte Menge an Operationen, so kann z. B. bei einem als Straße modellierten ADT die Länge abgefragt werden. Zum anderen bieten GIS eine physikalische Sicht, die das eigentliche, *räumliche Datenmodell* darstellt. Dessen Elemente werden als räumliche Objekte bezeichnet und können mit numerischen Werten dargestellt werden, die als *Punkte*, *Linien*, *Polylinien* und *Polygone* modelliert sind. Somit bildet das geografische Datenmodell mit seinen ADTs und den zugehörigen Operationen die Schnittstelle für Anwendungen zum physikalischen Datenmodell. Diese Schichtentrennung soll die Komplexität der physikalischen Datenhaltung vor der Anwendung verbergen und eine einheitliche Zugriffsschicht bieten. Abb. 2.5 greift die in Abschn. 2.2.4 eingeführte Middleware auf und zeigt eine detaillierte Sicht auf das GIS und räumliche DBS, die in diese Middleware integriert sind.

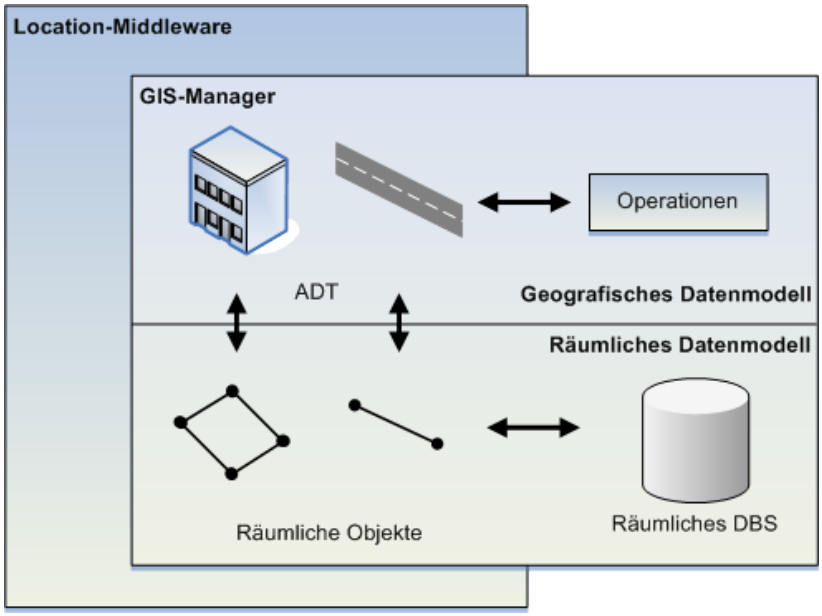

Abbildung 2.5: GIS und räumliches DBS einer Middleware für ortsbezogene Dienste

2.4.1 Räumliches Datenmodell

In diesem Unterabschnitt wird auf die Details der physikalischen Datenspeicherung räumlicher Objekte eingegangen.

2.4.1.1 Darstellung räumlicher Objekte

Für die Darstellung räumlicher Objekte existieren zwei mögliche Formen: Der Raster-Modus und der Vektor-Modus.

Raster-Modus Die Darstellung im *Raster-Modus* lässt sich mit der Darstellung von Objekten als Bitmap-Rastergrafik vergleichen. Die Daten werden in einem Raster aus Spalten und Zeilen, aus einer Menge von Pixeln beschrieben. Der Umriss von räumlichen Objekten wird somit durch die Anordnung benachbarter Pixel dargestellt. Die Position eines Objektes ergibt sich dabei implizit durch die einzelnen Positionen der Pixel im Raster. Entscheidend für eine gute Qualität der Daten ist somit eine hohe Auflösung des Rasters, wodurch jedoch der Speicherplatzbedarf und die Zeit zur Berechnung erhöht werden. Besonders ersteres ist bei der Kommunikation mit mobilen Rechnern ein Nachteil, da diese oft über drahtlose Schnittstellen mit geringer Bandbreite miteinander kommunizieren. Oftmals werden räumliche Objekte für die Darstellung im Raster-Modus auch direkt aus Bildern, wie z. B. Satellitenfotos, gewonnen.

Vektor-Modus In ortsbezogenen Diensten werden hauptsächlich Daten im *Vektor-Modus* gespeichert. Räumliche Objekte werden dabei durch ein Koordinatensystem und einen Wert beschrieben. Dies entspricht der in Abschn. 2.3.1 vorgestellten Darstellung numerischer Positionsinformationen. Demnach ist das simpelste räumliche Objekt ein einzelner Punkt, der dient als Ausgangspunkt für die Darstellung komplexerer Objekte dient. So lassen sich mit zwei Punkten eine gerade Linie und mit mehreren, geordneten Punkten eine Polylinie modellieren. Räumliche Objekte, die Oberflächenobjekte beschreiben, werden als Polygone modelliert. Ein Polygon ist eine geschlossene Polylinie, bei der erster und letzter Punkt identisch sind.

2.4.1.2 Datenspeicherung räumlicher Objekten

Zur Speicherung von Daten haben sich relationale Datenbankmanagementsysteme (RDBMS) bewährt. In RDBMS werden Daten in primitiven Datentypen gespeichert. Demnach müssen räumliche Objekte in einfache Datentypen überführt werden, wodurch geografische Daten über mehrere Relationen und mehrere Tupel gespeichert und bei entsprechenden Anfragen aggregiert werden müssen. Mit diesem Ansatz kann zwar die verbreitete Anfragesprache Structured Query Language (SQL) verwandt werden, allerdings ist die schlechtere Performanz durch die Verteilung auf viele Relationen und Tupel ein Nachteil.

Um diesem Nachteil zu begegnen, wird eine Erweiterung von RDBMS angestrebt. Geografische Objekte werden dabei in neu definierten Datentypen wie Punkte, Linien und Polygonen gespeichert. Dieser Ansatz folgt dem objekt-relationalen Ansatz von DBMS. Dazu ist die Nutzung einer erweiterten Anfragesprache notwendig (vgl. Egenhofer (1994)).

2.4.2 Geografisches Datenmodell

ADTs repräsentieren geografische Elemente aus der realen Welt. Die Darstellung und Struktur gleichartiger ADTs, sowie die Beziehungen der ADTs untereinander, werden in so genannten Themes zusammengefasst; es wird das Konzept der Überlagerung erreicht. So kann eine Karte z. B. aus Themes für Straßen und Orten von Interessen bestehen und somit dem Nutzer ein vollständiges Bild seiner Umgebung liefern. Dazu werden die einzelnen Themes überlagert angezeigt. Entscheidend für ADTs sind die entsprechend definierten Operationen, die je nach Typ des ADT ausgeführt werden können. Für eine genaue Beschreibung der einzelnen Operationen sei auf Rigaux u. a. (2001) verwiesen.

2.4.3 Datenbankanfragen mobiler Nutzer

In den vorherigen Unterabschnitten wurde die Darstellung und Speicherung räumlicher Daten diskutiert. In Abschnitt 2.3 wurden Verfahren zur Positionsbestimmung vorgestellt. An dieser Stellen sollen die Datenbankabfragen eingeführt werden, die ein mobiler Nutzer an diese Daten stellt. Dabei soll die Position des Nutzers berücksichtigt werden. Bei den in Abschn. 2.2.2 vorgestellten Informationsdiensten verändert der Nutzer seine eigene Position und holt Informationen über Objekte ein, deren Position sich nicht ändert. Im Gegensatz dazu stehen Datenbankanfragen an sich bewegende Objekte, wie sie in den in Abschn. 2.2.2 vorgestellten Tracking-Diensten vorkommen. Diese werden im Abschn. 2.4.4 diskutiert. Beiden Themen werden in Höpfner u. a. (2005) ausführlich behandelt.

2.4.3.1 Positionsbewusste und positionssabhängige Anfragen

Bei Datenbankanfragen mobiler Nutzer wird zwischen positionsbewussten und positionsabhängigen Anfragen unterschieden. Bei positionsbewussten Anfragen ist das Ergebnis unabhängig von der eigenen Position. Eine Positionsangabe ist expliziter Bestandteil der Anfrage, wobei die eigene Position und die Anfrageposition unterschiedlich sein können. Ein Beispiel ist: „Hat das (bestimmte) Restaurant einen barrierefreien Zugang?“. Anfragen dieser Art gehören im engeren Sinne nicht der Kategorie ortsbezogener Dienste an.

Ist Anfrageergebnis abhängig von der eigenen Position, spricht man von positionsabhängigen Anfragen. In der Anfrage findet sich keine explizite Positionsangabe. Die Positionsinformationen werden vom System transparent hinzugefügt. Beispiel: „Welche Restaurants (sind in der Nähe und) haben einen barrierefreien Zugang?“. Anfragen dieser Art sind typisch für ortsbezogene Dienste und sollen in den folgenden Unterabschnitten weiter betrachtet werden.

2.4.3.2 Anpassung von Positionsinformationen

In Abschn. 2.2 wurde erläutert, dass ortsbezogene Dienste verschiedene Informationen, in Abhängigkeit der aktuellen Position des Nutzers, bereitstellen. Um dies zu gewährleisten, müssen Operatoren zur Konvertierung und zum Vergleich positionsbezogenener Attribute bereitgestellt werden.

Konvertierung von Positionsinformationen Die Konvertierung von Positionsinformationen ist wichtig, damit verschiedene Formate zueinander kompatibel sind. Diese Funktionalität sollte von einer Middleware bereitgestellt werden, da die Formate der Positionsinformationen in verschiedenen Datenbanken heterogen sein können. Wie in Abschn. 2.3.1 erläutert, werden Positionsinformationen oft in numerischen Werten gespeichert. Diese Werte müssen häufig für den Nutzer in semantische Werte, z. B. Zeichenketten, übersetzt werden. Dies wird als *Geocoding* bezeichnet. So kann mittels geografischer Breite und Länge eine semantische Hausadresse ermittelt werden. Die Überführung semantischer in numerische Werte bezeichnet man entsprechend als *Reverse-Geocoding*.

Vergleich von Positionsinformationen Sind die Positionsinformationen mittels numerischer Werte in einer Datenbank gespeichert, können bekannte Vergleichsoperatoren auf diese angewandt werden. Bei positionsabhängigen Anfragen kann eine Abweichung des Nutzers zur Position mit hilfe einer Abstandsfunktion berücksichtigt werden. Diese Abstände beschreiben direkte, geradlinige Entfernungen zwischen Nutzer und Position und sind eventuell nicht für alle Anwendungsfälle geeignet. In solchen Fällen muss der Wertebereich in eine Hierarchie überführt werden, wodurch die Nutzung topologischer Distanzen und statistischer Methoden möglich wird.

2.4.4 Datenbankanfragen an bewegliche Objekte

Bei den in Abschn. 2.2.2 beschriebenen Tracking-Diensten stellt ein Nutzer Anfragen an sich bewegende Objekte. Der Nutzer selber kann eine feste Position haben oder diese ebenfalls verändern. Grundsätzlich ist es sinnvoll, Informationen über sich bewegende Objekte in einer Datenbank zu speichern. Konventionelle Datenbanken eignen sich dazu nur bedingt, da Informationen über die Position der Objekte permanent aktualisiert werden müssten. Dabei ist davon auszugehen, das eine Aktualisierung mit gewünschter Präzision und vertretbarem Aufwand nicht permanent durchgeführt werden kann. Die Kommunikation zwischen Datenbank und sich bewegendem Objekt sollte möglichst gering sein.

Bekannte Datenmodelle eignen sich zur Speicherung statischer Daten, ein geeignetes Datenmodell für sich bewegende Objekte muss dieses erweitern. Es reicht nicht aus, die Position eines Objektes zu Speichern; dynamische Aspekte, wie die Geschwindigkeit des Objektes

und die Zeit, müssen ebenfalls erfasst werden. Dadurch soll es möglich sein, die tatsächliche Position zu einem späteren Zeitpunkt abzuleiten. Ein Verfahren mit der Bezeichnung *Koppelnavigation* (engl.: deduced reckoning, dead reckoning) beschreibt die fortlaufende Ortung sich bewegender Objekte unter Berücksichtigung der letzten bekannten Position, sowie aktueller Messwerte der Bewegungsrichtung, der Geschwindigkeit und der Zeit. Da die Geschwindigkeit und die Bewegungsrichtung beim sich bewegenden Objekt gemessen werden, können diese nicht direkt für die Ableitung herangezogen werden; vielmehr werden hier, wie bei der Position, die letzten bekannten Werte herangezogen. Die Ergebnisse der abgeleiteten Positionsbestimmung sind daher mit einem Unsicherheitsfaktor behaftet, der ebenfalls berücksichtigt werden muss. Durch den Einsatz einer abgeleiteten Positionsbestimmung sind die Datenbankeinträge ungenau und nicht korrekt; sie sind nur für einen definierten Zeitpunkt gültig.

Diese Ungenauigkeit hat zwei Ursachen. Sie entsteht einerseits aus der ungenauen Positionsbestimmung (vgl. Abschn. 2.3.3), zum anderen werden bewusst Ungenauigkeiten in Kauf genommen, um die permanente Aktualisierung möglichst gering zu halten. In Höpfner u. a. (2005) wird zur Verdeutlichung die Funktion eines erweiterten Datenmodells für gleichförmige Bewegung eingeführt. Sei $L.updateValue$ die Position, $L.updateTime$ die Zeit und $L.velocity$ die Geschwindigkeit, dann ergibt sich folgende Funktion:

$$L.value(t) = L.updateValue + (t - L.updateTime) * L.velocity \tag{2.1}$$

Dazu wird zusätzlich $L.maxDeviation$ als Schwellwert für die maximale Abweichung eingeführt. Sobald die Abweichung von $L.value$ vom gemessen Wert den maximalen Schwellwert übersteigt, wird eine Aktualisierung durchgeführt. Im einfachsten Fall der Geschwindigkeitsabschätzung wird der Wert für $L.maxDeviation$ genau einmal festgelegt. In einer anpassbaren Abschätzung kann der Wert neu festgelegt werden. Dies ist sinnvoll, wenn sich die Rahmenbedingungen ändern, wenn z. B. die Aufwände für Aktualisierungen größer oder kleiner werden. Dies kann bei verschiedenen Möglichkeiten der drahtlosen Datenübertragung, z. B. beim Wechsel vom GSM-Netz in ein WLAN, gegeben sein.

Durch die sporadische Aktualisierung der Positionsinformationen ergibt sich aber ein weiteres Problem. Grundsätzlich besteht die Möglichkeit, dass der Aktualisierungsmechanismus durch Verbindungsprobleme zur Datenbank nicht verfügbar ist. Dies muss entsprechend berücksichtigt werden. Die Unsicherheit nach der letzten Aktualisierung steigt bis zur nächsten Aktualisierung stetig an. Um dem entgegenzuwirken, sinkt der Wert der maximalen Abweichung exponentiell. Hat diese einen hinreichend kleinen Wert erreicht, ist der Aktualisierungsmechanismus nicht mehr verfügbar.

2.5 Web 2.0

Wie in Abschn. 1.1 erläutert, hat das Web 2.0 aktuell eine sehr hohe Sichtbarkeit; es wird viel darüber berichtet. Der Begriff Web 2.0 selbst wurde im Aufsatz O'Reilly (2005) geschaffen. Ähnlich wie bei einer Software, soll die Versionsnummer einen Fortschritt suggerieren. Was sich genau hinter dem Web 2.0 verbirgt, ist nicht immer klar. Oftmals wird es nur mit der Technologie Asynchronous Java Script and XML (AJAX) in Verbindung gebracht, im schlimmsten Fall wird es allein darauf reduziert. Tatsächlich steckt hinter dem Web 2.0 vielmehr die Idee, das Verhalten der Nutzer zu ändern. Der Nutzer wird nicht mehr als reiner Konsument von Daten, sondern als Teil einer Gemeinschaft angesehen, die Daten über das Internet austauscht. Das Web 2.0 definiert sich somit als eine Ansammlung von Prinzipien und Praktiken.

2.5.1 Eigenschaften

Im Folgenden werden die elementaren Eigenschaften des Web 2.0, die in O'Reilly (2005) identifiziert werden, aufgezeigt. Darauf aufbauend wird eine Abgrenzung zum Internet der ersten Generation skizziert.

Web als Plattform Die typische Software im Web 2.0 nutzt das Internet als Plattform. Software ist nicht mehr der so genannten Desktop-Metapher nachempfunden. Sie wird nicht in einem Geschäft auf einem Datenträger verkauft und dann auf einzelnen Rechnern mit einem bestimmten Betriebssystem installiert; vielmehr ist die Software eine Dienstleistung. Die Nutzer zahlen direkt oder indirekt für den Gebrauch. Gängige Lizenzmodelle des Software-Vertriebs greifen nicht mehr. Dadurch kann die Software in kürzester Zeit eine breite Masse von Nutzern erreichen. Viele Nutzer sind wichtig, damit ein Dienst Erfolg hat. Eine Vielzahl von Diensten im Web 2.0 wird dem Nutzer kostenlos angeboten. Dies verlangt nach einem tragfähigen Geschäftsmodell.

Nutzung kollektiver Intelligenz Dienste im Web 2.0 stellen oftmals technische Rahmenbedingungen für Nutzer zur Verfügung. Die Nutzer sollen in diesem Rahmen agieren, indem sie Daten erstellen und diese miteinander verknüpfen. Dies wird im Web 2.0 unter anderem durch die Technik Really Simple Syndication (RSS) stark vereinfacht. Durch die RSS-Technologie können Teile von Web-Seiten in andere Web-Seiten integriert werden. Im Prinzip stellen RSS dynamische Querverweise dar, die es wie bei einem Abonnement ermöglichen, permanent neue Daten über den gleichen Verweis zu beschaffen. Diese Netzwerkeffekte durch die Nutzerbeteiligung scheinen ein Schlüssel zum Erfolg zu sein, wie die rasante Verbreitung dieser Technologie beweist.

Wichtigkeit der Daten Daten stellen den wichtigsten Bestandteil jeglicher Software im Web 2.0 dar. Generell ist die Kontrolle wichtiger Datenquellen bedeutsam, vor allem, wenn

die Gewinnung der Daten große Ressourcen verlangt. Die Software-Infrastruktur von Diensten im Web 2.0 wird häufig als Open-Source[21] bereitgestellt. So sind es oftmals die Daten, die einen wirklichen Wert haben und Umsätze generieren können. Als besonders wichtig im Web 2.0 werden geografische, persönliche und terminliche Daten angesehen. Es ist dabei vorteilhaft, wenn diese Daten von verschiedenen Diensten zusammengeführt werden und in einen neuen Dienst münden, der eine große Anzahl von Nutzern erreicht. Das Zusammenführen wird in diesem Zusammenhang als *Mash-Up* bezeichnet. Dabei sind allerdings die Privatsphäre der Nutzer und deren Rechte an den eigenen Daten fraglich. Beispiel eines derartigen Dienstes ist Yahoo pipes[22].

Software-Lebenszyklen Da Software im Web 2.0 als Dienst und nicht als Produkt ausgeliefert wird, greifen die typischen Lebenszyklen von Softwareprodukten nicht. Software wird als Dienstleistung betrachtet, dabei permanent weiterentwickelt und in sehr kurzen Zyklen, teilweise innerhalb weniger Stunden, bereitgestellt. Dazu wird auch der Nutzer in den Entwicklungsprozess mit einbezogen und seine Meinung zu Merkmalen der Software berücksichtigt. Dies wird dadurch deutlich, dass sich viele Software-Angebote im Web 2.0 im so genannten *Beta*-Status befinden.

Leichtgewichtige Programmiermodelle Erfolgreiche Dienste im Web 2.0 unterstützen leichtgewichtige Programmiermodelle wie Web-Services via Simple Object Access Protocol (SOAP) und Remote Procedure Calls (RPC) (W3C (2007)). Eine weiteres, einfaches Programmiermodell ist der so genannten Represential State Transfer (REST) (Fielding (2000)), eine Variante aus Extended Markup Language (XML) und Hyper Text Tranfer Protocol (HTTP). Ziel ist eine lose Kopplung verteilter Systeme. Daten sollen möglichst einfach verteilt und ausgetauscht werden können. Diese technische Grundlage soll innovative neue Dienste entstehen lassen, die sich einfach aus bestehenden Diensten zusammenbauen lassen, und deren Daten einen besonderen Wert haben.

Heterogene Geräte Dienste des Web 2.0 sind nicht ausschließlich als Web-Anwendungen konzipiert. Vielmehr nutzen sie das gesamte Web als einen Netzwerkverbund als integralen Bestandteil ihrer Infrastruktur. Beispielsweise nutzt der Dienst Apple iTunes[23] mobile Musikabspielgeräte, Personal Computer und Server im Internet, um dem Nutzer seinen Wünschen entsprechend Musik zu liefern.

Benutzerführung Die Benutzerführung von Diensten im Web 2.0 wird durch Technologien wie Macromedia Flash[24] und AJAX verbessert. Mit diesen Technologien ist es mög-

[21] Bei dieser Art von Software ist der Quelltext offen zugänglich und darf beliebig verändert und weitergegeben werden.
[22] http://pipes.yahoo.com/pipes/; Zugriffsdatum: 04.05.2007
[23] http://www.apple.com/itunes; Zugriffsdatum: 04.05.2007
[24] http://www.adobe.com/de/products/flash/; Zugriffsdatum: 04.05.2007

lich, Benutzerschnittstellen zu schaffen, die sich in der Bedienung kaum von Desktop-Anwendungen unterscheiden, jedoch in Web-Browsern ausgeführt werden können.

2.5.2 Abgrenzung

Wie oben erläutert, geht es beim Web 2.0 im Wesentlichen um Inhalte und Wissen und um Daten, die von den Nutzern miteinander verknüpft werden. Die Dienste im Web 2.0 unterstützen somit den Aufbau und die Pflege sozialer Netzwerke und Gemeinschaften, die diese Daten dann in Selbstorganisation generieren. Die Dienste unterstützen demnach menschliche Kommunikation und Kollaboration und werden entsprechend als soziale Software (engl.: social software) bezeichnet. Mit Blick auf die Vergangenheit des Webs wird deutlich, dass die Idee des Web 2.0, kollektive Intelligenz zu nutzen, nicht neu ist. So war das World Wide Web (WWW) in den 90er Jahren des letzten Jahrhunderts ebenfalls entwickelt worden, um Menschen und nicht Rechner miteinander zu verbinden (Berners-Lee (1996)). Mit der Jahrtausendwende begannen Firmen, das Internet zur Vermarktung von Informationen zu nutzen. Sie haben Inhalte bereitgestellt und die Nutzer zu passiven Konsumenten degradiert. Die Idee der rechnergestützten, verteilten Zusammenarbeit und der Aufbau und Austausch kollektiven Wissens lassen sich aber noch weiter zurückverfolgen. Bereits in den 1960er Jahren wurden die Grundsteine für die kollektive Wissensverarbeitung mit Hilfe von Rechnern gelegt (vgl. Engelbart (1962) und Engelbart (1972)).

2.5.3 Soziale Software

In Bächle (2006) und Hippner (2006) werden folgende Dienste für soziale Software im Internet skizziert. Letzterer beschreibt zudem ein Klassifikationsschema bezüglich der drei Zielsetzungen Beziehungsaufbau, Informationsaustausch und Kommunikation der Nutzer mittels sozialer Software (Abb. 2.6).

Forum Ein Forum ist eine Web-Anwendung und dient der Diskussion verschiedenster Themen. In aller Regel behandelt ein Forum ein bestimmtes Thema und wird dazu in Unterthemen bzw. Unterforen aufgeteilt. Die Nutzer können Einträge hinterlassen, die dann gelesen und beantwortet werden können. Mehrere Beiträge zum selben Thema werden zusammenfassend als Thread bezeichnet. Sehr häufig übernehmen ausgewählte Nutzer oder die Bereitsteller des Forums die Moderation und überwachen, ob einzelne Beiträge der Nutzer etwaigen Regeln entsprechen. Das Forum MyHAW[25] bietet Studenten der HAW Hamburg eine geeignete Plattform für Diskussionsthemen rund um das Studium.

[25] http://myhaw.de; Zugriffsdatum: 04.05.2007

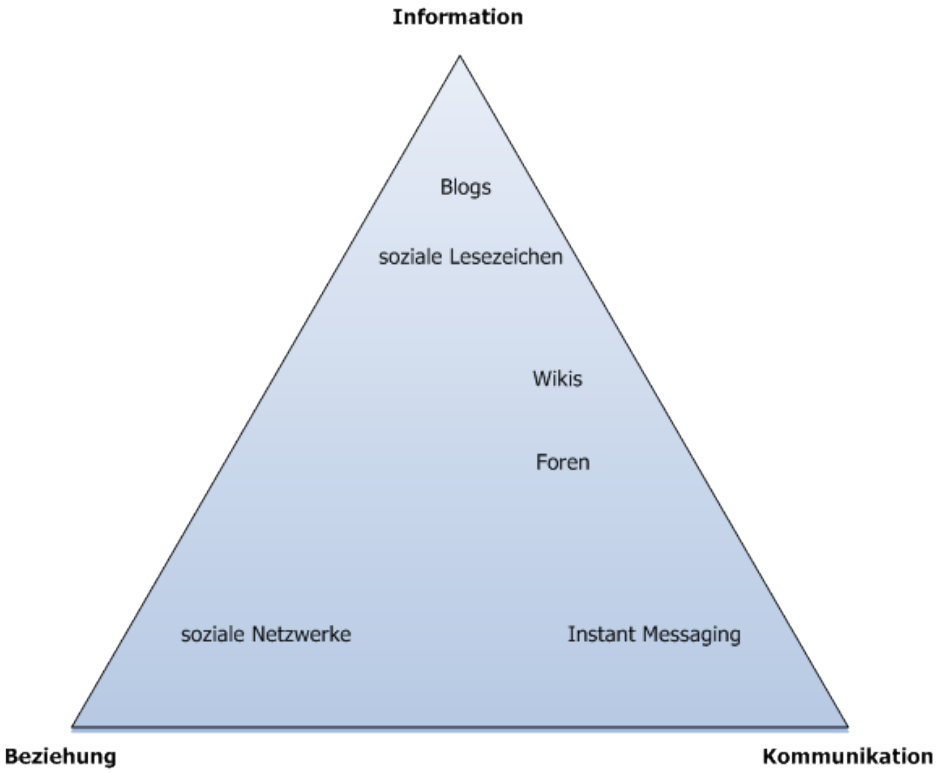

Abbildung 2.6: Klassifikationsschema sozialer Software (nach Hippner (2006))

Instant Messaging Dieser Client-Server-basierte Dienst ermöglicht die Kommunikation der Nutzer untereinander in Echtzeit. Die Kommunikation kann sowohl via Text als auch via Sprache durchgeführt werden. Letzteres gewinnt durch die Voice over Internet Protocol (VoIP)-Technologie immer mehr an Bedeutung. Die Kommunikation verläuft im Vollduplex-Modus. Oftmals sind auch zusätzliche Programme zur unterstützenden Zusammenarbeit in die Client-Anwendungen integriert. Die Client-Anwendung kann Web-basiert oder ein Smart-Client sein. Bekannte Dienste sind z. B. der Windows Live Messenger[26] der Firma Microsoft und der Yahoo Messenger [27] der Firma Yahoo. Obwohl diese Dienste von Firmen bereitgestellt werden, sind sie in aller Regel kostenlos.

Wiki Wikis sind Werkzeuge zum Wissensmanegement im Internet. Nutzer stellen Einträge in eine Art Seitensammlung ein. Wikis sind somit Content Manegement Systemen (CMS) sehr ähnlich. Das besondere dabei ist, das Nutzer diese Einträge jederzeit ändern können, auch wenn sie selber nicht Verfasser der Einträge sind. Einzelne Bei-

[26] http://get.live.com/messenger/overview; Zugriffsdatum: 04.05.2007
[27] http://de.messenger.yahoo.com/; Zugriffsdatum: 04.05.2007

träge sind dem Paradigma des WWW entsprechend durch Querverweise miteinander verbunden. Bekanntestes Wiki ist Wikipedia[28], eine frei Enzyklopädie.

Blog Ein Blog ist ein Tagebuch eines Nutzers, welches dieser im Internet veröffentlicht. Verfasser werden als so genannte Blogger bezeichnet. In ihren Beiträgen widmen sie sich in aller Regel einem spezifischen Thema. Leser können Kommentare zu den Beiträgen einstellen und die Autoren können auch Querverweise zu anderen Blogs herstellen. Dadurch können Netzwerke von Beiträgen und Kommentaren entstehen. Insbesondere das relativ unkomplizierte Aufsetzen und Pflegen der Blogs wird von vielen Nutzern als angenehm angesehen. Beiträge von Blogs dienen oftmals der Unterhaltung, aber auch der Wissensgewinnung; zur Zeit werden sie immer mehr als Marketinginstrumente genutzt. Einerseits erhoffen Unternehmen, dadurch einen besseren, nicht sofort als direkte Werbung ersichtlichen Zugang zu ihrer Zielgruppe zu bekommen; so wird mit Blogs vermehrt virales Marketing betrieben[29]. Zum anderen versuchen Unternehmen, Trends ihrer Kunden und Konsumenten aus Blogs zeitig zu erkennen, um entsprechend darauf reagieren zu können[30].

Soziale Lesezeichen Systeme für *soziale Lesezeichen*[31] dienen der Erfassung und Kategorisierung von Lesezeichen zu Web-Seiten. Diese Lesezeichensammlungen werden der Allgemeinheit zugänglich gemacht. Sie werden dabei mit Schlagworten, so genannten *Tags* annotiert. Das besondere dabei ist, dass durch Tags keine strenge Hierarchie der Begriffe erreicht wird; vielmehr steht es jedem Nutzer frei, sein eigenes Begriffssystem aufzubauen. Die Lesezeichen werden gemäß ihrer Annotierung miteinander verbunden. Dadurch schaffen sich Netzwerke von Informationen gleichgesinnter Nutzer, die über die Möglichkeiten von Suchmaschinen hinausgehen können. Ein beliebter Dienst für soziale Lesezeichen ist del.icio.us[32].

Soziale Netwerke *Soziale Netzwerke* dienen dem Auffinden gleichgesinnter Nutzer. So können zielgerichtet privat oder beruflich orientierte Beziehungen über das Internet aufgebaut werden. Jeder Nutzer hinterlegt ein einsehbares Profil über seine Person. Eine interessante Möglichkeit ist die Analyse der Beziehungen zwischen eigenen, direkten Kontakten und zwischen anderen Kontakten. Nachteilig bei einigen Ausprägungen dieser Software ist die Unterscheidung zwischen zahlenden und nicht zahlenden Nutzern, wobei letzteren wichtige Funktionen oftmals nicht zur Verfügung stehen. So können beim sozialen Netzwerk OpenBC[33] nur zahlende Mitglieder Nachrichten untereinander austauschen. Das Web 2.0 soll jedoch als Plattform genutzt werden, um eine

[28] http://de.wikipedia.org; Zugriffsdatum: 04.05.2007
[29] vgl. Spiegel Online (http://www.spiegel.de/netzwelt/web/0,1518,475822,00.html, Zugriffsdatum: 04.05.2007)
[30] vgl. golem.de (http://www.golem.de/0511/41487.html, Zugriffsdatum: 04.05.2007)
[31] Engl: social bookmarking
[32] http://del.icio.us; Zugriffsdatum: 23.02.2007
[33] http://www.openbc.com; Zugriffsdatum: 11.01.2007

möglichst breite Masse an Nutzern zu erreichen. Das soziale Netzwerk für Studierende studiVZ[34] ist für den Nutzer kostenlos und bietet jedem die gleichen Funktionalitäten. Allerdings stellt sich dabei erneut die Frage nach einem geeigneten Geschäftsmodell.

2.6 Fazit

Mobile Anwendungen haben besondere Eigenschaften und es entstehen Anforderungen, die bei der Entwicklung berücksichtigt werden müssen. Mobile Laufzeitumgebungen bieten eine geeignete Plattform für mobile Anwendungen. Vor allem dienen sie der Unterstützung heterogener Hardware und bieten ein hohes Maß an Basisfunktionalität. Die Konzepte der am weitesten verbreiteten Laufzeitumgebungen Java Plattform und .NET sind ähnlich.

Mobile Geräte und Anwendungen schaffen die nötigen Voraussetzungen für ortsbezogene Dienste. Diese Dienste können den Nutzer, in Abhängigkeit seiner aktuellen Position, gezielt mit Informationen versorgen und somit einen Mehrwert bieten. Die Dienste selber stellen erweiterte Anforderungen, welche entsprechend berücksichtigt werden müssen. Um ortsbezogene Dienste zu realisieren, bedarf es zwei verschiedener Komponenten, welche über eine Middleware integriert werden können. Es werden ein System zur Positionsbestimmung und ein geografisches System zur Speicherung räumlicher Daten benötigt.

Es existieren verschiedene Systeme zur Positionsbestimmung, wobei jedes System sowohl Vor- als auch Nachteile hat. Ein mobiles Informationssystem für ortsbezogene Dienste sollte daher so aufgebaut sein, dass es mehrere Systeme unterstützt.

Geografische Informationssysteme können ortsbezogene Daten, wie Karten und Orte von Interesse, speichern, verarbeiten und anzeigen. Dabei werden Positionsinformationen entsprechend berücksichtigt. Die Daten werden in Datenbanken gespeichert, welche bekannte objekt-relationale Konzepte erweitern. Insbesondere die Speicherung von sich bewegenden Objekten erfordert zusätzliche Maßnahmen.

Die einfache Nutzung entsprechender Software macht das Web 2.0 für eine große Anzahl an Nutzern interessant. Die Nutzer generieren mithilfe von Diensten Inhalte und bilden durch deren Veröffentlichung mit anderen Nutzern Netzwerke. Diese von Nutzern generierten Inhalte haben einen besonders hohen Wert und stellen ein Alleinstellungsmerkmal dar. Zudem werden die Inhalte von den Nutzern kostenlos erzeugt. Die Eigenschaft des Web 2.0, eine verteilte, vernetze Zusammenarbeit mittels Rechner zu schaffen, ist nicht neu; mit der aktuellen Technik ist es jedoch erstmals möglich, eine breite Masse von Nutzern zu adressieren.

[34] http://www.studivz.net; Zugriffsdatum: 09.04.2007

3 Analyse

Nachdem wichtige Grundlagen erläutert wurden, werden in diesem Kapitel die heraus gearbeiteten Anforderungen an das in dieser Arbeit entwickelte Informationssystem vorgestellt. Dazu wird zunächst die Trailblazers-Software beschrieben, die als Ausgangsszenario dient. Dieses soll in dieser Arbeit weiterentwickelt und somit als roter Faden und Anwendungsszenario dienen. Dazu wird diskutiert, wie ortsbezogene Dienste und das Paradigma des Web 2.0 gewinnbringend zusammengeführt werden können und es soll aufgezeigt werden, ob und wie ortsbezogene Dienste zum Schlüsselfaktor für mobile Anwendungen werden können. Aus den Erkenntnissen der Diskussion werden funktionale und nichtfunktionale Anforderungen an das Informationssystem abgeleitet. Der Entwurf und die Implementierung in den nachfolgenden Kapiteln sollen die in diesem Kapitel gewonnenen Erkenntnisse manifestieren.

3.1 Ausgangsszenario

Das in dieser Arbeit entwickelte mobile Informationssystem ist eine Erweiterung der in Stegelmeier u. a. (2006) vorgestellten Trailblazers-Software, die somit als Grundlage dient. Die Trailblazers-Software wird im Folgenden mit der Version *Alpha* gekennzeichnet, die in dieser Arbeit weiterentwickelte Version wird mit der Version *Beta* bezeichnet.

3.1.1 Motivation

Viele Städte sind nicht barrierefrei (vgl. ADAC (2003)). Insbesondere müssen Rollstuhlfahrer und andere, in ihrer Mobilität eingeschränkte Personen, Wege um Hindernisse herum finden. Hindernisse können dabei Treppen, Kopfsteinpflasterstraßen, aber auch Eingänge in Gebäude sein. Dies sind im weiteren Sinne Orte von Interesse für diese Nutzer. Wege ohne Hindernisse sind oft nicht ausgeschildert oder sofort ersichtlich. Derartige Informationen sind in bekannten Navigationssystemen nicht enthalten, da diese ausschließlich Straßenkarten enthalten und somit für die Navigation mit Kraftfahrzeugen optimiert sind (vgl. Navteq (2007)). Mit Trailblazers wurde eine Software entworfen, die diese zusätzlichen Informationen enthält.

3.1.2 Übersicht und Abgrenzung

Die Trailblazers-Software in der Version Alpha bietet Nutzern zwei grundlegende Funktionalitäten: Eine Plattform um eine Gemeinschaft zu bilden, sowie die mobile, barrierefreie Navigation. Mit der Gemeinschaftsplattform können sich die Nutzer im System registrieren, ein persönliches Profil anlegen, Blogs verfassen und Nachrichten austauschen. Mit der Navigation können sie sich auf einem barrierefreien Weg von einer Start- zu einer Zieladresse führen lassen.

Die Gemeinschaftsplattform soll in dieser Arbeit nicht weiter betrachtet werden. In der Masterarbeit von Piotr Wendt (Wendt (2007)) werden Fragestellungen in Bezug auf Web-Anwendungen im Web 2.0 diskutiert. Dabei dient die Gemeinschaftsplattform als Anwendungsszenario.

Trailblazers ist eine mobile Client-Server-Anwendung. Für die beiden fachlichen Anwendungsfunktionalitäten existieren zwei verschiedene Client-Anwendungen: Die Gemeinschaftsplattform ist ein Web-Client, die mobile Navigation ist eine Smart-Client-Anwendung für mobile Standardcomputer. Abb. 3.1 zeigt eine Übersicht.

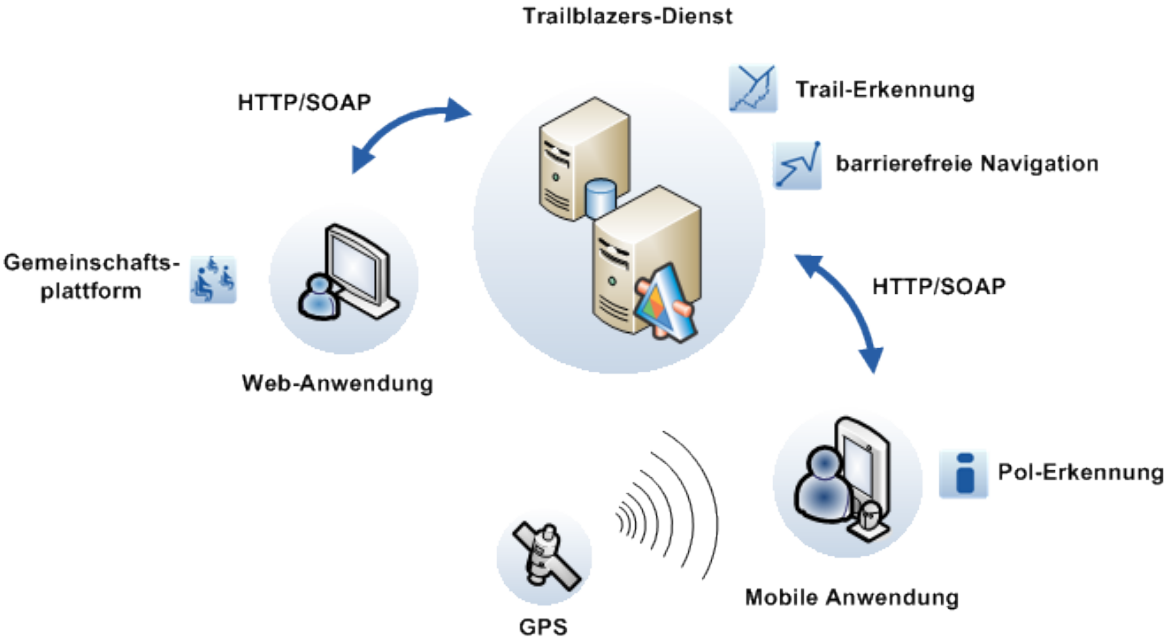

Abbildung 3.1: Trailblazers Alpha

Für die mobile Navigation kann der Nutzer Informationen zur barrierefreien Navigation von der Server-Anwendung abrufen, bekommt diese auf dem mobilen Gerät angezeigt und kann sie dort speichern. Die Client-Anwendung muss nicht permanent mit der Server-Anwendung

verbunden sein. Die Routenberechnung wird dennoch ausschließlich auf dem Server vorgenommen. Es werden unterschiedliche Nutzerprofile bzw. Nutzergruppen unterstützt. Diese enthalten Informationen über die verwendete Rollstuhlart, da sich dadurch unterschiedliche Anforderungen an eine Route ergeben können.

Karten, barrierefreie Wege und Informationen über Hindernisse werden von der Server-Anwendung via Web-Services an den Client übertragen. Während der Nutzung der mobilen Anwendung werden Positionsinformationen fortwährend gespeichert und bei Kommunikation mit der Server-Anwendung an diese übertragen. Aus den Positionsinformationen aller Nutzer werden neue barrierefreie Wege erstellt. Die Bestimmung der Position wird via GPS durchgeführt.

Die Server-seitige Erstellung der barrierefreien Wege aus den gesammelten Positionsinformationen, wird in dieser Arbeit nicht weiter betrachtet. Für das Konzept und die Implementierung, sowie für die damit verbundenen Herausforderungen sei auf Stegelmeier u. a. (2006) verwiesen.

3.1.3 Anwendungsfälle

In Abb. 3.2 skizzieren die Anwendungsfälle der Trailblazers-Software Alpha die Funktionalität des Informationssystems aus Sicht des Nutzers bei der Kommunikation mit dem System (vgl. Jacobsen (1995)). *Nutzer* ist der Hauptakteur des Systems. Dieser ist abgeleitet von dem abstrakten Akteur *Gemeinschaftsmitglied*, einer Person, die Informationen generiert, um diese der Gemeinschaft aller Nutzer zur Verfügung zu stellen. Der Akteur Nutzer ist eine Person, die Dienste aktiv nutzt, um gezielt Informationen für sich damit zu gewinnen. Kernanwendungsfall ist die *barrierefreie Navigation* für den Nutzer. Dieser Anwendungsfall wird durch die Anwendungsfälle *Trail-Erkennung* und *Point of Interest-Erkennung* erweitert.

3.1.3.1 Barrierefreie Navigation

Der Anwendungsfall barrierefreie Navigation basiert auf Anwendungsfällen von Navigationssystemen, wie sie z. B. in Jagoe (2003) vorgestellt werden. Der Nutzer gibt auf dem mobilen Gerät eine Start- und eine Zieladresse ein. Diese Anfrage wird an die Server-Anwendung übertragen, die barrierefreie Route berechnet, sowie Kartenmaterial, barrierefreie Wege und Orte von Interesse für die Route aus dem Datenbestand geladen. Diese Informationen werden zurück an die mobile Anwendung übertragen und für den Nutzer entsprechend visualisiert. Zusätzlich wird in der mobilen Anwendung die aktuelle Position des Nutzers auf der Karte angezeigt.

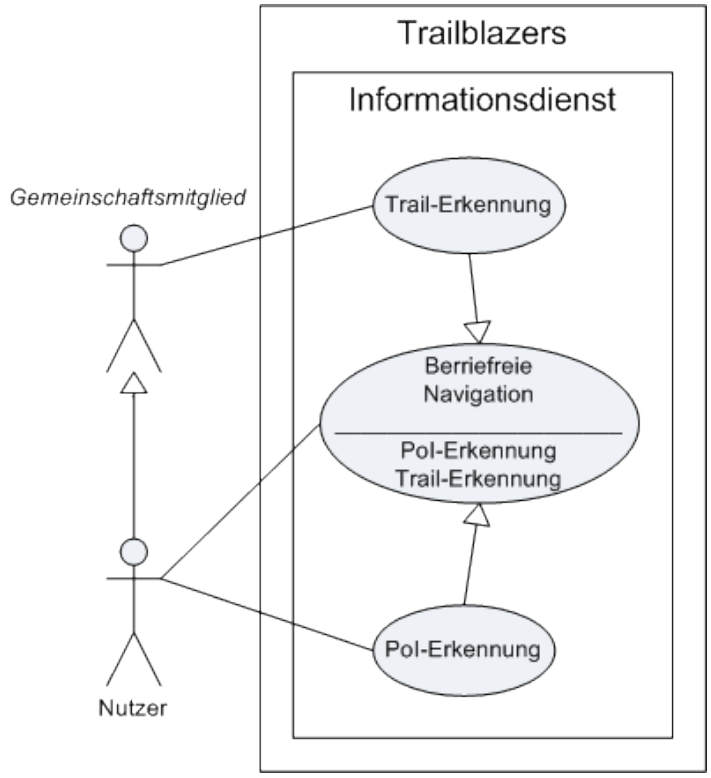

Abbildung 3.2: Anwendungsfalldiagramm Trailblazers Alpha

3.1.3.2 Trail-Erkennung

Wie oben erwähnt, sind keine Informationen über barrierefreie Wege in verfügbaren Navigationssystemen vorhanden. Im Trailblazers-System werden diese Wege von Nutzern gesammelt und der Gemeinschaft zu Verfügung gestellt. Bei jeder Person, die die Funktion zur barrierefreien Navigation nutzt, werden die GPS-Positionsdaten automatisch aufgezeichnet. Die Daten werden je nach Online- oder Offline-Szenario direkt an die Server-Anwendung übertragen oder auf dem mobilen Gerät zwischengespeichert. Aus den Daten aller Nutzer werden mittels Mustererkennung neue barrierefreie Wege erzeugt, die dann wieder allen Nutzern mit der barrierefreien Navigation zur Verfügung gestellt werden. Die Wege werden als Trampelpfade (engl.: trails) bezeichnet, um die Analogie zu Trampelpfaden auf einer Wiese in der realen Welt deutlich zu machen. Je mehr Personen einen Weg nutzen, desto ausgeprägter wird dieser Weg; wird er nicht mehr genutzt, wächst Gras darüber. Die Wege gelten als barrierefrei, da sie von Nutzern gesammelt werden, die sich, wie in ihrem Nutzerprofil hinterlegt, mit einem Rollstuhl fortbewegen.

3.1.3.3 Pol-Erkennung

Bei der barrierefreien Navigation müssen verschiedene Krankheitsbilder und Rollstuhltypen berücksichtigt werden. Dies trifft auf die Wege und im Besonderen auf die Barrieren an sich zu. Z.B. ist eine Treppe mit wenigen Stufen nicht für jeden Nutzer eine Barriere. Dies muss bei der Routenberechnung berücksichtigt werden, wozu entsprechende Informationen im Nutzerprofil hinterlegt sind. Die barrierefreien Wege können vom Nutzer zusätzlich mit so genannten Orten von Interesse annotiert werden. Der Nutzer kann mit seinem mobilen Gerät ein Foto von einer Barriere machen, diese klassifizieren und eine Beschreibung hinzufügen. Diese Informationen über eine Barriere werden ebenfalls an die Server-Anwendung übertragen und allen Nutzern der barrierefreien Navigation zur Verfügung gestellt. Bei der Routenberechnung werden diese Informationen zusammen mit dem Nutzerprofil entsprechend berücksichtigt.

3.1.4 Erweiterungen

Im weiteren Verlauf dieser Arbeit sollen mögliche Erweiterungen der mobilen Trailblazers-Software Alpha diskutiert werden, um diese in die Version Beta zu überführen. Möglichkeiten, die sich durch das Paradigma des Web 2.0 ergeben, sollen dabei in den Vordergrund gestellt werden. Die Erweiterungen können sich auf die bisher vorhandenen Anwendungsfälle beziehen; es soll aber ebenso diskutiert werden, ob neue Anwendungsfälle hinzukommen können.

Die oben beschriebene Motivation, sowie das gewählte Ausgangs- und das daraus resultierende Anwendungsszenario, seien lediglich stellvertretend für diese Arbeit aufgeführt. Grundsätzlich sollen sich die angestellten Überlegungen generalisieren und auf andere Anwendungsszenarien übertragen lassen. Die Trailblazers-Software könnte z. B. als mobiles Navigationssystem im Konsumentenmarkt speziell für Jogger, Fahrradfahrer, Eltern mit Kinderwagen oder Touristen in Städten optimiert werden. Ebenso sind Anwendungen im Industriemarkt, z. B. im Logistikbereich, möglich. Es sind jegliche Anwendungsszenarien denkbar, in denen eine personalisierte Navigation durch kontextbezogene Daten, die von anderen Nutzern generiert wurden, von Vorteil ist. Dieses Prinzip ist vergleichbar mit der so genannten *Social Navigation*, wie sie bei Web-Anwendungen zu finden ist (vgl. Dieberger u. a. (2001)).

3.2 Diskussion

In Martens u. a. (2006) wird erläutert, dass sich ortsbezogene Dienste trotz aller Vorhersagen bisher nicht etabliert haben. Zum einen wurden durch den Aufbau der Mobilfunknetze der dritten Generation sämtliche finanzielle und personelle Ressourcen gebunden. Zum anderen wurden keine geeigneten Infrastrukturen aufgebaut, die die Akzeptanz der Nutzer förderten. Insbesondere ein verbesserter Schutz der Privatsphäre, sowie die Unterstützung von Systemen, die eine genauere Positionsbestimmung ermöglichen, als das COO-Verfahren im GSM-Netz, wurden nicht praktisch verfolgt. Diese ungünstigen Rahmenbedingungen unterliegen gegenwärtig jedoch einem Wandel. Es entstehen ortsbezogene Dienste der nächsten Generation, die in ihrem Inhalt durch geografische Daten neuer Anbieter, wie z. B. Google, Yahoo und Microsoft, gekennzeichnet sind. Darüber hinaus ändern sich die Eigenschaften der angebotenen Dienste. Dienste können proaktiv sein, Nutzer können untereinander in Beziehung stehen, die Zielobjekte können mehrere Nutzer gleichzeitig sein, die Infrastruktur muss nicht zentralisiert aufgebaut sein und die Dienste können auch innerhalb von Gebäuden ausgeführt werden. Abb. 3.3 zeigt diese Eigenschaften ortsbezogener Diensten der neuen Generation. Darüber hinaus ändert sich auch hier das Verhalten von Nutzern der Dienste.

In der folgenden Diskussion werden die teilnehmenden Akteure, die Dienste an sich, die Daten, mögliche Geschäftsmodelle, sowie Erkenntnisse für die Privatsphäre der neuen Generation ortsbezogener Diensten im Zeitalter des Web 2.0 herausgearbeitet und entsprechende Konsequenzen für Trailblazers Beta besprochen.

3.2.1 Akteure

Nach Gasenzer (2001) beinhaltet die Wertschöpfungskette bei ortsbezogenen Diensten der ersten Generation die Akteure Inhalteanbieter, Dienstanbieter und Mobilfunknetzbetreiber. Bei einem ortsbezogenen Informationsdienst z. B. bestimmt der Mobilfunknetzbetreiber die Position des Nutzers und übernimmt die mobile Datenübertragung. Ein oder mehrere Inhalteanbieter liefern Orte von Interesse sowie geografische Informationen und übernehmen die Routenberechnung. Der Dienstanbieter führt alles zusammen und stellt den Dienst dem Nutzer zur Verfügung. Die Bezahlung kann dabei je nach System direkt beim Dienstanbieter oder beim Mobilfunknetzbetreiber erfolgen.

Insbesondere die Mobilfunknetzbetreiber hoffen, durch ortsbezogene Dienste neues Wachstum zu erlangen und durch den Aufbau neuer Geschäftsfelder ihrem Umsatz zu stabilisieren bzw. zu steigern. Zuletzt hat dieser durch Wettbewerb und Preissenkungen, sowie durch neue Kommunikationstechnologien im Kerngeschäft, der Telefonie, abgenommen (Gartner (2006b)). Dienste der zweiten Generation mit Offline-Funktionalität, kostenlose Dienste für

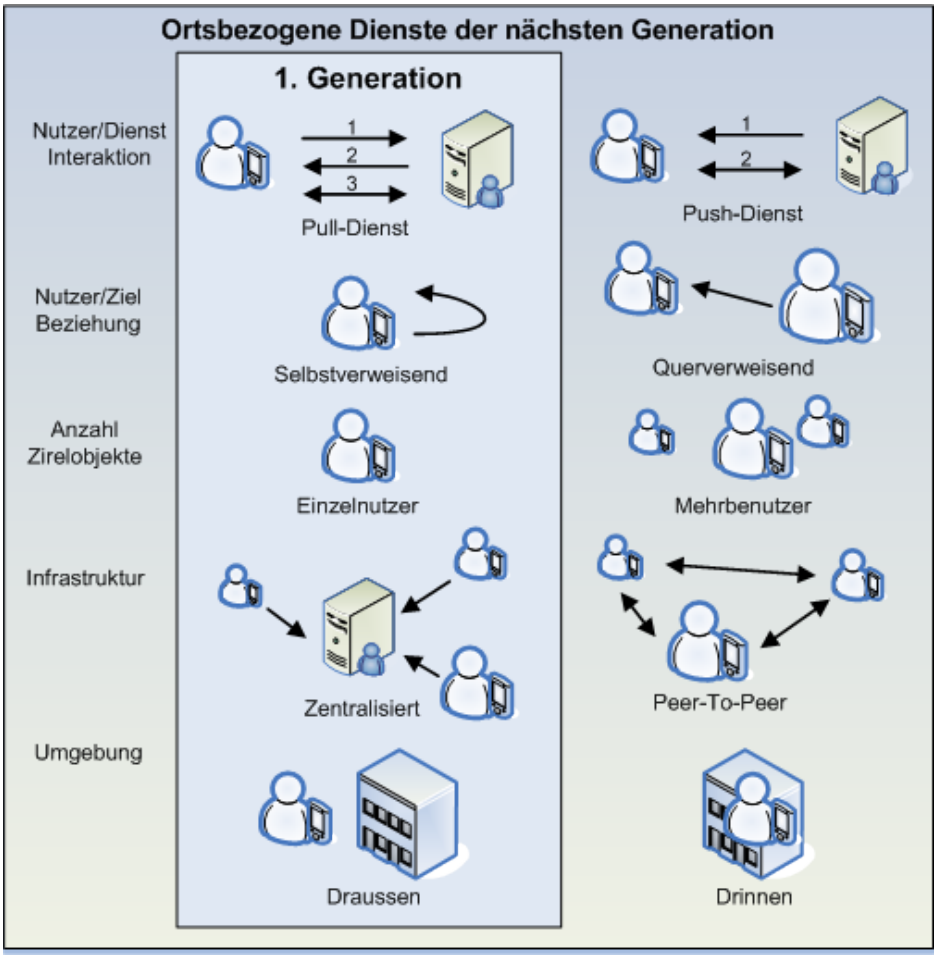

Abbildung 3.3: Eigenschaften ortsbezogener Dienste der zweiten Generation (Martens u. a. (2006))

geografische Daten, sowie die rasante Verbreitung lokaler, drahtloser Netze könnten die Mobilfunknetzbetreiber jedoch überflüssig machen. Sie werden lediglich für die Datenübertragung benötigt, wenn eine Verbindung ausschließlich via GSM oder UMTS zum Internet aufgebaut werden kann, oder wenn der Dienst nicht kostenlos und eine Bezahlung über den Mobilfunknetzbetreiber vorgesehen ist.

3.2.1.1 Konsequenzen

Ortsbezogene Dienste der zweiten Generation, neue Technologien und die Eigenschaften des Web 2.0 reduzieren die beteiligten Akteure auf den Trailblazers-Dienst und die Nutzer. Der Dienst stellt die gewünschte Funktionalität bereit, die Nutzer füllen den Dienst mit Inhal-

ten und die Nutzung des Dienstes kann kostenlos sein. Zusätzlich können dritte Dienste in das System integriert werden. Dies ist jedoch nicht unbedingt notwendig und erfolgt gemäß dem Mash-Up Paradigma des Web 2.0 relativ einfach. So müssen z. B. für die Nutzung eines Kartendienstes wie Microsoft Virtual Earth keine gesonderten vertraglichen Vereinbarungen getroffen, sondern lediglich Nutzungsbestimmungen[1] beachtet werden.

3.2.2 Dienste

Wie in Abschn. 2.5.1 dargelegt, wird Software im Web 2.0 als Dienst betrachtet, den die Nutzer in Anspruch nehmen, um Inhalte zu generieren und um daraus Wissen zu erlangen. Viele der oben skizzierten Dienste im Web 2.0 sind Web-basiert. Im Folgenden sollen Dienste aufgezeigt werden, die durch die Dimension der Mobilität ihren Nutzen auf mobilen Geräten voll entfalten können.

3.2.2.1 Mobile, soziale Netzwerke

Mobile, soziale Netzwerke bringen die in Abschn. 2.2.2 vorgestellten ortsbezogenen Gemeinschaftsdienste und die in Abschn. 2.5.2 sozialen Netzwerke des Web 2.0 zusammen.

Das Unternehmen Plazes[2] bietet mit ihrer Web-Anwendung gleiches wie die bisher skizzierten sozialen Netzwerke, erweitert diese aber um die Dimension der Position. Der Dienst bietet die Möglichkeit, andere Nutzer im Umkreis der eigenen Position ausfindig zu machen. Die Positionsbestimmung eines Nutzers wird dabei mittels geografischer Adressierung mit Domain Name Servern (DNS) (vgl. Roth (2005)) durchgeführt und unterscheidet sich damit signifikant von den in Abschn. 2.3.3 vorgestellten Systemen zur Positionsbestimmung. Darüber hinaus bietet Plazes diesen Dienst auch für Mobilfunktelefone an. Die Bestimmung der Position wird dabei mithilfe des GSM-Netzwerkes durchgeführt. Die eigene Position wird via Kurzmitteilung (SMS) an einen zentralen Server übertragen und von dort an definierte Personen, z. B. Freunde, übermittelt. So können Nutzer sehen, wo sich ihre Freunde aktuell aufhalten und sich mit ihnen kurzfristig treffen.

Ähnliche Dienste sind z. B. dodgeball[3] und sociallight[4].

[1] Nutzungsbestimmungen für Virtual Earth: http://local.live.com/Help/en-us/CodeOfConduct.htm; Zugriffsdatum: 29.04.2007
[2] http://beta.plazes.com/; Zugriffsdatum: 19.02.2007
[3] http://www.dodgeball.com/, Zugriffsdatum: 04.04.2007
[4] http://socialight.com/, Zugriffsdatum: 04.04.2007

3.2.2.2 Geografische, kollaborative Dienste

Bei geografischen, kollaborativen Diensten schließen sich Nutzer zusammen, die eine bestimmte räumliche Beziehung zueinander haben. Sie tauschen Informationen aus, um ein gemeinsames Ziel zu erreichen. Derartige Dienste stellen eine konsequente Weiterentwicklung der mobilen, sozialen Netzwerke dar. Die Nutzer gehören dabei einem, im Vergleich zu sozialen Diensten, oft nicht so eng vermaschten, sozialen Netzwerk an.

In Ibach u. a. (2004) wird ein kooperatives Verfahren vorgestellt, das die in Abschn. 2.3.3.2 skizzierte tabellenbasierte Positionsbestimmung im WLAN verbessert. Bei diesem Verfahren werden zusätzlich Positionsdaten und Signalstärken zwischen den einzelnen Client-Anwendungen ausgetauscht, so dass sich die Trainingsphase stark verkürzt und die Genauigkeit der Positionsmessung vergrößert.

Nova u. a. (2006) haben ein ortsbezogenes Spiel entwickelt, das die virtuelle mit der realen Welt verbindet. Drei Spieler müssen in der virtuellen Welt ein Objekt finden und es von drei Seiten in der realen Welt einkreisen. Dazu müssen sie zusammenarbeiten. Jeder Spieler hat ein mobiles Gerät, auf dem die Positionen der anderen Spieler und eine grobe Entfernung zum Objekt angezeigt werden. Sie können Nachrichten untereinander austauschen, um das gemeinsame Vorgehen zu besprechen, sowie gemeinsame Karten mit Notizen anreichern. Mit der Arbeit konnte somit u. a. aufzeigt werden, wie Spieler von einer Zusammenarbeit profitieren, um ein gemeinsames Ziel zu erreichen.

Ein anderes ortsbezogenes Spiel ist Geo-Caching (vgl. Geocaching.de (2007)), eine Art rechnergestützte Schnitzeljagd. Dabei agieren mindestens zwei Nutzer gegeneinander. Ein Nutzer ist der Initiator und versteckt in der realen Welt einen Behälter, den so genannten Cache. In diesem befindet sich ein beliebiger Inhalt und ein Notizbuch, das so genannte Log. Die Koordinaten des Verstecks werden im Internet veröffentlicht. Ziel des Spiels ist es, dass der andere Nutzer dieses Versteck findet, den Inhalt des Behälters tauscht und einen Eintrag im Log vornimmt. Die hohe Verbreitung und Akzeptanz des Spiels zeigt, dass auch solche Spieler zur Zusammenarbeit motiviert sind, die sich persönlich nicht kennen und nur virtuell in Kontakt treten.

3.2.2.3 Konsequenzen

Die Traiblazers-Version Alpha bietet dem Nutzer Offboard-Navigation. Dies bringt Nachteile mit sich. Die in den Abschn. 2.1.2 und 2.2.3 geforderte Unterstützung eines Offline-Szenarios lässt sich mit einer Offboard-Navigation nicht ausreichend berücksichtigen. Die Routenfunktionalität soll für die Version Beta auf ein hybrides System umgestellt werden. Die Daten für Karten, Trails und Orte von Interesse werden vom Trailblazers-Server bereitgestellt und auf das mobile Gerät übertragen. Der Nutzer kann diese Daten dort speichern und somit

auch in einem Offline-Szenario nutzen. Die Routenberechnung wird auf dem mobilen Gerät durchgeführt, die Darstellung der Route soll der Berechnung entsprechend angepasst werden. Die Route kann in bestimmten Fällen, z. B. wenn der Nutzer die vorgegebene Route verlässt, neu berechnet und angezeigt werden. Zusätzlich sollen Piktogramme die Navigation unterstützen. Die grundsätzliche Möglichkeit einer Sprachein- und ausgabe ist im Entwurf der Software zu berücksichtigen.

Die bisher skizzierten Anwendungen des Web 2.0 machen deutlich, dass es für Nutzer von Vorteil ist, wenn sie miteinander in Kontakt treten können. Interessant wird dieser Ansatz vor allem durch die bei mobilen Diensten hinzugekommene Dimension der Mobilität und der sich daraus ergebenen Gelegenheit, die virtuelle und die reale Welt zu verbinden. In Trailblazers-Beta sollen Nutzer sich gegenseitig Textnachrichten in Echtzeit zukommen lassen und die Position anderer Nutzer verfolgen können. Diese Funktionalitäten steigern den Mehrwert der Anwendung für den Nutzer besonders dann, wenn er sie innerhalb seines eigenen sozialen Netzwerkes anwendet. Darüber hinaus ergeben sich auch weitere Anwendungsmöglichkeiten, wie sie bei Tracking-Diensten üblich sind und z. B. in Schiller und Voisard (2004) für die Friend-Finder-Software beschrieben werden.

Die Genauigkeit und Qualität der kooperativen Positionsbestimmung im WLAN zeigt eine Korrelation mit der Anzahl der teilnehmenden Nutzer. Ortsbezogene Spiele zeigen ferner, dass Nutzer in der Lage sind, gemeinsam zu agieren, um ein gemeinsames Ziel zu erreichen. Besonders hervorzuheben ist dabei, dass das Erreichen des Ziels durch einen einzelnen Nutzer praktisch nicht möglich ist. Die Nutzer der Trailblazers-Software profitieren sowohl von der Quantität als auch der Qualität der Daten. Je mehr das System genutzt wird, desto höher ist die Qualität der Daten. Den Teilnehmern muss entsprechend deutlich gemacht werden, dass sie als Teil einer Gemeinschaft, agieren, und dass sie Daten bereitstellen müssen, um die Dienste im Gegenzug auch effektiv nutzen zu können. Wichtig ist, dass alle Nutzer immer ein gemeinsame Ziel vor Augen haben. Bei der Trailblazers-Software ist dieses Ziel die barrierefreie Navigation. Jedem Nutzer muss jederzeit klar sein, wie dieses Ziel zu erreichen ist.

Um die Aktivitäten eines einzelnen Nutzers quantifizieren und qualifizieren zu können, sollen Maßnahmen ergriffen werden, die bei Diensten des Web 2.0 vermehrt zum Einsatz kommen[5]. Für die Quantifizierung soll im System ein Aktivitätsindex pro Nutzer mithilfe eines Punktesystems festgehalten werden. So soll jedem Nutzer sofort ersichtlich werden, wie engagiert jeder einzelne innerhalb der Gemeinschaft ist. Jeder Nutzer erhält Punkte für Trails und Orte von Interesse, die von ihm an den Server übertragen werden. Zur Qualifizierung sollen alle Nutzer die Möglichkeit haben, die Orte von Interesse zu bewerten. Im Gegenzug können die Nutzer Vorteile, z. B. in Form von Sachprämien, aus den erarbeiteten Punkten

[5] So plant z. B. die Videoseite YouTube Produzenten origineller Videos mit Geld zu belohnen. Vgl. Spiegel Online (http://www.spiegel.de/netzwelt/web/0,1518,462883,00.html, Zugriffsdatum: 11.02.2007)

erhalten. Dies soll auch dem Missbrauch durch einzelne Nutzer entgegenwirken, wie er z. B. bei Wikipedia vereinzelt auftritt[6]. Die Nutzer müssen in der Lage sein, die Daten in dem System nach definierten Regeln[7] korrekt zu halten.

Ob diese Maßnahmen die erwünschte Wirkung zeigen, lässt sich an dieser Stelle nicht sagen. Dennoch wird dem einzelnen Nutzer zugetraut, als Teil einer Gemeinschaft etwas zu bewirken. So wurde der gemeine Internet-Nutzer in Grossman (2006) zur Person des Jahres 2006 ernannt.

3.2.3 Daten

In Abschn. 2.5.1 wird festgestellt, dass Daten im Web 2.0 ein besonders hoher Stellenwert zukommt. Es scheint daher vorteilhaft, Daten von einer Menge von Nutzern zu sammeln, insbesondere dann, wenn das Beschaffen dieser Daten teuer und aufwendig ist.

Nach Schiller und Voisard (2004) ist die Beschaffung geografischer Daten aufwändig und teuer. Die Autoren geben auch einen Überblick über die Anbieter geografischer Daten. Die Firmen fungieren als so genannte Datenbroker und verkaufen geografische Daten, damit diese in Dienste jeglicher Art integriert werden können. Stellvertretend sei die Firma TeleAtlas genannt (TeleAtlas (2007b)).

Diese Firma kartografiert u. a. Verkehrswege innerhalb Europas. Dazu wird das gesamte Straßennetz mithilfe von DGPS kartografiert und die Straßen fotografiert. Es werden mehr als 30 Fahrzeuge entsprechend ausgestattet und mit zwei Mitarbeitern besetzt. Neben den offensichtlich hohen Kosten und der verbrauchten Zeit ergeben sich durch diese Methode weitere Nachteile. Insbesondere müssen die Daten in regelmäßigen Abständen aktualisiert werden. Daher scheint es sinnvoll zu sein, derartige Daten von vielen, in der realen Welt verteilten Nutzern zu sammeln. Dies ermöglicht, dass Daten in kürzeren Abständen aktualisiert und bereitgestellt werden können.

3.2.3.1 Geo-Tracks

In Hariharan u. a. (2005) wird eine mobile Client-Server-Anwendung vorgestellt, mit der bestehende Karten mit den zurückgelegten Wegen des Nutzers annotiert werden können. Dazu werden die mittels GPS-System bestimmte, geografische Breite und Länge, in der mobilen

[6] Vgl. Spiegel Online (http://www.spiegel.de/netzwelt/web/0,1518,388801,00.html, Zugriffsdatum: 11.02.2007)

[7] Beispielsweise haben die Betreiber des sozialen Netzwerkes StudiVZ mit ihren Nutzern zusammen einen Verhaltenskodex erstellt, an den sich die Nutzer halten müssen.

Anwendung aufgezeichnet. Zu einem späteren Zeitpunkt werden die Daten dann an die stationäre Anwendung geladen und die Karten entsprechend annotiert. Der Nutzer bekommt so Statistiken zu von ihm zurückgelegten Wegen.

Das Projekt openstreetmap.org (2006) hat das Ziel, auf diese Weise vollständig erneuertes Kartenmaterial der Erde zu erzeugen, welches ohne Urheberrechtsbedingungen frei verfügbar sein soll. Um dieses Ziel zu erreichen, werden Nutzer aufgefordert, Daten zu sammeln und an das System zu übertragen. Dabei wird das GPS Exchange Format (GPX) genutzt. Die daraus erzeugten Kartendaten können als Web-Service abgerufen und somit in jegliche Anwendungen integriert werden.

In der Arbeit von Davies u. a. (2006) wird darüber hinaus ein Algorithmus zur verteilten Erstellung von Straßenkarten in Echtzeit vorgestellt. Die Grundidee des Algorithmus ist die gleiche, die der Trail-Erzeugung der Trailblazers-Software zugrunde liegt. Die Nutzer sammeln während des alltäglichen Gebrauchs GPS-Koordinaten ihrer zurückgelegten Positionen. Das Straßennetz wird als gerichteter Graf mit Metadaten beschrieben. Der Graf wird aus dem Histogramm der GPS-Koordinaten mehrerer Nutzer bzw. Fahrzeuge in einer definierten Umgebung erzeugt. Dadurch, dass die Koordinaten mehrerer Fahrzeuge zur Straßenkartenerzeugung herangezogen werden, werden Fehler durch ungenaue GPS-Koordinaten, z. B. aufgrund von Reflexion, reduziert, sowie Kreuzungen im Straßennetz sichtbar. In der Arbeit werden verschiedene, zentrale und dezentrale, Systemarchitekturen für den Algorithmus besprochen. Beim zentralen Ansatz sammeln Nutzer die GPS-Koordinaten und übertragen diese an einen Server, wo dann dann die Kartenerzeugung durchgeführt wird. Beim dezentralen Ansatz schließen sich die Nutzer zu einem Ad-hoc-Netzwerk zusammen, um GPS-Koordinaten und erzeugte Straßenkarten zu teilen.

3.2.3.2 Geo-Tags

In Abschn. 2.5.2 wurde erläutert, dass mithilfe von Tags Inhalte von Nutzern angereichert werden, um diese zu kategorisieren. Tags können auch aus geografischen Koordinaten bestehen. So genannte Geo-Tags werden genutzt, um Orte von Interesse zu annotieren. Viele Web-Seiten nutzen diese Mash-Up-Technik, um ihre eigenen Inhalte mit geografischen Daten anzureichern. Z.B. bietet der Internet-Fotodienst flickr[8] die Möglichkeit, Bilder entsprechend zu erweitern und mit anderen Nutzern zu teilen. Moderne digitale Fotokameras, wie z. B. das Modell T50 der Firma Sony, können mit einer GPS-Empfangseinheit ausgerüstet werden. Diese speichern automatisch zu jedem Foto die aktuellen geografischen Koordinaten des Fotografen. Die angereicherten Bilder können vom Nutzer auf entsprechende Web-Seiten eingestellt werden.

[8] http://www.flickr.com, Zugriffsdatum: 02.05.2007

Mobile Geräte mit integrierter Digitalkamera sind heutzutage Standard und bereits stark verbreitet. Nach Gershman und Fano (2005) werden im Jahr 2007 300 Millionen Mobilfunktelefone mit integrierter Kamera verkauft werden. Dies begünstigt die Annotierung von Kartendaten mit Fotos und zugehörigen Positionsinformationen enorm. Des weiteren wird beschrieben, dass jegliche visuelle Daten, wie z. B. Fotos, immer weiter an Bedeutung gewinnen. Aktuell werden diese Daten hauptsächlich in einem engen sozialen Kontext veröffentlicht. Mittlerweile etablieren sich jedoch auch Dienste, die über diesen Kontext hinausgehen. Yahoo - You Witness[9] oder Leser-Reporter von Bild.T-Online [10] stellen aktuelle Fotos und Kurznachrichten von Nutzern bereit. Diese werden vor der Veröffentlichung redaktionell geprüft und die Fotografen mit einem Honorar belohnt. Darüber hinaus unterscheiden sich diese Dienste von dem oben genannten Fotodienst insbesondere dadurch, dass Wert auf aktuelle Informationen mit Nachrichtencharakter gelegt wird.

Interessant für ortsbezogene Dienste ist diese Entwicklung dadurch, dass diese Nachrichten vor allem einen lokalen Ortsbezug zum Nutzer haben. Ziel der Betreiber ist es offensichtlich, neben globalen, allgemein interessanten Nachrichten, verstärkt lokale Neuigkeiten zu verbreiten, die entsprechend den lokalen Nutzern zugänglich gemacht werden.

3.2.3.3 Konsequenzen

Bisher existierten keine Daten über barrierefreie Wege, und Daten über Orte von Interesse wurden bislang von wenigen Personen beschafft. In der Stadt Hamburg werden derartige Informationen über Barrieren z. B. von der Landesarbeitsgemeinschaft für behinderte Menschen e.V. (vgl. LAG (2007)) gesammelt und jährlich veröffentlicht. Dies ist jedoch sehr aufwendig, und die Daten müssen in regelmäßigen Abständen neu erfasst werden, da Objekte in der realen Welt einer ständigen Veränderung unterliegen.

Das Sammeln geografischer Daten wie Geo-Trails und Geo-Tags wurde in der Trailblazers-Software Alpha bereits umgesetzt. Somit bietet der Dienst Daten an, die für Nutzer von großem Wert sind, und die es ohne die Beteiligung der Nutzer nicht geben kann. Die Daten bilden somit ein Alleinstellungsmerkmal für den Trailblazers-Dienst. Zudem werden sie permanent aktualisiert. Neue Wege werden als Geo-Trails erkannt, nicht mehr genutzte Wege werden gelöscht. Orte von Interesse können sofort oder sehr zeitnah von einzelnen Nutzern bereitgestellt werden.

Das Sammeln zeitnaher Daten durch einzelne Nutzer, vor allem das Erstellen von Fotos von Orten von Interesse, scheint nicht unproblematisch zu sein. Es besteht die Gefahr, dass Motive gewählt werden, die die Nachrichten im so genannten Sensationsjournalismus münden

[9] http://news.yahoo.com/you-witness, Zugriffsdatum: 02.05.2007
[10] http://www.bild.t-online.de/BTO/news/leser-reporter/startseite/leser-reporter.html, Zugriffsdatum: 02.05.2007

lassen. Die Nutzer würden dann zu offensiv agieren und die Privatsphäre von Mitmenschen verletzen[11]. Diese Entwicklung zeigt aber deutlich, dass durch die Verbreitung neuer Technologien und durch das Verständnis, diese zu Nutzen, neue Arten von Diensten entstehen. Daraus ergibt sich insgesamt die Forderung, dass Fotos von Orten von Interesse, durch die Nutzer kontrolliert werden. Dazu bietet sich der Mechanismus an, dass jeder Nutzer Fotos, wie oben erwähnt, zum einen bewerten, aber auch schädliche Fotos an eine Überwachungsinstanz melden kann. Diese kann dann Fotos aus dem System entfernen und die Urheber der Fotos verwarnen.

3.2.4 Geschäftsmodelle

Obwohl in älteren Studien (vgl. Fritsch und Muntermann (2005)) dem mobilen Handel (engl.: mobile commerce, M-Commerce) und ortsbezogenen Diensten als Katalysator hohe Umsatzerwartungen bescheinigt wurden, haben sich ortsbezogene Dienste der ersten Generation bisher nicht durchgesetzt. Durch die oben genannten Rahmenbedingungen kann sich das aktuell jedoch ändern. Dies wird auch durch die mittlerweile hohe Verbreitung mobiler Geräte unterstrichen. So sollen laut einer aktuellen Marktforschung (Gartner (2006c)) im Jahr 2007 über 900 Millionen Mobilfunktelefone, davon 120 Millionen Smartphones, sowie 18,8 Millionen Internet-fähige PDAs verkauft werden. In einer weiteren aktuellen Studie (Bhattacharyya und Scott Ellison (2006) wird ausgesagt, im Jahr 2010 werde jeder zweite Nutzer eines mobilen Gerätes im Endverbrauchersektor der USA ortsbezogene Dienste in Anspruch nehmen. Dadurch sollen ortsbezogene Dienste zu einem Markt heranwachsen, in dem Umsätze von mehreren Milliarden Dollar pro Jahr generiert werden.

3.2.4.1 Hinderungsgründe

In einer deutschen Studie (Fritsch und Muntermann (2005)) wurden potentielle Gründe für das bisherige Ausbleiben der Umsatzerwartungen für ortsbezogene Dienste genauer untersucht. Dabei wurden drei wesentliche Feststellungen gemacht. Erstens orientiere sich das Dienstangebot nicht an den Kundenwünschen; lediglich Dienste für Navigation und Notfalldienste wurden von Kunden nachgefragt. Zweitens seien die angebotenen Dienste aus Sicht der Nutzer zu teuer. Die Nutzer sind in der Regel nur bereit Kosten zu tragen, die auf dem Preisniveau einer SMS liegen. Erschwerend komme hinzu, dass einige Anbieter ihre Dienste mit einem unübersichtliches Preismodell aus Datenvolumenpreis, Transaktionspreis und

[11] Vgl. Spiegel Online (http://www.spiegel.de/netzwelt/web/0,1518,452442,00.html und http://www.spiegel.de/kultur/gesellschaft/0,1518,442492,00.html, Zugriffsdatum: 25.02.2007)

Grundgebühr anbieten, wodurch mangelnde Transparenz der tatsächlich anfallenden Kosten entstehe. Drittens wurden Datenschutzbedenken seitens der Nutzer genannt. Diesen sei dabei besonders wichtig, die Kontrolle über die Verarbeitung und Speicherung standortspezifischer, persönlicher Daten zu haben. Dies gilt vor allem bei Tracking-Diensten, die im Hintergrund ausgeführt werden. Insbesondere wird gefordert, dass die Nutzer vor jeder Ortung ihrer Position explizit ihre Zustimmung erteilen müssen. Die Privatsphäre der Nutzer wird im folgenden Unterabschnitt ausführlich diskutiert.

In Rannenberg u. a. (2005) wird vor allem den hohen Tarifpreisen für den mobilen Internet-Zugang eine hemmende Wirkung zugesprochen. Die Kunden müssen nicht nur die Dienste und Inhalte, sondern auch grundsätzlich Kosten für die mobilen Datenübertragungen zahlen. Diese sind höher als bei stationären Internet-Zugängen, wodurch die Nutzungsintensität erheblich gemindert wird.

In der Arbeit von Bazijanec und Turowski (2004) wurde zudem erläutert, dass die Zahlungsbereitschaft privater Nutzer vor allem für Dienste im Unterhaltungsbereich gegeben ist, was sich u. a. an dem hohen Umsatz für verkaufte Klingeltöne und Bilder für Mobilfunktelefone festmachen lässt.

3.2.4.2 Neue Geschäftsmodelle

Im Projekt von Skiera (2006) werden daher neue Geschäftsmodelle entwickelt, die es den Mobilfunknetzanbietern ermöglichen sollen, mobile Datenkommunikation für die Nutzer zu subventionieren. Als Lösungsansatz wird die personalisierte, ortsbezogene und im weiteren Sinne kontextbezogene Werbung als Erlösquelle eingeführt. Wie in den Abschn. 2.1.1 und 2.2.2 erläutert, ist dies ein besonderes Merkmal ortsbezogener, mobiler Informationssysteme und somit auch ein gewinnbringendes Alleinstellungsmerkmal im mobilen Handel. Die Kosten für die Nutzung des mobilen Internets sollen für den Kunden dadurch insgesamt niedriger werden und die Nutzungsintensität soll steigen. Dieses Geschäftsmodell scheint jedoch nicht auf Mobilfunknetzanbieter beschränkt zu sein, zumal diese in ortsbezogenen Diensten der zweiten Generation nicht notwendigerweise vertreten sein müssen. Vielmehr kann der Dienstanbieter ebenfalls ortsbezogene Werbung schalten und somit Einnahmen durch das Anbieten seines Dienstes generieren.

3.2.4.3 Mobiles Marketing

In Möhlenbruch und Schmieder (2001) wird mobiles Marketing untersucht, dessen Potential im sich ändernden Konsumverhalten der Kunden gesehen wird. Durch steigenden Wettbewerb und vorhandene Markttransparenz steigen die Ansprüche des Kunden an Produkte und Dienstleistungen. Der Kunde erwartet auf seine Bedürfnisse zugeschnittene Konsum- und

Interaktionsmöglichkeiten. Mobile Internet-Anwendungen können besonders dann Wertbeiträge liefern, wenn sich innovative Anwendungen deutlich von Lösungen des klassischen Internets unterscheiden, sowie eine Verbindung zwischen realer und virtueller Welt und einen konkreten Nutzen für den Konsumenten darstellen. Die unmittelbare Identifikation eines Kunden, sowie seines Standortes und seiner spezifischen Einkaufssituation, ermöglicht eine regionale, zeitliche, inhaltliche und personenspezifische Differenzierung eines Werbeangebotes, wie z. B. die gezielte Bewerbung eines Sonderangebotes, wenn der Kunde sich in der Nähe eines Geschäftes aufhält. Bei diesem so genannten One-to-One-Marketing ist zu beachten, dass die Kunden ihr Einverständnis für die persönliche Ansprache erteilen müssen. Dies wird als so genannte Option-In-Regelung bezeichnet. Mobiles Marketing sollte zudem nur Teil eines so genannten Multichannel-Marketing sein, welches andere Kanäle, wie traditionelle Medien und stationäres Internet, unterstützt.

3.2.4.4 Bezahlverfahren

In Herden u. a. (2006) wird eine prototypische Plattform für verschiedene ortsbezogene Dienste vorgestellt. Diese beinhaltet eine einheitliche Nutzerschnittstelle in Form einer Client-Anwendung, sowie eine Server-Anwendung, die verschiedene verfügbare Internet-Dienste integriert. In einem möglichen Geschäftsmodell soll eine gerechte Zahlung des Nutzers pro Dienstnutzung erfolgen; dies erfordert jedoch Billing- und Micropayment Bezahlverfahren (vgl. auch Klein (2005)). Daher wird eine pauschale Abrechnung in Form eines Abonnements für den Nutzer angestrebt. Des weiteren wird die Art der Vergütung der integrierten Dienste, die von der Server-Anwendung genutzt werden, skizziert. Dies kann jedoch nicht pauschal beantwortet werden, da die Art der Vergütung davon abhängig ist, ob der integrierte Dienst einen Vorteil durch die Integration, z. B. Erweiterung des Kundenkreises, erhält. Werbepartner hingegen können so in die Plattform integriert werden, dass sie zusätzliche Einnahmen erzielen. Problematisch ist es allerdings bei Werbepartnern der integrierten Dienste, da die Werbung nicht an die Client-Anwendung weitergereicht wird. In einem solchen Fall können Ausgleichzahlungen zu leisten sein.

3.2.4.5 Konsequenzen

Dienste im Web 2.0 erfordern zum Teil neue Geschäftsmodelle. Bei vielen aktuellen Diensten ist nicht ersichtlich, wie Umsatz oder gar Gewinn generiert werden soll.

Wie oben erläutert, sind vor allem Navigations- und Notfalldienste sehr nachgefragt und Navigationsanwendungen stark verbreitet. So besitzen aktuell bereits drei Millionen Personen in Deutschland ein Navigationsgerät (vgl. BITKOM (2007a)). Es wird deutlich, dass die Trailblazers-Software dadurch Marktpotential besitzt. Die Tatsache, dass die Daten im

Trailblazers-System ein Alleinstellungsmerkmal darstellen, grenzt diesen Dienst von anderen ab. Obwohl Nutzer der genannten Studie bereit sind, für einen solchen Dienst zu zahlen, soll der Navigationsdienst kostenlos angeboten werden, da das zugrunde liegende Geschäftsmodell dem Paradigma des Web 2.0 entsprechen soll. Der kostenlose Dienst kann eine möglichst breite Masse an Nutzern erreichen. Der Erfolg soll dadurch gesichert werden, da eine zirkuläre Abhängigkeit besteht: Denn je mehr Nutzer in das System integriert sind, desto qualitativ hochwertiger ist der Datenbestand, und desto mehr Nutzer sind am System interessiert. Demnach ist der initiale Start des Systems problematisch. Der Nutzer muss mit entsprechenden Marketing Maßnahmen auf den eigentlichen Dienst, sowie auf die Möglichkeit, zu gegenseitiger Hilfe, aufmerksam gemacht werden. Zusätzlich muss, wie oben erwähnt, jede aktive Mitgestaltung eines Nutzers belohnt werden.

Der Tracking- und der Echtzeit-Kommunikations-Dienst könnten dagegen als bezahlte Premium-Dienste angeboten werden, um Umsätze zu erzielen. Dies bringt jedoch die oben genannten Probleme des Payment mit sich. Diese Möglichkeit soll in dieser Arbeit daher nicht weiter verfolgt werden.

Neben der Möglichkeit, das Angebot durch neue Dienste zu erweitern, können auch weitere Zielgruppen angesprochen werden, um eine kritische Masse von Nutzern zu erreichen. Die Daten der einzelnen Gruppen sollen dabei getrennt voneinander behandelt werden, bzw. in einer möglichen Hierarchie angeordnet sein.

Insgesamt wird deutlich, dass profil- und kontextbezogene Werbung die vielversprechendste Möglichkeit für einen Web 2.0-Dienst ist, um Umsätze zu generieren. Dies wird zudem durch den weiter wachsenden Markt im Internet begünstigt. Der Online-Werbemarkt ist in Deutschland im 1. Quartal 2007 im Vergleich zum Vorjahr um 45 Prozent auf etwa 174 Millionen Euro (BITKOM (2007b)) gestiegen. Für den Nutzer darf die Werbung dabei nicht störend wirken. So darf eingeblendete Werbung nicht die volle Aufmerksamkeit des Nutzers erfordern, indem diese z. B. automatisch geöffnet und von ihm explizit geschlossen werden muss. Werbebotschaften sollen vielmehr permanent in einem kleinen Ausschnitt des Bildschirms eingeblendet werden und immer als Werbung identifizierbar sein. Aus Sicht des Werbetreibenden muss die Bereitstellung von Werbung in den Trailblazers-Dienst einfach möglich sein. Diese beiden Eigenschaften scheinen u. a. zum Erfolg des Werbemodells der Firma Google und ihrer Produkte AdWords und AdSense beigetragen zu haben.

Weiterhin könnten genaue Auswertungen über den Konsum der Werbung an den Werbetreibenden verkauft werden, so dass dieser Erkenntnisse bekommt, die er in seiner Marketingstrategie berücksichtigen kann. Dies wird beim Konzept der Kundenkarten bereits durchgeführt (vgl. Kaapke (2002)). Grundsätzlich entstehen durch diese Art der Werbung und der Auswertung datenschutzrechtliche Forderungen, die im nächsten Abschnitt weiter diskutiert werden.

3.2.5 Privatsphäre

Wie bereits erläutert, stellt die Privatsphäre eines der größten Probleme ortsbezogener Dienste dar. Bei Anwendungen im Web 2.0 scheint diese dagegen kaum eine Bedeutung zu haben. So geben z. B. viele Nutzer detaillierte persönliche Informationen in Blogs preis oder veröffentlichen Videos über die eigene Person. In der virtuellen Welt werden dabei häufig auch virtuelle Identitäten genutzt. Ist dies nicht der Fall, kann es zum Nachteil werden, wenn die Informationen von Leuten wahrgenommen werden, die in der realen Welt in einer Beziehung zum Verfasser stehen. Als Beispiel sei ein potentieller Arbeitgeber genannt, der den Blog eines Bewerbers liest, um Informationen über den Kandidaten zu erhalten. Ausschließlich virtuelle Identitäten stellen auch dann keinen probarten Lösungsansatz da, wenn fremde Personen sich einander, z. B. mittels ortsbezogener Tracking-Dienste, in der realen Welt finden können.

3.2.5.1 Nutzer und ihre Privatsphäre

In einer US-amerikanischen Studie (Burak und Sharon (2004)) wurden Untersuchungen zum Nutzungsverhalten ortsbezogener Dienste angestellt. Dabei wurde festgestellt, dass junge Erwachsene ihre eigene Position preisgeben, um miteinander in Kontakt zu treten. Die Einschränkung der Privatsphäre wird dabei bewusst in Kauf genommen. Da die Studie an einer geringen Anzahl von Probanden durchgeführt wurde, kann diese nicht als repräsentativ angesehen werden. Zudem zeigt die Studie ein anderes Ergebnis als die oben genannte Studie von Frietsch, was in einem divergenten Verständnis von Datenschutz in den USA und Deutschland begründet werden kann. Letztlich zeigt sich jedoch, dass es grundsätzlich möglich ist, Nutzer dazu zu bewegen, ihre Position anderen Nutzern mitzuteilen, insbesondere dann, wenn dadurch ein eigener Vorteil entsteht.

3.2.5.2 Maßnahmen zur Sicherung der Privatsphäre

Folgende Maßnahmen können zur Sicherung der Privatsphäre ergriffen werden:

Sichere Kommunikation Als Grundvoraussetzung für die Wahrung der Privatsphäre muss eine sichere Kommunikation zwischen mobiler Client- und stationärer Server-Anwendung bestehen. Insbesondere die Authentifizierung, die Integrität und die Vertraulichkeit müssen gewährleistet sein (vgl. Eckert (2004) und Küpper (2005)).

Zugriffsregeln Der Zugriff von Dritten auf die Positionsdaten eines Nutzers kann nach definierten Regeln gewährt bzw. verweigert werden (Myles u. a. (2003)). In der einfachsten Regel wird der Nutzer immer explizit um sein Einverständnis gebeten. Wie in ersterer

Studie gefordert, bestimmt der Nutzer somit selber, ob und wann er seine Positionsinformationen freigibt. Andererseits würde eine angestrebte Automatisierung den Nutzer entlasten. Entsprechende Regeln können z. B. aus zeitlicher und örtlicher Beschränkung und aus genau definierten Akteuren und Dienstarten zusammengesetzt werden. So wird bei vielen Diensten im Internet die Bedingung der genau definierten Akteure genutzt. Diese Dienste erlauben die Weitergabe von Positionsinformationen nur an Personen, die der Nutzer in seinem sozialen Netzwerk, z. B. in seiner Freundesliste, hinzugefügt hat.

Anonymisierung Eine weitere wichtige Maßnahme zum Schutz der Privatsphäre ist die Anonymisierung der Nutzer oder des Inhaltes. Ersteres wird unter anderem von Beresford und Stajano (2004) im Mix Zones-Konzept vorgestellt. Dabei werden Orte in *Application Zones* und *Mix Zones* eingeteilt. Ortsbezogene Dienste können von den Nutzern nur konsumiert werden, wenn sie sich in einer Application Zone aufhalten. Sobald ein Nutzer eine Mix Zone betritt, wird ihm eine neue Identität zugewiesen. Dadurch wird es erschwert, von einem bestimmten Nutzer Bewegungsprofile zu erstellen.

Durch die Anonymisierung entstehen jedoch zwei Zielkonflikte. Zum einen entsteht ein Zielkonflikt mit dem oben genannten Anwendungsfall der Trail-Erkennung. Auch erfordert der Trailblazers-Dienst die eindeutige Identifikation der Nutzer, obgleich diese nicht der realen Person des Nutzers entsprechen muss. Zum anderen entsteht ein Zielkonflikt mit dem oben genannten Geschäftsmodell. Dieses sieht profil- und kontextbezogene Werbung vor. Der Kunde möchte auf seine Bedürfnisse gezielt beworben werden, dabei jedoch seine Anonymität wahren. Das werbetreibende Unternehmen möchte seine Werbung möglichst ohne Streuverlust an potentielle Kunden platzieren und daher genaue Informationen über diese erhalten. Wichtig ist, ob die Profilierung des Nutzers ausschließlich auf ein direktes oder zusätzlich auf ein indirektes Nutzerprofil zurückzuführen ist (vgl. Abschn. 2.2.3), denn ersteres kann der Nutzer bewusst beeinflussen.

Die eindeutige Identifikation der realen Person eines Nutzers in ortsbezogenen Diensten sollte dennoch vermieden werden. Dementsprechend sollte die Identität eines Nutzers in ein Pseudonym überführt werden. In Bulander u. a. (2005) werden Identität, Pseudonymität und Anonymität voneinander abgegrenzt. Abb. 3.4 zeigt eine Übersicht mit steigendem Anonymisierungsgrad von links nach rechts. Bei einem Pseudonym werden Identifikationsmerkmale des Nutzers ersetzt, um seine Bestimmung zu erschweren (vgl. Bundesministerin der Justiz (1990)).

3.2.5.3 Konsequenzen

Es ist sehr wichtig, die Privatsphäre des Nutzers zu erhalten, um sein Vertrauen nicht zu enttäuschen und sein Nutzungsverhalten nicht negativ zu beeinflussen. Dafür gilt es, Nutzer

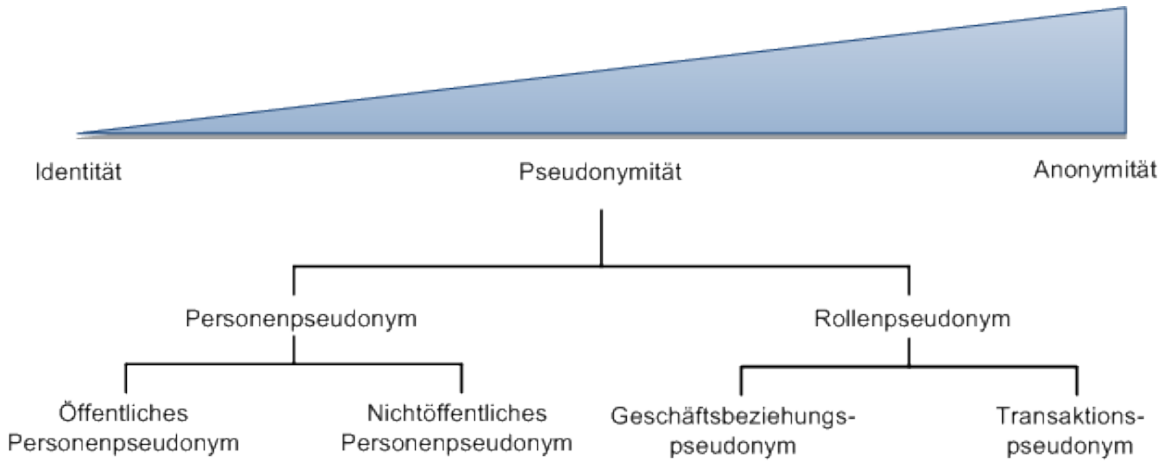

Abbildung 3.4: Kontinuum zwischen Identität und Anonymität (Bulander u. a. (2005))

durch Pseudonyme zu anonymisieren, sowie den Zugriff auf Positionsinformationen beim Tracking-Dienst durch Dritte bei Bedarf zu unterbinden.

Grundsätzlich müssen sich die Nutzer im System registrieren und bei jeder Anmeldung authentifizieren. Für die Registrierung wird eine gültige E-Mail-Adresse benötigt. Jeder Nutzer kann bei der Registrierung Nutzernamen und Passwort beliebig wählen; die jeweilige Authentifizierung erfolgt mit diesen Informationen. Des weiteren kann jeder Nutzer zusätzliche Informationen in seinem Profil hinterlegen. Weitere bekannte Konzepte für die Anonymisierung aus dem stationären Internet können nicht für alle Anwendungsfälle übertragen werden. So scheint die Anonymisierung für den Anwendungsfall barrierefreie Navigation ausreichend, für den echtzeit Tracking-Dienst dagegen nicht, da hierbei die Vermaschung der virtuellen mit der realen Welt ausgeprägter ist. Um für den Nutzer vorteilhafte ortsbezogene Dienste bereitstellen zu können, ist eine vollständige Anonymisierung zudem nicht zweckmäßig. Die Nutzerprofile sollen in der Trailblazers Beta-Software entsprechend in nichtöffentliche Personenenpseudonyme für den Tracking-Dienst, sowie in Geschäftsbeziehungspseudonyme für den Werbedienst und die Trail-Erkennung überführt werden.

Um die Weitergabe der Positionsinformationen an Dritte zu unterbinden, soll zusätzlich das Konzept der definierten Regeln integriert werden. Die Nutzer können in ihrem persönlichen Profil bestimmen, welche anderen Nutzer wann auf ihre Positionsdaten zugreifen können. Dabei soll der Bestätigungsprozess sowohl manuell, als auch für definierte Regeln automatisch erfolgen können.

Wie oben erwähnt, ist die Blockierung der Weitergabe der eigenen Positionsinformationen an den Trailblazers-Dienst ein Zielkonflikt, da dadurch nicht mehr das gemeinsame Ziel der barrierefreien Navigation erreicht werden kann. Dennoch scheint es sinnvoll zu sein, ein Grundkonzept der Mixed Zones in den Trailblazers-Dienst zu integrieren. Die Nutzer müssen

die Möglichkeit haben, Areale zu definieren, an denen keine Informationen über ihre Position an den Dienst übertragen werden können. Dadurch werden keine Bewegungsprofile an solchen Orten erstellt, an denen sich die Nutzer häufig aufhalten. Derartige Orte können z. B. die Umgebung der Arbeitsstätte oder des Wohnortes sein.

Da zudem die in Abschn. 2.3.3.1 vorgestellte GPS-Technik zur Positionsbestimmung genutzt wird, liegt die Position beim Nutzer und nicht im Netzwerk vor. Dadurch hat dieser immer die Kontrolle über die Positionsbestimmung und kann durch einfaches Abschalten des GPS-Empfängers bzw. durch Nutzung des Offline-Modus möglichen Missbrauch unterbinden.

Für alle personalisierten Dienste müssen die Nutzer vorab ihr Einverständnis erteilen und dieses jederzeit widerrufen können. Grundsätzlich müssen die Nutzer stets die Kontrolle über ihre persönlichen Informationen, inklusive aller Zeit- und Positionsinformationen, haben (vgl. BMWi (2006)); dies muss im Dienst berücksichtigt sein.

3.3 Funktionale Anforderungen

Ausgehend von den in den Grundlagen herausgearbeiteten Anforderungen an mobile Informationssysteme und ortsbezogene Dienste, sowie vom Anwendungsszenario und der Diskussion, ergeben sich funktionale Anforderungen an die Beta Version der Trailblazers-Software. Diese beschreiben die Funktionalität der Software (Kahlbrandt (2001)).

3.3.1 Anwendungsfälle

Die folgenden Anwendungsfälle sollen die funktionalen Anforderungen verdeutlichen. In dieser Arbeit werden nur diejenigen Anwendungsfälle betrachtet, die die Nutzer der mobilen Client-Anwendung betreffen. Insbesondere werden die Anwendungsfälle und Akteure der Web-basierten Trailblazers-Gemeinschaftsplattform hier nicht aufgeführt. Abb. 3.5 zeigt das zugehörige Anwendungsfalldiagramm. Zugehörige Aktivitätsdiagramme befinden sich im Anhang (A.1). An den Anwendungsfällen sind die beiden Akteure Gemeinschaftsnutzer und Nutzer der Version Alpha beteiligt.

3.3.1.1 Navigation

Dieser Anwendungsfall beschreibt die erweiterte barrierefreie Navigation von einem Start- zu einem Zielort.

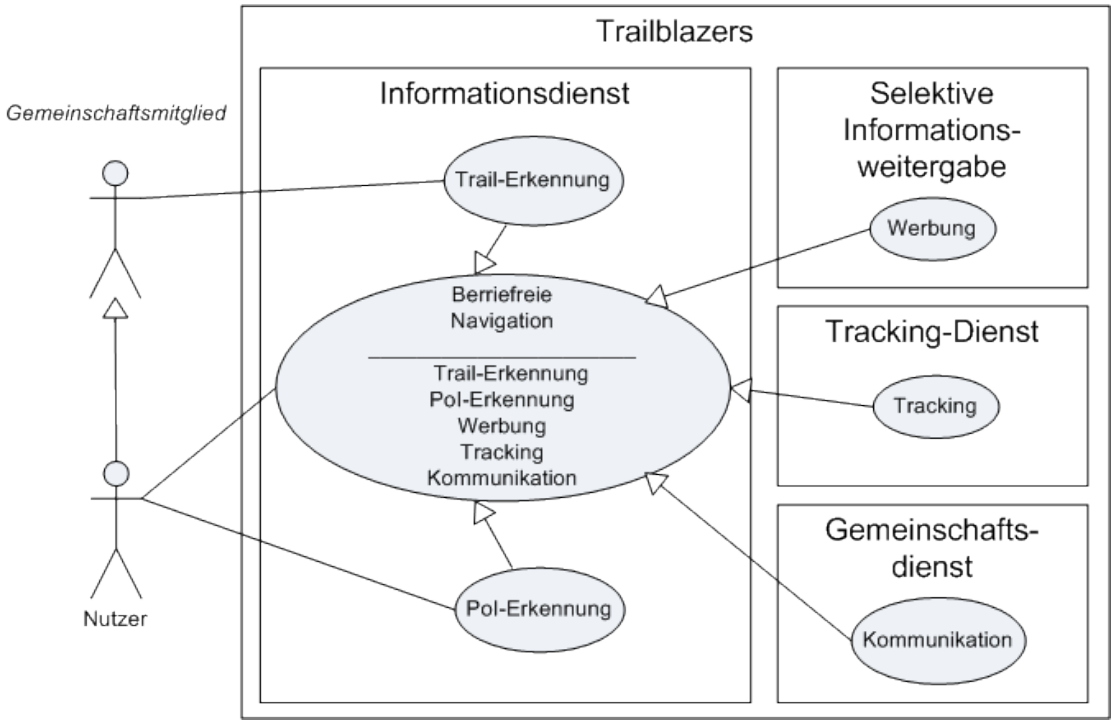

Abbildung 3.5: Anwendungsfalldiagramm Trailblazers Beta

Vor- und Nachbedingung Der Nutzer ist im System registriert. Am Ende des Anwendungsfalls wird die Route vom Start- zum Zielort dem Nutzer in der mobilen Anwendung angezeigt.

Beschreibung Der Nutzer stellt mit seinem mobilen Gerät eine Verbindung zum Internet her (A1). Der Akteur meldet sich durch Eingabe seines Nutzernamens und seines Passworts am System an und gibt dann die Start- und die Zieladresse ein (A2). Die Eingabe erfolgt via Text oder via Sprache. Die Adressen werden an die Server-Anwendung übertragen und in geografische Länge und Breite übersetzt. Die Kartendaten, die Trail-Daten, die Daten der Orte von Interesse, orts- und personenbezogene Werbung, sowie Informationen über private Areale für den Bereich vom Start- zum Zielpunkt werden geladen. Diese Daten werden zurück an die Client-Anwendung übertragen, anschließend wird in der Client-Anwendung die Routenberechnung durchgeführt und die Karte angezeigt. Der Nutzer startet die Bestimmung seiner eigenen Position, diese wird auf der Karte entsprechend dargestellt und er wird während der Fortbewegung entlang der Route zum Ziel geführt (A3). Dazu werden ihm eine Karte und Piktogramme angezeigt.

Sonderfälle

A1 Eine Verbindung zum Internet wird nicht hergestellt, die Client-Anwendung befindet sich im Offline-Modus. Die Anmeldung ist nur möglich, wenn Nutzerinformationen auf dem Gerät gespeichert sind. Die Navigation kann nur mit Daten durchgeführt werden, die auf dem Gerät gespeichert sind. Diese werden in einer Auswahlliste angezeigt. Der Nutzer wählt eine Karte aus, definiert Start- und Endpunkt durch direkte Auswahl auf der Karte und der Anwendungsfall wird fortgesetzt.

A2 Der Nutzer gibt keine Adresse ein, sondern lässt sich eine Karte der aktuellen Umgebung anzeigen. Da keine Start- und Endpunkte definiert sind, ist eine Routenberechnung dieser Karte nicht möglich.

A3 Wird die Route verlassen, wird eine Neuberechnung der Route durchgeführt. Der Anwendungsfall wird fortgesetzt.

3.3.1.2 Trail-Erkennung

Dieser Anwendungsfall beschreibt die erweiterte Trail-Erkennung während der barrierefreien Navigation.

Vor- und Nachbedingung Als Vorbedingung muss der Anwendungsfall barrierefreie Navigation vom Nutzer ausgeführt werden. Der Anwendungsfall hat keine Nachbedingung.

Beschreibung Dieser Anwendungsfall wird vom Anwendungsfall barrierefreie Navigation genutzt. Während der Ausführung werden die aktuellen Positionsdaten in geografischer Länge und Breite sowie Höhe über dem Meerespiegel in regelmäßigen Abständen aufgenommen und an die Server-Anwendung übertragen (A1). Sobald ein Nutzer während der Fortbewegung ein Areal erreicht, das in seinem Profil als privates Areal hinterlegt ist, wird die Aufzeichnung unterbunden, bis das Areal wieder verlassen wird. Der Anwendungsfall endet mit dem Anwendungsfall barrierefreie Fortbewegung.

Sonderfälle

A1 Eine Verbindung zum Internet wird nicht hergestellt, die Client-Anwendung befindet sich im Offline-Modus. Die Positionsdaten werden lokal auf dem mobilen Gerät gespeichert und bei der nächsten Synchronisation mit der Server-Anwendung an diese übertragen.

3.3.1.3 Werbung

Dieser Anwendungsfall beschreibt die Anzeige von Werbung für den Nutzer während der barrierefreien Navigation.

Vor- und Nachbedingung Als Vorbedingung muss der Anwendungsfall barrierefreie Navigation vom Nutzer ausgeführt werden. Der Anwendungsfall hat keine Nachbedingung.

Beschreibung Dieser Anwendungsfall wird vom Anwendungsfall barrierefreie Navigation genutzt. Sobald ein Nutzer während der Fortbewegung ein Areal erreicht, für welches Werbung hinterlegt ist, wird die Werbebotschaft in der Client-Anwendung angezeigt. Die Anzeige hebt sich somit von der generellen Anzeige von Werbung auf der Karte ab. Der Anwendungsfall endet mit dem Anwendungsfall barrierefreie Fortbewegung.

3.3.1.4 Tracking

Dieser Anwendungsfall beschreibt die Auffindung und Verfolgung der Position anderer Nutzer.

Vor- und Nachbedingung Der Anwendungsfall barrierefreie Navigation muss im Online-Modus ausgeführt werden. Am Ende des Anwendungsfalls wird die Position anderer Nutzer in der mobilen Anwendung angezeigt.

Beschreibung Ein Nutzer A öffnet eine Liste mit Namen anderer Nutzer aus seinem sozialen Netzwerk. Neben dem Namen wird der Status dieser Nutzer angezeigt. Nutzer A wählt aus der Liste einen Nutzer B mit Status *Online* aus, von dem er die Position angezeigt haben möchte (A1). Die Informationen werden an die Server-Anwendung übertragen. Dort werden die letzten Positionen des ausgewählten Nutzers B automatisch überprüft (A2, A3), zurück an den Client übertragen und dort auf einer Karte angezeigt. Aus diesen vergangenen Positionen wird für einen definierten Zeitraum eine abgeleitete Position errechnet und die Ansicht aktualisiert. Nach Ablauf des Zeitraums wird der gesamte Vorgang in regelmäßigen Abständen wiederholt (A4).

Ausnahmen

A1 Nutzer A hat während der Sitzung bereits einen Nutzer B ausgewählt, der Tracking-Vorgang wurde bereits ausgeführt und die Position von Nutzer B soll aktualisiert werden. Der Anwendungsfall wird normal fortgesetzt.

A2 Für den Tracking-Vorgang ist eine manuelle Bestätigung des Zugriffs durch Nutzer B erforderlich. Dessen Client-Anwendung zeigt eine entsprechende Meldung an. Nutzer B kann den Zugriff erlauben oder verweigern. Bei Erlaubnis wird der Anwendungsfall normal fortgesetzt. Bei Verweigerung wird nicht die Position, sondern eine entsprechende Hinweismeldung in der Anwendung von Nutzer A angezeigt.

A3 Der Zugriff wurde verweigert. Eine entsprechende Hinweismeldung wird angezeigt.

A4 Die Client-Anwendung von Nutzer B befindet sich nicht mehr im Online-Modus oder kann nicht mehr erreicht werden. Die Position kann nicht aktualisiert werden. Eine entsprechende Hinweismeldung wird angezeigt.

3.3.1.5 Kommunikation

Dieser Anwendungsfall beschreibt die Kommunikation mit anderen Nutzern.

Vor- und Nachbedingung Der Anwendungsfall barrierefreie Navigation muss im Online-Modus ausgeführt werden. Am Ende des Anwendungsfalls wird die Nachricht in der mobilen Anwendung eines anderen Nutzers angezeigt.

Beschreibung Ein Nutzer A öffnet eine Liste mit Namen anderer Nutzer aus seinem sozialen Netzwerk. Neben dem Namen wird der Status dieser Nutzer angezeigt. Der Nutzer wählt eine Person aus der Liste aus, der er eine Nachricht schicken möchte und welcher den Status *Online* hat. Der Nutzer gibt eine Nachricht ein und versendet diese. Die Informationen werden an die Server-Anwendung und anschließend an die Client-Anwendung des anderen Nutzers übertragen und ihm dort angezeigt.

3.4 Nichtfunktionale Anforderungen

Die nichtfunktionalen Anforderungen beschreiben Rahmenbedingungen, unter denen die funktionalen Anforderungen erbracht werden müssen, Anforderungen die unabhängig von einem speziellen Anwendungsfall sind, oder die aus anderen Gründen nicht als Funktion

beschrieben werden können (Kahlbrandt (2001)). Diese sind bei der Entwicklung des mobilen Informationssystems zu berücksichtigen und haben entscheidende Auswirkungen auf den Entwurf der Software. Dabei sind nichtfunktionale Anforderungen von mobilen Informationssystemen (vgl. Abschn. 2.1.2) im Allgemeinen, sowie von ortsbezogenen Diensten im Speziellen (vgl. Abschn. 2.2.3) zu beachten. Diese nichtfunktionalen Anforderungen werden im Folgenden in Leistungsanforderungen, Qualitätsanforderungen und Randbedingungen unterteilt. Abb. 3.6 zeigt eine Übersicht.

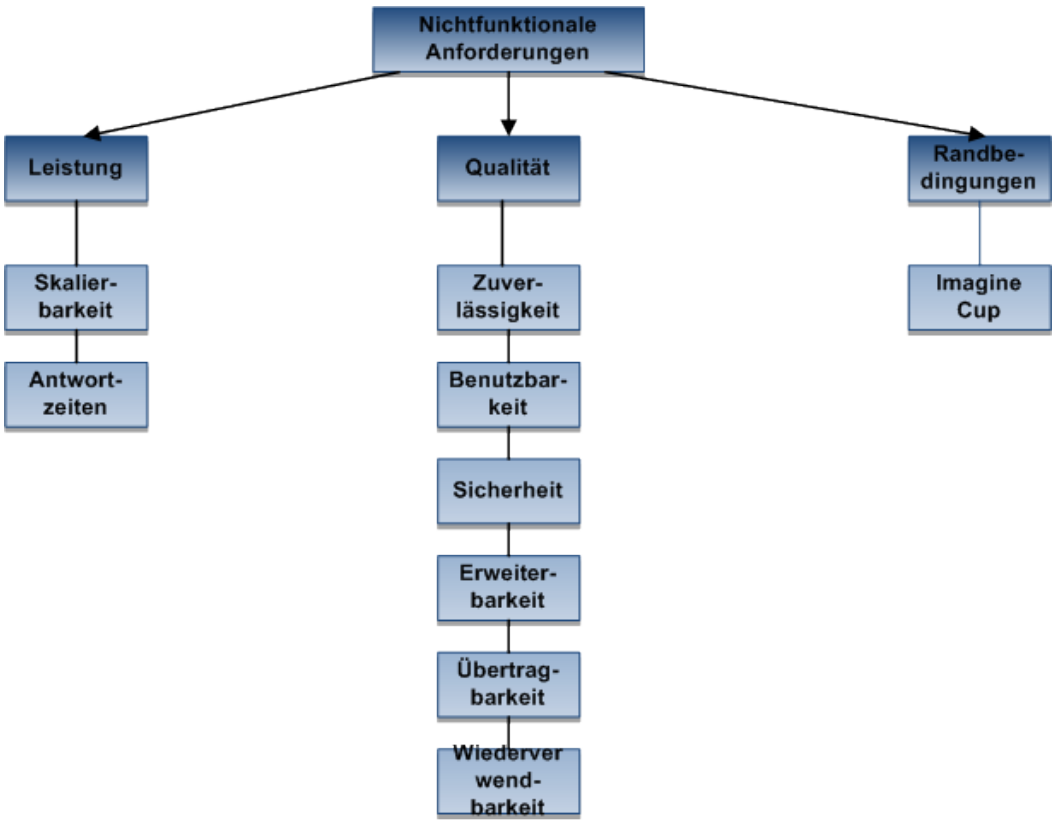

Abbildung 3.6: Nichtfunktionale Anforderungen an Trailblazers Beta

3.4.1 Leistungsanforderungen

Die Leistungsanforderungen beschreiben Anforderungen bzgl. Zeiten, Mengen, Geschwindigkeiten und Raten wie z. B. Laufzeit und Platzbedarf.

Skalierbarkeit Die Skalierbarkeit des gesamten Systems muss gegeben sein. Die Server-Anwendung muss, gemäß der unten genannten Anforderungen zur Verfügbarkeit, bis zu 1000 Client-Anwendungen gleichzeitig bedienen können. Die Architektur der

Server-Anwendung soll durch Hinzufügen von Hardware an größere Nutzerzahlen adaptierbar sein. Aufgrund der Unterstützung für ein Offline-Szenario werden Daten auf dem mobilen Geräten zwischengespeichert. Die Client-Anwendung muss die Ressourcen der verfügbaren Hardware berücksichtigen und unabhängig von der Datenmenge effizient arbeiten. Die maximale Kapazität der zwischengespeicherten Daten soll dabei 5 MB nicht überschreiten bzw. vom Nutzer festgelegt werden können.

Antwortzeiten Die Kommunikation zwischen Client- und Server-Anwendung muss für den Nutzer in akzeptabler Zeit erfolgen. Dazu soll die Verarbeitung einer Anfrage von der Client- an die Server-Anwendung asynchron erfolgen, so dass die Client-Anwendung nicht blockiert, der Nutzer diese jederzeit bedienen und die Anfrage auch abbrechen kann. Die Antwort der Server-Anwendung kann dabei je nach Anwendungsfall und Übertragungsgeschwindigkeit des Kommunikationsnetzwerkes variieren. Das Laden einer Karte beim Anwendungsfall barrierefreie Navigation darf 60 Sekunden nicht überschreiten. Die Client-Server-Kommunikation für die anderen Anwendungsfälle darf maximal 20 Sekunden dauern.

3.4.2 Qualitätsanforderungen

Da nichtfunktionale Anforderungen die Qualitätseigenschaften des Systems betreffen, werden die in Kahlbrandt (2001) aufzeigten Qualitätseigenschaften herangezogen, um die nichtfunktionalen Anforderungen zu konkretisieren.

Zuverlässigkeit Für die Zuverlässigkeit der Software soll die in Abschn. 2.2.3 geforderte Verlässlichkeit von den Daten und der Positionsbestimmung berücksichtigt werden. Der Nutzer muss Hinweise über die Qualität der Informationen bekommen, damit er entsprechend handeln kann. Der Dienst muss für die Nutzer zuverlässig und 24 Std. am Tag, 7 Tage die Woche und 365 Tage im Jahr, erreichbar sein. Angestrebt wird dabei eine Verfügbarkeit von 99 Prozent. Dies erlaubt eine Ausfallzeit von 88 Std. im Jahr.

Benutzbarkeit Die Bedienbarkeit der Client-Anwendung soll einfach und intuitiv sein. Entgegen der gewünschten Plattformunabhängigkeit, sind für die Benutzerschnittstelle möglichst an die Hardware angepasste Funktionalitäten einzubauen, die Text- und Spracheingabe unterstützen sollte. Fehlerhafte Eingaben müssen als solche erkannt werden, dürfen das System nicht negativ beeinflussen und dem Nutzer kenntlich gemacht werden.

Sicherheit Die Client-Server-Kommunikation muss gesichert sein. Die Authentifizierung des Nutzers, die Integrität der Daten, sowie die Vertraulichkeit müssen gegeben sein. Des weiteren gilt es, die Privatsphäre des Nutzers zu schützen. Dazu sind die oben genannten Konsequenzen umzusetzen.

Erweiterbarkeit Das System soll kostengünstig und einfach erweiter- und veränderbar sein. Die einzelnen Komponenten müssen genau definierte Schnittstellen anbieten. Diese beschreiben die von einer Komponente angebotenen Dienste. Dazu gilt es beim Software-Entwurf einzelne Komponenten zu identifizieren und entsprechend zu entwerfen. Einzelne Komponenten sollen dem System, wenn erforderlich auch zur Laufzeit, hinzugefügt und wieder entfernt werden können. Wie oben erwähnt, kann der Dienst auch für andere Zielgruppen Mehrwert bringen. Dementsprechend ist Mandantenfähigkeit, insbesondere bei der Datenhaltung, zu integrieren.

Übertragbarkeit Das Informationssystem wird aus einer Vielzahl zueinander heterogener, technischer Hardware-Komponenten aufgebaut. So soll die mobile Client-Anwendung auf verschiedenen, in Abschn. 2.1.5 vorgestellten Geräten, ausführbar sein. Die Software muss entsprechend an die Hardware-Plattformen angepasst sein. Dies kann durch mobile Laufzeitsysteme als Plattform erreicht werden (vgl. Abschn. 2.1.4). Um den Dienst zum Erfolg zu führen, muss dieser von vielen Nutzern konsumiert werden. Da die Java- und die .NET-Plattformen stark verbreitet sind, dienen diese als ideale Plattformen für die mobile Client-Anwendung. Dies erfordert eigenständige Implementierungen, der Entwurf soll dabei jedoch generisch gehalten sein. Die Server-Anwendung dagegen kann auf einer einzigen Plattform realisiert werden, muss die Heterogenität der mobilen Plattformen jedoch berücksichtigen.

Wiederverwendbarkeit Da das mobile Informationssystem auf verschiedenen Plattformen ausgeführt werden soll, müssen Software-Komponenten identifiziert und in Module gekapselt werden. Redundante Implementierungen gilt es zu verhindern. Des weiteren sollen vorhandene Software-Komponenten eingekauft und nicht neu implementiert werden.

3.4.3 Randbedingungen

Die Randbedingungen beschreiben nichtfunktionale Anforderungen, die sich weder durch Leistungs-, noch durch Qualitätsanforderungen beschreiben lassen.

Imagine Cup Wie oben beschrieben, ist die Trailblazers-Software Alpha im Rahmen des Imagine Cups in der Kategorie Software Design entstanden. In diesem Wettbewerb sollte von den Teilnehmern eine Software entworfen werden, die definierten, nichtfunktionalen Anforderungen genügt. Die Software musste einen selbst erstellten Web-Service enthalten und unter Verwendung der Entwicklungs- und Laufzeitumgebung .NET 2.0 implementiert worden sein. Die bisherigen, dementsprechenden Prototypen sollten weiterverwendet werden, da sie als Grundlage für diese Arbeit dienten. In Zukunft können, wie oben gefordert, weitere Client-Anwendungen für die Java-Plattform entwickelt werden.

3.5 Fazit

Die im Rahmen des Imagine Cup-Wettbewerbs entwickelte Trailblazers Alpha-Software zeigt, dass ein derartiger Ansatz zur Gewinnung ortsbezogener Daten auf positive Resonanz stößt. Die Erkenntnisse der Diskussion lassen die Schlussfolgerung zu, dass ortsbezogene Dienste im Allgemeinen von aktuellen Entwicklungen profitieren.

Speziell die Generierung von Kartenmaterial durch eine Nutzergemeinschaft hat enormes Potential und kann das Verständnis von Karten grundlegend verändern. Dies haben auch die kommerziellen Kartenanbieter erkannt (vgl. TeleAtlas (2007a)). Mit den hinzugekommenen Anwendungsfällen Nutzer-Tracking und Kommunikation bietet Trailblazers Beta ein mobiles, soziales Netzwerk. Der gesamte Trailblazers-Dienst fördert darüber hinaus die ortsbezogene Kollaboration aller Nutzer.

Diese Entwicklungen führen jedoch auch zu neuen Herausforderungen; die Nutzer sollen zusammenarbeiten, sich des gemeinsamen Ziels stets bewusst sein und sich aktiv dafür einsetzen. Dabei ist es zweckmäßig, wenn die Nutzer sich gegenseitig in bestimmten Aspekten kontrollieren. Aktivitäten einzelner Nutzer sollen gewürdigt und für alle sichtbar gemacht werden.

Dienste im Web 2.0 brauchen neue Geschäftsmodelle, um erfolgreich zu sein. Die Nutzung der Dienste sollte kostenlos sein; darüber hinaus sollten die Beiträge der Nutzer entsprechend honoriert werden. Werbung ist oftmals das einzige Mittel für einen Web 2.0-Dienst, um Umsätze zu generieren. Ortsbezogene Dienste können Nutzer mithilfe des skizzierten Anwendungsfalls Werbung gezielt adressieren. Insgesamt können derartige Dienste zerstörend für etablierte Dienste und Geschäftsmodelle sein; insbesondere die Geschäftsmodelle der oben genannten Kartenanbieter müssen sich radikal verändern, sobald sie ernst zunehmende Konkurrenz durch entsprechende Web 2.0-Dienste erfahren.

Bisher war die mangelnde Akzeptanz der ortsbezogenen Dienste hauptsächlich mit dem geringen Vertrauen der Nutzer in die Sicherung der Privatsphäre begründet. Der Gedanke des Web 2.0 könnte wesentlich dazu beitragen, dass Nutzer ihre Privatsphäre weniger stark geschützt haben wollen bzw. genau definierte Informationen über sich preisgeben. Diese Informationen sollten nach Möglichkeit anonymisiert sein. Der Nutzer muss in jedem Fall darüber informiert werden, wann, wie und welche seiner personenbezogenen Daten erfasst, verarbeitet oder gespeichert werden. Ihm muss jederzeit die Möglichkeit gegeben werden, dieses zu unterbinden.

4 Entwurf

In diesem Kapitel wird der Software-Entwurf des Trailblazers Beta Informationssystems besprochen. Dazu werden zunächst Architekturalternativen diskutiert und bewertet und im Anschluss die gewählte Anwendungsarchitektur des Informationssystems beschrieben. Außerdem werden der Entwurf der ortsbezogenen Dienste der stationären Server-Anwendung und der Entwurf der mobilen Client-Anwendung detailliert aufgezeigt.

4.1 Architekturalternativen

Die Software-Architektur ist maßgeblich dafür verantwortlich, dass die herausgearbeiteten funktionalen und nichtfunktionalen Anforderungen erfüllt werden. Insbesondere die Qualitätsmerkmale der nichtfunktionalen Anforderungen werden mit dem Entwurf der Architektur umgesetzt. Genaue Definitionen für Software-Architekturen sind z. B. in Dunkel und Holitschke (2003), Dustdar u. a. (2003) und Starke (2005) zu finden.

Trailblazers Alpha basiert gemäß den Rahmenbedingungen auf einer einfachen Client-Server-Architektur mit HTTP-basierten Web-Services als Kommunikationsmittel. Die Architektur entspricht jedoch nicht den übrigen, in dieser Arbeit herausgearbeiteten nichtfunktionalen Anforderungen. Daher werden in diesem Abschnitt Alternativen für den Entwurf der Software-Architektur diskutiert. Dazu werden generelle Architekturmuster skizziert, Musterarchitekturen vorgestellt und die Verteilung der Software-Komponenten zwischen mobiler Client- und stationärer Server-Anwendung herausgearbeitet. Ausgehend von der Verteilung wird die Kommunikation zwischen stationärer und mobiler Anwendung skizziert; dazu werden mögliche Arten der Kommunikation aufgezeigt und bewertet.

4.1.1 Architekturmuster

Das Informationssystem besteht aus einer mobilen und einer stationären Anwendung. Diese verteilte Anwendung nutzt ein verteiltes System, um dem Nutzer die in den funktionalen Anforderungen beschriebene, fachliche Funktionalität zur Verfügung stellen zu können.

Für verteilte Anwendungen existieren verschiedene Architekturmuster. Diese beschreiben die Rolle der einzelnen Anwendungskomponenten und ihre Beziehung zueinander. Im Folgenden werden drei elementare Muster vorgestellt:

Client-Server-Architekturen Die Eigenschaften von Client-Server-Architekturen wurden bereits in Abschn. 2.1.3 skizziert. Aus technischer Sicht handelt es sich dabei um mehrere, kurzlebige Client-Anwendungen, die auf Anfragen an eine langlebige Server-Anwendung entsprechende Antworten erhalten. Verteilte Anwendungen werden häufig mit einer Client-Server-Architektur umgesetzt; jede Web-basierte Anwendung im Internet basiert auf einer Client-Server-Architektur.

Peer-To-Peer-Architekturen Bei ortsbezogenen Diensten der nächsten Generation basiert die genutzte Infrastruktur auf einem so genannten Peer-To-Peer-Netzwerk (vgl. Abschn. 3.2). Bei Anwendungen, die nach dem Peer-To-Peer-Ansatz verteilt sind, interagieren mehrere, gleichberechtigte Anwendungen miteinander; eine Anwendung ist sowohl Client als auch Server. Weiterführende Informationen sind z. B. in Dustdar u. a. (2003) zu finden.

Push-Architekturen Neben der Nutzung eines Peer-To-Peer-Architekturmodells unterscheiden sich ortsbezogene Dienste der nächsten von der aktuellen Generation auch dadurch, dass diese vermehrt Push-Diensten entsprechen. Wie bereits erwähnt, wird der Nutzer bei derartigen Diensten informiert, sobald ein Ereignis eintritt. Dazu muss sich der Nutzer vorab beim Dienst registrieren. Push-Dienste werden dann eingesetzt, wenn Informationen gezielt an definierte Empfänger weitergeleitet werden sollen. In Dustdar u. a. (2003) werden Push-Architekturen detailliert beschrieben und von Peer-To-Peer-Architekturen abgegrenzt.

4.1.2 Musterarchitekturen

Muster- oder Referenzarchitekturen beschreiben in einer spezifischen Fachdomäne den vollständigen Entwurf einer Anwendung. Unter der Annahme, dass weitere Anwendungen in der gleichen Fachdomäne ähnliche Entwurfsprobleme beinhalten, können Referenzarchitekturen als Vorlage für den architektonischen Entwurf einer Anwendung gelten (vgl. Starke (2005)).

Für verteilte Smart-Client-Anwendungen, die auf dem .NET Framework basieren, existieren verschiedene Referenzarchitekturen (vgl. Microsoft (2006e) und Microsoft (2006a)) der Firma Microsoft. Die Architekturen sind ausführlich beschrieben und zugehörige Anwendungen beispielhaft implementiert. Diese sind der Fachdomäne betriebswirtschaftlicher Informationssysteme zuzuordnen. Obgleich dessen, wurden die Architekturen in dieser Arbeit entsprechend berücksichtigt.

In dem Bereich der ortsbezogenen Dienste gibt es einige Arbeiten und Projekte die sich speziell mit der Entwicklung geeigneter, generischer Architekturen beschäftigen. So wird z. B. in Herden u. a. (2006) die Architektur eines personalisierten und ortsbezogenen Dienstes exemplarisch beschrieben. Das Projekt LOMS (2007) hat das Ziel eine offene Dienstarchitektur für ortsbezogene Dienste zu definieren und zu entwickeln. Erkenntnisse aus diesen Arbeiten wurden ebenfalls in den Entwurf mit einbezogen.

4.1.3 Verteilung der Architekturschichten

Bei einer Schichtenarchitektur hat jede Schicht eine dedizierte Aufgabe. Jede Schicht hat eine starke Kohärenz, bietet lose Kopplung zu anderen Schichten und hat nur Zugriff auf die jeweils tiefer liegende Schicht. Dadurch werden die Komponenten einer Software-Schicht vor der Außenwelt gekapselt und bilden eine schmale, definierte Schnittstelle. Applikationen, die auf einer Drei-Schichten-Architektur aufbauen, sind relativ robust, da sich Änderungen innerhalb einer Schicht nicht auf andere Schichten auswirken. Diese Kriterien dienen den in den nichtfunktionalen Anforderungen genannten Qualitätskriterien für Wartbarkeit und Erweiterbarkeit (vgl. Abschn. 3.4.2).

Ein weiterer Vorteil der Drei-Schichten-Architektur ist die Möglichkeit, einzelne Schichten auf physikalisch unterschiedliche Rechner zu verteilen. Für die Architektur von Trailblazers Beta ist die Verteilung der einzelnen Komponenten von elementarer Bedeutung. Ausgehend von einer Drei-Schichten-Architektur, wird im Folgenden die Verteilung der einzelnen Schichten zwischen Client- und Server-Anwendung diskutiert. Um eine Einteilung vornehmen zu können, wurden Kriterien mit Einfluss auf die Verteilung skizziert. In Book u. a. (2005) wurden die drei Kriterien Benutzermobilität, Gerätemobilität und Dienstkonnektivität herausgearbeitet. Diese abstrahieren von fachlichen und technischen Kriterien für die Anwendung und konzentrieren sich auf die Anforderungen durch die Benutzer. Sie bieten so, zusammen mit den oben ausgearbeiteten funktionalen Anforderungen, ein adäquates Mittel zur Identifizierung der Verteilung aus konzeptioneller Sicht.

Benutzermobitität Die Benutzermobilität beschreibt den Grad der Orts- und Bewegungsfreiheit, die ein Nutzer bei Anwendung eines Dienstes erfährt. Diese kann allgemein *lokal*, *verteilt*, *mobil* und *in Bewegung* sein. Demnach wird der Trailblazers-Dienst von Personen genutzt, die sich in Bewegung befinden.

Gerätemobilität Die Gerätemobilität beschreibt die physische Mobilität der Geräte in Bezug zu den Zugängen eines Kommunikationsnetzwerkes, wie z. B. z. B. GSM-Mobilfunk- oder WLAN-Zellen. Mobilität beschreibt in diesem Zusammenhang Geräte, die sich zwischen den Abdeckungsgebieten der Netzwerke bewegen. Es gibt *lokal*, *verteilt*, *mobil* und *in Bewegung* funktionierende Geräte. Die mobile Trailblazers-Client-Anwendung soll für mobile in Bewegung funktionierende Geräte, wie z. B. Smartpho-

nes, konzipiert werden. Grundsätzlich muss bedacht werden, welches Netzwerk als Bezugssystem in Frage kommt.

Dienstkonnektivität Die Dienstkonnektivität ist stark an die Gerätemobilität und die skizzierten Abdeckungsgebiete der Netzwerkzugangspunkte gebunden. Die Abdeckungsgebiete müssen sich nicht notwendigerweise überlappen, es können auch Gebiete ohne Netzwerkzugangspunkt existieren, was der Dient bei mobilen und in Bewegung funktionierenden Geräten berücksichtigen muss. Grundsätzlich kann ein Dienst ein *Offline-Dienst*, ein *Hybrid-Offline-Dienst*, ein *Hybrid-Online-Dienst* oder ein *Online-Dienst* sein. Die in dieser Arbeit bisher gestellten Anforderungen an mobile Informationssysteme, ortsbezogene Dienste und Navigationssysteme (vgl. Abschn. 2.1.2, 2.2.3 und 2.2.5) nach einem Online-Offline-Szenario stellen einen Hybrid-Offline-Dienst dar. Diese Art von Diensten erlaubt dem Nutzer, Anwendungsfälle auch ohne Netzverbindung durchzuführen. Die Verbindung wird nur gelegentlich benötigt, um Daten mit einer Server-Anwendung zu synchronisieren. Hybrid-Online-Dienste sind dagegen im Regelfall mit einem entfernten Dienst verbunden, erlauben jedoch die Überbrückung von Verbindungsabbrüchen.

Abb. 4.1 zeigt ein Verteilungsmuster für verteilte Informationssysteme in Abhängigkeit dieser drei Kriterien, insbesondere der Dienstkonnektivität. Dieses Muster erweitert das in Abschn. 2.1.3 eingeführte Muster für leichtgewichtige und schwergewichtige Clients mobiler Anwendungen. Für jeden Grad der Konnektivität wird die Verteilung der einzelnen Schichten zwischen Client und Server gezeigt. Der Verteilung werden die minimalen Client- und die maximalen Server-Anforderungen zugrunde gelegt.

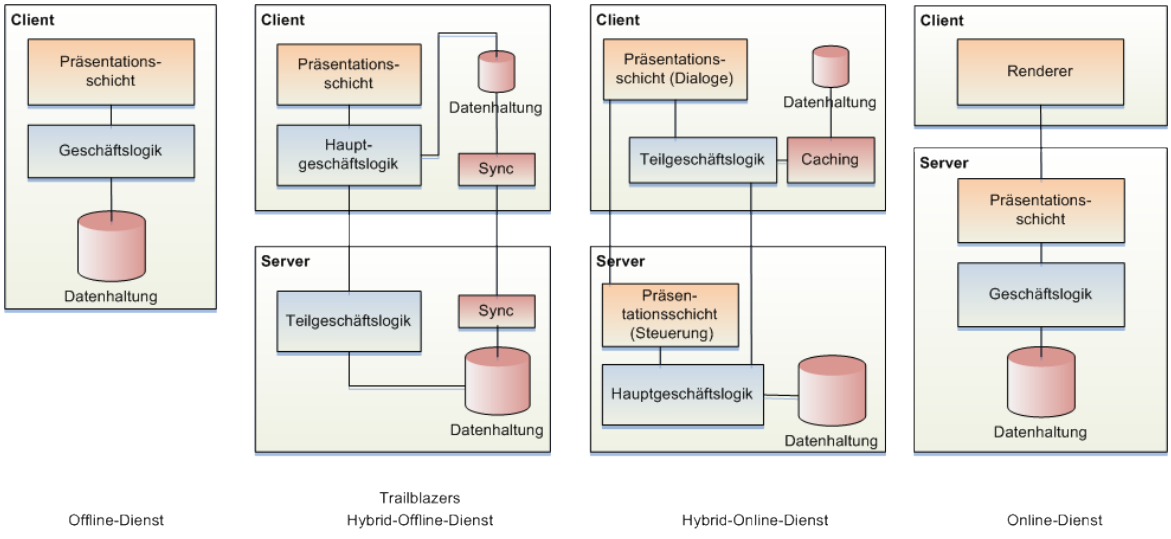

Abbildung 4.1: Verteilungsmuster für Informationssysteme in Abhängigkeit der Dienstkonnektivität (nach Book u. a. (2005))

4.1.4 Client-Server-Kommunikation

Nachdem die Verteilung der einzelnen Software-Schichten zwischen der Client- und der Server-Anwendung diskutiert wurde, wird im Weiteren die Kommunikation zwischen Client- und Server-Anwendung aufgezeigt.

4.1.4.1 Kommunikationsmodelle

Für die Kommunikation zwischen Client und Server bzw. zwischen Sender und Empfänger stehen grundsätzlich das synchrone und das asynchrone Kommunikationsmodell zur Verfügung.

Synchrone Kommunikation Bei der synchronen Kommunikation schickt der Sender eine Nachricht an den Empfänger und wartet, bis dieser auf die Nachricht antwortet. Synchrone Kommunikation wird bei verteilten Anwendungen mit hoher Interaktionsrate und enger Kopplung genutzt.

Asynchrone Kommunikation Bei der asynchronen Kommunikation wartet der Sender nicht, bis der Empfänger antwortet, und ist dadurch nicht blockiert. Wenn ihn die Antwort erreicht, wird er benachrichtigt. Dadurch wird eine gute Entkopplung zwischen Sender und Empfänger erreicht. Ein derartiges Kommunikationsmodell wird oft bei schlechter Netzwerkverbindung und hoher Ausfallwahrscheinlichkeit eingesetzt.

Es ist möglich, synchrone mittels asynchroner Kommunikation umzusetzen und umgekehrt. Im ersten Fall wartet der Sender explizit, bis der Empfänger eine Antwort auf die Nachricht sendet. Diese wird ebenfalls explizit losgeschickt. Im zweiten Fall wird ein zusätzlicher Prozess erzeugt, der die Nachrichten zwischen Sender und Empfänger zwischenspeichert.

4.1.4.2 Kommunikationsmöglichkeiten der .NET-Plattform

Das .NET Framework und das .NET Compact Framework sind anwendungsorientierte Middlewares, die die Entwicklung verteilter Anwendung vereinfachen und die Abläufe der Anwendungen unterstützen (vgl. Hammerschall (2005)). Die .NET-Plattform bietet grundsätzlich folgende Möglichkeiten der verteilten Client-Server-Kommunikation (vgl. Hill u. a. (2004) und Salmre (2005)):

Sockets Als rudimentärste Möglichkeit bietet .NET die direkte programmatische Nutzung von Protokollen der Anwendungsschicht (z. B. HTTP oder FTP) und der Transportschicht (TCP und UDP) des ISO-OSI-Schichtenmodells. Dadurch können eine größere Flexibilität und eine Steigerung der Leistung erreicht werden. Sämtliche Vorteile einer anwendungsorientierten Middleware gehen jedoch verloren.

.NET Remoting .NET Remoting bietet entfernte, synchrone Methodenaufrufe (engl.: Remote Method Invocation, RMI) für Objekte einer verteilten Anwendung. Es können diverse Kommunikationsprotokolle verschiedener Schichten, wie TCP oder HTTP, und verschiedene Techniken zur Objektserialisierung, wie XML- oder Binärformat, eingesetzt werden. Dazu wird auf Client-Seite ein Stellvertreterobjekt (engl.: proxy object) eingeführt, das sich wie das Server-Objekt verhält und die Methodenaufrufe intern an das eigentliche Server-Objekt weiterleitet. .NET Remoting bietet dazu eine geeignete Infrastruktur, sowie die Verwaltung der lokalen und entfernten Objekte.

Microsoft Message Queue Nachrichtenwarteschlangen (engl.: message queue) basieren auf dem asynchronen Kommunikationsmodell. Die Microsoft Message Queue (MSMQ) ist eine Implementierung für .NET-basierte Anwendungen, die auf Betriebssystemen der Windows-Familie ausgeführt werden. Sender und Empfänger müssen nicht gleichzeitig miteinander verbunden sein; die Übertragung der Nachricht wird dennoch garantiert, indem die Nachrichten bzw. die Antworten in Warteschlangen vorgehalten werden. Zusätzliche Sicherheitsaspekte, wie die Verschlüsselung und Authentifizierung der Nachrichten, stellen weitere Vorteile dar. Nachteilig hingegen ist im Besonderen, dass die Message Queue-Komponente auf jedem mobilen Gerät installiert werden muss und die Interaktion mit anderen Systemen außerhalb der .NET-Plattform nicht immer problemlos möglich ist.

Web-Services Sowohl das .NET Framework, als auch das .NET Compact Framework unterstützen den Konsum von Web-Services, das .NET Framework unterstützt außerdem die Implementierung und Bereitstellung derartiger Dienste in einem Web-Server, den so genannten ASP.NET Web-Services.

WCF Mit der Windows Communication Foundation (WCF) werden die oben skizzierten Kommunikationsmöglichkeiten unter einer einheitlichen Programmierschnittstelle zusammengefasst. Die WCF ist ab der Version 3.0 Bestandteil des .NET Frameworks (vgl. Abschn. 2.1.4.1).

4.1.5 Bewertung

Der Peer-To-Peer Ansatz bringt einige Vorteile mit sich, dennoch ist er für mobile Anwendungen und die skizzierten funktionalen Anforderungen nicht geeignet. Die eingeschränkten Ressourcen mobiler Geräte, sowie eine nicht permanent vorhandene Internet-Verbindung erfordern einen zentralen Server. Des weiteren können mobile Peer-To-Peer-Anwendungen ihren Nutzen vor allem in lokalen Ad-Hoc-Netzwerken entfalten. Für die Trailblazers-Software dient das Internet als zugrunde liegende Infrastruktur. Die Infrastruktur des Internet unterscheidet sich signifikant von der Infrastruktur von Ad-Hoc-Netzwerken. Client-Server-Architekturen haben sich zudem bewährt und sind mit vorhandenen Werkzeugen problemlos

umzusetzen. Das in dieser Arbeit entwickelte mobile Informationssystem basiert daher auf einer derartigen Architektur.

Aufgrund der zur Zeit unzureichenden Unterstützung mobiler Push-Dienste, sollte die Trailblazers-Software Beta lediglich Pull-Dienste anbieten. Die Client-Anwendung muss bei einer derartigen Architektur allerdings das lokale Zwischenspeichern (engl.: caching) von Daten in der Client-Anwendung, die Synchronisierung der Daten mit der Server-Anwendung und die damit verbundene Nebenläufigkeit der Daten berücksichtigen.

Web-Services bieten die geeignetste Möglichkeit der Interoperabilität zwischen heterogenen Client-Server-Anwendungen. Die Schnittstellen des Web-Services müssen dabei so entworfen werden, dass diese Interoperabilität gewährleistet bleibt. So dürfen insbesondere keine plattformspezifischen Datentypen[1] genutzt werden.

4.2 Anwendungsarchitektur

In Starke (2005) werden verschiedene Sichten auf die Architektur eines Software-Systems beschrieben. Die so genannte Kontext- oder konzeptionelle Sicht beschreibt das System und ihre Schnittstellen auf einem hohen Abstraktionsniveau. Nachfolgend wird die Makroarchitektur des Informationssystems mit dieser Sicht beschrieben, um einen abstrakten Überblick über die Schichten und Komponenten der Server- und der Client-Anwendung zu geben.

4.2.1 Server-Anwendung

Der Server übernimmt die Datenhaltung und Teile der Geschäftslogik. Daten werden, mittels definierter Schnittstellen, verschiedenen Clients über die Dienstfassade zur Verfügung gestellt. Die konzeptionelle Sicht auf die Architektur der Server-Anwendung ist in Abb. 4.2 dargestellt.

Die `Dienstfassadenschicht` beschreibt den Einstiegspunkt für alle Konsumenten des Trailblazers-Dienstes. In dem `Dienstvertrag` werden die Operationen und die Entitäten des Dienstes, sowie die Nachrichten für die Operationen definiert. Der `Dienstadapter` implementiert diesen Vertrag und bietet einen Endpunkt, basierend auf ASP.NET Web-Services.

[1] Oftmals nutzen in .NET implementierte Web-Service den Datentyp DataSet. Dieser ist z. B. in Java nicht vorhanden. Der Web-Service kann dann nicht von Java-Anwendungen konsumiert werden.

Wie in Abschn. 4.1.3 gezeigt, wird die Anwendungslogik auf die Client- und die Server-Anwendung verteilt. Die `Anwendungsschicht` enthält die Server-seitige Anwednungslogik. Die `Anwendungsfassade` bietet dazu einen zentralisierten Zugriff. Die Komponenten `Route` und `Karte` dienen der Zusammenstellung geeigneter Karteninformationen für den Anwendungsfall berrierefreie Navigation (vgl. Abschn. 3.3.1.1). Die Komponente `Position-Log` ermöglicht das Hinzufügen von Potitions-Logs in das System (vgl. Abschn. 3.3.1.2). Die Komponenten `Kommunikation` und `Tracking` enthalten die Funktionalität für die Anwendungsfälle Kommunikation und Tracking (vgl. Abschn. 3.3.1.5 und Abschn. 3.3.1.4).

Die `Datenzugriffsschicht` ist für jegliche Kommunikation mit der Datenbank zuständig. Mit dem `Datenzugriff` werden Dienst-Entitäten in die Datenbank geschrieben bzw. aus ihr extrahiert. Um den Datenzugriff von einer konkreten Datenbank zu abtrahieren, können verschiedene `Datenbankzugriffsanbieter` implementiert werden.

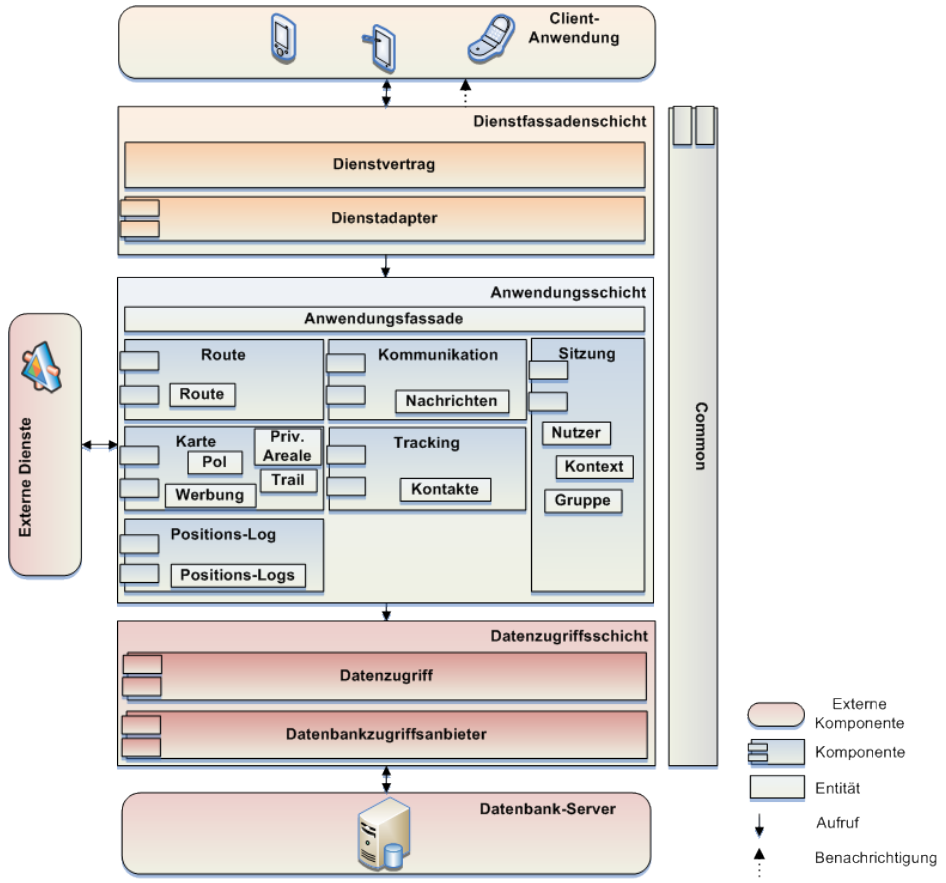

Abbildung 4.2: Konzeptionelle Sicht auf die Architektur der Server-Anwendung

Diese Architektur basiert auf einer Kombination aus dem *Publish-Subscribe*-Konzept und

dem auf einer Datenbank basierenden Konzept für ortsbezogene Dienste. Beide Konzepte werden in verfügbarer Middleware für derartige Dienste genutzt. Alternativ dazu existieren die Konzepte *Subject Spaces* und *Tupel Spaces* (vgl. Schiller und Voisard (2004)).

4.2.2 Mobile Client-Anwendung

Abb. 4.3 zeigt die konzeptionelle Sicht auf die Architektur der mobilen Client-Anwendung.

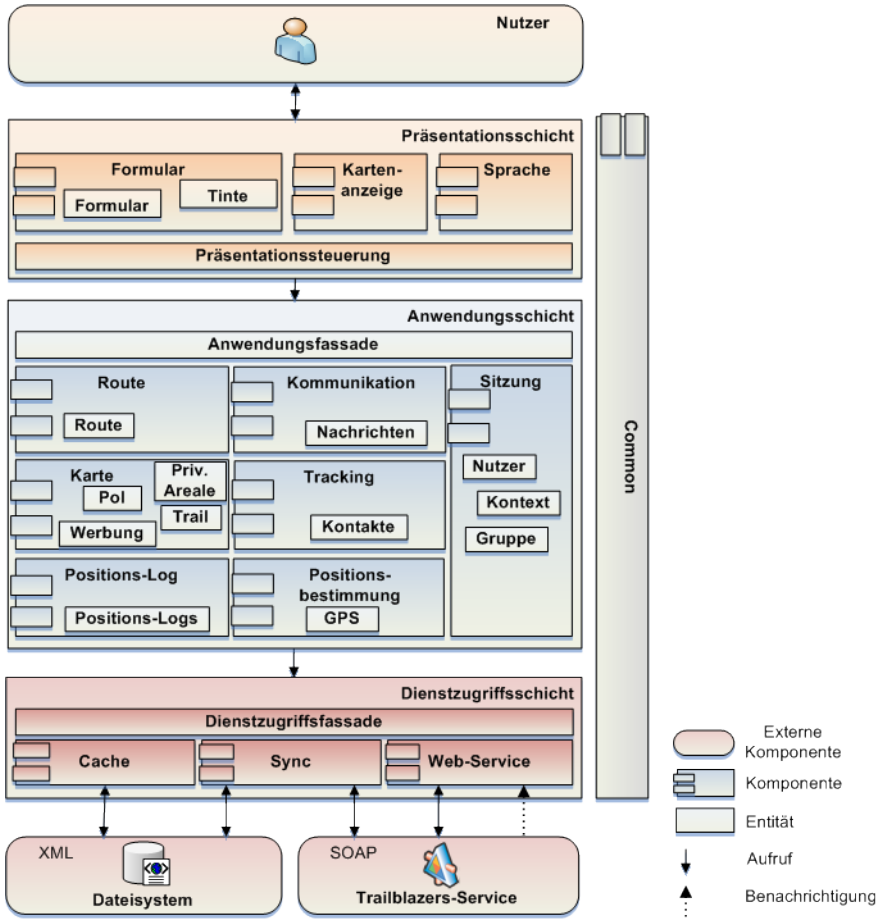

Abbildung 4.3: Konzeptionelle Sicht auf die Architektur der mobilen Client-Anwendung

Der Nutzer hat direkten Zugriff auf die `Präsentationsschicht`. In dieser werden Ereignisse durch Benutzereingaben ausgelöst, sowie fachliche Daten angezeigt. Ereignisse vom Nutzer werden durch die `Präsentationssteuerung` an die `Anwendungsschicht` weitergeleitet, um dort einen entsprechenden fachlichen Ablauf auszulösen. Die Benutzereingaben erfolgen standardmäßig via Tastatur. Optional kann die Eingabe auch via digitaler `Tinte` und `Sprache` erfolgen, sofern dies vom mobilen Gerät unterstützt wird. Die

Anzeige der fachlichen Daten der in Abschn. 3.3.1 beschriebenen Anwendungsfälle erfolgt in einem `Formular`. Optional kann für den Anwendungsfall Navigation die Darstellung der Route via Sprache erfolgen.

Die `Anwendungsschicht` der mobilen Client-Anwendung implementiert die mobilen Teile der fachlichen Anwendungslogik. Die `Anwendungsfassade` bietet der Präsentationsschicht Zugriff auf fachliche Abläufe. Die `Positions`-Komponente liefert via GPS Informationen über die letzte bekannte Position des Nutzers. Des weiteren werden die Positionen kontinuierlich im System als `Positions-Logs` gespeichert, um den Anwendungsfall Trail-Erkennung zu realisieren (vgl. Abschn. 3.3.1.2). Die `Karten`-Komponente bietet zu entsprechend vorgegebenen Positionen Informationen in Form von Karten an. Diese Karten enthalten `Trail`, `Orte von Interesse`, `private Areale` sowie ortsbezogene `Werbung`. Mit der `Routen`-Komponente wird, in Abhängigkeit der durch die `Positions`-Komponente bestimmte Position, eine aktuelle Route von Start- zu Zieladresse berechnet. Die `Positions-`, `Karten-` und `Routen`-Komponenten werden für die Anwendungsfälle Navigation, Werbung, Trail- sowie PoI-Erkennung genutzt (vgl. Abschn. 3.3.1.1, Abschn. 3.3.1.3, Abschn. 3.3.1.2 sowie Abschn. 3.1.3.3). Die `Tracking`-Komponente dient der Positionsbestimmung anderer Nutzer, sowie der Weitergabe eigener Positionsinformationen an andere Nutzer (vgl. Abschn. 3.3.1.4). Die `Kommunikations`-Komponente erweitert die `Tracking`-Komponente um die Möglichkeit, Nachrichten mit anderen Nutzern auszutauschen (vgl. Abschn. 3.3.1.5). Die `Sitzungs`-Komponente wird von anderen Komponenten genutzt, sie bietet Zugriff auf die `Nutzer-` und `Gruppen`-Informationen. Des weiteren überwacht sie die Einhaltung der Privatsphäre des Nutzers.

Die `Dienstzugriffsschicht` bietet der Anwendungsschicht, über die `Dienstzugriffsfassade`, einen geeigneten Zugriff auf den Trailblazers-Dienst. Dabei wird das geforderte Online-Offline-Szenario gewährleistet. Je nach Zustand der Anwendung erfolgt der Zugriff auf den entfernten oder den lokalen Dienst. Entfernte Dienstaufrufe erfolgen via Web-Services, lokale Zugriffe erfolgen auf lokale, im Dateisystem gespeicherte Informationen. Durch den Synchronisations-Mechanismus wird sichergestellt, dass Informationen, die während einer Offline-Sitzung dem lokalen Dienst hinzugefügt wurden, bei der nächsten Online-Sitzung an den entfernten Dienst übermittelt werden. Weiterhin werden mit diesem Mechanismus lokal gespeicherte Daten aktualisiert. Durch einen Benachrichtigungsmechanismus erhält die mobile Anwendung zudem von der Server-Anwendung Nachrichten, wenn bestimmte Ereignisse eintreten.

4.3 Server-Anwendung

In diesem Abschnitt wird die Mikroarchitektur der stationären Server-Anwendung aus der Implementierungssicht detailliert beschrieben. Dabei werden nur die für ortbsezogenen Diens-

te des Informationssystems relevanten Komponenten aufgezeigt. Der Datenbankentwurf und die einzelnen Schichten der Anwendung werden gesondert vorgestellt. Beim Entwurf der Mikroarchitektur wurden Entwurfsprobleme gezielt identifiziert und der Einsatz von Entwurfsmustern (Gamma u. a. (1995)) geprüft und ggfs. umgesetzt. Zur Visualisierung werden UML-Diagramme (Jacobson u. a. (1999)) eingesetzt.

4.3.1 Datenbankentwurf

Räumliche und nicht-räumliche Daten des Informationssystems werden in einem objekt-relationalen Datenbankmanagementsystem (ORDBMS) gespeichert. Wie in Abschn. 2.4.1.2 erläutert, wird für die Speicherung räumlicher Daten eine Erweiterung der Datentypen und der Anfragesprache für relationale Datenbanksysteme vorgeschlagen. Datentypen lassen sich bei ORDBMS entsprechend als komplexe Konstrukte (Objekte) entwerfen.

4.3.1.1 Entity-Relationship-Modell

Ausgehend von den funktionalen Anforderungen, wurde ein Entity-Relationship-Modell (ER-Modell) für die im Informationssystem zu speichernden Daten herausgearbeitet (siehe Abb. 4.4). In diesem Modell wird die in Shekhar und Chawla (2002) vorgestellte Erweiterung räumlicher Datentypen genutzt; die Erweiterungen sind durch Piktogramme für die räumlichen Basisformen Punkt, Linie und Polygon gekennzeichnet.

Nutzer Die Entität `Nutzer` entspricht dem in Abschn. 3.3.1 beschriebenen Akteur Nutzer. Alle Attribute dienen, den Anforderungen entsprechend, ausschließlich der Speicherung anonymisierter Daten (vgl. Abschn. 3.2.5.3). Der `Nutzer` ist durch eine künstliche `Identifikationsnummer (ID)` eindeutig identifizierbar. Das Attribut `Login` ist ein vom Nutzer frei wählbarer, künstlicher Name, die `E-Mail-Adresse` wird ebenfalls vom Nutzer festgelegt. Das Attribut `Punkte` speichert die in Abschn. 3.2.2.3 beschriebenen Punkte für einen Aktivitätsindex, die ein Nutzer für das Aufnehmen bzw. Erzeugen von `Positions-Logs` bzw. von `Orten von Interesse` erhält. Des weiteren gehört ein `Nutzer` einer oder mehreren `Gruppen` an und hat ein oder mehrere `Private Areale` für sich hinterlegt. Ein `Nutzer` kennt keine oder mehrere andere `Nutzer`. Diese rekursive Beziehung ist durch einen `Typ` definiert, um z. B. die Nutzer in einem sozialen Netzwerk einzuordnen (vgl. Abschn. 3.3.1.4 und Abschn. 3.3.1.5).

Gruppe Die Entität `Gruppe` beschreibt eine allgemeine Nutzergruppe, der keine oder mehrere `Nutzer` zugeordnet werden können. Sie dient der Kategorisierung einzelner Nutzer in Gruppen. Dadurch lassen sich Daten der Entitäten `Trail` und `Werbung` bestimmten Nutzergruppen, in dieser Arbeit z. B. Rollstuhlfahrern, zuordnen.

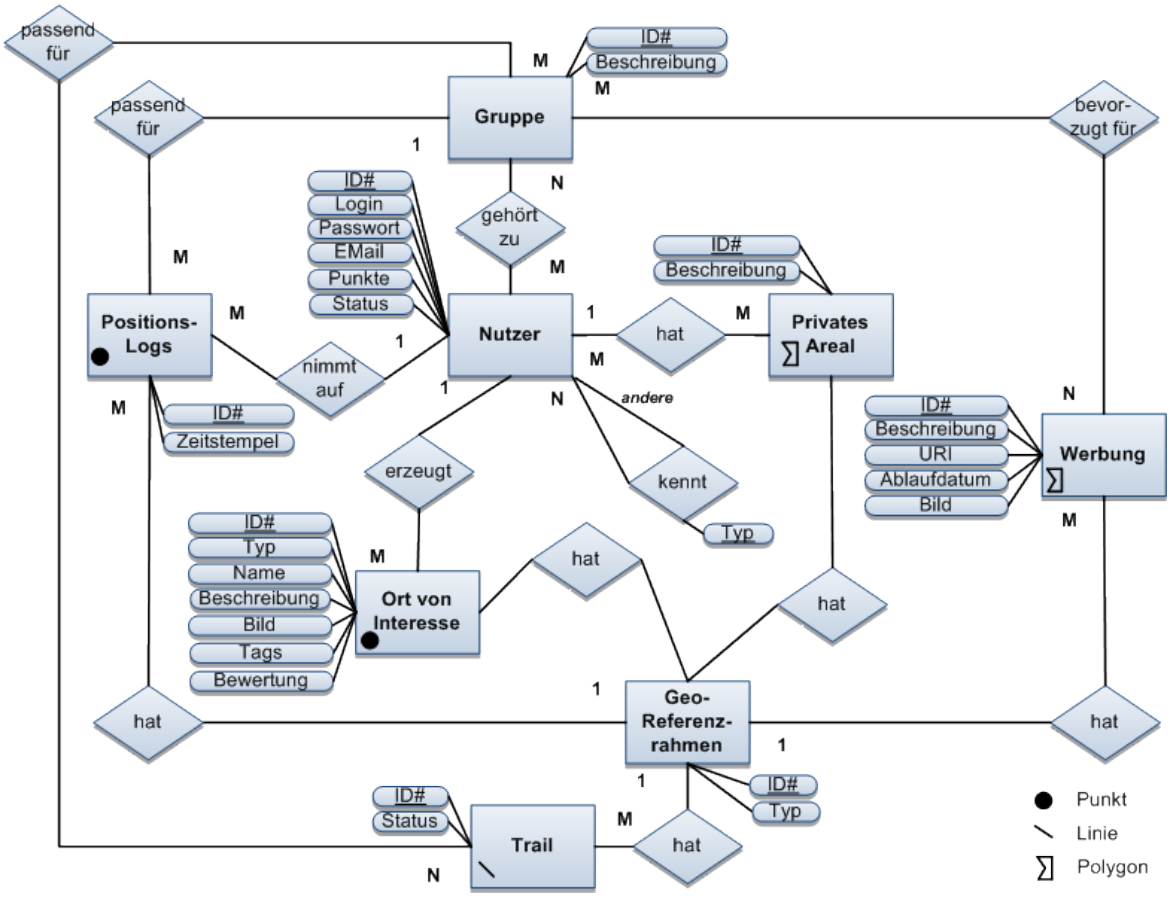

Abbildung 4.4: Enitiy-Relationship-Modell

Positions-Logs Die Entität `Postions-Log` beschreibt ein in Abschn. 3.3.1.2 skizziertes Positions-Datum. Die Entität ist vom räumlichen Datentyp Punkt, der die genaue geografische Position definiert. Ein `Positions-Log` ist durch eine `ID` identifizierbar und wird mit einem `Zeitstempel` versehen. Ein `Positions-Log` wird immer von einem bestimmten `Nutzer` hinzugefügt, der zu diesem Zeitpunkt einer bestimmten `Gruppe` angehört. Dieser `Positons-Log` ist dadurch für die gesamte `Gruppe` von `Nutzern` geeignet.

Ort von Interesse Ein `Ort von Interesse` entspricht den in Abschn. 3.1.3.3 skizzierten Orten von Interesse, die von einzelnen `Nutzern` hinzugefügt werden. Ein `Ort von Interesse` ist ein räumlicher Datentyp Punkt, der eine geografische Position beschreibt. Ein `Ort von Interesse` ist durch eine `ID` identifizierbar. Ein `Name`, eine `Beschreibung`, `Tags` und ein `Bild` werden vom `Nutzer` vergeben. Durch das Attribut `Typ` wird ein `Ort von Interesse` in vorgebende Kategorien eingeordnet. `Bewertung` beschreibt die durchschnittliche Bewertung des

Ortes von Interesse durch die Nutzer. Dadurch wird die in Abschn. 3.2.2.3 beschriebene Qualifizierung der Beiträge eines Nutzers unterstützt.

Privates Areal Das Private Areal beschreibt das in Abschn. 3.3.1.2 beschriebene private Areal, in dem die Aufnahme von Position-Logs unterbunden wird. Die Eckpunkte des Areals werden vom räumliche Datentyp Polygon definiert. Ein Areal ist durch eine ID identifizierbar und enthält eine Beschreibung.

Werbung Werbung beschreibt die in Abschn. 3.3.1.3 vorgestellte Werbung, die einem Nutzer angezeigt wird. Werbung ist durch eine ID identifizierbar und hat eine Beschreibung und ein Bild, die dem Nutzer angezeigt werden sollen. Außerdem enthalten ist ein so genannter Uniform Ressource Identifier (URI), der auf eine genauere Informationsquelle der Werbung im Internet verweist. Ein Ablaufdatum grenzt die Gültigkeit der Werbung ein. Die Entität Werbung ist vom räumlichen Datentyp Polygon; dieses beschreibt ein räumliches Areal, in dem die Werbung angezeigt werden soll, wenn der Nutzer sich in diesem aufhält. Werbung ist bevorzugt für eine oder mehrere Gruppen definiert; pro Gruppe ist mehr als eine Werbung möglich.

Trail Die Entität Trail ist vom räumlichen Typ Linie. Diese beschreibt ein Teilstück eines barrierefreien Weges. Eine Linie besteht aus einem Start- und einem Endpunkt. Ein Trail ist durch eine ID identifizierbar und hat einen Status, der beschreibt, ob der Trail für eine Routenberechnung herangezogen werden kann. Ein Trail ist passend für eine Gruppe von Nutzern.

Geo-Referenzrahmen Jede der oben genannten räumlichen Entitäten hat einen geografischen Referenzrahmen. Dieser wird durch die Entität Geo-Referenzrahmen beschrieben, durch das Attribut ID identifiziert und durch das Attribut Typ kategorisiert. Eine mögliche Kategorie eines Referenzsystems ist der in Abschn. 2.3.1 vorgestellte UTM. Weitere Koordinatensysteme, insbesondere für Positionsinformationen innerhalb eines Gebäudes, sind hier denkbar.

4.3.1.2 Objekt-relationales Datenmodell

Nach einem definierten Algorithmus (vgl. Heuer u. a. (2001)) wurde das ER-Modell in das objekt-relationale Datenmodell überführt und dort in die dritte Normalform gebracht. Abb. 4.5 zeigt das entsprechende Datenbankschema. Die räumlichen Basisformen werden als neue Datentypen definiert.

Damit die Entitäten im gesamten System immer eindeutig identifizierbar sind, werden Primärschlüssel vom Datentyp Global Unique Identifier (GUID) genutzt. Dadurch können auch in der Client-Anwendung neue Daten einer Entität, inklusive eines gültigen Primärschlüssels, erzeugt werden.

Das Attribut `Passwort` der Entität `Nutzer` wird durch eine Hash-Funktion als verschlüsselte Zeichenkette gespeichert.

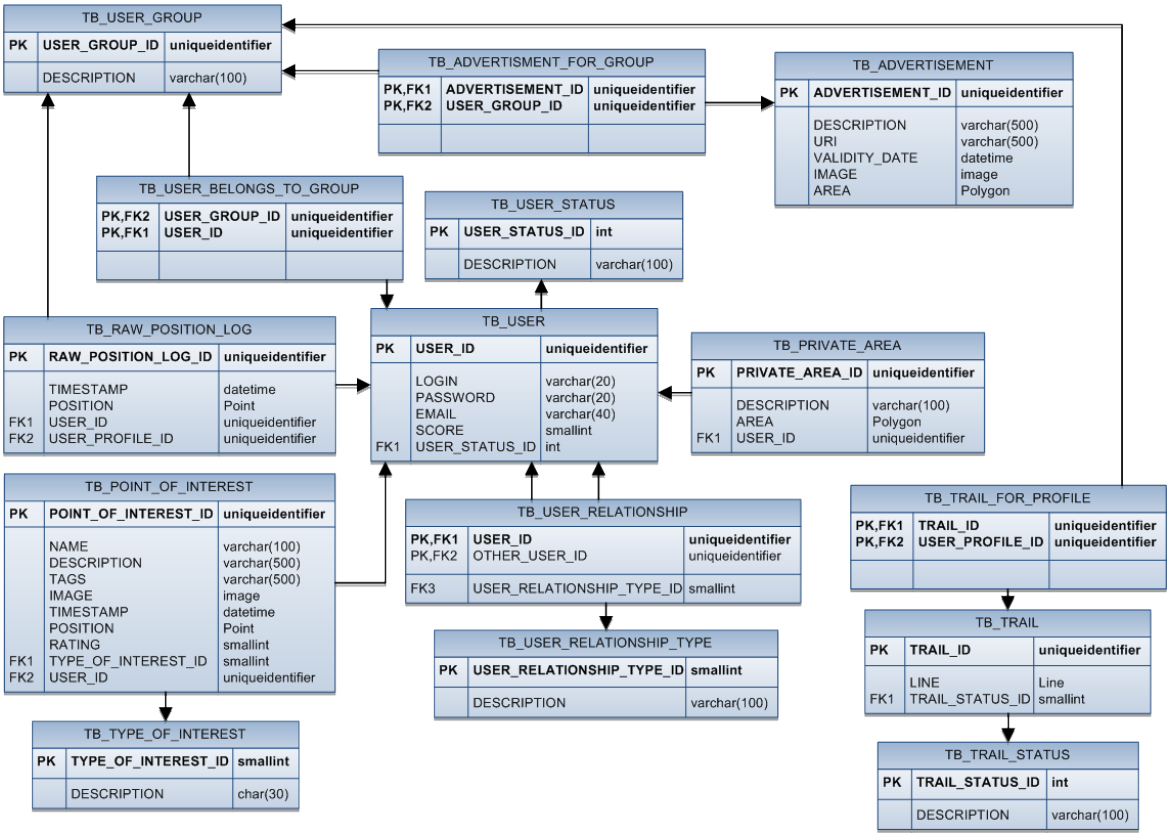

Abbildung 4.5: Objekt-relationales Datenmodell

4.3.2 Dienstfassadenschicht

Die Dienstfassade ermöglicht Konsumenten den Zugriff auf den Trailblazers-Dienst. Die Funktionalität des Dienstes wurde im so genannten *Contract-First-Ansatz* (vgl. Microsoft (2006h)) definiert. Dazu wurde zunächst mithilfe eines Datenvertrags festgelegt, wie die zwischen der Client- und Server-Anwendung zu übertragenden Daten aufgebaut sind (`DataContract`). Gemäß einer dienstorientierten Schnittstelle repräsentieren diese lediglich einen Container für fachliche Daten. Danach wurde mittels eines Nachrichtenvertrags festgelegt, wie die Nachrichten aufgebaut sind, die diese Daten übertragen (`MessageContract`). Als Beschreibungssprache wurde XML-Schema gewählt. Aus den XML-Dokumenten wurden entsprechende Klassen generiert (vgl. Anhang A.2).

Die Schnittstelle `ITrailblazersService` beschreibt alle Methoden, die der Dienst anbieten muss. Diese orientieren sich am so genannten *CRUD-Muster* (Create, Retrieve, Update, Delete), um kleine, atomare Operationen für die Entitäten zu gewährleisten. Dies kommt der z. B. in Bieberstein u. a. (2006) beschriebenen Eigenschaft für dienstorientierte Architekturen (engl.: Service Oriented Architecture, SOA) nach, mit der die Dienste beliebig verwendet und kombiniert werden können. Die Klasse `TrailblazersServiceImplementation` implementiert diese Schnittstelle und ist von der Klasse `System.Web.Services.WebService` abgeleitet. Diese beschreibt eine Basisklasse für ASP.NET Web-Services.

In der Klasse `TrailblazersServiceImplementation` wird zudem die Gültigkeit eines Dienstaufrufes überprüft. Dazu wird im Kopf der SOAP-Nachricht geprüft, ob dieser eine gültige Sitzungsinformationen beinhaltet. Die Informationen werden als ein Objekt der Klasse `SessionHeader` dem Kopf der SOAP-Nachricht hinzugefügt, wenn die Methode `LogOn(...)` der Klasse `TrailblazersServiceImplementation` aufgerufen wird. Die Informationen werden permanent zwischen Client und Server übertragen. Jede Methode des Dienstes führt dabei eine entsprechende Überprüfung durch. Weiterhin wird der Dienst an einem HTTPS-Endpunkt zur Verfügung gestellt; dadurch werden die SOAP-Nachrichten verschlüsselt übertragen.

4.3.3 Anwendungsschicht

Die Anwendungsschicht beinhaltet die Server-seitige Anwendunglogik des Trailblazers-Informationssystems, Abb. 4.3.3 zeigt das Klassendiagramm. Für jede der im vorausgegangen Abschnitt skizzierten Entitäten gibt es eine eigene Klasse, die die Geschäftslogik kapselt. Dienstaufrufe werden von der Klasse `TrailblazersService` an Methoden der Fassadenklasse `TrailblazersBusiness` weitergeleitet. Diese ruft die Methoden der entsprechenden Geschäftslogikklasse einer Entität auf. Dort werden entweder Methoden anderer Geschäftslogikklassen aufgerufen oder die Aufrufe an die Datenzugriffsschicht delegiert.

4.3.3.1 Kartenkomponente

Die Klasse `TrailblazersMapManager` vereint normale Straßenkarten mit von Nutzern generierten Informationen. Der Datenvertrag beschreibt dazu die Entität `TrailblazersMap`. Diese enthält einerseits Straßenkarten, die von einem externen Kartendienst abgerufen werden, und andererseits die zusätzlichen Informationen zur Annotierung der Karten. Abb. 4.3.3.1 zeigt dazu ein detailierteres Klassendiagramm. Der Aufbau der Entität wird in Abschn. 4.4.2 weiter aufgezeigt.

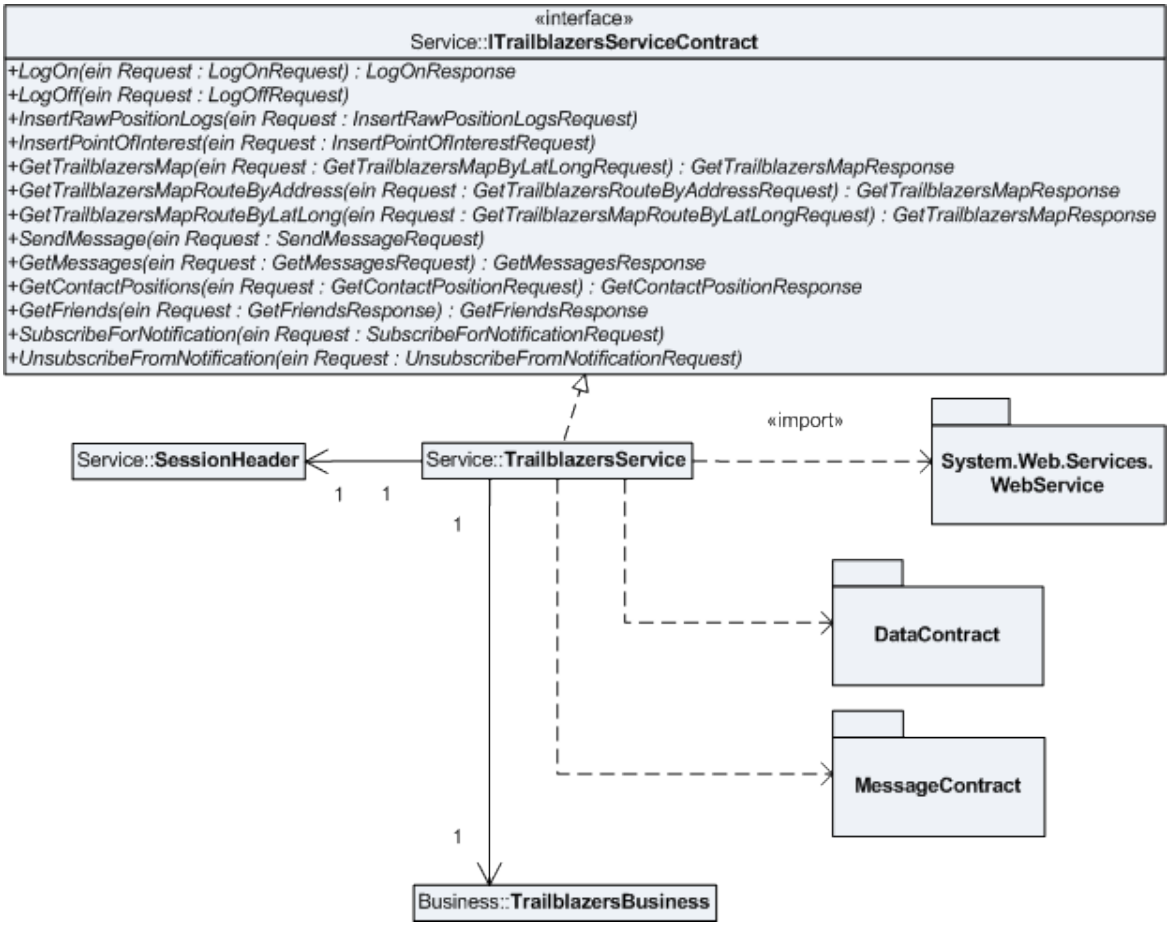

Abbildung 4.6: Klassendiagramm der Dienstfassadenschicht

Die Klasse `TrailblazersMapBusiness` enthält die Logik, um Straßenkarten mit Trailblazers-Daten zu annotieren. Mit dem Fabrikmethoden-Entwurfsmuster bietet die Klasse `MapServiceFactory` zur Laufzeit verschiedene Referenzen auf Implementierungen externer Dienste für Straßenkarten an. Dazu beschreibt die Schnittstelle `IMapService` jene Methoden, die ein derartiger Dienst anbieten muss, um in die Anwendung integriert werden zu können. Die Methode `GetLatLong(...)` bietet Geocoding, die Methode `GetAddress(...)` bietet Reverse-Geocoding auf Basis von Hausadressen sowie geografischer Länge und Breite an. Die Klassen `MapPointProxy`, `VirtualEarthProxy` und `GoogleMapsProxy` implementieren diese Schnittstelle. Sie dienen dabei, gemäß dem Stellvertreter-Entwurfsmuster, als lokale Stellvertreter für entfernte Dienste. In der Klasse `MapPointWebServiceProxy` ist der Aufruf des Microsoft MapPoint Web-Services (Microsoft (2006c)) implementiert, dazu wird eine entsprechende Referenz auf den Dienst integriert. Die beiden anderen Stellvertreterklassen implementieren den Zugriff auf Microsoft Virtual Earth bzw. Google Maps. Dazu werden, nach dem Web 2.0 Mash-Up Paradigma,

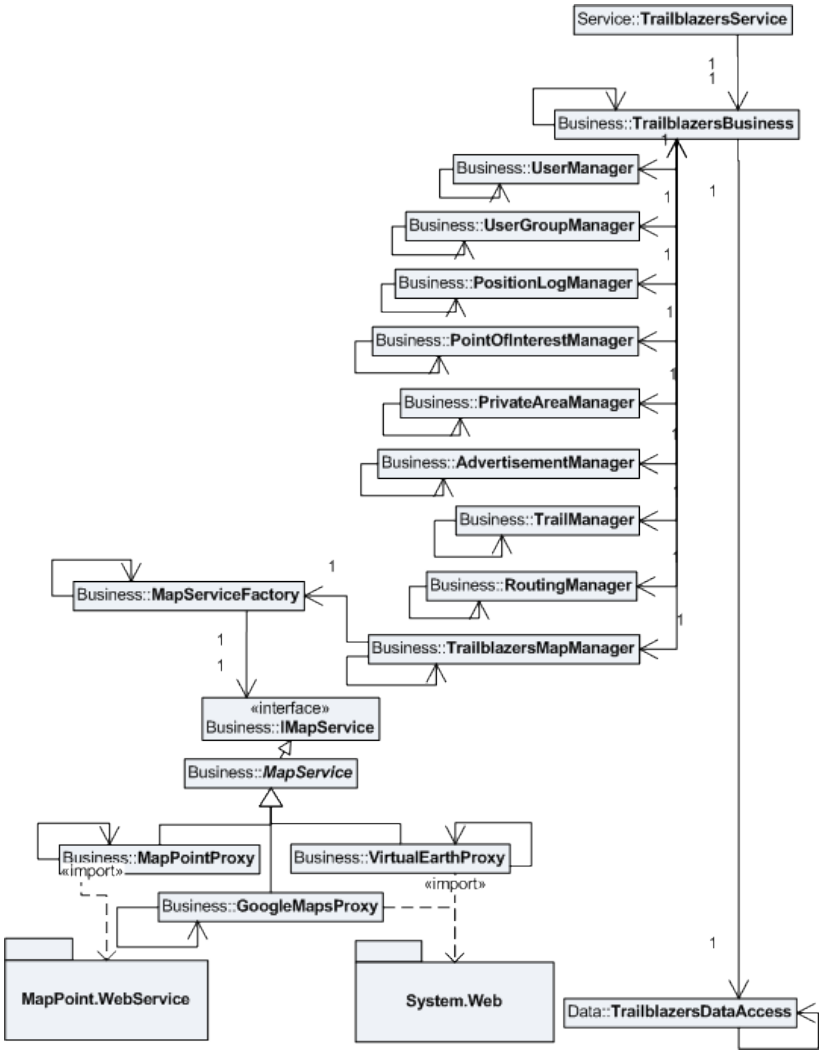

Abbildung 4.7: Klassendiagramm der Anwendungsschicht der Server-Anwendung

einfache HTTP-Aufrufe an die entsprechenden Web-Seiten gestellt und die Antworten ausgewertet. Implementierungsdetails dazu sind im nächsten Kapitel beschrieben.

Bei allen Klassen der Anwendungsschicht wird das Singleton-Entwurfsmuster genutzt. Ein Vorteil gegenüber Klassen mit statischen Methoden ist das dadurch gebotene Objektverhalten. Nur dadurch ist auch der beschriebene Austausch verschiedener Implementierungen für den externen Kartendienst zur Laufzeit möglich.

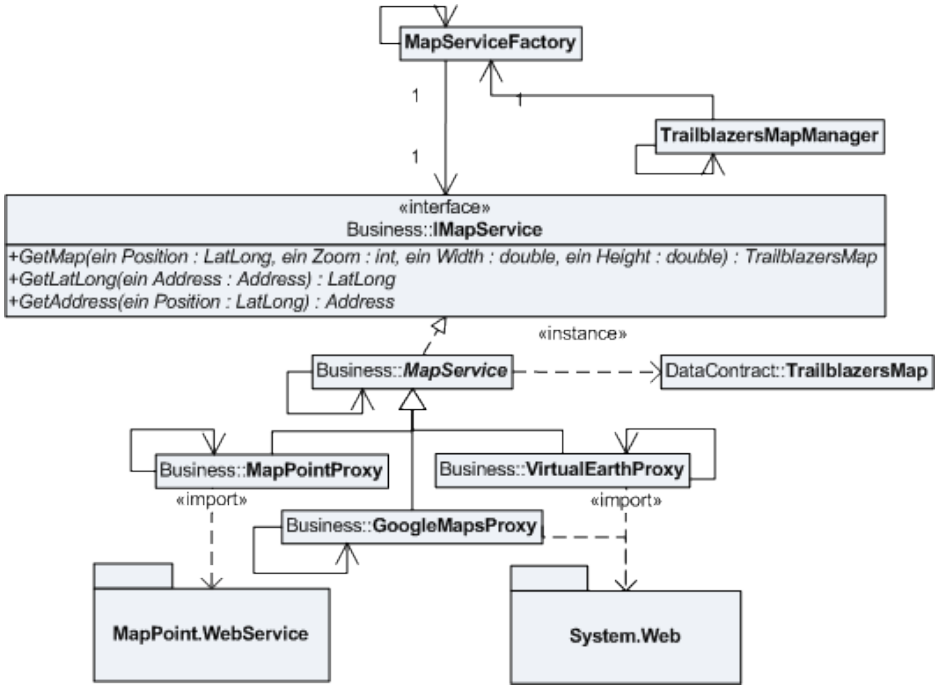

Abbildung 4.8: Klassendiagramm der Kartenkomponente

4.3.4 Datenzugriffsschicht

Die Hauptaufgabe der Datenzugriffsschicht ist es, die oben definierten Entitäten in der Datenbank zu speichern und aus ihr zu lesen. Dabei soll möglichst von der konkreten Datenbank abstrahiert werden.

Abb. 4.9 zeigt das Klassendiagramm der Datenzugriffsschicht. Der Datenzugriff auf die Datenbank ist in separaten Klassen gekapselt. Die Klassen der Anwendungsschicht nutzen jeweils nur einfache CRUD-basierte Methoden der Klasse `TrailblazersDataAccess`. Diese Klasse dient, gemäß dem Fassaden-Entwurfsmuster, der Anwendungsschicht als einziger Zugangspunkt für die Datenzugriffsschicht. Die Aufrufe werden an eine konkrete Implementierung der Schnittstelle `IDataAcessProvider` delegiert. Durch das Fabrikmethoden-Entwurfsmuster können zur Laufzeit verschiedene Implementierungen der Schnittstelle genutzt werden. Dadurch wird vom Zugriff auf die konkrete Datenbank abstrahiert, und dennoch können datenbankspezifische SQL-Dialekte genutzt werden. Die Klasse `SqlServerDataAccessProvider` implementiert diese Schnittstelle für den Zugriff auf einen Microsoft SQL Server. Für jede oben skizzierte Entität wurde eine eigene Klasse entworfen, die die Zuordnung von objektorientierten Datentypen auf objekt-relationale Datentypen der Datenbank (Object-Relational-Mapping), sowie den Zugriff auf die Datenbank

implementiert. Die Klasse `SqlServerDataAccessProvider` greift auf die Methoden dieser Klassen entsprechend zu.

Bei allen Klassen der Datenzugriffsschicht wird das Singleton-Entwurfsmuster eingesetzt, um den beschriebenen Austausch verschiedener Implementierungen für den Datenzugriff zur Laufzeit zu ermöglichen.

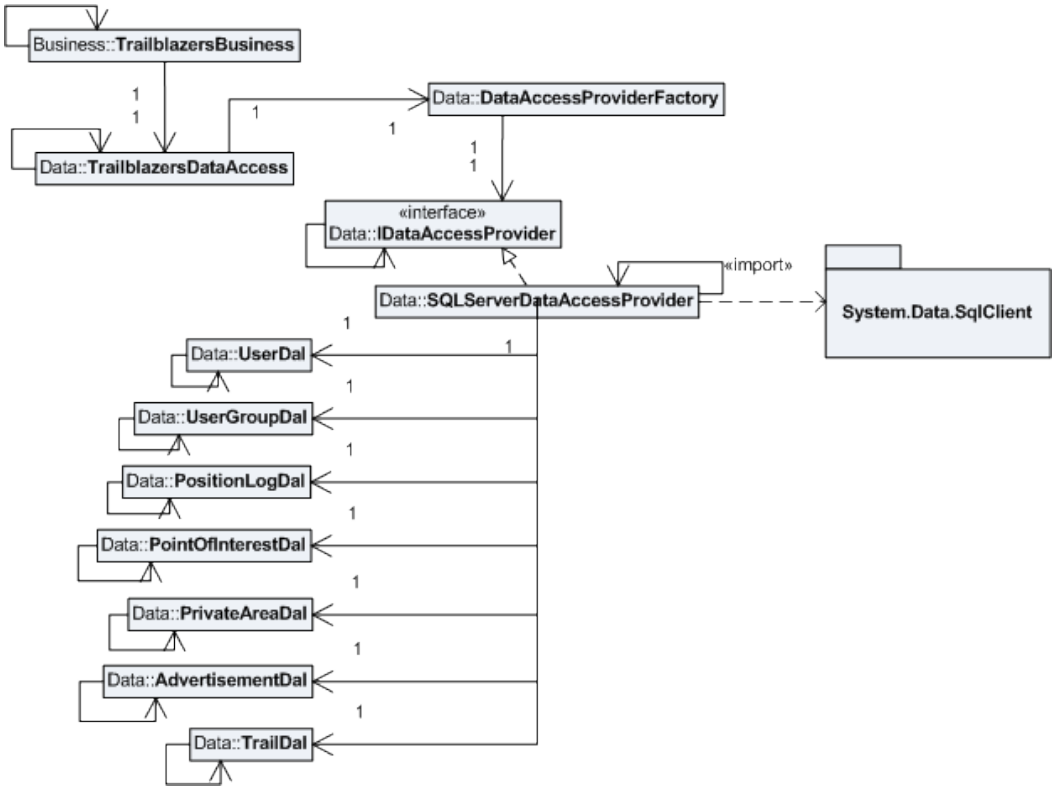

Abbildung 4.9: Klassendiagramm der Datenzugriffsschicht

Vorhandene Persistenz-Rahmenwerke wie NHibernate (Hibernate (2007)) für .NET können nicht verwendet werden, da diese zur Zeit keine Unterstützung für räumliche Datentypen und Anfragesprachen bieten. Implementierungsdetails für den Datenzugriff werden im folgenden Kapitel gesondert besprochen.

4.4 Mobile Client-Anwendung

In diesem Abschnitt wird die Mikroarchitektur der mobilen Client-Anwendung detailliert beschrieben. Beim Entwurf wurden ebenfalls Entwurfsmuster eingesetzt, um bekannte Entwurfsprobleme zu lösen.

4.4.1 Präsentationsschicht

Die Präsentationsschicht repräsentiert die Schnittstelle zwischen Nutzer und mobilem Informationssystem und hat zwei grundlegende Aufgaben: Zum einen die Darstellung der fachlichen Entitäten aus der Anwendungsschicht, zum anderen die Steuerung der Dialogabfolge. Diese beiden Aufgaben werden voneinander getrennt, um eine lose Kopplung sowie die Trennung der Verantwortlichkeiten zu erreichen. Die Umsetzung erfolgt mit dem *Model-View-Controller (MVC)-Paradigma* (vgl. z. B. Starke (2005)). Abb. 4.10 zeigt das Klassendiagramm der Präsentationschicht, das im Folgenden weiter beschrieben wird.

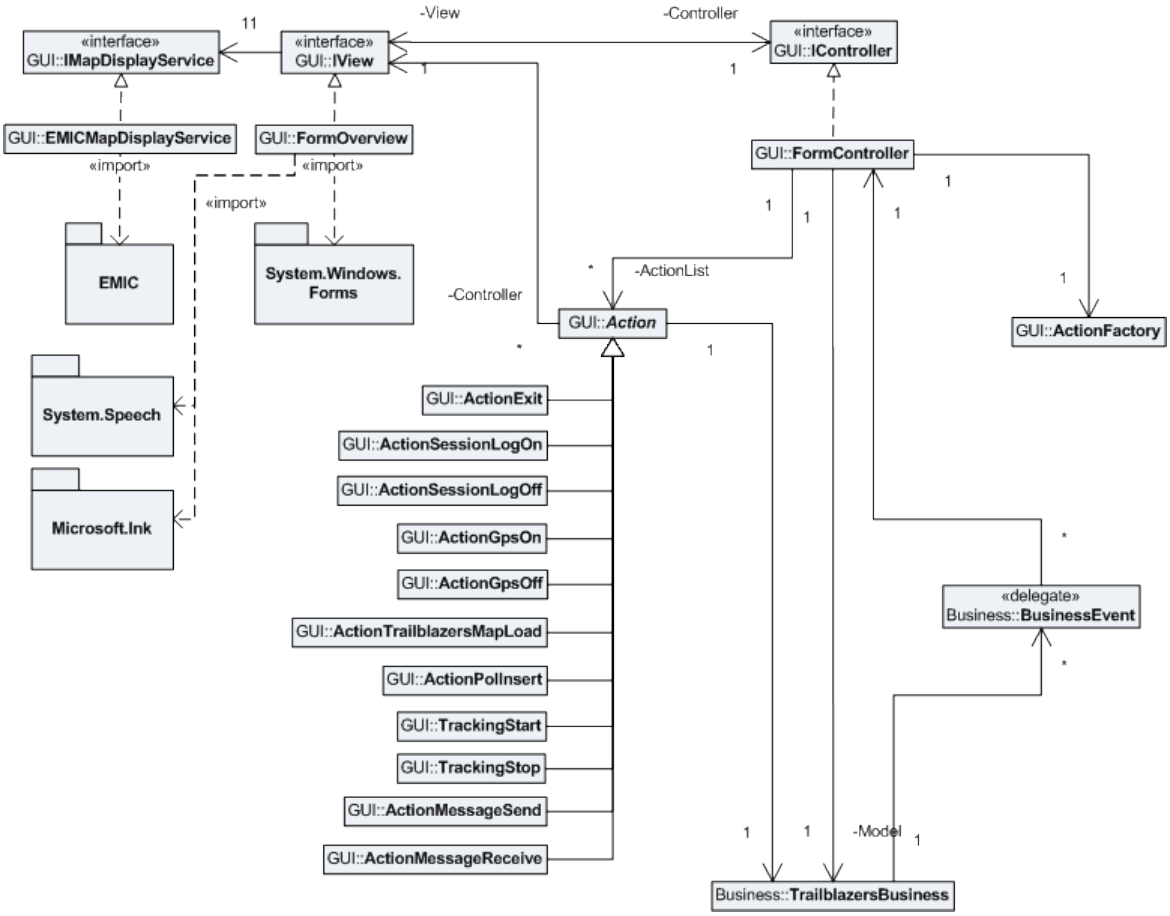

Abbildung 4.10: Klassendiagramm der Präsentationsschicht

4.4.1.1 Darstellung

Die Darstellung fachlicher Daten sollte nach Möglichkeit mit Standardbausteinen der jeweiligen Plattformen erfolgen. Für die verschiedenen Varianten der .NET-Plattformen existieren

zahlreiche Bausteine. Ein Grundmenge, wie z. B. Eingabefelder, Textfelder und Knöpfe (engl.: button) ist bei allen Plattformen vorhanden. Dennoch sind die Klassen, die die Darstellung implementieren, für die einzelnen Plattformen, und eventuell für verschiedene mobile Geräte, anzupassen. Durch die Nutzung des MVP-Paradigmas entfällt die Anpassung lediglich auf diese Klassen, die fachlichen Abläufe müssen nicht redundant implementiert werden. In Abschn. 4.4.1.4 werden Besonderheiten bei der Visualisierung von Kartendaten diskutiert. Die Klasse `FormOverview` stellt das Hauptformular der Anwendung dar. Die Klasse implementiert die Schnittstelle `IView`. Die Controller-Klassen referenzieren lediglich diese Schnittstelle, damit unbenötigte Details von der Formularklasse verborgen bleiben.

4.4.1.2 Dialogkontrolle

Ein durch den Nutzer ausgelöstes Ereignis führt häufig zu einer Client-Server-Kommunikation. Das Ergebnis wird anschließend dem Nutzer angezeigt. Dieser Kontrollfluss der Benutzerschnittstelle soll asynchron erfolgen, wie in den nichtfunktionalen Anforderungen (vgl. Abschn. 3.4.1) gefordert. Zur Umsetzung wird das Befehl-Entwurfsmuster genutzt, um den Kontrollfluss an einer zentralen Stelle und in separaten Klassen zu implementieren. Dadurch werden Redundanzen vermieden und die Darstellung von der Dialogkontrolle entkoppelt. Das befehlsauslösende Objekt, z. B. ein durch den Nutzer ausgelöstes Ereignis in der Darstellungssicht, und das befehlsausführende Objekt, z. B. die Verarbeitung im Model, werden dabei voneinander getrennt. Die Kapselung der Befehle als eigene Objekte ermöglicht zudem deren asynchrone Ausführung, sowie deren Speicherung in Befehlslisten, um die Ausführung einzelner Befehle abbrechen zu können. Da die Verarbeitung asynchron erfolgt, werden durch die Befehle zudem bestimmte Aktionen auf den Sichten ausgeführt. Z.B. wird der Knopf, der durch den Nutzer gedrückt wurde, für die Zeit der Ausführung deaktiviert. Die Dialogsteuerung wird mit der Klasse `FormController` umgesetzt; diese implementiert die Schnittstelle `IController`, die der Darstellungssicht bekannt ist. Die auszuführenden Aktionen werden in einzelnen, konkreten Klassen der abstrakten Klasse `Action` umgesetzt. Die Steuerung delegiert die Ausführung an die jeweilige konkrete Klasse. Zur Erzeugung von Objekten der entsprechenden konkreten Klasse dient die `ActionFactory`. Diese kapselt, mittels des Fabrikmethoden-Entwurfsmusters, die Erzeugung der Objekte in einer separaten Klasse.

4.4.1.3 Schnittstelle zur Anwendungsschicht

Die Interaktion zwischen der Präsentations- und der Anwendungsschicht findet über eine klar definierte Schnittstelle statt. Die `Aktion`-Klassen der Dialogsteuerung greifen mithilfe des Fassaden-Entwurfsmusters über eine schmale Schnittstelle auf die Anwendungsschicht

zu, um dort die asynchron laufende Verarbeitung anzustoßen. Die Anwendungsschicht informiert die Präsentationsschicht mittels Beobachter-Entwurfsmuster, sobald die Verarbeitung abgeschlossen ist. Die Anwendungsschicht nutzt somit keine konkreten Objekte der Präsentationsschicht; die Schichtentrennung ist gewährleistet. Weiterhin implementiert jede fachliche Entitätsklasse eine Schnittstelle. Die Darstellungsklassen greifen für die Anzeige von Daten nur über die Schnittstellen auf eine Entität zu; die konkrete Implementierung der Klasse bleibt verborgen.

Das Sequenzdiagramm in Abb. 4.11 zeigt das Laden einer Karte durch eine Anfrage des Nutzers und soll die beschriebene Funktionsweise der einzelnen Klassen verdeutlichen.

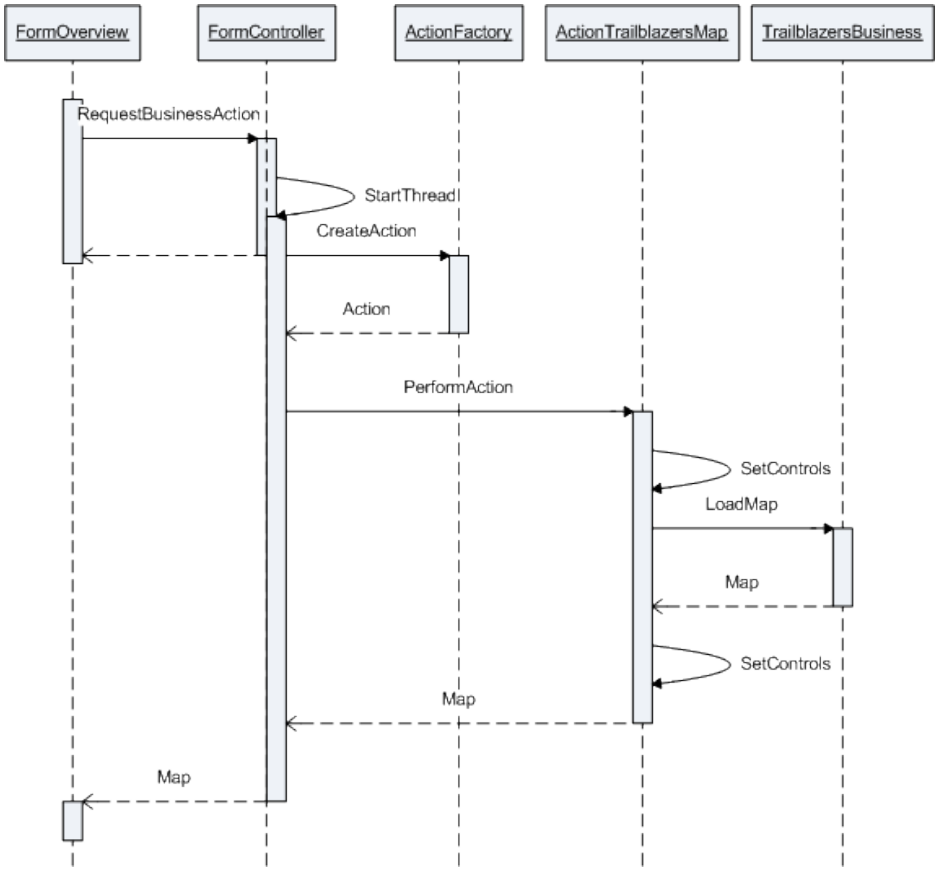

Abbildung 4.11: Sequenzdiagramm der Präsentationsschicht beim Laden einer Karte

4.4.1.4 Kartendarstellungskomponente

Die Anzeige der Routeninformationen für den Anwendungsfall barrierefreie Navigation ist ein elementarer Bestandteil der Anwendung. Die Software-Komponente für die Anzeige der

Kartendaten wurde in dieser Arbeit nicht entwickelt, vielmehr wurden vorhandene Komponenten evaluiert. Wie in Abschn. 4.3.3.1 beschrieben, werden auf dem Rasterformat basierende Straßenkarten eingesetzt. Diese werden mit zusätzlichen Trailblazers-Informationen annotiert. So muss die Karten-Software einerseits eine Karte mit Bezug zur aktuellen Nutzerposition anzeigen, außerdem sollen Trails, die aktuelle Route und Orte von Interesse angezeigt werden können. Des weiteren müssen Areale für ortsbezogene Werbung auf die Karte abgebildet und ein entsprechendes Ereignis ausgelöst werden, sobald sich die Position des Nutzers innerhalb dieses Areals befindet. Um die konkrete Implementierung der Komponente von der Sicht zu trennen, wird das Adapter-Entwurfsmuster eingesetzt. Die Darstellungssicht referenziert die Schnittstelle `IMapDisplayService`. Adapter-Klassen können diese Schnittstelle implementieren und damit die jeweils verwendete Komponente für die Kartendarstellung in die Darstellungssicht integrieren.

Für den im Rahmen dieser Arbeit entwickelten Prototyp wurde das EMIC Location und Mapping Framework (EMIC (2007)) eingesetzt. Die Adapter-Klasse `EMICMapDisplayService` integriert dieses in die Anwendung. Abb. 4.12 zeigt das Klassendiagramm. Ein Austausch der Komponente für die Kartendarstellung würde lediglich zur Einführung einer neuen Adapterklasse, zu deren Implementierung der Schnittstelle und zur Anpassung an die etwaige neue Komponente führen. Die Schnittstellen `ITrailblazersMap`, `ITrailblazersRoute` und `ILocationInformation` werden im folgenden Abschnitt vorgestellt.

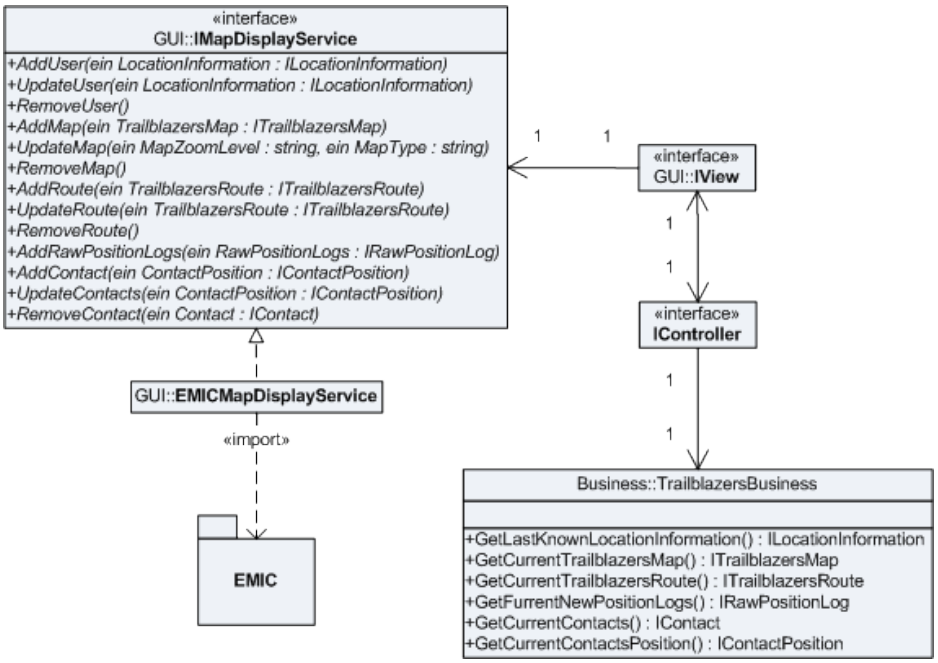

Abbildung 4.12: Klassendiagramm der Kartendarstellung

4.4.2 Anwendungsschicht

Die Anwendungsschicht enthält die Client-seitige Anwendungslogik des mobilen Informationssystems. Für jede im Architekturentwurf skizzierte Komponente wurde eine eigene `Business`-Klasse definiert. Diese Klassen bieten Operationen auf Entitäten an, die vom Trailblazers-Dienst geladen wurden, indem sie den aktuellen Zustand der Anwendung halten. Für jede der Entitäten wurde nach dem Hüllen-Entwurfsmuster eine Geschäftsklasse (Business-Object, BO) entworfen, die zusätzliche Operationen bereitstellt. Dazu implementieren die Geschäftsklassen Schnittstellen, durch die in allen drei Schichten der Client-Anwendung auf diese Klassen zugegriffen werden kann. Klassen, die eine räumliche Entität kapseln, sind von den Klassen `Point`, `Line` und `Polygon` der externen Komponente `NetTopolgySuite` abgeleitet, um geografische Operationen zu ermöglichen. Abb. 4.13 zeigt das Klassendiagramm der Anwendungsschicht.

4.4.2.1 Kartenkomponente

Die Klasse `TrailblazersMapBusiness` kapselt die Anwendungslogik für Trailblazers-Karten, die dem Nutzer angezeigt werden. Dazu hält sie eine Referenz auf die aktuell vom Dienst geladene Karte. Die Schnittstelle `ITrailblazersMap` ist ein Vertrag für die Geschäftsobjekt-Klasse `BOTrailblazersMap`. Diese Klasse kapselt die Dienstentität `TrailblazersMap`. Eine `ITrailblazersMap` besteht aus einer oder mehreren Straßenkarten (`IMap`), keinem oder mehreren Arealen für ortsbezogene Werbung (`IAdvertisement`), keinem oder mehreren Orten von Interesse (`IPointOfInterest`), sowie einer Start- und einer Zieladresse (`IAddress`).

4.4.2.2 Routenkomponente

Die Klasse `TrailblazersRoutingBusiness` beinhaltet die Anwendunsglogik der Routenkomponente. Die Berechnung einer Route wird innerhalb der Client-Anwendung durchgeführt (vgl. Abschn. 3.3.1.1).

Eine Route ist ein gerichteter Teilgraf $H = (V', E')$ eines Grafen $G = (V, E)$, wobei V (Knoten, engl.: vertex) die Menge aller Knoten und E (Kanten, engl.: edge) die Menge aller ein- oder zweielementiger Teilmengen von V ist. Der Graf entspricht einem Pfadnetzwerk, bestehend aus Trail-Entitäten. Die Berechnung einer Route basiert auf bekannten Algorithmen der Grafentheorie. Unter anderem in Shekhar und Chawla (2002) werden der *Dijkstra*- und der *A-Stern-Algorithmus* für diesen Zweck vorgestellt.

Mittels Strategie-Entwurfsmuster wird mit der Schnittstelle `IRoutingAlgorithm` eine Familie derartiger Algorithmen definiert. Die Routenberechnung findet in einer konkreten

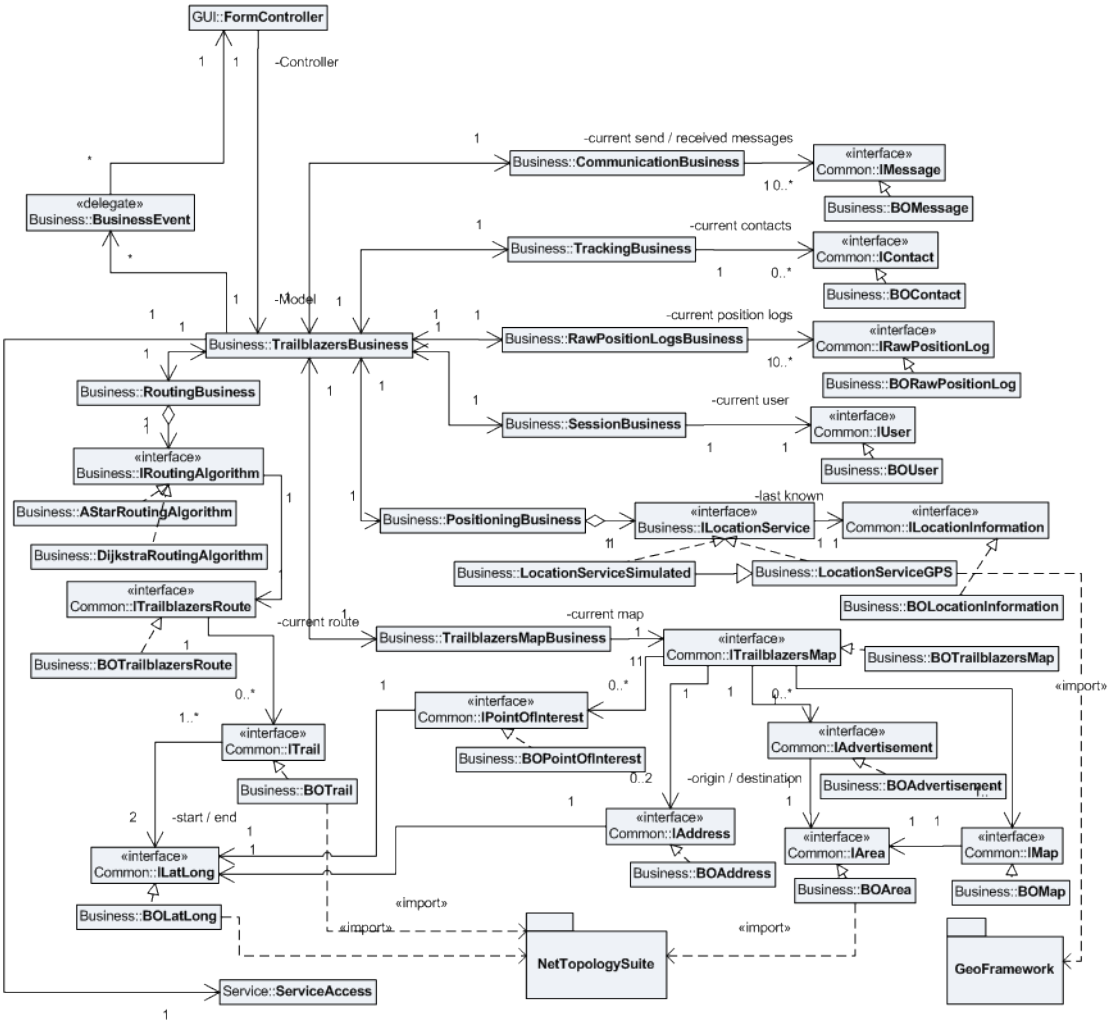

Abbildung 4.13: Klassendiagramm der Anwendungsschicht der mobilen Client-Anwendung

Implementierung `AStarRoutingAlgorithm` bzw. `DijkstraRoutingalgorithm` der Schnittstelle statt. Dies erhöht die Flexibilität und ermöglicht den Austausch des Algorithmus zur Laufzeit. Abb. 4.14 zeigt das zugehörige Klassendiagramm.

Während der Navigation wird die Route permanent aktualisiert, um ein Verlassen der selbigen durch den Nutzer zu berücksichtigen. Die Schnittstelle `ITrailblazersRoute` zeigt die Route in Form eines aus `ITrail` bestehenden Feldes und bietet auch Informationen über die selbige an. Die Methode `GetRemainingDistance(...)` gibt die verbleibende Distanz von der Position des Nutzers zum Zielpunkt an. Die Distanz d zwischen zwei geografischen Koordinaten a und b wird dabei mit der Formel

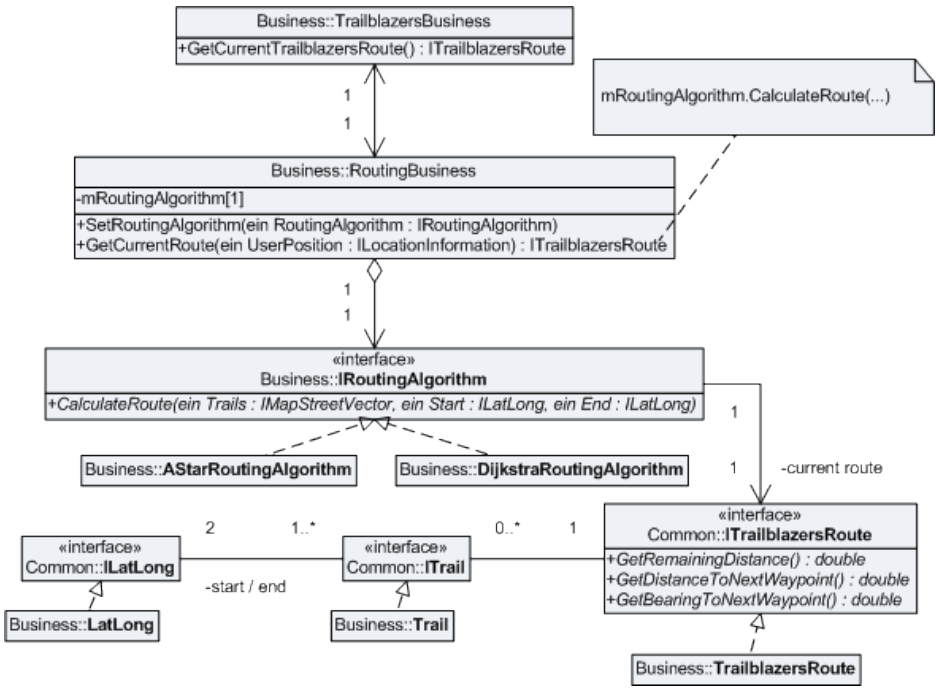

Abbildung 4.14: Klassendiagramm der Routenkomponente

$$d = r \cdot \arccos(\sin lat_a \cdot \sin lat_b + \cos lat_a \cdot \cos lat_b \cdot \cos(long_b - long_a)) \tag{4.1}$$

berechnet, wobei r den Erdradius in Kilometern beschreibt. Die Methode `GetDistanceToNextWaypoint(...)` gibt die Entfernung zum nächsten Wegpunkt zurück. Ein Wegpunkt ist dabei der Knoten, der zur Menge V' gehört und der aktuellen Position des Nutzers am nächsten liegt. Die Methode `GetBearingToNextWaypoint(...)` gibt, in Abhängigkeit der Bewegungsrichtung des Nutzers, die Himmelsrichtung zu diesem Punkt zurück. Dabei sind zwei unterschiedliche Fälle möglich: Der Nutzer befindet sich auf der Route, oder der Nutzer hat die Route verlassen. Befindet er sich auf der Route, wird die Richtung zum nächsten Wegpunkt angezeigt (Abb. 4.15 (a)). Sobald er sich dem Ende eines Pfadvektors nähert, wird das Ende des nächsten Pfadvektors der Route als nächster Wegpunkt genommen (Abb. 4.15 (b)). Hat der Nutzer die Route verlassen, wird derjenige Wegpunkt als Referenz genommen, der am dichtesten an der Position des Nutzers liegt (Abb. 4.15 (c)). Entsteht durch das Verlassen eine neue, kürzere Route, wird diese angezeigt und der Nutzer zum Ziel geführt (Abb. 4.15 (d).

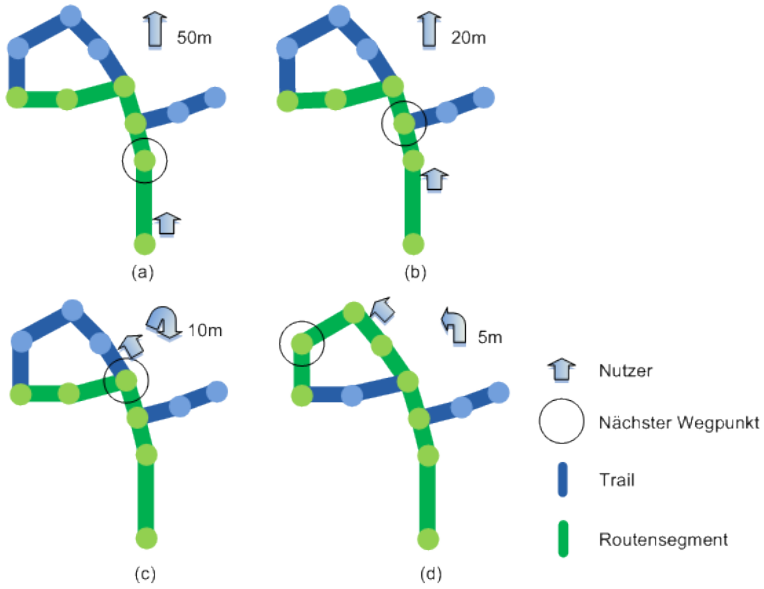

Abbildung 4.15: Routeninformationen der Routenkomponente

4.4.2.3 Positionskomponente

Die Klasse `PositioningBusiness` implementiert die Bestimmung der aktuellen Position des Nutzers. Die Positionsbestimmung wird in eigenen Klassen gekapselt. Durch Nutzung des Strategie-Entwurfsmusters kann das System zur Positionsbestimmung während der Laufzeit ausgetauscht werden. Eine Klasse zur Bestimmung der Position muss die Schnittstelle `ILocationService` implementieren. Die Klasse `LocationServiceGPS` implementiert diese für die Positionsbestimmung mittels GPS. Dazu dient diese Klasse gemäß dem Adapter-Entwurfsmuster als Adapter für die externe Komponente `GeoFramework`. Die Klasse `LocationServiceSimulated` erweitert diese Klasse um die Möglichkeit, Positionskoordinaten aus einer Textdatei einzulesen, um die Positionsbestimmung zu simulieren. Weitere Implementierungen, z. B. für die Positionsbestimmung via WLAN, können integriert werden.

4.4.2.4 Tracking-Komponente und Position-Log-Komponente

Die Klasse `PositionLogBusiness` implementiert die Anwendungslogik für die Aufzeichnung und Weiterleitung der eigenen Positionsdaten an den Trailblazers-Dienst; die Klasse `TrackingBusiness` implementiert die Logik, mit der Positionsdaten anderer Nutzer vom Dienst abgefragt werden können. Wie in Abschn. 2.4.4 erläutert, soll die Aktualisierung der Daten nicht permanent durchgeführt werden, um die Netzbelastung gering zu halten.

Entsprechend werden die eigenen Positionsdaten (`IPostionLog`) zwar kontinuierlich aufgezeichnet, jedoch nur in größeren Abständen komplett an den Trailblazers-Dienst übertragen. Dort kann dann durch die Vielzahl der Daten eine zukünftige Position abgeleitet werden. In der Klasse `TrackingBusiness` wird die abgeleitete Position eines anderen Nutzers berücksichtigt. Zusätzlich wäre hier auch denkbar, die Routeninformationen eines anderen Nutzers einmalig zu übertragen und ebenfalls zur abgeleiteten Positionsbestimmung heranzuziehen.

In der Klasse `TrackingBusiness` werden auch die in Abschn. 3.2.5 definierten Regeln zur Sicherung der eigenen Privatsphäre umgesetzt.

4.4.3 Dienstzugriffsschicht

Die Dienstzugriffsschicht ist die Client-seitige Schnittstelle zwischen Client- und Server-Anwendung. Die Anwendungsschicht greift über die Fassadenklasse `ServiceAccess` auf den Trailblazers-Dienst zu, um Daten an den Dienst zu übermitteln, oder um Daten vom Dienst zu erhalten. Wie oben erwähnt, wird dabei, je nach Anwendungszustand, auf den entfernten oder den lokalen Dienst zugegriffen. Der Anwendungszustand wird durch die Klasse `ConnectionMonitor` bestimmt, die permanent die verfügbaren Netzwerkverbindungen überwacht. Verwaltet werden die Netzwerkverbindungen in der Klasse `ConnectionCollection`; die abstrakte Klasse `Connection` kann von konkreten Klassen implementiert werden, die verschiedene Arten von Verbindungen überwachen. Die Klasse `NICConnection` kontrolliert eine Netzwerkverbindung zum Internet, die Verbindung `CellConnection` überwacht eine Mobilfunkverbindung. Wenn sich der Verbindungsstatus ändert, werden angemeldete Beobachter mittels Beobachter-Entwurfsmuster informiert. Diese können dann entsprechende Maßnahmen durchführen.

Die Schnittstelle für den Zugriff auf den Trailblazers-Dienst wird mit `IServiceAccessProvider` definiert. Die Implementierung `CacheServiceAccessProvider` bietet Zugriff auf den lokalen, die Implementierung `WebServiceAccessProvider` bietet Zugriff auf den entfernten Dienst. Mittels Dekorierer-Entwurfsmuster wird die Funktionalität letzterer Klasse mit der Funktionalität der ersteren Klasse transparent erweitert.

Die Klasse `WebServiceAccessProvider` implementiert zudem die Schnittstelle `IServiceListener`. Dadurch kann die Client-Anwendung SOAP-Nachrichten von der Server-Anwendung empfangen.

Das Klassendigramm der Dienstzugriffssschicht ist in Abb. 4.16 gezeigt. Der entfernte und der lokale Dienstzugriff, sowie die Zwischenspeicherung von Dienstaufrufen und Daten, werden in den folgenden Unterabschnitten weiter beschrieben.

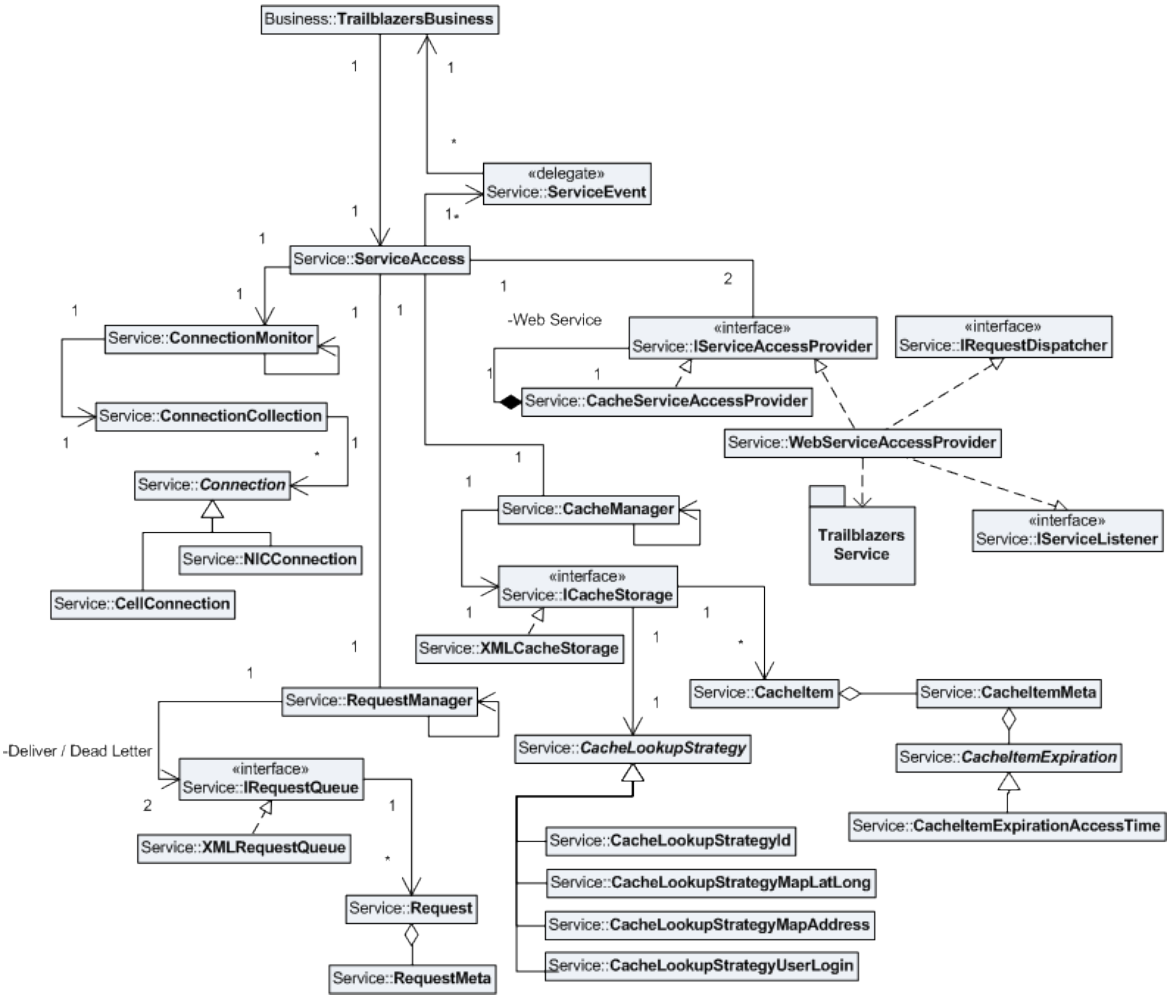

Abbildung 4.16: Klassendiagramm der Dienstzugriffsschicht

4.4.3.1 Lokale Dienstaufrufe

Die mobile Anwendung ist nicht permanent mit dem entfernten Dienst verbunden, sie sollte dennoch eine gewisse Funktionalität für den Nutzer zur Verfügung stellen. Für ein derartiges Offline-Szenario ist ein asynchrones Kommunikationsmodell gut geeignet. Wie oben beschrieben, steht dieses jedoch nicht zur Verfügung. Daher wird mit diesem Entwurf ein asynchrones Kommunikationmodell mittels synchroner Kommunikation eingeführt. Zur Umsetzung ist es erforderlich, die Dienstaufrufe und die von dem Dienst empfangenen Daten auf dem mobilen Gerät zwischenzuspeichern.

Wenn keine Verbindung zum entfernten Dienst besteht, werden die Aufrufe in einer Warteschlange (engl.: queue) zwischengespeichert und bei bestehender Verbindung übertragen.

Dieses so genannte *Store-And-Forward*-Prinzip ist elementar für ein Offline-Szenario. Es entspricht der oben skizzierten Kommunikation mittels Nachrichtenwarteschlange.

Bei Dienstaufrufen ist grundsätzlichen zu unterscheiden, ob Daten an den Dienst übertragen oder Daten vom Dienst abgefragt werden. Bei Übertragungen an den Dienst kann die Anwendung im Offline-Betrieb normal weiterlaufen. Beim Abfragen von Daten muss die Anwendung entsprechend entworfen sein, um die gewünschte Funktionalität bereitzustellen. Dazu wird ein Zwischenspeicher (engl.: cache) eingeführt, der Daten des entfernten Dienstes lokal speichert. Bei derartigen Anfragen wird zunächst geprüft, ob die Daten bereits im Zwischenspeicher vorhanden sind und ggfs. aus diesem ausgelesen. Das Aktivitätsdiagramm in Abb. 4.17 verdeutlicht diese Vorgänge.

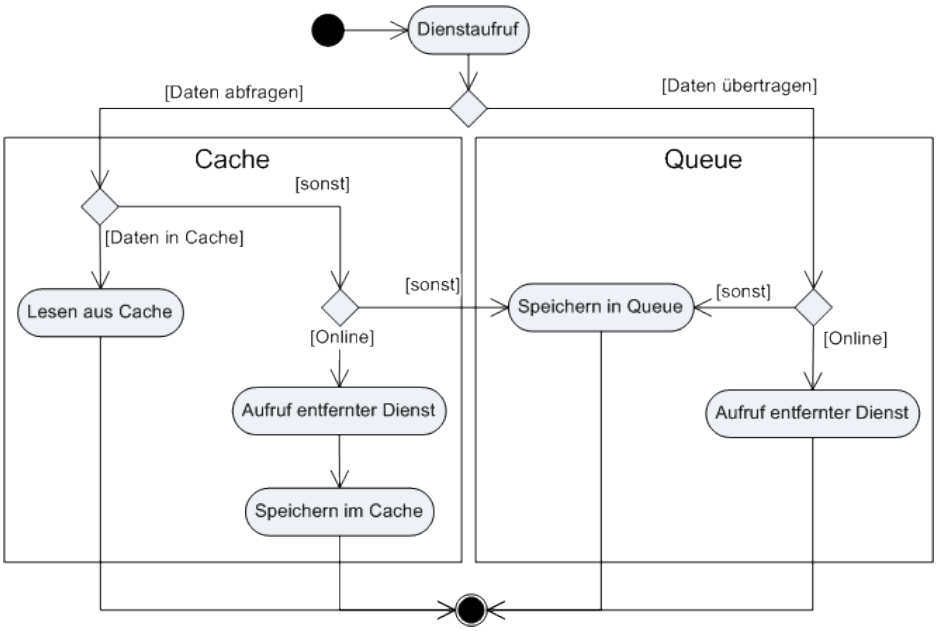

Abbildung 4.17: Aktivitätsdigramm für Dienstaufrufe

4.4.3.2 Zwischenspeicherung von Dienstentitäten

Im Folgenden werden Dienstaufrufe beschrieben, die Daten vom Trailblazers-Dienst abfragen und lokal zwischenspeichern. Die Daten entsprechen den in Abschn. 4.3.2 vorgestellten Dienstentitäten. In den Anwendungsfällen wurden entsprechende Funktionalitäten und damit die Entitäten identifiziert, die ein Offline-Szenario gewährleisten.

Die Klasse `ServieAccess` greift über die Schnittstelle `IServiceAccessProvider` auf die gewünschte Funktionalität des Dienstes zu. In der Klasse `CacheServiceAccess-Provider` werden die Zugriffe auf den entfernten bzw. lokalen Dienst berücksichtigt.

In jeder Dienstmethode der Klasse `CacheServiceProvider` wird der Dienstaufruf zunächst an die Klasse `CacheManager` weitergeleitet. Diese Klasse implementiert das Strategie-Entwurfsmuster für verschiedene Vergleichstrategien, die auf den zwischengespeicherten Daten ausgeführt werden können. Die Strategien können zur Laufzeit dynamisch gesetzt werden; so implementiert z. B. die Klasse `CacheLookupStrategyMapLatLong` eine Vergleichstrategie, die für die Suche im Zwischenspeicher nach einer Kartenentität mit Längen- und Breitengrad als Eingabe- bzw. Vergleichsparameter geeignet ist. Dazu wird mit der Methode `Lookup(...)` geprüft, ob die gewünschte Dienstentität im Zwischenspeicher vorhanden ist und ggfs. aus diesem ausgelesen. Die Prüfung wird dabei an die Schnittstelle `ICacheStorage` delegiert, welche die Funktionalität für den Zwischenspeicher definiert. Ist die Entität nicht vorhanden und eine Verbindung zum entfernten Dienst möglich, wird der Aufruf an die dekorierte Klasse von `CacheServiceAccessProvider` delegiert. Zur Feststellung, ob eine Verbindung verfügbar ist, kann die Eigenschaft `IsOnline` der Klasse `ConnectionMonitor` abgefragt werden. Nach erfolgtem Aufruf wird das Ergebnis entgegengenommen und dem Zwischenspeicher durch Ausführung der Methode `Add(...)` hinzugefügt. Ist der entfernte Dienst nicht verfügbar, wird der Dienstaufruf in der Warteschlange zwischengespeichert. Die Anwendung muss dann entsprechend reagieren und den Nutzer informieren; in diesem Fall ist die gewünschte Funktionalität eventuell nicht gewährleistet.

Die zwischengespeicherten Dienstentitäten werden in Objekten der Klasse `CacheItem` gekapselt. Dabei werden Metainformationen, gekennzeichnet durch die Klasse `CacheItemMeta`, hinzugefügt. Diese beschreiben die Priorität der Entiät und ihre Gültigkeitsdauer für den Zwischenspeicher. Letztere wird in die abstrakte Klasse `CacheItemExpiration` ausgelagert. Die Klasse `CacheItemExpirationAccessTime` implementiert, dass eine Entität im Zwischenspeicher nach einem definierten Zeitraum nach letztem Zugriff ihre Gültigkeit verliert.

Daten im Zwischenspeicher, die ihre Gültigkeit verloren haben, können über die Methode `Update` der Klasse `CacheManager` aus dem Cache entfernt werden. Zusätzlich ist hier ein Aktualisierungsmechanismus möglich, der Daten im Zwischenspeicher mit Daten vom entfernten Dienst aktualisiert.

Die Funktionalität für den Zwischenspeicher wird in der Schnittstelle `ICacheStorage` definiert und von der Klasse `XMLCacheStorage` implementiert. Dabei werden Objekte der Klasse `CacheItem` in einzelne XML-Dateien serialisiert und auf dem Dateisystem gespeichert. Weitere Implementierungen, die z. B. eine mobile Datenbank als Persistenzspeicher nutzen, wären denkbar.

Das Sequenzdiagramm in Abb. 4.18 soll diese Vorgänge verdeutlichen. Dazu wird das oben angeführte Beispiel zum Laden einer Karte vom entfernten Dienst fortgesetzt. Abb. 4.19 zeigt das zugehörige, detailliertere Klassendiagramm.

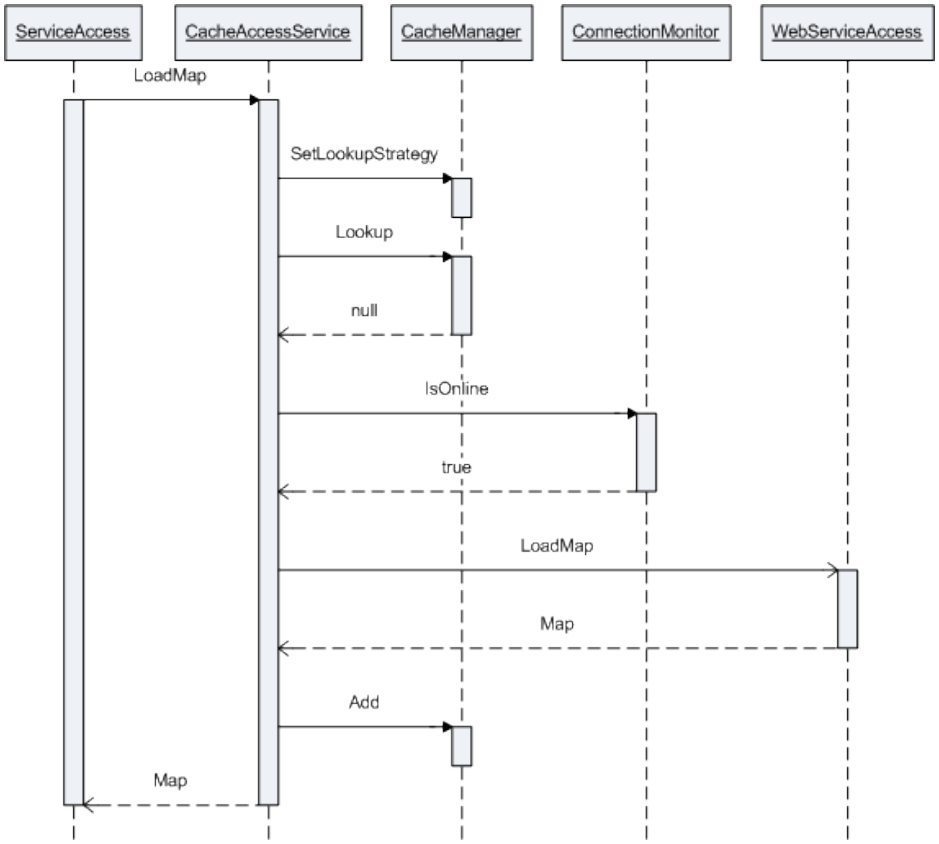

Abbildung 4.18: Sequenzdiagramm der Dienstzugriffsschicht beim Laden einer Karte

4.4.3.3 Zwischenspeicherung von Dienstaufrufen

Wenn Aufrufe an den entfernten Dienst aufgrund fehlender Verbindung nicht durchgeführt werden können, werden diese auf dem mobilen Gerät zwischengespeichert. Wie oben erwähnt, wird in der entsprechenden Dienstmethode der Klasse `CacheServiceProvider` entschieden, ob der lokale oder der entfernte Dienst aufgerufen wird. Bei lokalen Aufrufen wird der Aufruf in einem Objekt der Klasse `Request` gespeichert. Die Attribute `ServiceMethodName` und `ServiceMethodParameters` geben dabei den Namen der Dienstmethode sowie die Parameter an, die bei dem Aufruf genutzt werden sollen. Das Request-Objekt wird dann mithilfe der Methode `Enqueue(...)` der Klasse `RequestManager` in der lokalen Warteschlange gespeichert. Die Warteschlange wird durch die Schnittstelle `IRequestQueue` definiert. Dessen Implementierung `XMLRequestQueue` serialisiert die Objekte und speichert so die Dienstaufrufe in einzelnen XML-Dateien.

Durch die Methode `DispatchAllPendingReuests()` der Klasse `RequestManager`

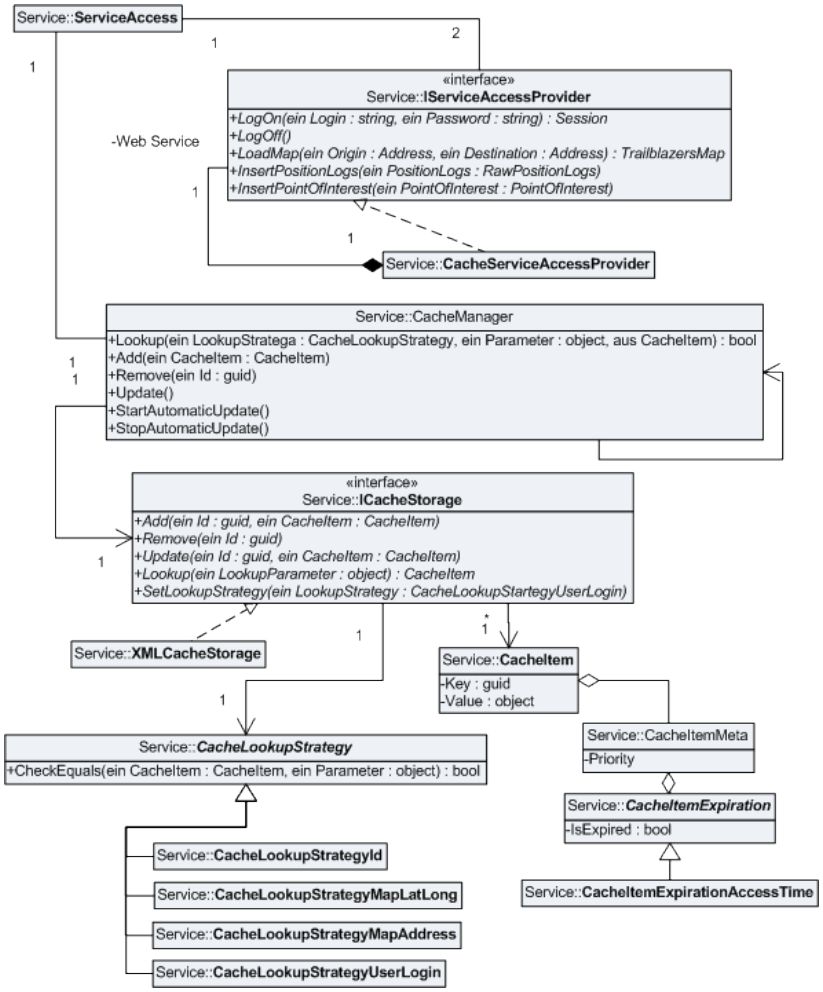

Abbildung 4.19: Klassendiagramm des lokalen Zwischenspeichers

werden alle Dienstaufrufe übertragen. Dazu ruft der `RequestManager` die Methode `Dispatch(...)` der Schnittstelle `IDispatchRequester` auf. Die Schnittstelle wird von der Klasse `WebServiceAccessProvider` implementiert. In dieser Methode wird via eines Reflektion-Mechansimus', in Abhängigeit der oben genannten Attribute, die entsprechende Dienstmethode aufgerufen und dort ein Aufruf an den entfernten Dienst gestartet.

Die Klasse `CacheManager` ermöglicht weiterhin eine automatisierte Weiterleitung der gespeicherten Dienstaufrufe in Abhängigkeit der verfügbaren Verbindung. Dabei wird über einen Delegaten-Mechanismus die private Methode `OnConnectionDispatch()` aufgerufen, sobald sich der Verbindungsstatus ändert. In der Methode wird daraufhin ein eigener Thread gestartet, der die Dienstaufrufe der Warteschlange entnimmt und sie an `IRequestDispatcher` weiterleitet. Dieser Mechanismus wird mit den Methoden

`StartAutomaticDispatch()` und `StopAutomaticDispatch()` gestartet bzw.. gestoppt. Das Attribut `Price` der Klasse `RequestMeta` definiert die Kosten für einen derartigen, entfernten Dienstaufruf. Äquivalient dazu werden Kosten für die verfügbaren Netzwerkverbindungen in den Klassen `NICConnection` und `CellConnection` definiert. Durch diesen Mechanismus ist es z. B. möglich, bei bestehender Verbindung zum Trailblazers-Server via GSM und GPRS, nur Dienstaufrufe zum Einfügen von Positions-Logs direkt auszuführen, Dienstaufrufe zum Einfügen eines Ortes von Interesse jedoch in der Warteschlange zwischenzuspeichern. Wenn der Nutzer zu einem späteren Zeitpunkt z. B. eine WLAN-Verbindung zum Trailblazers-Server herstellt, werden diese Dienstaufrufe durchgeführt.

Gespeicherte Dienstaufrufe, die nach einer definierten Anzahl von Versuchen nicht weitergeleitet werden konnten, werden in einer separaten Warteschlange gespeichert. Der Nutzer wird entsprechend darüber informiert.

Abb. 4.20 zeigt ein detailierteres Klassendiagramm.

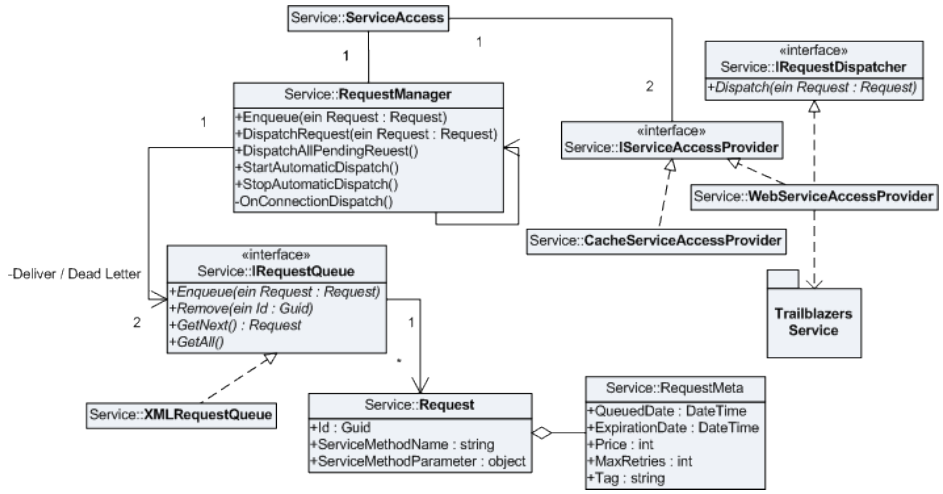

Abbildung 4.20: Klassendigramm der lokalen Dienstaufrufwarteschlange

Wenn dieselben Daten an mehrere mobile Anwendungen repliziert und dort in einem Offline-Szenario von verschiedenen Nutzern geändert werden, kann es beim Zusammenführen der Daten zu Konflikten kommen. Mobile Datenbanksysteme, die häufig für ein derartiges Szenario eingesetzt werden, können Konflikte erkennen und ggfs. automatisch auflösen (Gray u. a. (1996)). Die bisher erarbeiteten Anwendungsfälle sehen die Änderung mobiler Daten nicht vor. Daten können jederzeit hinzugefügt werden, sind jedoch im gesamten System eindeutig und können daher problemlos mit dem gesamten Datenbestand konsolidiert werden.

4.4.3.4 Entfernte Dienstaufrufe

In der Klasse `WebServiceAccessprovider` werden die Aufrufe an den entfernten Dienst implementiert. Die Klasse hat mit dem Stellvetreter-Entwurfmuster Zugriff auf lokale Klassen des Trailblazers-Web-Service. Diese wurden aus der WSDL-Beschreibung des Dienstes generiert.

Beim Aufruf einer Methode einer lokalen Stellvertreter-Klasse wird eine SOAP-Nachricht erzeugt und diese über das Kommunikationsnetzwerk an den Trailblazers-Dienst übertragen.

4.4.3.5 Benachrichtigung vom entfernten Dienst

Für die Anwendungsfälle Tracking und Kommunikation muss der entfernte Dienst die Client-Anwendung bei bestimmten Ereignissen benachrichtigen, z. B. dann, wenn ein Nutzer eine Nachricht an einen anderen Nutzer versendet. Dafür können keine propritären Protokolle, wie z. B. bei Instant-Messanger-Anwendungen üblich, eingesetzt werden, da dies den nichtfunktionalen Anforderungen widerspricht.

Dustdar u. a. (2003) schlägt vor, für diesen Fall entweder einen *Client-Pull* oder einen *Server-Push* zu verwenden. Bei ersterem ruft die Client-Anwendung in regelmäßigen Intervallen den entfernten Dienst auf und prüft, ob ihre Sicht auf die Daten noch aktuell ist. Bei zweitem benachrichtigt die Server-Anwendung alle angemeldeten Client-Anwendungen.

Jedoch ist ein Server-Push mit dem bei den entfernten Dienstaufrufen genutzten, RMI-basierten Web-Service, nicht möglich. Auch kann das Protokoll HTTP dafür nicht genutzt werden, da die mobilen Anwendungen in der Regel keinen HTTP-Endpunkt zur Verfügung stellen. Dennoch soll dieser Mechanismus angewandt werden; der Client-Pull ist bei mobilen Anwendungen nicht akzeptabel, da die Netzlast steigt und eine zeitgerechte Benachrichtigung nicht erreicht werden kann.

Allerdings werden die entsprechenden Web-Service-Nachrichtenmuster *Solicit Response* und *Notifikation* für einen Server-Push werden aktuell noch nicht von ASP.NET Web-Services unterstützt[2](Microsoft (2006e)). Die entsprechenden Standards *WS-Notifikation* und *WS-Eventing* wurden von der Normierungsorganisation OASIS (OASIS (2006)) im Oktober 2006 verabschiedet.

[2]Mit den Web-Service Enhancements (WSE) existiert eine Klassenbibliothek, mit der Web-Service mit verschiedenen Nachrichtenmustern umgesetzt werden können; diese ist jedoch nicht für das .NET Compact Framework verfügbar.

Die Benachrichtigung vom entfernten Dienst wird daher mithilfe des Publish-Subscribe-Konzeptes realisiert. Die Client-Anwendung meldet sich über den entfernten Dienstaufruf `SubscribeForNotifiaction(...)` bei der Server-Anwendung als Abonnent für den Empfang von Benachrichtigungen an. Tritt in der Server-Anwendung ein Ereignis ein, werden alle angemeldeten und betroffenen Abonnenten benachrichtigt. Dazu werden SOAP-Nachrichten über TCP vermittelt. Die Klasse `WebServivceAccessProvider` der Client-Anwendung implementiert die Schnittstelle `IServiceListener`; deren Implementierung stellt einen TCP-Endpunkt bereit. Die Nachrichten enthalten lediglich Informationen über das eingetretene Ereignis und somit geringe Datenmengen. Das Ereignis wird in der Client-Anwendung ausgewertet, und der entsprechende entfernte Dienst wird aufgerufen, um die Sicht auf die Daten zu aktualisieren. Dabei werden auch die aktuellen Verbindungskosten berücksichtigt, die durch die Klasse `ConnectionMonitor` bereitgestellt werden. Dieses Verfahren entspricht einem hybriden Ansatz aus Client-Pull und Server-Push.

Darüber hinaus wird Client-seitig in der Klasse `ConnectionMonitor` auch ein Wechsel der eigenen IP-Adresse, z. B. bei einem so genannten vertikalen Handover, berücksichtigt. Nach einem Wechsel wird die Anfrage als Abonnent erneut durchgeführt, damit die Client-Anwendung auch weiterhin für Benachrichtigungen erreichbar ist.

4.5 Fazit

Die Weiterentwicklung der Trailblazers-Software wurde mit dem in diesem Kapitel beschriebenen Software-Entwurf umgesetzt. Trailblazers Beta ist zum einen ein kompletter Neuentwurf, inklusive Strukturverbesserung (engl. : refactoring), der Funktionalitäten von Trailblazers Alpha, sowie eine konsequente Weiterentwicklung zur Integration von Eigenschaften des Web 2.0. Mit dem beschriebenen Entwurf werden alle funktionalen und nichtfunktionalen Anforderungen umgesetzt. Abb. 4.21 zeigt eine Übersicht über Trailblazers Beta.

Die gewählte Client-Server-Architektur des Informationssystems wird den Anforderungen an ein mobiles Informationssystem für ortebezogene Dienste gerecht. Das Informationssystem basiert auf einer dienstorientierten Architektur. Die Dienste der Server-Anwendung sind lose gekoppelte Anwendungskomponenten, die durch den Daten- und Nachrichtenvertrag universell beschrieben sind und plattformunabhängig genutzt werden können. Des Weiteren ist eine spätere Migration auf WCF-basierte Web-Services problemlos möglich. Die Anwendungslogik ist auf die mobile und die stationäre Anwendung verteilt, einzelne Schichten dienen engem Zusammenhalt und loser Kopplung; die Trennung der Verantwortlichkeiten ist gegeben.

Abbildung 4.21: Trailblazers Beta

Der Benachrichtigungsmechnismus, basierend auf dem Publish-Subsribe-Konzept und dem Versand von SOAP-Nachrichten, stellt eine befriedigende Lösung für eine Benachrichtigung der Client-Anwendung durch die Server-Anwendung dar. Bei Verfügbarkeit scheint hier die Nutzung des Standards WS-Notification sinnvoll.

Der Datenbankentwurf basiert auf einer objekt-relationalen Datenbank. Dabei wurden räumliche Datentypen genutzt, um die Vorteile einer objekt-relationalen Datenbank und einer erweiterten Anfragesprache für derartige Datentypen zu nutzen. Informationen über Nutzer werden in einem direkten und einem indirekten Profil in der Datenbank gespeichert.

Der Entwurf der mobilen Smart-Client-Anwendung ist generisch, und kann für verschiedene mobile Plattformen implementiert werden. Die Darstellungskomponente muss in der Regel speziell an eine Zielplattform angepasst werden. Daher ist diese vom Rest der Anwendung entkoppelt. Die Client-seitige Zwischenspeicherung von Dientsaufrufen und Dienst-Entitäten ermöglicht ein Online-Offline-Szenario. Dieses wird durch die hybride Onboard-Navigation unterstützt. Bei der Navigation werden vorhandene Straßenkarten gezielt mit Informationen von Nutzern annotiert.

Weiterhin wurden Komponenten identifiziert und nach Möglichkeit bestehende, externe Komponenten in die Anwendung integriert. Der Einsatz von Entwurfsmustern unterstützt dabei

die lose Kopplung und macht den Austausch einzelner Komponenten, teilweise auch zur Laufzeit, sowie Erweiterungen der Funktionalität möglich. Es werden durchgehend verbreite Web-Standards sowie Standards ortsbezogener Dienste genutzt. Der Entwurf kann mit aktuellen Technologien umgesetzt werden.

5 Implementierung

In diesem Kapitel wird die konkrete Implementierung des Software-Entwurfs des Trailblazers Beta Informationssystems besprochen. Dazu wird zunächst der Status der Implementierung aufgezeigt. Im Anschluss werden die zur Entwicklung verwendete Hard- und Software, sowie die in das Informationssystem eingebundenen, externen Software-Komponenten aufgeführt; des weiteren werden die bei der Entwicklung durchgeführten Maßnahmen zur Qualitätssicherung skizziert. Abschließend werden ausgesuchte Details zur Implementierung einzelner Komponenten ausführlich diskutiert.

5.1 Implementierungsstatus

Ziel der Arbeit war die prototypische Implementierung des Software-Entwurfs des Trailblazers Beta Informationssystems. Im Rahmen dieser Arbeit wurde der Software-Entwurf nahezu komplett umgesetzt. Das Ergebnis ist eine prototypische Client- und Server-Anwendung, die sämtlichen funktionalen Anforderungen genügt. Die Client-Anwendung wurde in einer Version für mobile Standardcomputer implementiert und im Anschluss in eine Version für Smartphones portiert. Der Quelltext der Anwendungen ist auf der dieser Arbeit beiliegenden CD-ROM gespeichert (siehe Anhang A.3).

Im Folgenden werden die Fertigstellungsgrade aller Komponenten aufgezeigt und auf noch zu vervollständigende Implementierungsarbeiten hingewiesen. Bei der Implementierung stand die Verarbeitungsgeschwindigkeit der Anwendungen nicht im Vordergrund. Ebenso wurde auf eine Komponente für eine automatische Bereitstellung (engl.: deployment) und Aktualisierung einzelner Komponenten verzichtet. Beide Punkte sollten vor dem Übergang in eine Produktionsumgebung jedoch berücksichtigt werden. In (Microsoft (2006g)) werden dazu mögliche Lösungsvorschläge für .NET-basierte Anwendungen aufgezeigt.

5.1.1 Server-Anwendung

Dienstfassadenschicht Die Dienstfassadenschicht wurde dem Entwurf entsprechend implementiert.

Anwendungsschicht Die Kartenkomponente ist noch nicht vollständig implementiert; der Zugriff auf den Google Maps Kartendienst ist noch fehlerhaft. Die Sitzungsinformationen einzelner Nutzer in der Sitzungskomponente werden zur Zeit nicht in einer Datenbank, sondern lediglich im Hauptspeicher gehalten. Die Überprüfung des berechtigten Zugriffs auf die Positionsdaten anderer Nutzer in der Tracking-Komponente ist nicht implementiert. Alle anderen Komponenten wurden nach dem Entwurf vollständig implementiert.

Datenzugriffsschicht Die Datenzugriffsschicht ist dem Entwurf entsprechend vollständig implementiert.

5.1.2 Mobile Client-Anwendung

Präsentationsschicht Die Präsentationsschicht wurde vollständig implementiert.

Anwendungsschicht Bei der Tracking-Kompenente wurde die Vorhersage für die zukünftige, abgeleitete Position noch nicht umgesetzt, ebenso die automatisierte Aktualisierung der Positionsinformationen in bestimmten Zeitintervallen durch einen eigenen Thread. Alle anderen Komponenten wurden vollständig implementiert.

Dienstzugriffsschicht Der automatische Aktualisierungsmechanismus für Daten im lokalen Zwischenspeicher wurde bisher nicht implementiert. Das automatische Versenden zwischengespeicherter Dienstaufrufe in einem eigenen Thread ist ebenfalls noch nicht fertiggestellt. Des weiteren fehlt die Implementierung für eine Warteschlange, die alle nicht zustellbaren Nachrichten sammelt. Davon abgesehen wurde sämtliche Funktionalität in der Dienstzugriffsschicht implementiert.

5.2 Verwendete Hardware

Für die Entwicklung des mobilen Informationssystems wurde folgende Hardware genutzt:

Nemerix GPS-Empfänger Für die Positionsbestimmung wurde ein BT77 Nemerix Bluetooth GPS-Empfänger genutzt. Dieser unterstützt das NMEA-Protokoll und beinhaltet einen Nemerix Chipsatz. Der Anschluss an ein mobiles Gerät erfolgt via Bluetooth mit dem Profil der seriellen Schnittstelle.

Q1 UMPC Die auf dem .NET Framework basierende mobile Anwendung wurde zu Zwecken der Qualitätssicherung und der Evaluierung, in regelmäßigen Abständen auf einem Samsung Q1 UMPC bereitgestellt.

Qtek Smartphone Die .NET Compact Framework-basierte Variante der mobilen Anwendung wurde zu Testzwecken auf einem htc Qtek 8500 Smartphone installiert.

Thinkpad T60 Notebook Ein Lenovo Thinkpad T60 Notebook diente als Entwicklungs- und Testrechner sowohl für die Client-, als auch für die Server-Anwendung. Als Betriebssystem wurde zu Beginn der Arbeit Microsoft Windows XP Professional genutzt; im weiteren Verlauf wurde dieses auf Windows Vista Business Edition umgestellt.

5.3 Verwendete Software

Folgende Software-Werkzeuge wurden im Rahmen dieser Arbeit für die Entwicklung des mobilen Informationssystems genutzt:

SQL Server 2005 Für die Speicherung der Daten des mobilen Informationssystems wurde ein Microsoft SQL Server 2005 Developer Edition eingesetzt.

Visio 2003 Die UML-Digramme und das objekt-relationale Datenmodell des Software-Entwurfs wurden mit Microsft Visio 2003 Enterprise Architect erstellt. Aus den Diagrammen wurden Quelltexte bzw. SQL-Anweisungen erzeugt, die eine Grundlage für die Implementierung des Software-Entwurfs boten.

Visual Studio 2005 Für die Implementierung des mobilen Informationssystem wurde die integrierte Entwicklungsumgebung Microsoft Visual Studio 2005 Team Edition for Architects genutzt. Diese stellt auch Emulatoren für mobile Geräte, wie Smartphones und PDAs, bereit.

Abb. 5.1 zeigt ein Bildschirmfoto der Entwicklungsumgebung mit dem geöffneten Projekt der mobilen Anwendung.

5.4 Verwendete Software-Komponenten

Für die Implementierung wurden die Programmiersprache C#, sowie folgende Microsoft .NET-Technologien genutzt:

.NET Compact Framework 2.0 Das .NET Compact Framework 2.0 wurde für die Smartphone-Version der Client-Anwendung genutzt.

.NET Framework 2.0 Die Server-Anwendung basiert auf dem .NET Framework 2.0.

.NET Framework 3.0 Die mobile Client-Anwendung für UMPCs wurde mit dem .NET Framework 3.0 implementiert.

Weiterhin wurden bei der Implementierung folgende externe Software-Komponenten von Drittanbietern genutzt:

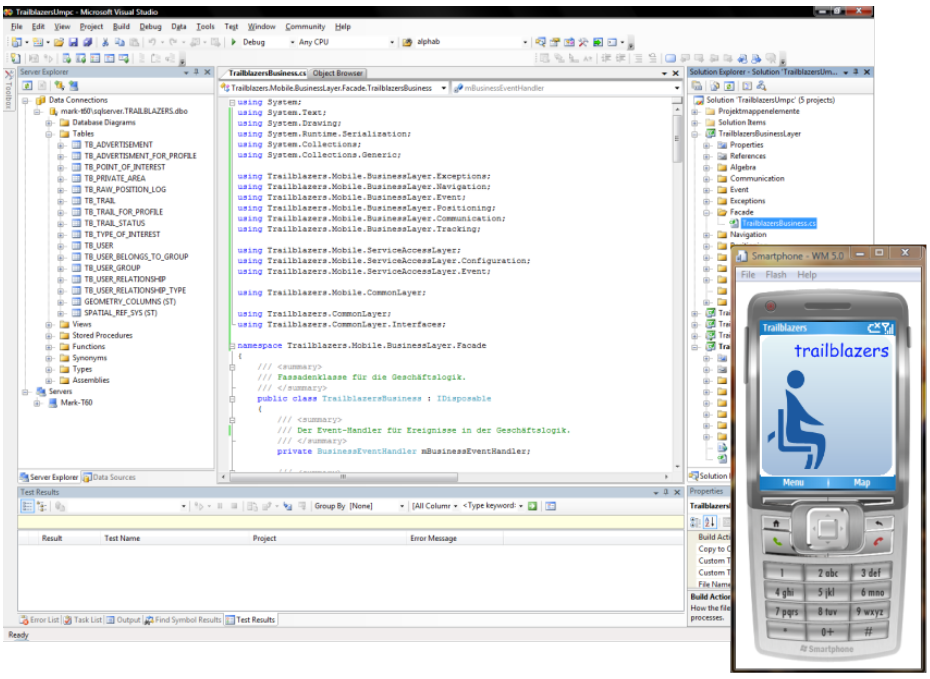

Abbildung 5.1: Entwicklungsumgebung mit geöffnetem Projekt der mobilen Anwendung und mit Emulator

EMIC Location und Mapping Framework Die Anzeige annotierter Karten für die barrierefreie Navigation wurde mit dem EMIC Location und Mapping Framework (EMIC (2007)) implementiert. Dieses ist sowohl für das .NET Framework, als auch für das .NET Compact Framework erhältlich und kann im Rahmen von Forschung und Lehre frei genutzt werden.

GeoFramework Die kommerzielle Komponente GeoFrameworks (GeoFrameworks (2007)) wurde in der Client-Anwendung genutzt, um die GPS-Positionsbestimmung umzusetzen. Dazu stellt diese Komponente die Verbindung mit einem Hardware-GPS-Empfänger über eine serielle Schnittstelle her und wertet die Signale, basierend auf dem NMEA-Protokoll, aus. Über einen Delegaten-Mechanismus werden diese Signale an die Anwendung weitergegeben. Die Komponente ist sowohl für das .NET-Framework, als auch für das .NET Compact Framework erhältlich.

Janus Forms Janus WinForms Control Suite (Janus (2007)) ist eine kommerzielle .NET Framework-basierte Sammlung von Bausteinen für Benutzerschnittstellen. Mit dieser Komponente wurde die Benutzerschnittstelle für die UMPC-basierte Client-Anwendung implementiert.

Microsoft.Ink Microsoft.Ink (Microsoft (2006d)) ist eine .NET Framework-basierte Klassenbibliothek für Benutzerschnittstellen, welche die Eingabe mittels digitaler Tinte ermöglicht. Diese wurde für die UMPC-Version der Client-Anwendung genutzt.

MsSqlSpatial Der SQL Server 2005 verfügt über eine eigene .NET-Laufzeitumgebung. MsSqlSpatial (MsSqlSpatial (2007)) ist eine Open-Source-Erweiterung für räumliche Datentypen und Operationen, die in dieser Laufzeitumgebung ausgeführt werden kann, und die damit beim Datenbankentwurf genutzten, räumlichen Datentypen bereitstellt.

NetTopologySuite NetTopolgySuite (Sourceforge (2007)) ist eine Open-Source-Klassenbibliothek für räumliche Datentypen und geografische Operationen, gemäß den Spezifikationen des Open Geospatial Consortiums (OGC) (vgl. OGC (2007a)). Sie wurde in der Anwendungsschicht sowohl der Client-, als auch der Server-Anwendung genutzt. Die Bibliothek exisiert in Versionen für das .NET Framework und für das .NET Compact Framework.

NGenerics NGenerics (NGenerics (2007)) ist eine Open-Source-Klassenbibliothek aus Datenstrukturen und Algorithmen, die im .NET Framework nicht enthalten sind. Sie beinhaltet u. a. eine Implementierung des Dijktra-Routenalgorithmus.

System.Speech System.Speech ist eine Klassenbibliothek des Net Frameworks der Version 3.0, die Zugriff auf die Sprachein- und -ausgabegfunktionen der Windows Betriebssysteme bietet. Damit wurde die Sprachausgabe von Routeninformationen der UMPC-Version der mobilen Anwendung realisiert.

5.5 Qualitätssicherung

Zur Qualitätssicherung der Implementierung wurden Modultests, so genannte Unit-Tests, herangezogen. Diese wurden, gemäß dem XP-Paradigma, parallel zu der Entwicklung relevanter Komponenten umgesetzt (Beck (2002)). Dazu wurde die integrierte Unit-Test Funktionalität der oben genannten Entwicklungsumgebung genutzt.

Um die ortsbezogenen Anwendungsteile unter Laborbedingungen testen zu können, implementiert die Klasse `LocationServiceSimulated` in der Client-seitigen Anwendungsschicht eine simulierte GPS-Positionsbestimmung. Dazu wurde eine Textdatei erzeugt, die Positionsdaten im NMEA-Format enthält. Die Textdatei wurde von der Klasse kontinuierlich eingelesen und simulierte so eine praktische Poitionsbestimmung, bei der Daten im NMEA-Format über eine serielle Schnittstelle empfangen werden.

5.6 Implementierungsdetails

In diesem Abschnitt werden Implementierungsdetails ausgesuchter Komponenten des Software-Entwurfs gesondert beschrieben.

5.6.1 Kartenkomponente und externe Kartendienste

Für den Anwendungsfall barrierefreie Navigation werden in der Server-seitigen Kartenkomponente Straßenkarten externer Dienstanbieter abgerufen, mit Informationen aus der Trailblazers-Datenbank annotiert und an die Client-Anwendung weitergereicht.

5.6.1.1 Auswahl des Kartenkorridors

Im Aktivitätsdiagramm des Anwendungsfalls (vgl. Anhang A.1) lassen sich zwei erforderliche Dienstaufrufe identifizieren. Ein dritter wird benötigt, wenn der Nutzer eine Route so verlassen hat, dass seine Position nicht mehr auf den geladen Straßenkarten anzeigbar ist:

Umgebungskarte mit geografischer Länge und Breite Möchte der Nutzer nur eine Umgebungskarte zu seiner aktuellen Position angezeigt bekommen, wird ein Areal um diese Position für die zu ladenden Karteausschnitte gewählt.

Karte mit Route von Start- zu Zieladresse Möchte der Nutzer eine Karte angezeigt bekommen, auf der eine Route mit Start- und Zieladresse dargestellt ist, wird ein Korridor entlang der Route für die zu ladenden Kartenausschnitte gewählt. Die Adressen werden in geografische Koordinaten übersetzt.

Karte mit Route von Start- und Zielpunkt und eigener Position Verlässt der Nutzer eine Route so, dass seine Position nicht mehr in dem zuvor geladenen Kartenkorridor enthalten ist, muss der Korridor neu geladen werden. Die Start- und Zieladresse und die eigene Position sind bereits als geografische Koordinaten vorhanden. Letztere bezeichnet auch die Koordinate, an dem der Nutzer den Korridor verlassen hat. Hier sind erweiterte Aktualisierungstrategien denkbar, die z. B. die Bewegungsmuster des Nutzers berücksichtigen und damit die entsprechenden Karten im Voraus vom Dienst laden (vgl. Schiller und Voisard (2004)).

Für die Auswahl des Kartenkorridors wird zunächst die Route vom Start- zum Zielpunkt berechnet, im Anschluss daran werden einzelne Straßenkarten entlang der Route von einem externen Dienst geladen. Abb. 5.2 zeigt eine schematische Übersicht. Ist eine komplette Berechnung einer Route vom Start- zum Zielpunkt nicht möglich, z. B. weil erforderliche Daten über Trails nicht in der Datenbank gespeichert sind, wird als Umgehungslösung eine direkte Verbindung zwischen zwei entsprechenden Trail-Segmenten gewählt.

5.6.1.2 Karten des externen Kartendienstes

Die Kartendienste liefern Rasterkarten in einem Grafikformat, wie z. B. in Bitmap. Die Rasterakarten können sowohl aus dem Vektor-Modus erzeugte Grafiken, als auch Satellitenfotos

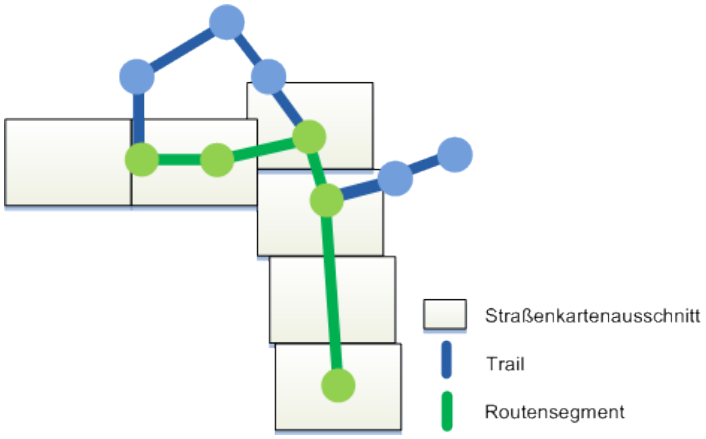

Abbildung 5.2: Straßenkartenausschnitte entlang einer Route

darstellen. Die Karten zeichnen sich durch eine Höhe und Breite in Bildpunkten aus. Jeder einzelne Bildpunkt ist dabei einer Koordinate im gewählten geografischen Referenzsystem zugeordnet. Dabei wird der Maßstab der Karte entsprechend berücksichtigt. Abb. 5.3 zeigt dazu eine schematische Übersicht. Eine Karte zeigt unabhängig vom Maßstab jeweils den gleichen Ausschnitt; je kleiner der Maßstab ist, desto mehr Bildpunkte werden benötigt. Eine Karte darf dabei eine vorgegebene Auflösung von Bildpunkten nicht überschreiten und wird daher ggfs. in Ausschnitte unterteilt. Diese werden als so genannte Kacheln (engl.: tiles) aneinander gefügt. Ein Kartensegment mit einer Auflösung von 800 mal 600 Bildpunkten hat dabei eine Dateigröße von ca. 20 KB.

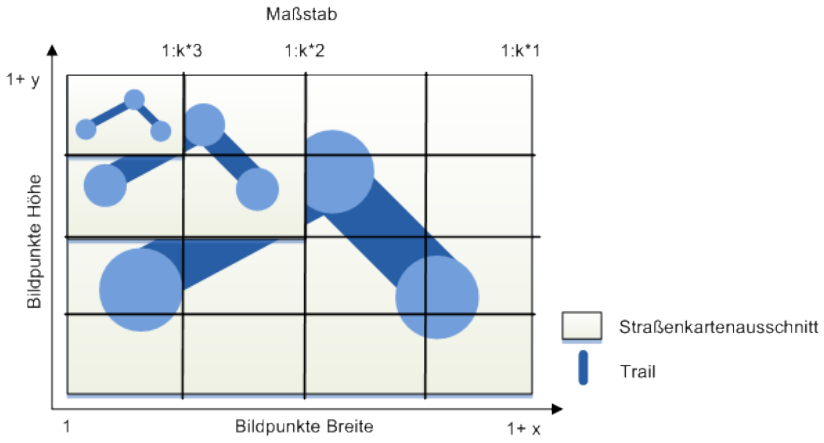

Abbildung 5.3: Berücksichtigung verschiedener Maßstäbe einer Straßenkarte

5.6.1.3 Mash-up mit einem externen Kartendienst

Die Kartendienste Google Maps und Virtual Earth stellen auf Web-Servern Straßenkarten und Satellitenfotos im Rasterformat zur Verfügung. Die Karten sind bereits fertig generiert und werden nicht bei jedem Aufruf neu erstellt. Sie sind in verschiedenen Maßstäben vorhanden und in einzelne Kacheln unterteilt, der Referenzrahmen ist UTM. Mit einer passenden URL können einzelne Kartenausschnitte vom Server geladen werden (vgl. Gibson und Erle (2006) und Microsoft (2007)). Listing (5.1) zeigt das Laden eines Kartenausschnitts von Virtual Earth für eine definierte geografische Position und einen definierten Maßstab.

```
1
2  // Lade Kartenkachel von Virtual Earth.
3  public Bitmap GetMapTile(int pLat, int pLong, int pZoomLevel, MapStyle pMapStyle)
4  {
5   string lUrl = BuildUrl(pLat, pLong, pZoomLevel, pMapStyle);
6   Bitmap lMapTile = LookupBitmapFromServer(lUrl);
7   return lMapTile;
8  }
9
10 // Baue die entsprechende URL zusammen.
11 private override string BuildUrl(int pLat, int pLong, int pZoomLevel, MapStyle pMapStyle)
12 {
13  string lKey = TileToQuadKey(pLat, pLong, pZoomLevel);
14  char pMapTypeChar = (pMapStyle == MapStyle.Road) ? 'r' : 'a');
15  StringBuilder lUrl = new StringBuilder();
16  lUrl.Append("http://");
17  lUrl.Append(pMapTypeChar);
18  lUrl.Append(lKey[lKey.Length - 1]);
19  lUrl.Append(".ortho.tiles.virtualearth.net/tiles/");
20  lUrl.Append(pMapTypeChar);
21  lUrl.Append(lKey);
22  lUrl.Append(pMapStyle == MapStyle.Road ? ".png" : ".jpeg", "?g=");
23  lUrl.Append(15);
24  return lUrl.ToString();
25 }
26
27 // Lade mit der URL die Kartenkachel von dem Web-Server.
28 private Bitmap LookupBitmapFromServer(string pUrl)
29 {
30  WebRequest lRequest = WebRequest.Create(pUrl);
31  Stream lStream = lRequest.GetResponse().GetResponseStream();
32  return new Bitmap(lStream);
33 }
```

Listing 5.1: Virtual Earth Mash-Up

5.6.2 Datenzugriffskomponente und räumliche Daten

Für die Speicherung von ADTs werden die räumlichen Datentypen Punkt, Linie und Polygon genutzt. Weiterhin wird für den Datenzugriff eine erweiterte SQL-Anfragesprache verwendet,

die den Spezifikationen des OGC entspricht (vgl. OGC (2007b)). Beides wird von der oben genannten Erweiterung MsSqlSpatial bereitgestellt.

Die Anfragesprache unterstützt zum einen die Nutzung räumlicher Datentypen, zum anderen stellt sie geografische Operationen auf diesen bereit. Insbesondere wird eine Unterstützung des Dimension Extended-9 Intersection Model (DE-9IM) (vgl. Shekhar und Chawla (2002)) geboten. Dadurch können räumliche Daten in der Datenbank zueinander in Beziehung gesetzt werden. Durch einfache Datenbankanfragen können so bestimmte geografische Daten ausgelesen werden. Entsprechende Anfragen werden für alle räumlichen Entitäten genutzt; folgendes Beispiel soll dies verdeutlichen:

Wenn ein Nutzer eine Karte mit Start- und Zieladresse anfragt, wird in der Server-Anwendung zunächst ein Korridor entlang der Route definiert (vgl. Abschn. 5.6.1.1). Danach gilt es, alle relevanten geografischen Informationen entlang des Korridors aus der Datenbank zu lesen. Für Orte von Interesse wird demnach die SQL-Anweisung in Listing 5.2 formuliert. Dabei ist der Eingabeparameter `@AREA` ein geografischer Ausschnitt in Form eines Polygons entlang des Korridors. Als Ergebnis werden sämtliche Orte von Interesse zurückgegeben, deren Position sich innerhalb dieses Areals befinden.

```
1 SELECT
2  *
3 FROM
4  RelateQuery('TB_POINT_OF_INTEREST', 'POSITION', @AREA, 'contains') AS q,
5  TB_POINT_OF_INTEREST AS poi
6 WHERE
7  q.POINT_OF_INTEREST_ID = poi.POINT_OF_INTEREST_ID
```

Listing 5.2: SQL-Anfrage zum Auslesen aller Orte von Interesse innerhalb eines Areals

5.7 Fazit

Der Software-Entwurf wurde soweit umgesetzt, dass alle skizzierten funktionalen Anforderungen, sowie die Qualitätsanforderungen und Rahmenbedingungen erfüllt werden. Die Leistungsanforderungen konnten im Rahmen dieser Arbeit nicht verifiziert werden, da die Anwendung in einer Test-Umgebung eingesetzt wurde, die keiner Produktionsumgebung entsprach. Bei der Implementierung sind insgesamt keine nennenswerten Probleme aufgetreten.

Beim Software-Entwurf wurde darauf geachtet, alle externen Software-Komponenten so in die beiden Anwendungen zu integrieren, dass diese leicht durch Komponenten anderer Anbieter austauschbar sind. Zur Überprüfung wurden u. a. verschiedene externe Komponenten für die Kartendarstellung und für die GPS-Positionsbestimmung in die Client-Anwendung

integriert. In die Server-Anwendung wurden u. a. verschiedene externe Dienste für Straßenkarten eingebunden, die zur Laufzeit austauschbar sind.

Die Implementierung des Software-Entwurfs für mobile Standardcomputer und die anschließende Portierung auf eine Version für Smartphones hat gezeigt, dass der Software-Entwurf generisch und zwischen den verschiedenen .NET-Plattformen übertragbar ist. Eine Portierung auf Java-Plattformen konnte in dieser Arbeit nicht durchgeführt werden, sollte aber problemlos möglich sein.

6 Evaluierung

Um den Nutzen des im Rahmen dieser Arbeit entworfenen und implementierten Informationssystems herauszustellen, und um damit die Synergieeffekte durch die Zusammenführung von Web 2.0 und ortsbezogenen Diensten zu verifizieren, wurde eine erste, formative Evaluierung[1] durchgeführt.

6.1 Versuchsziele

Für die Überprüfung der Umsetzung sämtlicher an das mobile Informationssystem gestellten Anforderungen (vgl. Abschn. 3.3 und 3.4) wurden folgende Versuchsziele abgeleitet:

Anwendungsfälle Die Überprüfung der in den Anwendungsfällen skizzierten Funktionalitäten stellt den Hauptbestandteil der Evaluierung dar. Da sämtliche, für die Funktionserfüllung relevanten Komponenten implementiert worden sind, sind alle funktionalen Anforderungen zu erfüllen.

Online-Offline-Funktionalität Die Überprüfung der Nutzbarkeit in einem Online-Offline-Szenario soll zeigen, ob der Entwurf und die Implementierung des lokalen Dienstzugriffs ausreichend sind. Die mobile Anwendung muss in einem Offline-Szenario funktionieren, in dem der Nutzer über einen längeren Zeitraum nicht mit dem Trailblazers-Dienst verbunden ist.

GPS-Positionsbestimmung Die Überprüfung der Positionsbestimmung via GPS zeigt, ob diese Art der Positionsbestimmung für den alltäglichen Betrieb geeignet ist. Insbesondere soll die Bestimmung in verschiedenen Umgebungen durchgeführt werden, um eine subjektive Aussage über die Genauigkeit treffen zu können.

Sicherung der Privatsphäre Die Überprüfung der skizzierten und implementierten Maßnahmen zur Sicherung der Privatsphäre soll zeigen, ob diese ausreichend sind.

[1] Der Autor setzt den Begriff *Evaluierung* mit dem Begriff *Evaluation* gleich.

6.2 Versuchsaufbau

Im Gegensatz zur Qualitätssicherung bei der Implementierung, wurde diese erste Evaluierung außerhalb einer Laborumgebung realisiert. Der Trailblazers-Dienst war dabei nicht über das Internet, sondern lediglich über ein drahtloses, lokales Netzwerk erreichbar. Die Online-Funktionalität konnte daher nur örtlich eingeschränkt getestet werden.

Die Evaluierung wurde in den Städten Hamburg, Lübeck und London durchgeführt. An den Versuchsreihen haben insgesamt vier Testpersonen teilgenommen, es waren jedoch nie mehr als zwei Nutzer gleichzeitig im System angemeldet. Zu Beginn der Evaluierung wurden Testdaten, insbesondere Trail-Datensätze, manuell in die Datenbank eingefügt, um die Nutzung des Systems durch eine Vielzahl von Anwendern zu simulieren.

Abb. 6.1 zeigt Bildschirmfotos der mobilen Client-Anwendung. Links im bild ist die Version für UMPCs, rechts die Version für Smartphones dargestellt. Bei der UMPC-Variante ist ein Satellitenfoto der Innenstadt Londons als Straßenkarte eingeblendet. Trails sind als blaue Linien, die Route als ein grüne Linie gekennzeichnet. Aufgenommene Positions-Logs werden als blaue Punkte kontinuierlich gezeichnet. Die Position des Nutzers ist durch ein Piktogramm auf der Karte gekennzeichnet. Navigationsanweisungen werden links unten auf der Karte angezeigt, rechts oben wird ein Bild für ortsbezogene Werbung eingeblendet. Die Position anderer Nutzer, sowie deren Name und die Uhrzeit der Positionsbestimmung, werden an der jeweiligen Stelle der Karte angezeigt. Der Nutzer kann Nachrichten mit digitaler Tinte verfassen und diese an andere Nutzer seines sozialen Netzwerks schicken; dort werden die Nachrichten entsprechend angezeigt.

6.3 Versuchsergebnisse

Folgende Ergebnisse haben sich durch die Evaluierung ergeben:

Anwendungsfälle Die Nutzer konnten sämtliche Funktionalität des Informationssystems nutzen. Sie konnten entlang einer Route von einem Start- zu einem Zielpunkt gelangen. Weiterhin konnten sie Nachrichten in Echtzeit austauschen, die Position des jeweils anderen Nutzers verfolgen und sich so gemeinsam treffen. Eine Vermaschung von virtueller mit realer Welt war gegeben. Werbung wurde orts- und gruppenbezogen angezeigt und dabei nicht als störend empfunden. Allerdings konnte durch ein einzelnes Bild kaum die Aufmerksamkeit des Nutzers auf die Werbung gelenkt werden.

Online-Offline-Funktionalität Der Offline-Modus ermöglichte die barrierefreie Navigation, sowie das Hinzufügen von Positions-Logs bzw. von Orten von Interesse, wenn keine Verbindung zum Trailblazers-Dienst zur Verfügung stand. Der spätere Transfer der

Abbildung 6.1: Bildschirmfoto der mobilen Client-Anwendung

lokalen Daten funktionierte einwandfrei. Wurde im Offline-Modus ein neues Ziel gewählt, war dies nur durch direkte Markierung auf der Karte möglich; in der mobilen Anwendung werden, mit Ausnahme zwischengespeicherter Start- und Zieladressen, keine Adressen gespeichert. Der Nutzer musste demnach wissen, wo sein Ziel liegt; dies hat sich als nachteilig herausgestellt.

GPS-Positionsbestimmung Die Positionsbestimmung via GPS war immer problemlos möglich und lieferte stets genaue Ergebnisse. Die Abweichung zwischen gemessener und tatsächlicher Position betrug weniger als 10 Meter. Obwohl ein so genanntes Map-Matching[2] nicht möglich war, da die Nutzer sich bewusst abseits von Straßen bewegt haben, war eine Orientierung durch die Kartendarstellung gegeben.

Sicherung der Privatsphäre Die ergriffenen Maßnahmen zur Sicherung der Privatsphäre waren ausreichend. Aufgenommene Positions-Logs konnten keinem einzelnen Nutzer zugeordnet werden. Durch private Areale konnten die Nutzer eine Aufnahme in bestimmten geografischen Bereichen unterbinden. Die Weitergabe der eigenen Position ausschließlich an Nutzer, die dem eigenen sozialen Netzwerk angehören, war den

[2] Die gemessene Position wird dabei mit der digitalen Karte abgeglichen und eventuell auf eine wahrscheinlichere Position, z. B. die Straßenmitte, korrigiert.

Nutzern eine bekannte und ausreichende Möglichkeit zum Schutz der eigenen Privatsphäre. Unterstützt wurde dies dadurch, dass ein Nutzer stets informiert wurde, wenn seine Position weitergegeben wurde. Dadurch konnte und er die Weitergabe auch jederzeit unterbinden.

Die Versuchsreihen haben insgesamt gezeigt, dass die prototypischen Implementierungen der Anwendungsfälle den Nutzern Mehrwehrt bieten. Die Erweiterungen der Beta Version weisen deutliche Eigenschaften einer Web 2.0 Anwendung auf, die den Nutzern zu Gute kommen. Die Nutzer konnten sich helfen, indem sie ortsbezogene Daten generierten und anderen Nutzern zur Verfügung stellten. Ein mobiles, soziales Netzwerk ließ sie Teil einer Gemeinschaft werden in der sie sich austauschen konnten. Der tatsächliche Nutzen des Informationssystems lässt sich jedoch erst mit einer großen Anzahl von Nutzern verifizieren. Das ist zudem notwendig, um etwaigen Missbrauch durch die Nutzer festzustellen, sowie die geeigneten Kontrollmaßnahmen zu validieren.

7 Fazit

Zu Beginn dieses Kapitels wird eine Zusammenfassung der gesamten Arbeit gegeben und die gewonnenen Erkenntnisse der Arbeit bewertet. Zum Abschluss werden mögliche technische und organisatorische Fortsetzungen des entwickelten mobilen Informationssystems, sowie allgemein zu erwartende Fortschritte in den Bereichen Ortsbezogene Dienste und Web 2.0 angesprochen.

7.1 Zusammenfassung

In dieser Arbeit wurden aktuelle Fragestellungen bezüglich mobiler, ortsbezogener Dienste und der Paradigmen des Web 2.0 untersucht, deren Einsatz durch aktuelle technologische Entwicklungen und ein sich änderndes Verständnis von Diensten und Nutzern im Internet begünstigt wird. Dies konnte im Rahmen dieser Arbeit durch den Entwurf und die Implementierung eines mobilen Informationssystems für ortsbezogene Dienste - auf Basis aktueller, verfügbarer Technologien - exemplarisch verifiziert werden.

Der Leser wurde in Kap. 2 mit den Grundlagen mobiler Informationssysteme vertraut gemacht. Deren Eigenschaften, an sie gestellte Anforderungen, mögliche Anwendungsarchitekturen, sowie dazu verwendete mobile Geräte und Betriebssysteme wurden aufgezeigt. Smart-Clients wurden dabei als geeignete Anwendungsarchitektur, das .NET Framework als eine adäquate Plattform für das mobile Informationssystem identifiziert. Ortbezogene Dienste wurden skizziert und Navigationssysteme als dafür geeignete mobile Informationssysteme vorgestellt. Weiterhin wurden verschiedene Technologien und Systeme zur Positionsbestimmung voneinander abgegrenzt, wobei die GPS-Technologie für den Einsatz im mobilen Informationssystem als zweckmäßig angesehen werden konnte. Abschließend wurden wesentliche Merkmale des Web 2.0 herausgearbeitet, um diese für nachfolgende Fragestellungen heranziehen zu können.

Kap. 3 beginnt mit der Beschreibung eines vorhandenen mobilen Informationssystem, das als Ausgangsszenario für diese Arbeit angesehen wurde. Im Anschluss wurde diskutiert, in

welcher Form ortsbezogene Dienste im Allgemeinen und der Dienst des gewählten Ausgangsszenarios im Speziellen, mit den Eigenschaften des Web 2.0 gewinnbringend zusammengeführt werden können. Dabei wurde festgestellt, dass mobile Anwendungen für ortsbezogene Dienste von den aktuellen Entwicklungen des Web 2.0 u. a. dadurch profitieren, dass Nutzer durch Zusammenarbeit ein gemeinsames Ziel erreichen können. Die Erkenntnisse der Diskussion wurden in funktionale und nichtfunktionale Anforderungen an eine erweiterte, neue Version des mobilen Informationssystems überführt.

Die herausgearbeiteten Anforderungen wurden in Kap. 4 in einem Software-Entwurf umgesetzt. Dazu wurden zunächst mögliche Architekturen voneinander abgegrenzt und es wurde festgestellt, dass eine dienstorientierte Client-Server-Architekur sowie Web-Services als Kommunikationtechnologie für diese Art von Informationssystemen gut geeignet sind. Der Datenbankentwurf, als Fundament des Informationssystems, wurde im ER-Modell und in einem daraus resultierenden, objekt-relationalen Schema beschrieben. Die Speicherung ortsbezogener Daten fand durch räumliche Datentypen und eine angepasste Datenbankanfragesprache statt. Im detailliert beschriebenen, objektorientierten Entwurf von Client- und Server-Anwendung wurden verwendete Entwurfsmuster vorgestellt, die u. a. die lose Kopplung und Integration externer Komponenten ermöglichen.

In Kap. 5 wurde die Implementierung des mobilen Informationssystems besprochen, wozu vorab eine Übersicht über den Implementierungsstatus der erstellten Prototypen gegeben wurde; der Software-Entwurf wurde bei der Implementierung nahezu vollständig umgesetzt. Die für die Implementierung eingesetzte Hard- und Software und genutzte externe Komponenten wurden aufgezeigt und die wesentlichen Maßnahmen zur Qualitätssicherung erläutert. Abschließend wurden Implementierungsdetails für den Zugriff auf externe Kartendienste und auf die räumliche Datenbank beschrieben.

Im Anschluss an die Implementierung wurde eine erste Evaluierung des mobilen Informationssystems durchgeführt. Kap. 6 enthält die Beschreibung der Versuchsziele, des Versuchsaufbaus und der Versuchsergebnisse. Es konnte gezeigt werden, dass das mobile Informationssystem die beschriebenen Anforderungen erfüllt. Die Nutzer können sich gegenseitig helfen, indem sie ortsbezogene Daten generieren und austauschen; darüber hinaus können sie ein mobiles soziales Netzwerk bilden und miteinander in Kontakt treten.

7.2 Bewertung

Das Ziel der Arbeit, der Software-Entwurf eines mobilen Informationssystems für ortsbezogene Dienste im Web 2.0, kann als erreicht angesehen werden. Das entworfene System entstand unter Berücksichtigung der Konstruktionsregeln des Software-Engineering. Die entworfene Software ist modular aufgebaut, der Funktionsumfang kann problemlos erweitert

werden. Der Software-Entwurf ist für die .NET-Plattformen optimiert, jedoch generell auch auf andere Plattformen übertragbar. Die entworfene Architektur bietet eine dienstorientierte Sichtweise. Die mobile Anwendung kann in einem Online-Offline-Szenario genutzt werden, in der die Anwendung nur sporadisch mit dem entfernten Dienst verbunden ist. Beim Entwurf wurden geeignete technische und fachliche Funktionalitäten identifiziert und gezielt durch bestehende Komponenten umgesetzt. Der Datenbankentwurf basiert auf dem objektrelationalen Ansatz für räumliche Datenbanken und bietet damit gute Möglichkeiten, derartige Daten zu verarbeiten. Die beiden Anwendungen des Informationssystems wurden mithilfe aktueller Technologien prototypisch umgesetzt und konnten ausführlich getestet werden. Dadurch können die funktionalen und die nichtfunktionalen Anforderungen als erfüllt betrachtet werden.

Das gewählte Anwendungsszenario ist grundsätzlich generalisierbar, was im Software-Entwurf weitestgehend berücksichtigt wurde. Demnach ist es nicht nur problemlos möglich, den Entwurf für andere Zielgruppen als Rollstuhlfahrer zu nutzen; vielmehr können diese Zielgruppen bereits in die implementierte Software integriert werden, da die Mandantenfähigkeit gewährleistet ist.

Das Informationssystem bietet Nutzern eine geeignete Plattform, um in Zusammenarbeit ein gemeinsames Ziel zu erreichen. Dabei werden individuelle Nutzerpräferenzen durch eine direkte und indirekte Personalisierung berücksichtigt. Die Evaluierung hat dabei gezeigt, dass Eigenschaften des Web 2.0 auf mobile Anwendungen übertragbar sind und zusammen mit ortsbezogenen Dienste hohe Synergieeffekte aufweisen. Dennoch lässt sich über die Wirtschaftlichkeit und über den Effekt des mobilen Informationssystems erst nach längerer Nutzung, durch viele Teilnehmer, eine Aussage treffen.

7.3 Ausblick

In Folgenden werden einige technische und organisatorische Weiterentwicklungsmöglichkeiten von Trailblazers Beta, sowie mögliche Fortschritte ortsbezogener Dienste und des Web 2.0 aufgezeigt.

7.3.1 Trailblazers Gamma

Die Trailblazers-Software ist als Teil einer Web 2.0-Plattform konzipiert. Die Entwicklung der Software, ist gemäß den Eigenschaften des Web 2.0 bezüglich der Software-Lebenszyklen, einer ständigen Weiterentwicklung unterworfen.

7.3.1.1 Veröffentlichung und Evaluierung

Die durchgeführte Evaluierung hat bereits gezeigt, dass ortbeszogene Dienste im Paradigma des Web 2.0 viel versprechende Möglichkeiten für mobile Anwendungen bieten. Um empirische Aussagen über diese These machen zu können, bedarf es jedoch einer Versuchsreihe mit mehreren hundert Probanden. Dazu soll der gesamte Trailblazers-Dienst im Internet[1] frei zugänglich gemacht werden, sowie die Client-Anwendung kostenlos geladen und genutzt werden können.

7.3.1.2 Portierung auf die Java-Plattform

Um eine kritische Masse von Nutzern zu erreichen, ist es unabdingbar, die mobile Client-Anwendung auf die Java-Plattform zu migrieren. Dies kann, nach Meinung des Autors, mit dem in dieser Arbeit beschriebenen Entwurf der mobilen Anwendung leicht umgesetzt werden.

7.3.1.3 Technische Anpassungen

Der Software-Entwurf basiert auf aktuell verfügbaren Technologien, die sich im Bereich mobiler Informationssysteme kontinuierlich weiter entwickeln. Die Spezifizierung weiterer Web-Service-Standards (vgl. OASIS (2006) und W3C (2007)), wie z. B. WS-Notification, schafft neue technologische Möglichkeiten, die für zukünftige Versionen der Trailblazers-Software berücksichtigt werden sollten. Dies betrifft auch Sicherheitsaspekte, die im Rahmen dieser Arbeit nur rudimentär betrachtet werden konnten. Die entworfene dienstorientierte Architektur bietet dazu eine geeignete Grundlage.

Aktuell werden verschiedene Middlewares und Rahmenwerke für orts- und kontextbezogene Dienste der nächsten Generation, wie z. B. Trax (Martens u. a. (2006)) oder Nimbus (Roth (2004)) entwickelt. Sobald diese nutzbar sind, scheint eine Überprüfung der Einsetzbarkeit in der Trailblazers-Software sinnvoll.

7.3.1.4 Erweiterungen

Die in dieser Arbeit vorgestellten Anwendungsfälle decken nur einen Teil der möglichen Funktionalitäten des mobilen Informationssystems ab. Nach Meinung des Autors bieten sich in naher Zukunft folgende Erweiterungen an:

[1] http://www.trailblazers.de, Zugriffsdatum: 04.05.2007

Integration personalisierter Karten Die Integration personalisierter Karten ist ein möglicher Schritt, um Nutzer an die Trailblazers-Plattform zu binden. Die Erstellung durch die Nutzer soll einfach möglich sein, so dass z. B. Karten, die das innere eines Gebäudes repräsentieren, in das Informationssystem integriert werden können. Im Software-Entwurf wurde dies bereits weitestgehend berücksichtigt: Die genutzte Komponente für die Kartendarstellung unterstützt Grafikdateien als Karten, die Positionsbestimmung für WLAN kann integriert werden, der Kartendienst in der Server-Anwendung ist zur Laufzeit austauschbar und das Datenmodell unterstützt verschiedene Geo-Referenzrahmen. Weiterhin wäre die Integration spezialisierter, nicht-linearer Karten denkbar, die z. B. einen Nahverkehrsplan darstellen. Mit der Tracking-Funktionalität könnten diese dem Nutzer Informationen in Echtzeit über den Nahverkehr in seiner Stadt geben.

Trailblazers-SKD Viele erfolgreiche Web 2.0 Anwendungen, wie Google Maps oder Microsoft Virtual Earth bieten ein so genanntes Software Development Kit (SDK) an. Durch dieses können gewillte Nutzer die Dienste mit eigenen Funktionalitäten erweitern und diese wiederum anderen Nutzern zur Verfügung stellen. Eine Trailblazers-SDK für die mobile Anwendung ist daher denkbar. So sind z. B. Erweiterungen für Jogger möglich, die Statistiken über Kalorienverbrauch und gelaufene Kilometer verwalten, oder Erweiterungen, die eine Geo-Caching-Funktionalität bereitstellen. Die Erweiterungen können mit einem SDK als so genannte Plug-In-Module erstellt werden. Dazu würde die mobile Client-Anwendung einmalig so erweitert werden, dass diese bestimmte Schnittstellen für generische Funktionalitäten bereitstellt. Mit dem SDK werden diese Schnittstellen in eigenen, ausführbaren Modulen (Assemblies) implementiert, welche zur Laufzeit in die Anwendung eingebunden werden können. Die Entwicklung von Plug-In-Modulen für .NET-basierte Anwendungen ist z. B. in Nowak und Weber (2006) beschrieben.

Online-Dienst Die steigende Verfügbarkeit mobiler Internetverbindungen könnte Trailblazers in den nächsten Jahren in einen Hybrid-Online-Dienst verändern, bei dem die Nutzung einer Internet-Verbindung stärker ausgeprägt ist. Dieses bietet eine Reihe neuer Anwendungsfälle; so könnten dann z. B. lokale Internet-Suchdienste, gemäß dem Mash-Up-Paradigma direkt in die mobile Anwendung integriert werden.

7.3.2 Ortsbezogene Dienste der übernächsten Generation

Ortsbezogene Dienste gehören einem weiterhin stark wachsenden Forschungsgebiet an. Der technologische Fortschritt wird die Nutzung derartiger Dienste zudem weiter vorantreiben. Aktuelle Forschungsthemen sind dabei insbesondere Systeme zur Positionsbestimmung, Standards und Protokolle sowie Datenschutz (vgl. Küpper (2005), Martens u. a. (2006)). So wird es nach den in dieser Arbeit beschriebenen ortsbezogenen Diensten der

nächsten Generation (vgl. Abschn. 3.2) auch eine nachfolgende Generation geben. Derartige Dienste der übernächsten Generation könnten sich vor allem dadurch auszeichnen, dass diese den gesamten Kontext eines Nutzers, nicht nur die aktuelle Position, bei einer möglichen Entscheidungsfindung berücksichtigen (vgl. z. B. Raubal (2006)).

7.3.3 Web 3.0

Die Frage, ob das Web 2.0 die Informationstechnologie nachhaltig verändert (vgl. Studie in Abschn. 1.1), kann zum jetzigen Zeitpunkte noch nicht beantwortet werden. In nichtwissenschaftlicher Populärliteratur[2] werden bereits die Eigenschaften des Web 3.0 skizziert, die zum jetzigen Zeitpunkt jedoch nur vermutet werden können. Sicher scheint hingegen eine Fortsetzung der durch das Web 2.0 eingeschlagenen Entwicklung. Sehr wahrscheinlich wird das so genannte *Semantic Web* (Berners-Lee u. a. (2001)) als Web 3.0 bezeichnet werden. Beim Semantic Web existieren neben den eigentlichen Daten zusätzliche Metadaten, welche die Semantik der Inhalte formal festlegen. Dabei werden Ontologien für semantische Begriffe und Zusammenhänge gebildet. Die Generierung derartiger Metadaten kann ähnlich komplex und aufwendig sein, wie z. B. das Sammeln ortsebzogener Daten für barrierefreie Wege und kann daher mit dem aus dem Web 2.0 bekannten *Bottom-To-Top-Ansatz* gut umgesetzt werden.

[2] Vgl. z. B. Artikel „Extreme Informatik“ in dem Magazin Technology Review, Ausgabe 05/07.

Literaturverzeichnis

[ADAC 2003] ADAC: *ADAC-Studie zur Barrierefreiheit in deutschen Städten*. 2003. – URL http://www.handicap-info.de/DesktopDefault.aspx?tabid=1238&tabindex=0&centermoduleid=10218&dm=true&path=admin//ADAC%20Stadt.html. – Zugriffsdatum: 09.02.2007

[Akasaka und Onisawa 2004] AKASAKA, Yuta (Hrsg.) ; ONISAWA, Takehisa (Hrsg.) ; IEEE Computer Society (Veranst.): *Construction of Pedestrian Navigation Svstem and Its Evaluation*. IEEE, 2004

[Barbará 1999] BARBARÁ, Daniel: Mobile Computing and Databases - A Survey. In: *IEEE Transactions on Knowledge and Data Enginerring* 11 (1999), Nr. 1

[Bazijanec und Turowski 2004] BAZIJANEC, Bettina (Hrsg.) ; TUROWSKI, Klaus (Hrsg.) ; Gesellschaft für Informatik e.V. (Veranst.): *Potenziale des Mobile Commerce*. Mobile Economy - Transaktionen, Prozesse, Anwendungen und Dienste, Proceedings zum 4. Worshop Mobile Commerce, Februar 2004

[Bächle 2006] BÄCHLE, Michael: Social Software. In: *Informatik Spektrum* 29 (2006), April, Nr. 2

[Beck 2002] BECK, Kent: *Test Driven Development. By Example.* Addison-Wesley Longman, Amsterdam, 2002

[Beresford und Stajano 2004] BERESFORD, Alastair (Hrsg.) ; STAJANO, Frank (Hrsg.) ; IEEE (Veranst.): *Mix Zones: User Privacy in Location-aware Services*. IEEE, 2004. (roceedings of the Second IEEE Annual Conference On Pervasive Computing and Communications Workshops)

[Berners-Lee 1996] BERNERS-LEE, Tim: *The World Wide Web: Past, Present and Future*. August 1996. – URL http://www.w3.org/People/Berners-Lee/1996/ppf.html. – Zugriffsdatum: 01.04.2007

[Berners-Lee u. a. 2001] BERNERS-LEE, Tim ; HENDLER, James ; LASSILA, Ora: The Semantic Web : a new form of Web content that is meaningful to computers will unleash a revolution of new possibilities. In: *Scientific American* 284 (2001), May, S. 34–43

[B'Far 2005] B'FAR, Reza: *Mobile Computing Principles - Designing and Developing Mobile Applications with UML and XML*. Cambridge University Press, 2005

[Bhattacharyya und Scott Ellison 2006] BHATTACHARYYA, Rena ; SCOTT ELLISON, IDC R.: *U.S Wireless Business Location-Based Services 2006-2010 Forecast*. September 2006. – Zugriffsdatum: 09.01.2007

[Bieberstein u. a. 2006] BIEBERSTEIN, Norbert ; BOSE, Sanjay ; FIAMMANTE, Marc ; JONES, Keith ; SHAH, Rawn ; FOO, Linda (Hrsg.): *Service Oriented Architecture Compass*. IBM Press, 2006

[BITKOM 2007a] BITKOM: *3 Millionen Deutsche besitzen bereits ein mobiles Navigationsgerät*. April 2007. – URL http://www.bitkom.org/45523_45415.aspx. – Zugriffsdatum: 18.01.2007

[BITKOM 2007b] BITKOM: *Online-Werbung im 1. Quartal 2007*. April 2007. – URL http://www.bitkom.org/45177_45097.aspx. – Zugriffsdatum: 18.01.2007

[BMWi 2006] BMWI, Bundesministerium für Wirtschaft und Technologie: Rechtsfragen beim E-Business. In: *e-f@cts* 09 (2006), S. 1 – 8

[Book u. a. 2005] BOOK, Matthias (Hrsg.) ; GRUHN, Volker (Hrsg.) ; HÜLDER, Malte (Hrsg.) ; SCHÄFER, Clemens (Hrsg.) ; Gesellschaft für Informatik e.V. (Veranst.): *Der Einfluss verschiedener Mobilitätsgrade auf die Architektur von Informationssystemen*. Mobile Business - Processes, Platfroms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), 2005

[Borriello 2005] BORRIELLO, Gaetano: Delivering Real-World Ubiquitous Location Systems. In: *Communications of the ACM* 45 (2005), Nr. 3

[Bulander u. a. 2005] BULANDER, Rebecca (Hrsg.) ; SCHIEFER, Gunther (Hrsg.) ; DECKER, Michael (Hrsg.) ; Gesellschaft für Informatik e.V. (Veranst.): *Anonymity by Design - Eine Architektur zur Gewährleistung von Kundenschutz im mobilen Marketing*. Mobile Business - Processes, Platfroms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), 2005

[Bundesministerin der Justiz 1990] BUNDESMINISTERIN DER JUSTIZ: *Bundesdatenschutzgesetz*. 1990. – URL http://www.gesetze-im-internet.de/bdsg_1990/index.html. – Zugriffsdatum: 31.01.2007

[Burak und Sharon 2004] BURAK, Asaf (Hrsg.) ; SHARON, Taly (Hrsg.) ; ACM International Conference Proceeding Series (Veranst.): *Usage patterns of FriendZone: mobile location-based community services*. Bd. *83*. Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia, 2004

[Chappell 2006] CHAPPELL, David: *Einführung in .NET Framework 3.0*. August 2006. – URL http://www.microsoft.com/germany/msdn/library/net/EinfuehrungInNETFramework30.mspx?mfr=true. – Zugriffsdatum: 02.05.2007

[Claus und Schwill 2003] CLAUS, Volker ; SCHWILL, Andreas: *Duden. Informatik.* Dudenverlag Mannheim Leipzig Wien Zürich, 2003

[Coutaz 2005] COUTAZ, Joelle: Context is Key. In: *Communications of the ACM* 45 (2005), Nr. 3

[Davies u. a. 2006] DAVIES, Jonathan J. ; BERESFORD, Alastair R. ; HOPPER, Andy: Scalable, Distributed, Real-Time Map Generation. In: *IEEE Pervasive Computing* 5 (2006), Nr. 4, S. 47–54. – ISSN 1536-1268

[Dieberger u. a. 2001] DIEBERGER, Andreas ; HÖÖK, Kristina ; SVENSSON, Martin ; LÖNNQVIST, Peter: *Social Navigation Research Agenda*. 2001. – URL http://people.dsv.su.se/~peterl/publications/chi_2001_short_paper.pdf. – Zugriffsdatum: 16.02.2007

[Dix u. a. 2000] DIX, Alan ; RODDEN, Tom ; DAVIES, Nigel ; TREVOR, Jonathan ; FRIDAY, Adrian ; PALFREYMAN, Kevin: Exploiting space and location as a design framework for interactive mobile systems. In: *ACM Transactions on Computer-Human Interaction (TOCHI)* 7 (2000), Nr. 3, S. 285–321

[Domer u. a. 2004] DOMER, Jason ; NANJA, Muthi ; SRINIVAS, Suresh: Comparative performance analysis of mobile runtimes on Intel XScale® technology. In: *Proceedings of the 2004 workshop on Interpreters, virtual machines and emulators* 3 (2004), S. 51–57

[Dröge u. a. 2006] DRÖGE, Ruprecht ; NOWAK, Peter ; WEBER torsten: *Programmieren mit dem .NET Compact Framework*. Microsoft Press Unterschleißheim, 2006

[Dürr u. a. 2004] DÜRR, Frank ; HÖNLE, Nicola ; NICKLAS, Daniela ; BECKER, Christian ; ROTHERMEL, Kurt: Nexus - A Platform for Context-Aware Applications. In: *1. Fachgespräch Ortsbezogene Anwendungen und Dienste der GI-Fachgruppe KuVS* 1 (2004), S. 15–18

[Dunkel und Holitschke 2003] DUNKEL, Jürgen ; HOLITSCHKE, Andreas: *Softwarearchitektur für die Praxis*. Springer-Verlag Berlin Heidelberg New York, 2003

[Dustdar u. a. 2003] DUSTDAR, Scharam ; GALL, Harald ; HAUSWIRTH, Manfred: *Software-Architekturen für Verteilte Systeme*. Springer-Verlag Berlin Heidelberg New York, 2003

[Eckert 2004] ECKERT, Claudia: *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, 2004

[Ecma 2006] ECMA: *Ecma International*. 01 2006. – URL http://www.ecma-international.org/default.htm. – Zugriffsdatum: 02.09.2006

[Egenhofer 1994] EGENHOFER, Max (Hrsg.) ; IEEE Computer Society (Veranst.): *Spatial SQL: A Query and Presentation Language*. IEEE Computer Society Transactions on Knwoledge and Data Engineering 6, 1994

[EMIC 2007] EMIC: *EMIC Location and Mapping Framework*. January 2007. – URL http://www.microsoft.com/emea/emic/downloadcenter.mspx. – Zugriffsdatum: 03.01.2007

[Engelbart 1962] ENGELBART, Douglas C.: Augmenting Human Intellect: A Conceptual Framework / Stanford Research Institute. URL http://www.bootstrap.org/augdocs/friedewald030402/augmentinghumanintellect/ahi62index.html, 1962. – Forschungsbericht. Zugriffsdatum: 27.02.2007

[Engelbart 1972] ENGELBART, Douglas C.: Coordinated Information Services for a Discipline - Or Mission-Oriented Community / Bootstrap Institute. URL http://www.bootstrap.org/augdocs/augment-12445.htm, 1972. – Forschungsbericht. Zugriffsdatum: 27.02.2007

[Ericcsson 2006] ERICCSSON: *Mobile Positioning System*. November 2006. – URL http://www.ericsson.com/mobilityworld/sub/open/technologies/mobile_positioning/index.html. – Zugriffsdatum 28.09.2006

[EU 2003] EU, Europäische Union: *Empfehlung der Kommission vom 25. Juli 2003 zur Übermittlung von Angaben zum Anruferstandort in elektronischen Kommunikationsnetzen an um Standortangaben erweiterte Notrufdienste*. Juli 2003. – URL http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:32003H0558:DE:HTML. – Zugriffsdatum 26.09.2006

[FCC 2006] FCC, Federal Communications Commission: *Enhanced 911 - Wireless Services*. 2006. – URL http://www.fcc.gov/911/enhanced/. – Zugriffsdatum: 15.11.2006

[Fielding 2000] FIELDING, Roy T.: *Architectural Styles and the Design of Network-based Software Architectures*, University of California, Irvine, Dissertation, 2000. – URL http://www.ics.uci.edu/%7Efielding/pubs/dissertation/fielding_dissertation.pdf. – Zugriffsdatum: 03.11.2006

[Fritsch und Muntermann 2005] FRITSCH, Lothar (Hrsg.) ; MUNTERMANN, Jan (Hrsg.) ; Gesellschaft für Informatk e. V. (Veranst.): *Aktuelle Hinderungsgründe für den Erfolg von Location Based Service-Angeboten*. Mobile Business - Processes, Platfroms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), Februar 2005

[Galileo-Industries 2006] GALILEO-INDUSTRIES: *Galileo Satellitensystem*. 2006. – URL http://www.galileo-industries.net/galileo/galileo.nsf/pages/CONT-5ZRH9H?openDocument&e. – Zugriffsdatum: 07.11.2006

[Gamma u. a. 1995] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional Computing Series, 1995

[Gartner 2006a] GARTNER: *Gartner's 2006 Emerging Technologies Hype Cycle Highlights Key Technology Themes*. September 2006. – URL http://www.gartner.com/it/page.jsp?id=495475. – Zugriffsdatum 27.09.2006

[Gartner 2006b] GARTNER: *Predicts 2007: Carriers Will Spend Billions to Try to Survive IP Revolution*. November 2006. – Zugriffsdatum: 18.02.2007

[Gartner 2006c] GARTNER: *Quarterly Statistics: PDA and Smartphone Shipment Forecast*. Dezember 2006. – Zugriffsdatum: 16.01.2007

[Gasenzer 2001] GASENZER, Rolf: Positionsbasierte Leistungsangebote für den mobilen Handel. In: *HMD - Praxis der Wirtschaftsinformatik - Mobile Commerce* 220 (2001), S. 37 – 51

[Geocaching.de 2007] GEOCACHING.DE: *Geocaching.de*. 2007. – URL http://www.geocaching.de/. – Zugriffsdatum: 09.02.2007

[GeoFrameworks 2007] GEOFRAMEWORKS: *GPS.NET*. März 2007. – URL http://www.geoframeworks.com/. – Zugriffsdatum: 05.03.2007

[Gershman und Fano 2005] GERSHMAN, Anatole ; FANO, Andrew: Examples of commercial applications of ubiquitous computing. In: *Commun. ACM* 48 (2005), Nr. 3, S. 71. – ISSN 0001-0782

[Gibson und Erle 2006] GIBSON, Rich ; ERLE, Schuyler: *Google Maps Hacks*. O'Reilly, 2006

[Google 2006a] GOOGLE: *Google Earth*. September 2006. – URL http://earth.google.de/. – Zugriffsdatum 14.09.2006

[Google 2006b] GOOGLE, Inc.: *Google Maps*. September 2006. – URL http://maps.google.de. – Zugriffsdatum 14.09.2006

[Gray u. a. 1996] GRAY, Jim (Hrsg.) ; HELLAND, Pat (Hrsg.) ; O'NEIL, Patrick (Hrsg.) ; SHASHA, Dennis (Hrsg.) ; ACM (Veranst.): *The dangers of replication and a solution*. SIGMOD '96: Proceedings of the 1996 ACM SIGMOD international conference on Management of data, 1996

[Grossman 2006] GROSSMAN, Lev: Time's Person of the Year: You. In: *Time Magazine* (2006), Dezember. – URL http://www.time.com/time/magazine/article/0,9171,1569514,00.html. – Zugriffsdatum: 11.02.2007

[Hammerschall 2005] HAMMERSCHALL, Ulrike: *Verteilte Systeme und Anwendungen - Architekturkonzepte, Standards und Middleware-Technologien*. Pearson Studium, 2005

[Hansen und Neumann 2005] HANSEN, Hans R. ; NEUMANN, Gustaf: *Wirtschaftsinformatik 1*. Lucius + Lucius, UTB für Wissenschaft, 2005

[Hansmann u. a. 2003] HANSMANN, Uwe ; MERK, Lothar ; NICKLOUS, Martin S. ; STOBER, Thomas: *Pervasive Computing Handbook*. Sppringer Verlag, Berlin, 2003

[Hariharan u. a. 2005] HARIHARAN, Ramaswamy (Hrsg.) ; KRUMM, John (Hrsg.) ; HORVITZ, Eric (Hrsg.): *Web-Enhanced GPS*. Proceedsings of First International Workshop, LoCA 2005, Mai 2005

[Helal u. a. 2004] HELAL, Abdelsalam (. (Hrsg.) ; MOORE, Steven E. (Hrsg.) ; RAMACHANDRAN, Balaji (Hrsg.) ; IEEE Computer Society (Veranst.): *Drishti: An Integrated Navigation System for Visually Impaired and Disabled*. Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04) 2004, 2004

[Helal 2002a] HELAL, Sumi: Pervasive Java. In: *IEEE Pervasive Computing* 1 (2002), Nr. 1, S. 82–85

[Helal 2002b] HELAL, Sumi: Pervasive Java, Part II. In: *IEEE Pervasive Computing* 1 (2002), Nr. 2, S. 85–89

[Herden u. a. 2006] HERDEN, Sebastian ; GÓMEZ, Jorge M. ; RAUTENSTRAUCH, Claus ; ZWANZIGER, André: *Software-Architekturen für das E-Busiiness*. Springer-Verlag Berlin Heidelberg, 2006

[Heuer u. a. 2001] HEUER, Andreas ; SAAKE, Gunter ; SATTLER, Kai-Uwe: *Datenbanken kompakt*. mitp-Verlag, 2001

[Hibernate 2007] HIBERNATE: *NHibernate*. April 2007. – URL http://www.hibernate.org/. – Zugriffsdatum: 04.02.2007

[Hill u. a. 2004] HILL, David ; WEBSTER, Brentor ; JEZIERSKI, Edward A. ; VASIREDDY, Srinath ; AL-SABT, Mo: Smart Client Architecture and Design Guide. In: *Microsoft patterns & practices* (2004)

[Hippner 2006] HIPPNER, Hajo: Bedeutung, Anwendung und Einsatzpotentiale von Social Software. In: *HMD - Praxis der Wirtschaftsinformatik - Social Software* 252 (2006), S. 6 – 16

[Höpfner u. a. 2005] HÖPFNER, Hagen ; TÜRKER, Can ; KÖNIG-RIES, Birgitta: *Mobile Datenbanken und Informationssysteme*. dpunkt.verlag Heidelberg, 2005

[Ibach u. a. 2004] IBACH, Peter (Hrsg.) ; HÜBNER, Tobias (Hrsg.) ; SCHWEIGERT, Martin (Hrsg.) ; Chaos Communications Congress (Veranst.): *MagicMap*. Chaos Communications Congress 2004, Dezember 2004

[Jacobsen 1995] JACOBSEN, Ivar: *Object-Oriented Software Engineering: A Use Case Driven Approach*. Addison-Wesley, 1995

[Jacobson u. a. 1999] JACOBSON, Ivar ; BOOCH, Grady ; RUMBAUGH, James: *Unified Software Development Process*. Addison Wesley, 1999

[Jagoe 2003] JAGOE, Andrew: *Mobile Location Service - The Definite Guide*. Pearson Education, Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2003

[Janus 2007] JANUS: *Janus WinForms*. April 2007. – URL http://www.componentsource.com/products/janus-winforms-controls-suite/index-de.html. – Zugriffsdatum: 27.02.2007

[JCP 2006] JCP: *Java Community Process*. 01 2006. – URL http://jcp.org/en/home/index. – Zugriffsdatum: 04.11.2007

[Kaapke 2002] KAAPKE, Andreas: *Kundenbindung mit System*. Kap. Kundenkarten als Instrument der Kundenbindung, S. 177–192, Deutscher Fachverlag, 2002

[Kahlbrandt 2001] KAHLBRANDT, Bernd: *Software-Engineering mit der Unified Modeling Language*. Springer-Verlag Berlin Heidelberg New York, 2001

[Klein 2005] KLEIN, Andreas (Hrsg.) ; Gesellschaft für Informatik e.V. (Veranst.): *Systematisierung und Beurteilung von Micropayment-Systemen aus Nachfragersicht*. Mobile Business - Processes, Platfroms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), 2005

[Kray u. a. 2003] KRAY, Christian (Hrsg.) ; ELTING, Christian (Hrsg.) ; LAAKSO, Katri (Hrsg.) ; COORS, Volker (Hrsg.): *Presenting route instructions on mobile devices*. ACM International Conference on Intelligent User Interfaces, Proceedings of the 8th international conference on Intelligent user interfaces, 2003

[Küpper 2005] KÜPPER, Axel: *Location-Based Services - Fundamentals and Operation*. Wiley Ltd. West Sussex, 2005

[LAG 2007] LAG, Hamburger Landesarbeitsgemeinschaft für behinderte Menschen e.V.: *Hamburger Stadtführer für Rollstuhlfahrer*. 2007. – URL http://www.lagh-hamburg.de/. – Zugriffsdatum: 09.02.2007

[Loeffler u. a. 2005] LOEFFLER, William ; VASIREDDY, Srinath ; PACE, Eugenio: Smart Client - Composite UI Application Block. In: *Microsoft patterns & practices* (2005), Dezember. – URL http://msdn2.microsoft.com/en-us/library/aa480450.aspx. – Zugriffsdatum: 07.03.2007

[LOMS 2007] LOMS: *LOMS: Local Mobile Services*. Mai 2007. – URL http://www.loms-itea.org/index.php

[Martens u. a. 2006] MARTENS, Johannes ; TREU, Georg ; RUPPEL, Peter ; WEISS, Diana ; KÜPPER, Axel ; LINNHOFF-POPIEN, Claudia: Eine Plattform zur Unterstützung von proaktiven ortsbezogenen Mehrbenutzer-Anwendungen. In: *3. GI/ITG KuVS Fachgespräch Örtsbezogene Anwendungen und Dienste", Freie Universität Berlin* (2006), September, S. 65–68

[Möhlenbruch und Schmieder 2001] MÖHLENBRUCH, Dirk ; SCHMIEDER, Ulf-Marten: Gestaltungsmöglichkeiten und Entwicklungspotentiale des Mobile Marketing. In: *HMD - Praxis der Wirtschaftsinformatik - Mobile Commerce* 220 (2001), S. 15 – 26

[Microsoft 2006a] MICROSOFT: *Adventure Works Cinema 1.0*. Januar 2006. – URL http://www.microsoft.com/germany/msdn/library/vs2005/samples/default.mspx?mfr=true. – Zugriffsdatum: 06.12.2006

[Microsoft 2006b] MICROSOFT: *Imagine Cup 2006*. Mai 2006. – URL http://www.imaginecup.info. – Zugriffsdatum 26.09.2006

[Microsoft 2006c] MICROSOFT: *MapPoint*. November 2006. – URL http://www.microsoft.com/mappoint/default.mspx. – Zugriffsdatum 24.06.2006

[Microsoft 2006d] MICROSOFT: *Microsoft.Ink*. März 2006. – URL http://www.microsoft.com/downloads/details.aspx?FamilyID=84bbefa4-7047-41df-8583-e3bdbf9d805f&DisplayLang=de. – Zugriffsdatum: 05.05.2007

[Microsoft 2006e] MICROSOFT: *Mobile Client Software Factory*. Juli 2006. – URL http://www.microsoft.com/downloads/details.aspx?familyid=f9176708-9f57-4c0f-97fb-f9c65a9bbf22&displaylang=en. – Zugriffsdatum: 16.04.2007

[Microsoft 2006f] MICROSOFT: *NET Compact Framework*. 01 2006. – URL http://msdn.microsoft.com/smartclient/understanding/netcf. – Zugriffsdatum: 05.02.2007

[Microsoft 2006g] MICROSOFT: *Smart Client Software Factory*. June 2006. – URL http://msdn2.microsoft.com/en-us/library/aa480482.aspx. – Zugriffsdatum: 19.02.2007

[Microsoft 2006h] MICROSOFT: *Web Service Software Factory*. Dezember 2006. – URL http://msdn2.microsoft.com/en-us/library/aa480534.aspx. – Zugriffsdatum: 11.12.2006

[Microsoft 2007] MICROSOFT: *Virtual Earth Interactive SDK*. April 2007. – URL http://dev.live.com/virtualearth/sdk/. – Zugriffsdatum: 03.04.2007

[Microsoft-Research 2006] MICROSOFT-RESEARCH: *SLAM - Social Location Annotation Mobile*. Oktober 2006. – URL http://www.msslam.com/faq.aspx. – Zugriffsdatum 18.10.2006

[MsSqlSpatial 2007] MSSQLSPATIAL: *MsSqlSpatial - Spatial Extensions for SQL Server 2005*. April 2007. – URL http://www.codeplex.com/MsSqlSpatial. – Zugriffsdatum: 02.04.2007

[Mutschler und Specht 2004] MUTSCHLER, Bela ; SPECHT, Günther: *Mobile Datenbanksysteme*. Springer-Verlag Berlin Heidelberg, 2004

[Myles u. a. 2003] MYLES, Ginger ; FRIDAY, Adrian ; DAVIES, Nigel: Preserving Privacy in Environments with Location-Based Applications. In: *IEEE Pervasive Computing* 2 (2003), S. 56 – 64

[Navteq 2007] NAVTEQ: *Navteq*. 2007. – URL http://www.navteq.com/index2.html. – Zugriffsdatum: 09.02.2007

[Neable 2002] NEABLE, Craig: The .NET Compact Framework. In: *IEEE Pervasive Computing* 1 (2002), Nr. 4, S. 84–87

[NGenerics 2007] NGENERICS: *A class library providing generic data structures and algorithms not implemented in the standard .NET Framework*. April 2007. – URL http://www.codeplex.com/NGenerics. – zugriffsdatum: 09.04.2007

[Nova u. a. 2006] NOVA, Nicolas (Hrsg.) ; GIRARDIN, Fabien (Hrsg.) ; MOLINARI, Gaëlle (Hrsg.) ; DILLENBOURG, Pierre (Hrsg.): *The Underwhelming Effects of Location-Awareness on Collaboration in a Pervasive Game*. International Conference on the Design of Cooperative Systems, May 2006

[Nowak und Weber 2006] NOWAK, Peter ; WEBER, Torsten: Server-Kommunikation per Instant Messaging. In: *dot.net Magazin* (2006), Dezember

[OASIS 2006] OASIS: *OASIS Web Services Notification (WSN) TC*. Oktober 2006. – URL http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsn. – Zugriffsdatum: 18.03.2007

[OGC 2007a] OGC: *Open Geospatial Concortium*. April 2007. – URL http://www.opengeospatial.org/. – Zugriffsdatum: 29.03.2007

[OGC 2007b] OGC: *OpenGIS® Implementation Specification for Geographic information - Simple feature access - Part 2: SQL option*. 2007. – URL http://www.opengeospatial.org/standards/sfs. – Zugriffsdatum 29.03.2007

[openstreetmap.org 2006] OPENSTREETMAP.ORG: *OpenStreetMap*. Dezember 2006. – URL http://wiki.openstreetmap.org/index.php/Main_Page. – Zugriffsdatum: 30.12.2006

[Openwave 2006] OPENWAVE: *Location Studio*. November 2006. – URL http://www.openwave.com. – Zugriffsdatum 28.10.2006

[O'Reilly 2005] O'REILLY, Tim: *What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software*. O'Reilly Network. September 2005. – URL http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html. – Zugriffsdatum: 11.10.2006

[Pashtan 2005] PASHTAN, Ariel: *Mobile Web Services*. Cambridge University Press Cambridge, 2005

[Pfeifer u. a. 2006] PFEIFER, Tom (Hrsg.) ; HAUGHNEY, Hélène (Hrsg.) ; DOWNES, Barry (Hrsg.) ; IEEE Computer Society (Veranst.): *UGetMobile End-user Mobile Publishing Platform*. Proceedings of the 8th IEEE International Conference on E-commerce Technology and the 3th IEEE International Conference on Enterprise Computing, 2006

[Ranganathan u. a. 2004] RANGANATHAN, Anand (Hrsg.) ; AL-MUHTADI, Jalal (Hrsg.) ; CHETAN, Shiva (Hrsg.) ; CAMPBELL, Roy (Hrsg.) ; MICKUNAS, M. D. (Hrsg.) ; ACM (Veranst.): *MiddleWhere: a middleware for location awareness in ubiquitous computing applications*. Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware, 2004

[Rannenberg u. a. 2005] RANNENBERG, Kai (Hrsg.) ; ALBERS, Andreas (Hrsg.) ; FIGGE, Stefan (Hrsg.) ; RADMACHER, Mike (Hrsg.) ; ROSSNAGEL, Heiko (Hrsg.) ; Gesellschaft für Informatik e.V. (Veranst.): *Mobile Commerce - Forschungsfragen am Scheideweg der Mobilfunkgenerationen*. Mobile Business - Processes, Platfroms, Payments, Proceedings zur 5. Konferenz Mobile Commerce Technologien und Anwendungen (MCTA 2005), 2005

[Raubal 2006] RAUBAL, Martin: Location-Based Decision Services - eine neuartige Form mobiler räumlicher Entscheidungsunterstützung. In: *Vermessung und Geoinformation* 1+2 (2006), S. 30–37

[Röckl 2005] RÖCKL, Matthias: *Konzeption und Implementierung einer Location-based Push-Architektur für mobile Navigationsdienste*, Insitut für Informatik der Ludwig-Maximilians Universität München, Diplomarbeit, Juni 2005

[Retscher 2004] RETSCHER, Günther (Hrsg.) ; IEEE Computer Society (Veranst.): *Pedestrian Navigation Systems and location-based services*. 5th IEEE International Conference on 3G Mobile Communication Technologies, 2004

[Rigaux u. a. 2001] RIGAUX, Philippe ; SCHOLL, Michel O. ; VOISARD, Agnes: *Spatial Databases. With Application to GIS*. Morgan Kaufmann Publishers, 2001

[Roth 2005] ROTH, Jörg: *Mobile Computing, 2.Auflage*. dpunkt.verlag Heidelberg, 2005

[Roth 2004] ROTH, Jörg: *A Decentralized Location Service Providing Semantic Locations*, Fernuniversität Hagen, Dissertation, Oktober 2004. – URL http://www.wireless-earth.de/paper/habilFinal_opt.pdf. – Zugriffsdatum: 28.02.2007

[Salmre 2005] SALMRE, Ivo: *Writing Mobile Code - Essential Software Engineering for Building Mobile Applications*. Addison-Wesley, Pearson Education, Upper Saddle River, NJ, 2005

[Schiller und Voisard 2004] SCHILLER, Jochen (Hrsg.) ; VOISARD, Agnes (Hrsg.): *Location-Based Services*. Morgan Kaufmann San Francisco, 2004

[Schmatz 2004] SCHMATZ, Klaus-Dieter: *Java 2 Micro Edition*. dpunkt.verlag Heidelberg, 2004

[Shekhar und Chawla 2002] SHEKHAR, Shashi ; CHAWLA, Sanjay: *Spatial Databases: A Tour*. Prentice Hall, 2002

[Siedersleben 2004] SIEDERSLEBEN, Johannes: *Moderne Software-Architekturen*. dpunkt.verlag Heidelberg, 2004

[Skiera 2006] SKIERA, Bernd: PREMIUM: Preis- und Erlösmodelle im Internet - Umsetzung und Marktchancen. In: *it - Information Technology* 48 (2006), August, S. 200 – 207

[Sourceforge 2007] SOURCEFORGE: *NetTopolgySuite*. Mai 2007. – URL http://nts.sourceforge.net/. – Zugriffsdatum: 28.03.2007

[Starke 2005] STARKE, Gernot: *Effektive Software-Architekturen - Ein praktischer Leitfaden*. Carl Hanser Verlag München Wien, 2005

[Stegelmeier u. a. 2006] STEGELMEIER, Sven (Hrsg.) ; STEIN, Martin (Hrsg.) ; THOMÉ, Mark (Hrsg.) ; WENDT, Piotr (Hrsg.) ; ASK-IT (Veranst.): *Trailblazers - A Community Driven Navigation System For Mobility Impaired People*. ASK-IT International Conference, October 2006. – URL http://www.ask-it.org/

[Sun 2006] SUN, Microsystems: *Java Plattform Mobile Edition*. 2006. – URL http://java.sun.com/javame/index.jsp. – Zugriffsdatum: 30.11.2006

[Tanenbaum und van Steen 2003] TANENBAUM, Andrew S. ; STEEN, Martin van: *Verteilte Systeme - Grundlagen und Paradigmen*. Pearson Studium, 2003

[TeleAtlas 2007a] TELEATLAS: *Community - The Key to Developing Maps of the Future*. January 2007. – URL http://www.teleatlas.com/Pub/Markets/MarketingMaterials/Whitepapers/index.htm. – Zugriffsdatum: 24.02.2007

[TeleAtlas 2007b] TELEATLAS: *TeleAtlas*. 2007. – URL http://www.teleatlas.com/Pub/Home. – Zugriffsdatum: 24.02.2007

[Thomé 2005a] THOMÉ, Mark: Java 2 Micro Edition und .NET Compact Framework - Laufzeitumgebungen für mobile Anwendungen / HAW Hamburg. 2005. – Forschungsbericht

[Thomé 2005b] THOMÉ, Mark: Mobile Informationssysteme für ortsbezogene Dienste / HAW Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06/thome/abstract.pdf, Dezember 2005. – Forschungsbericht. Zugriffsdatum: 02.09.2006

[Thomé 2006] THOMÉ, Mark: Pocket Task Timer - A Personal Approach On Location-Based Services / HAW Hamburg. URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master05-06-proj/thome/paper.pdf, Dezember 2006. – Forschungsbericht. Zugriffsdatum: 02.09.2006

[Tilson und Baxter 2004] TILSON, David (Hrsg.) ; BAXTER, Kalle Lyytinen R. (Hrsg.) ; IEEE Computer Society (Veranst.): *A Framework for Selecting a Location Based Service (LBS) Strategy and Service Portfolio*. Bd. *3*. Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04), 2004

[Tsalgatidou u. a. 2003] TSALGATIDOU, Aphrodite (Hrsg.) ; VEIJALAINEN, Jari (Hrsg.) ; MARKKULA, Jouni (Hrsg.) ; KATASONOV, Artem (Hrsg.) ; HADJIEFTHYMIADES, Stathes

(Hrsg.) ; ScanGIS (Veranst.): *Mobile E-Commerce and Location-Based Services: Technology and Requrements*. ScanGIS'2003 - Proceedings of the 9th Scandinavian Research Conference on Geographical Information Science, June 2003

[W3C 2007] W3C: *Web Service Activity*. Mai 2007. – URL http://www.w3.org/2002/ws/. – Zugriffsdatum: 03.11.2006

[Weiser 1991] WEISER, Mark: The Computer for the 21st Century. In: *Scientific American* (1991), September

[Wendt 2007] WENDT, Piotr: *Trailblazers - Eine Plattform zur Community-gestützten kollaborativen Erweiterung von Navigationskarten*, HAW Hamburg, Diplomarbeit, Mai 2007

[Westphal 2002] WESTPHAL, Ralf: *.NET Kompakt*. Spektrum Akademischer Verlag, Heidelberg, 2002

[White 2001] WHITE, James: An introduction to Java 2 micro edition (J2ME); Java in small things. In: *IEEE Proceedings of the 23rd International Conference on Software Engineering* (2001), S. 724–725

[Wigley und Wheelwright 2003] WIGLEY, Andy ; WHEELWRIGHT, Stephen: *Microsoft .NET Compact Framework - Core Reference*. Microsoft Press Redmond, Washongton, 2003

[Wolfe 2003] WOLFE, Alexander: Microsoft's Compact Framework Targets Smart Devices. In: *ACM Queue* 1 (2003), Nr. 7, S. 10–12

A Anhang

A.1 Aktivitätsdiagramme der Anwendungsfälle

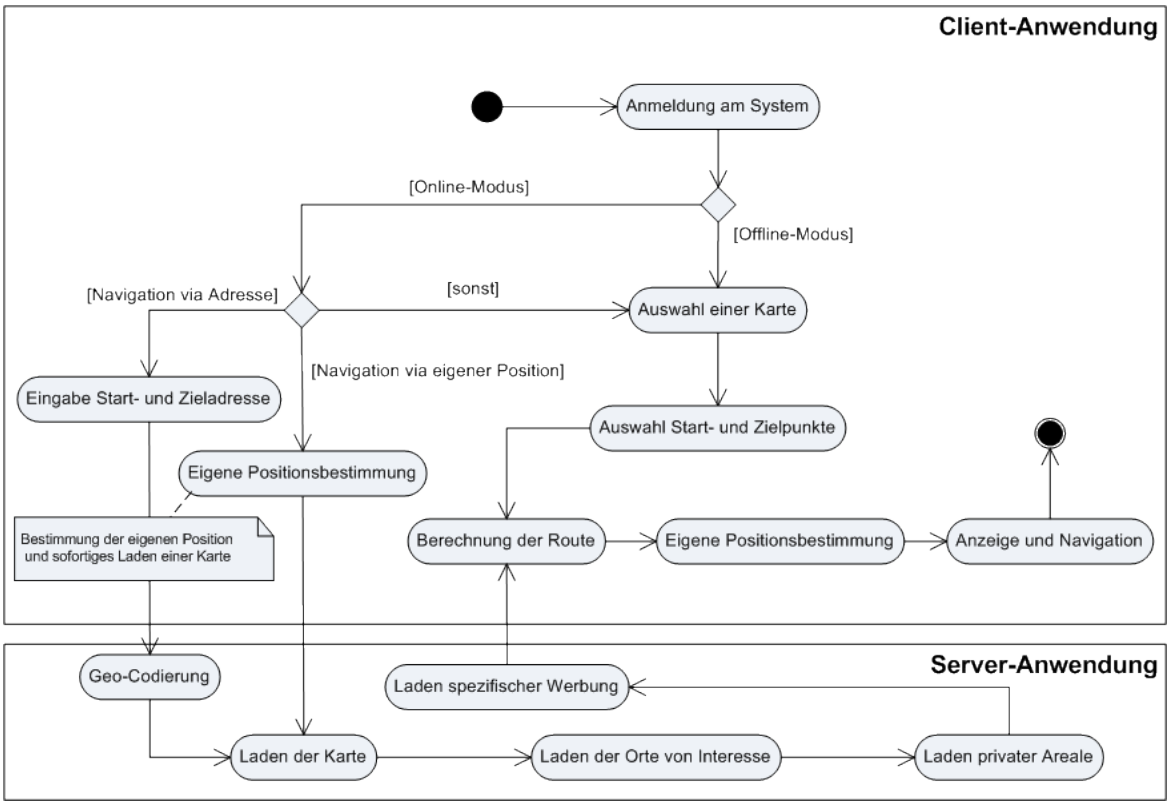

Abbildung A.1: Aktivitätsdiagramm barrierefreie Navigation

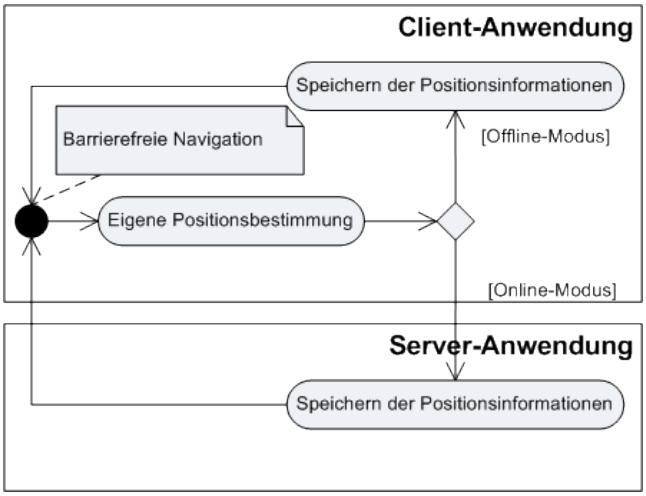

Abbildung A.2: Aktivitätsdiagramm Trail-Erkennung

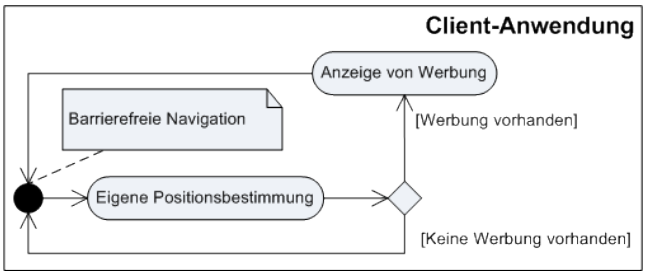

Abbildung A.3: Aktivitätsdiagramm Werbung

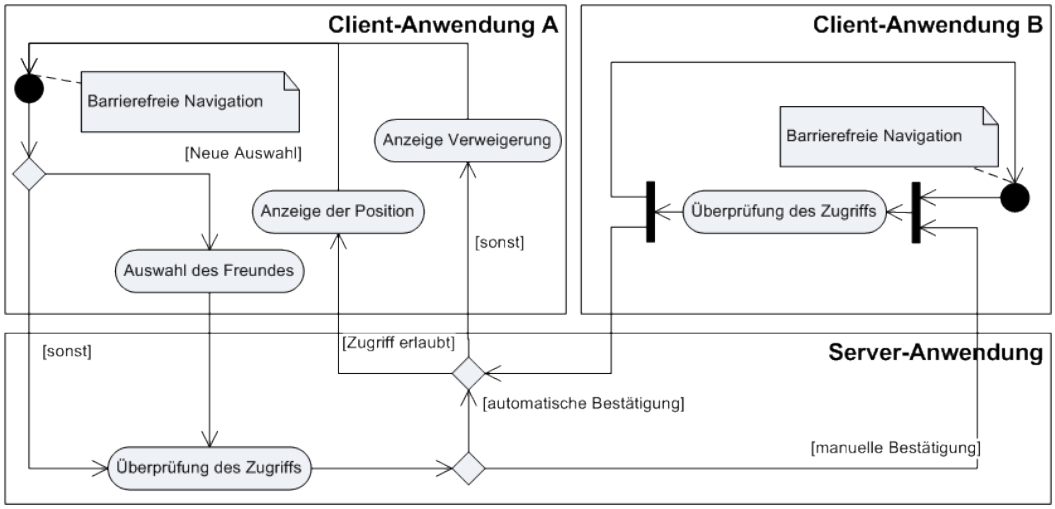

Abbildung A.4: Aktivitätsdiagramm Tracking

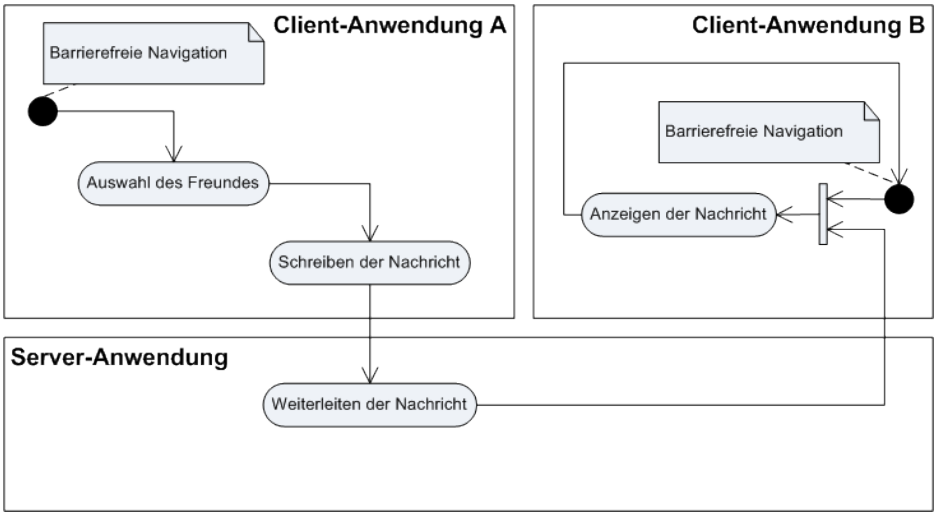

Abbildung A.5: Aktivitätsdiagramm Kommunikation

A.2 XML-Schema der Entitäten des Trailblazers-Dienstes

In den folgenden Listings werden die XML-Schemas der Entitäten aus dem Datenvertrag gezeigt.

```
<xs:complexType name="User">
 <xs:sequence>
  <xs:element name="UserId" type="clr:guid" />
   <xs:element name="Login" type="xs:string" />
   <xs:element name="Password" type="xs:string" />
   <xs:element name="EMail" type="xs:string" />
   <xs:element name="Score" type="xs:int" />
    <xs:element name="UserGroups" type="UserGroup" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
```

Listing A.1: XML-Schema der Dienst-Entität User

```
<xs:complexType name="UserGroup">
 <xs:sequence>
  <xs:element name="UserGroupId" type="clr:guid" />
  <xs:element name="Description" type="xs:string" />
 </xs:sequence>
</xs:complexType>
```

Listing A.2: XML-Schema der Dienst-Entität UserGroup

```
<xs:complexType name="Contact">
 <xs:sequence>
  <xs:element name="UserId" type="clr:guid" />
  <xs:element name="Login" type="xs:string" />
  <xs:element name="EMail" type="xs:string" />
 </xs:sequence>
</xs:complexType>
```

Listing A.3: XML-Schema der Dienst-Entität Contact

```
<xs:complexType name="ContactPosition">
 <xs:sequence>
  <xs:element name="Contact" type="Contact" />
  <xs:element name="Position" type="LatLong" />
  <xs:element name="Timestamp" type="xs:dateTime" />
 </xs:sequence>
</xs:complexType>
```

Listing A.4: XML-Schema der Dienst-Entität ContactPosition

```
<xs:complexType name="Message">
 <xs:sequence>
  <xs:element name="SenderContact" type="Contact" />
   <xs:element name="ReceiverContact" type="Contact" />
   <xs:element name="MessageText" type="xs:string" />
   <xs:element name="Timestamp" type="xs:dateTime" />
 </xs:sequence>
</xs:complexType>
```

Listing A.5: XML-Schema der Dienst-Entität Message

```
<xs:complexType name="PositionLog">
 <xs:sequence>
  <xs:element name="PositionLogsId" type="clr:guid" />
  <xs:element name="Timestamp" type="xs:dateTime" />
  <xs:element name="Position" type="LatLong" />
  <xs:element name="UserId" type="clr:guid" />
  <xs:element name="UserGroupId" type="clr:guid" />
 </xs:sequence>
</xs:complexType>
```

Listing A.6: XML-Schema der Dienst-Entität PositionLog

```
<xs:complexType name="PointOfInterest">
 <xs:sequence>
  <xs:element name="PointOfInterestId" type="clr:guid" />
  <xs:element name="Name" type="xs:string" />
  <xs:element name="Description" type="xs:string" />
  <xs:element name="Tags" type="xs:string" />
  <xs:element name="Image" type="xs:unsignedByte" minOccurs="0" maxOccurs="unbounded"/>
  <xs:element name="Timestamp" type="xs:dateTime" />
  <xs:element name="Type" type="xs:string" />
  <xs:element name="Position" type="LatLong" />
  <xs:element name="UserId" type="clr:guid" />
 </xs:sequence>
</xs:complexType>
```

Listing A.7: XML-Schema der Dienst-Entität PointOfInterest

```
<xs:complexType name="PrivateArea">
 <xs:sequence>
  <xs:element name="PrivateAreaId" type="clr:guid" />
  <xs:element name="UserId" type="clr:guid" />
  <xs:element name="Description" type="xs:string" />
  <xs:element name="Area" type="Area" />
 </xs:sequence>
</xs:complexType>
```

Listing A.8: XML-Schema der Dienst-Entität PrivateArea

```
<xs:complexType name="Trail">
 <xs:sequence>
  <xs:element name="TrailId" type="clr:guid" />
  <xs:element name="Start" type="LatLong" />
  <xs:element name="End" type="LatLong" />
 </xs:sequence>
</xs:complexType>
```

Listing A.9: XML-Schema der Dienst-Entität Trail

```
<xs:complexType name="Area">
 <xs:sequence>
  <xs:element name="Min" type="LatLong" />
  <xs:element name="Max" type="LatLong" />
 </xs:sequence>
</xs:complexType>
```

Listing A.10: XML-Schema der Dienst-Entität Area

```
<xs:complexType name="LatLong">
 <xs:sequence>
  <xs:element name="Latitude" type="xs:double" />
  <xs:element name="Longitude" type="xs:double" />
 </xs:sequence>
</xs:complexType>
```

Listing A.11: XML-Schema der Dienst-Entität LatLong

```
<xs:complexType name="Address">
 <xs:sequence>
  <xs:element name="AddressId" type="clr:guid" />
  <xs:element name="Country" type="xs:string" />
  <xs:element name="City" type="xs:string" />
  <xs:element name="Street" type="xs:string" />
  <xs:element name="StreetNumber" type="xs:string" />
  <xs:element name="Postalcode" type="xs:string" />
  <xs:element name="Position" type="LatLong" />
 </xs:sequence>
</xs:complexType>
```

Listing A.12: XML-Schema der Dienst-Entität Address

```
<xs:complexType name="Advertisement">
 <xs:sequence>
  <xs:element name="AdvertisementId" type="clr:guid" />
  <xs:element name="Description" type="xs:string" />
  <xs:element name="URI" type="xs:string" />
  <xs:element name="ValidityDate" type="xs:dateTime" />
  <xs:element name="Image" type="xs:unsignedByte" maxOccurs="unbounded" minOccurs="0"/>
  <xs:element name="Position" type="LatLong" />
 </xs:sequence>
</xs:complexType>
```

Listing A.13: XML-Schema der Dienst-Entität Advertisement

```
<xs:complexType name="Map">
 <xs:sequence>
  <xs:element name="Image" type="xs:unsignedByte" maxOccurs="unbounded" minOccurs="0">
  </xs:element>
  <xs:element name="MapId" type="clr:guid" />
  <xs:element name="TopLeft" type="LatLong" />
  <xs:element name="TopRight" type="LatLong" />
  <xs:element name="BottomLeft" type="LatLong" />
  <xs:element name="BottomRight" type="LatLong" />
  <xs:element name="Category" type="xs:string" />
  <xs:element name="Style" type="xs:string" />
 </xs:sequence>
</xs:complexType>
```

Listing A.14: XML-Schema der Dienst-Entität Map

```
1  <xs:complexType name="TrailblazersMap">
2   <xs:sequence>
3    <xs:element name="Map" type="Map" maxOccurs="unbounded" minOccurs="0" />
4    <xs:element name="TrailblazersMapId" type="clr:guid" />
5    <xs:element name="Origin" type="Address" />
6    <xs:element name="Destination" type="Address" />
7    <xs:element name="PointsOfInterest" type="PointOfInterest" maxOccurs="unbounded" minOccurs="0" />
8    <xs:element name="Trails" type="Trail" maxOccurs="unbounded" minOccurs="0" />
9   </xs:sequence>
10 </xs:complexType>
```

Listing A.15: XML-Schema der Dienst-Entität TrailblazersMap

A.3 Inhalt der CD-ROM

Dieser Arbeit ist eine CD-ROM beigefügt. Unter dem Wurzelverzeichnis der CD-ROM befinden sich **Ordner** und `Dateien` mit folgendem Inhalt:

Masterarbeit Dieses Dokument der Arbeit als Datei `Masterarbeit.pdf`.

Vorvertrag Den Vorvertrag zu dieser Arbeit in der Datei `Vorvertrag.pdf`.

Positionspapier Ein Positionspapier[1] über diese Arbeit in der Datei `Positionspapier.pdf`.

Software Enthält folgende - im Rahmen dieser Arbeit entwickelten - Software-Anwendungen in den entsprechenden Unterordnern. Eine Installationsanleitung ist in der Datei `Installationsanweisung.txt` zu finden.

- **TrailblazersUMPC** Das Visual Studio 2005-Projekt für die mobile Client-Anwendung basierend auf der .NET Framework-Version für UMPCs.
- **TrailblazersSmartphone** Das Visual Studio 2005-Projekt für die mobile Client-Anwendung basierend auf der .NET Compact Framework-Version für Smartphones.
- **TrailblazersServer** Das Visual Studio 2005-Projekt für die stationäre Server-Anwendung.
- **TrailblazersService** Das Visual Studio 2005-Projekt für den Web-Service der stationären Server-Anwendung.
- **TrailblazersDatabase** Die Datei `TrailblazersDatabase.sql` ist ein SQL-Skript zum Aufsetzen des objekt-relationalen Datenmodells für einen SQL Server 2005.

Externe komponenten Enthält frei verfügbare Software-Komponenten, die für die Installation des mobilen Informationssystems benötigt werden.

[1] Das Papier wurde im Rahmen der Software-Engineering Konferenz 2007 veröffentlicht (vgl. http://www.se2007.de/, Zugriffsdatum: 11.04.2007).

Glossar

API Siehe **Application Programming Interface**.

Application Programming Interface (API) Ein API ist eine Programmierschnittstelle, um die beschriebene Funktionalität einer Komponente oder eines Moduls für andere Anwendungen zur Verfügung zu stellen.

Delegate In der Programmiersprache C# ist ein Delegate eine Referenz auf eine Methode. Diese lässt sich als Rückruffunktion nutzen, um eine Aktion bei einem Ereignis auszulösen.

Dienst-Entität Eine Entität die vom Trailblazers-Dienst generiert, sowie aus diesem ausgelesen und wieder an diesen übertragen werden kann.

Dienstorientierte Architektur Technisch stellen sich die Dienste in einer dienstorientierten Architektur als lose gekoppelte Anwendungskomponenten dar, deren Schnittstellen universell beschrieben sind und die plattformunabhängig genutzt werden können.

Dienst Eine Software-Anwendung, die eine definierte Leistung erbringt und über ein Kommunikationsnetzwerk genutzt werden kann.

Entität In dieser Arbeit wird eine Entität als ein Objekt der realen oder der Vorstellungswelt definiert, das Informationen enthält.

eXtensible Markup Language (XML) XML ist eine Auszeichnungssprache zur Darstellung hierarchisch strukturierter Daten in Form von Textdateien.

Global Positioning System (GPS) Ein globales System zur satellitenbasierten Positionsbestimmung.

GPS Siehe **Global Positioning System**.

HTTP Siehe **Hypertext Transfer Protocol**.

Hypertext Transfer Protocol (HTTP) Ein Protokoll der ISO-OSI-Anwendungsschicht zum Übetragen von Daten in einem Netzwerk.

ISO-OSI-Schichtenmodell Das ISO-OSI-Schichtenmodell beschreibt modellhaft die Art der Datenübertragung für die Kommunikation offener, informationsverarbeitender Systeme, wie z. B. im Internet.

Komponente Eine (Software-)Komponente dient zur Kapselung von fachlichen oder technischen Funktionalitäten. Ähnlich wie eine Klasse besteht sie aus Daten und Methoden, ist jedoch größer. Der Zugriff von Außen auf die Interna erfolgt ausschließlich über eine definierte Schnittstelle. Komponenten sind häufig austausch- und wiederverwendbar.

Middleware Eine Middleware ist eine vermittelnde Software in vernetzten Systemen. Mit einer Middleware lassen sich Komponenten nutzen und zusammenfassen, sowie Dienste auffinden und ansprechen, die über klar definierte Schnittstellen verfügen.

Modul Eine funktional und technisch abgeschlossene Komponente einer Software.

NMEA Bezeichnet das NMEA 0183 Datenformat der National Marine Electronics Association. Dieses Standardformat wird u. a. für die Kommunikation zwischen GPS-Empfängern und einer Software-Schnittstelle für den GPS-Empfang genutzt.

Nutzer Der Nutzer des mobilen Trailblazers-Informationssystems. In dieser Arbeit sei damit eine Person gemeint, welche die mobile Anwendung des Systems unter Zuhilfenahme eines mobilen Gerätes nutzt.

Schnittstelle Eine Schnittstelle dient als Vertrag zwischen Komponenten. Sie sind eine formale Deklaration der vorhandenen Funktionen und definiert, wie diese Funktionen angesprochen werden können.

Simple Object Access Protocol (SOAP) SOAP ist ein Netzwerkprotokoll zum Übertragen von Daten und zum Durchführen von entfernten Methodenaufrufen. SOAP basiert auf Standardtechniken wie z. B. XML.

SOAP Siehe **Simple Object Access Protocol**.

SOA Siehe **dienstorientierte Architektur**.

TCP Siehe **Transmission Control Protocol**.

Trailblazers-Dienst Die stationäre Server Anwendung des Informationssystems. Diese stellt ortbezogene Dienste in Form von Web-Services bereit.

Trailblazers-Karte Eine Straßenkarte, welche mit zusätzlichen Informationen des Trailblazers-Dienstes annotiert wurde.

Transport Control Protocol (TCP) TCP ist ein verbindungsorientiertes Transportprotokoll der Transportschicht im ISO-OSI-Schichtenmodell.

Verteilte Anwendung Eine verteilte Anwendung ist eine Software-Anwendung, die in einem verteilten System abläuft.

Verteiltes System Ein verteiltes System ist ein Zusammenschluss mehrerer unabhängiger Rechner.

Web Services Description Language (WSDL) WSDL ist eine Metasprache, mit deren Hilfe die angebotenen Funktionen, Daten, Datentypen und Austauschprotokolle eines Web-Service beschrieben werden können.

Web-Service Ein Web-Service ist eine Software-Anwendung die mittels bekannter Internet-Standards beschrieben, gefunden und verwendet werden kann.

WSDL Siehe **Web Services Description Language**.

XML-Schema XML-Schema ist eine Sprache zur Beschreibung der Struktur eines XML-Dokumentes. Für die Beschreibung wird XML genutzt.

XML Siehe **eXtensible Markup Language**.

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, 8. Mai 2007	
Ort, Datum	Unterschrift