Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Fatih Keles

Ubiquitäre Annotationen und ortsbezogene Dienste im Zusammenspiel

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Fatih Keles

Ubiquitäre Annotationen und ortsbezogene Dienste im Zusammenspiel

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

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. Jörg Raasch

Abgegeben am 08. Februar 2008

Fatih Keles

Thema der Masterarbeit

Ubiquitäre Annotationen und ortsbezogene Dienste im Zusammenspiel

Stichworte

Ubiquitäre Annotationen, Ubiquitous Computing, Ortsbezogene Dienste, Mobile Computing

Kurzzusammenfassung

Die Themenbereiche Ubiquitous Computing und ortsbezogene Dienste sind in den vergangenen Jahren immer stärker in den Fokus der Informatik gerückt. Dabei beschäftigt sich das Ubiquitous Computing mit der Allgegenwärtigkeit von Informationssystemen in der Alltagswelt der Menschen. Ortsbezogene Dienste dagegen verwenden die geographische Position der Nutzer um ihnen Dienste ortsabhängig anzubieten. In dieser Arbeit werden zunächst die Grundlagen beider Themenbereiche beleuchtet und ausgewählte Projekte und Entwicklungen vorgestellt. Anschließend werden beide Bereiche in einem Anwendungsszenario miteinander verknüpft. Ziel ist es zu überprüfen, ob aus der Kombination beider Technologien ein Mehrwert für die Nutzer erzielbar ist. Zu diesem Zweck wird ein Anwendungssystem entworfen und prototypisch implementiert.

Fatih Keles

Title of the paper

Ubiquitous Annotations and Location-based Services combined

Keywords

Ubiquitous Annotations, Ubiquitous Computing, Location-based Services, Mobile Computing

Abstract

Over the past years, the sub-topics of Ubiquitous Computing and Location-based Services have been more and more intensively focused upon within the field of computer sciences. Ubiquitous Computing deals with the ubiquitous nature of information systems which form an integral part of the users' everyday life. As opposed to that location-based services make use of the users' geographic position in order to offer them services that are related to a specific position. The objective of this thesis is to discuss the basics of these two areas as a first step, presenting some selected projects and research results. As a second step, both areas of application will be combined in a scenario to be used for specific purposes. The aim is to scrutinise whether a combination of these two technologies actually provides an added value to the user. To that end, an application system will be developed and implemented in the form of a prototype.

Inhaltsverzeichnis

1 Einleitung

Ubiquitäre Annotationen und ortsbezogene Dienste sind zwei Forschungsgebiete, die in den letzten Jahren stark an Bedeutung gewonnen haben. Sie lassen sich in den Themenbereich des Mobile und Ubiquitous Computing einordnen [HMNS03]. Auch diese Arbeit kann den genannten Themengebieten zugeordnet werden. Sie beinhaltet jedoch auch einen Software-Entwurf und eine Software-Entwicklung. Insofern kann man sie auch im Bereich Software-Engineering ansiedeln. Auf den folgenden Seiten werden die Motivation und die Zielsetzung dieser Arbeit erörtert. Anschließend folgen Informationen über den inhaltlichen Aufbau der Arbeit.

1.1 Motivation

Mobile Anwendungen und Geräte erfreuen sich bereits seit vielen Jahren größter Popularität. Die Auswahl an mobilen Geräten, wie z. B. Handys und PDAs, steigt. Zudem sind die Geräte mittlerweile für den Massenmarkt bezahlbar, und den Nutzern stehen immer mehr Software-Produkte zur Verfügung [Rou06].

Eine Gruppe dieser Software-Produkte stellen die ortsbezogenen Dienste dar [Küp05]. Sie kennen den Standort des Nutzers und ermöglichen eine Reihe neuer Anwendungen. Navigationssysteme für Fahrzeuge und sog. Friend-Finder-Applikationen sind zwei Beispiele der vielen Ausprägungen. Eine Reihe von Studien zeigt, dass ortsbezogene Dienste in Zukunft mehr und mehr an Bedeutung gewinnen werden [Hel06].

Parallel zu dieser Entwicklung findet man immer häufiger Anwendungen, in denen die Nutzer nicht mehr nur reine Informationskonsumenten sind. Vielmehr treten die Nutzer selbst als Informationsanbieter auf [Alb07]. Diese Entwicklung konnte man in den vergangenen Jahren insbesondere im Internetbereich beobachten. Unter dem Stichwort „Web 2.0“ entstanden beispielsweise Wissens- und Community-Portale. Immer mehr Nutzer verfassten ihre eigenen Blogs und veröffentlichten ihre Bilder und selbstgedrehten Videos im Internet.

Konsequent weitergedacht führen diese Entwicklungen zu den ubiquitären Annotationen. Sie ermöglichen nicht nur die Kollaboration verschiedener Nutzer über das Internet, sondern transportieren sie in die Realwelt. Objekte der Realwelt können von Nutzern beschrieben und mit Informationen verknüpft werden. Andere Nutzer haben Zugriff auf diese Informationen und können sie verändern oder ergänzen. So können Wissen, Erfahrungen und Erlebnisse realitätsnah ausgetauscht und vermittelt werden.

Ausgehend von diesen Überlegungen entstand im Sommersemester 2007 das Projekt *UbiZoo*[1] an der HAW-Hamburg. Im Rahmen des Wettbewerbs *Imagine Cup 2007*[2] entwickelte der Autor gemeinsam mit seinen Kommilitonen Eike Falkenberg, Jan Napitupulu und Thomas Schmidt ein Anwendungssystem zur Unterstützung von Zoobesuchen von Schulklassen. Hierbei erhalten Lehrer und Schüler während eines Zoobesuches mobile Geräte, die sie durch den Zoo führen. Sie können Informationen zu den Tieren im Zoo auf den Geräten abrufen und durch spielerische Anwendungen mehr über Flora und Fauna erfahren. Des Weiteren gibt das Anwendungssystem dem Lehrer die Möglichkeit, den Zoobesuch zuvor zu planen. Die Schüler wiederum können den Zoobesuch nach der Durchführung virtuell über eine Webseite erneut erleben und z. B. ihren Eltern und Freunden zeigen (vgl. Abbildung 1.1).

(a) Planung des Zoobesuchs (b) Durchführung des Zoobesuchs (c) Nachbereitung des Zoobesuchs

Abbildung 1.1: Screenshots: *UbiZoo*-Anwendung

Im *UbiZoo*-System kamen sowohl Technologien aus dem Bereich der ortsbezogenen Dienste als auch aus dem der ubiquitären Annotationen zum Einsatz. Die Vereinigung beider Technologiebereiche versprach völlig neuartige Dienste und Anwendungssyteme. Diese Feststellung war Anlass und Motivation dafür, die Idee des *UbiZoo*-Systems fortzuführen und in der vorliegenden Arbeit zu thematisieren.

1.2 Zielsetzung

Diese Arbeit greift den Ansatz des *UbiZoo*-Projekts auf und entwickelt ihn weiter. Das speziell auf einen Zoobesuch zugeschnittene Szenario wird ersetzt durch ein Szenario, das sich auf Stadtrundgänge bezieht. Das Szenario soll eine anschauliche Darstellung ermöglichen und dient darüber hinaus als Vorlage für die prototypische Entwicklung eines Anwendungssytems. Im Wesentlichen sollen Technologien und Entwicklungen aus den Bereichen „Ortsbezogene Dienste“ und „Ubiquitäre Annotationen“ in einer Anwendung verknüpft werden, um so einen Mehrwert zu erzielen. Das Szenario ist so gewählt worden, dass beide Technologiebereiche beim Systemdesign und der Implementierung Anwendung finden werden.

Dem Nutzer des Anwendungssystems soll die Möglichkeit gegeben werden, individuell auf ihn zugeschnitte Stadtrundgänge durchzuführen. Er soll aktiv am Geschehen teilnehmen und seine

[1] UbiZoo: http://www.ubizoo.de
[2] Imagine Cup: http://www.imaginecup.com

Eindrücke mit anderen Touristen teilen können. Des Weiteren soll er die Rundgänge vor der Durchführung planen und im Nachhinein rekapitulieren können.

Derzeitige Anwendungen für Stadtrundgänge sind meist sehr statisch und bieten nur einen begrenzten Funktionsumfang [KB03]. Insbesondere sind in der Regel nur vordefinierte Routen auswählbar. Die Möglichkeit der Vorbereitung eines Stadtrundgangs sehen diese Anwendungen ebenso wenig vor wie eine Rekapitulation des Erlebten.

Darüber hinaus bieten schon heute einige Museen ihren Besuchern elektronische Informationssysteme an [HF05]. Dabei sind die Systeme für Museen nicht integriert mit den Systemen für Stadtrundgänge. Wünschenswert wäre daher eine Anwendung, die einerseits Stadtrundgänge ermöglicht und andererseits detailliertere Informationen über einzelne Objekte in oder bei Sehenswürdigkeiten zur Verfügung stellt. So könnte man sich als Tourist während des Stadtrundgangs zu einem Museum führen lassen und auch innerhalb des Museums Informationen zu einzelnen Exponaten aufrufen. Die Möglichkeiten der Umsetzung solcher integrierter Anwendungssysteme sollen hier untersucht und umgesetzt werden.

1.3 Aufbau der Arbeit

Diese Arbeit ist in sechs Kapitel gegliedert. Ihre Reihenfolge spiegelt im Wesentlichen die tatsächliche Abfolge der Umsetzung wieder. Die Tatsache, dass im Detail nachfolgende Kapitel durchaus auch vorangehende Kapitel beeinflusst haben, wird in Abschnitt 5.1 näher erörtert.

Im Anschluss an die Einleitung werden in Kapitel 2 die Grundlagen, die für diese Arbeit von Bedeutung sind, vorgestellt. Dabei werden zunächst in Abschnitt 2.1 ortsbezogene Dienste diskutiert. Anschließend folgen die Abschnitte 2.2 und 2.3, in denen die Themen „Ubiquitous Computing“ und „Ubiquitäre Annotationen“ erörtert werden. Das Kapitel beschreibt die Anforderungen und Anwendungsbereiche der genannten Themenbereiche. Des Weiteren werden die technischen Voraussetzungen und Möglichkeiten aufgezeigt.

In Kapitel 3 werden eine Reihe von Arbeiten vorgestellt, die sich im Umfeld der ortsbezogenen Dienste und ubiquitären Annotationen bewegen. Anhand von fünf Beispielprojekten soll dem Leser ein Einblick in konkrete Entwicklungen gegeben werden.

In Kapitel 4 wird der Entwurf eines Anwendungssystems für Stadtrundgänge diskutiert. Es wird ein Szenario vorgestellt (Abschnitt 4.1), auf dem der Entwurf und die spätere Entwicklung basieren. Nach der Aufstellung der fachlichen sowie technischen Anforderungen in Abschnitt 4.1 wird in den folgenden drei Abschnitten die Anwendungsarchitektur präsentiert. Einzelne Teilsysteme, Komponenten und Module, aus denen das Gesamtsystem besteht, werden erläutert und ihr Zusammenspiel geschildert.

Im Anschluss werden in Kapitel 5 Details der Implementierung betrachtet. Es beginnt mit Abschnitt 5.1, in dem der Entwicklungsprozess in seiner Gesamtheit beschrieben wird. Danach folgt in Abschnitt 5.2 ein Blick auf den aktuellen Stand der Implementierung. Dort finden sich auch Erläuterungen zum Stand der Umsetzung der zuvor definierten fachlichen Anforderungen. Hiernach werden bei der Implementierung verwendete Hard- und Software-Produkte vorgestellt.

Das Kapitel schließt mit Ausführungen zum Test der entwickelten Software (Abschnitt 5.6) und zu einigen Implementierungsdetails (Abschnitt 5.7) ab.

Im letzten Kapitel (6) wird die Arbeit zusammengefasst und ein Ausblick auf mögliche Weiterentwicklungen gegeben.

2 Grundlagen

Im vorliegenden Kapitel werden eine Reihe von Grundlagenthemen diskutiert. Sie sind für das Verständnis der folgenden Kapitel notwendig und sollen einen Überblick über die Hauptthemen dieser Arbeit geben. Eines dieser Themen sind ortsbezogene Dienste. Diese werden in Abschnitt 2.1 besprochen. Anschließend folgt eine Erörterung der Themen „Ubiquitous Computing“ und „Ubiquitäre Annotationen“ in den Abschnitten 2.2 und 2.3.

2.1 Ortsbezogene Dienste

Ortsbezogene Dienste *(engl.: Location-based Services)* können nach [Spi04] als Dienste bezeichnet werden, die den Ort oder die Position von mobilen Geräten mit weiteren Informationen verknüpfen, und so dem Anwender einen zusätzlichen Nutzen bieten. Dies ist eine der häufigsten Definitionen für ortsbezogene Dienste, die man in der Literatur findet. Sie wird in ähnlicher Form auch von [SV04] verwendet. Im Gegensatz zu [Spi04] spricht [SV04] dabei aber nicht ausschließlich von mobilen Geräten, sondern von Applikationen im Allgemeinen.

Weitere Definitionen zu ortsbezogenen Diensten sind u. a. in [Küp05], [Rot05] und [VMG+01] zu finden.

Ortsbezogene Dienste werden oft als eine Untermenge der sog. kontextbewussten Dienste *(engl.: Context-aware services)* betrachtet [Küp05]. Die allgemein anerkannte Definition von kontextbewussten Diensten in [DA00] macht den Grund für diese Klassifikation deutlich:

> *A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the users's task.*

Diese Definition ähnelt sehr stark der Definition von ortsbezogenen Diensten, mit dem Unterschied, dass hier nicht von dem Ort im Speziellen, sondern vom Kontext des Benutzers im Allgemeinen die Rede ist. Der Ort wird dabei als eine Ausprägung des Kontextes betrachtet. Weitere Ausprägungen können z. B. die Zeit oder die aktuelle Tätigkeit und das soziale Umfeld des Benutzers sein [SBG99].

Nach dieser kurzen Einleitung werden auf den folgenden Seiten einige wichtige Aspekte von ortsbezogenen Diensten diskutiert. Zunächst wird in Abschnitt 2.1.1 der Begriff des Orts näher erläutert. Abschnitt 2.1.2 beschreibt anschließend alternative Techniken zur Positionsbestimmung. Nachdem in Abschnitt 2.1.3 die Anforderungen an ortsbezogene Dienste diskutiert werden, folgt

in den Abschnitten 2.1.4 und 2.1.5 eine Übersicht über die Klassifikation ortsbezogener Dienste und mögliche Anwendungsszenarien. Abschnitt 2.1.6 behandelt die Themen Architektur und Middleware. Das Kapitel schließt mit der Betrachtung einiger Privacy-Aspekte in Abschnitt 2.1.7 ab.

2.1.1 Orte

Der Begriff „Ort“ bezeichnet in der Regel einen bestimmten Platz in der Realwelt. Ein Ort kann aber auch ein virtueller Treffpunkt sein, beispielsweise eine Webseite im Internet. Um hier eine Klassifizierung vornehmen zu können, wird im Folgenden der Begriff des Orts aus der Sicht von ortsbezogenen Diensten erläutert. Hierbei kann man zwischen drei Kategorien von Orten unterscheiden [Küp05]:

Beschriebener Ort:
Ein beschriebener Ort ist immer geknüpft an natürliche geographische Objekte, wie z. B. Berge, Flüsse und Landschaften, oder an geographische Objekte, die vom Menschen geschaffen wurden, wie z. B. Städte, Straßen, Gebäude und Räume. Alle diese Orte haben gemeinsam, dass sie durch Namen oder Nummern beschrieben werden.

Ort im Raum:
Ein Ort in einem Raum kann durch einen Punkt in einem zwei- oder dreidimensionalen Koordinatensystem beschrieben werden. Solch einen Punkt bezeichnet man als Position eines Objekts in einem Raum. Positionen werden im Alltag von Menschen nicht verwendet, um Orte zu beschreiben. Dies liegt daran, dass sich Menschen eher an geographischen Objekten orientieren [TS06]. Allerdings lassen sich Orte über Koordinaten präziser beschreiben. Für ortsbezogene Dienste spielen sie deshalb eine wichtige Rolle.

Netzwerk-Ort:
Netzwerk-Orte beziehen sich auf Topologien von Kommunikationsnetzwerken, wie z. B. das Internet oder zellenbasierte Systeme, wie GSM und UMTS. Ein Ort wird hierbei beschrieben durch eine Netzwerkadresse, die jedes Objekt im Netzwerk erhält. Im Internet ist dies z. B. die IP-Adresse.

Ortsbezogene Dienste können auf allen drei Kategorien von Orten basieren. Häufig findet man eine Kombination mehrerer Kategorien vor. Der Ort einer Person, die sich mit einem GPS-fähigen Gerät in einer Stadt aufhält, kann beispielsweise sowohl über die GPS-Position als auch über die Adresse, an der sie sich aufhält, beschrieben werden. Geocoding-Dienste, wie sie z. B. von Google angeboten werden [gooa], versuchen Ortsbeschreibungen in geographische Koordinaten zu übertragen.

2.1.2 Positionierung

Die zuverlässige Bestimmung des Orts ist eine Voraussetzung dafür, dass ortsbezogene Dienste überhaupt angeboten werden können. Gleichzeitig ist die exakte Bestimmung des Orts kein einfaches Unterfangen [DRW02]. Für die Lösung des Problems der Positionierung existieren verschiedene Verfahren, die später diskutiert werden.

2.1.2.1 Tracking/Positioning

Positionierungssysteme lassen sich grob in zwei Kategorien unterteilen: Tracking und Positioning [Rot05].

Beim *Tracking* wird die Position eines Benutzers von einem Netzwerk bestimmt. Der Benutzer kennt die eigene Position daher zunächst nicht. Falls er sie benötigt, muss die Position nach der Ermittlung durch das Netzwerk an den Benutzer übermittelt werden.

Vom *Positioning* spricht man in Fällen, in denen die Position vom Benutzer selbst ermittelt wird. Das Netzwerk kennt die Position des Benutzers nicht. Dies hat den Vorteil, dass die Position privat bleibt, und damit die Privatsphäre des Benutzers geschützt wird. Erfordert allerdings die Anwendung, dass das Netzwerk die Position kennen muss, wird eine Übermittlung der Position notwendig.

2.1.2.2 Basistechniken zur Positionsbestimmung

Bevor die Verfahren zur Positionsbestimmung näher erläutert werden, erfolgt hier eine Auflistung der Basistechniken, die bei der Ermittlung der Position verwendet werden können [DRW02].

- ***Cell of Origin (CEO):*** Diese Methode kann dann verwendet werden, wenn das Netzwerk eine Zellenstruktur aufweist. Basisstationen im Mobilfunkbereich stellen z. B. solche Zellen dar. Die Position ergibt sich einfach über die ID der Zelle, an der der Benutzer gerade angemeldet ist. Da solche Zellen relativ weit voneinander entfernt stehen können, kann die Positionsermittlung unter Umständen ungenau ausfallen.
- ***Time of Arrival (TOA):*** Bei dieser Technik wird ein Signal an mindestens drei Basisstationen gesendet. Aus der Differenz der Zeit, zu der das Signal bei den Basisstationen eintrifft, kann dann die Position des Benutzers ermittelt werden.
- ***Angle of Arrival (AOA):*** Diese Methode funktioniert ähnlich wie die TOA-Technik. Im Unterschied dazu wird allerdings nicht die Laufzeit der Signale gemessen, sondern der Winkel, aus dem die Signale bei den Basisstationen eintreffen.

Neben den hier genannten Techniken existieren auch Ansätze, die Position über die Messung der Signalstärke und durch die Auswertung von Videobildern zu ermitteln [Rot04].

2.1.2.3 Systeme zur Positionsbestimmung

In den vergangenen Jahren sind eine Reihe von Verfahren zur Positionsbestimmung entwickelt worden [Han06]. Diese lassen sich laut [Rot05] in drei Kategorien unterteilen: Verfahren, die auf Satellitennavigation basieren, Verfahren, die Innerhalb von Gebäuden eingesetzt werden und Verfahren, die Netzwerkgestützt arbeiten. Abbildung 2.1 zeigt eine Übersicht über die Kategorien und deren Ausprägungen.

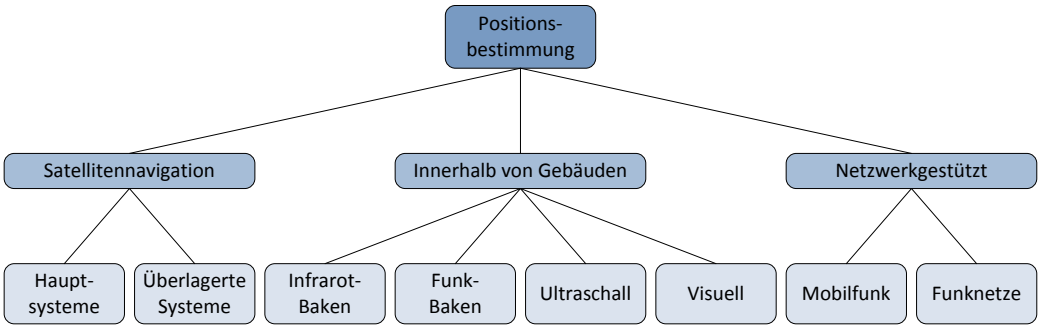


Abbildung 2.1: Systeme zur Bestimmung der Position (vgl. [Rot05])

Satellitennavigation
Satellitennavigationssysteme können sehr große geographische Bereiche abdecken [Küp05]. Das bekannteste und meist genutzte System ist das Global Positioning System (GPS). GPS arbeitet mit 24 Satelliten, die in sechs Umlaufbahnen zu je vier Satelliten in einer Höhe von 20.200 km um die Erde kreisen. Damit lässt sich die Position von Menschen oder Objekten an jedem Ort der Erde bestimmen. Die Satelliten strahlen dauernd ihre Position und die genaue Uhrzeit aus. Aus den Signallaufzeiten können GPS-Empfänger ihre eigene Position ermitteln. Typischerweise besteht dabei ein Kontakt zu fünf bis zehn Satelliten. Mindestens drei Satelliten werden jedoch in jedem Fall benötigt. Heutige GPS-Empfänger geben die Position als Längen- und Breitengrad an.

Wenn ein GPS-Empfänger keine Information über die Position der Satelliten hat, muss er die Satelliten zunächst auffinden. Das als „cold startup“ bezeichnete Verfahren kann mehrere Minuten dauern und verzögert die initiale Bestimmung der Position.

Die Genauigkeit, mit der die Position bestimmt werden kann, beträgt für die zivile Nutzung im Optimalfall 25 Meter in der Horizontalen und 43 Meter in der Vertikalen. Die Genauigkeit kann in Großstädten mit hohen Gebäuden abnehmen, da der Empfang der Satellitensignale durch Gebäude erheblich gestört werden kann. Innerhalb von Gebäuden ist deshalb eine Positionsermittlung mittels GPS kaum möglich [Rot05].

GPS gehört zu den Hauptsystemen der Positionsermittlung über Satelliten. Darüber hinaus existieren sog. überlagerte Systeme. Diese nutzen die Hauptsysteme als Basis, führen aber zusätzliche Verfahren ein, um die Positionsgenauigkeit zu erhöhen [Rot04]. Eines der überlagerten Systeme nennt sich „Differential GPS“ (DGPS). Wie man am Namen erkennen kann, verwendet

DGPS als Grundlage die GPS-Signale. Zusätzlich werden Basisstationen auf der Erde aufgestellt, deren exakte Position bekannt ist. Die Basisstationen ermitteln ihre Position außerdem über GPS. Die Abweichung zwischen der ermittelten und der bekannten Position wird als Korrekturwert für DGPS-Empfänger verwendet, die sich in der Nähe der Basisstation befinden. Mit DGPS ist eine Genauigkeit der Positionsbestimmung von bis zu einem Zentimeter möglich [Han06]

Weitere Informationen über GPS und Satellitennavigationssysteme findet man in [Küp05] und [Rot05]. Hier werden insbesondere die technischen Details der Verfahren zur Positionsbestimmung näher erläutert. Projekte, die auf GPS basieren werden z. B. in [AAH+97], [HBC+04], [MS00] und [PBC+01] beschrieben.

Positionierung innerhalb von Gebäuden

Da GPS innerhalb von Gebäuden praktisch unbrauchbar ist wurden in den vergangenen Jahren im Rahmen von Forschungsprojekten für diesen Zweck verschiedene Verfahren entwickelt und erprobt [Han06]. Eine Auswahl dieser Systeme wird nachfolgend exemplarisch beschrieben.

- *Infrarot-Baken:* Bei dieser Art der Indoor-Positionierung tragen die Benutzer kleine Infrarot-Sender mit sich, die periodisch Infrarot-Signale senden. Im Gebäude befinden sich Infrarot-Sensoren, die die Signale empfangen können. Da Infrarot-Signale keine Wände durchdringen können, kann so bestimmt werden, in welchem Raum sich eine Person aufhält. Die Bestimmung der Position ist damit Raumgenau [WHFG92].
- *Funk-Baken:* Anstelle von Infrarot-Signalen werden bei diesen Verfahren Funk-Signale verwendet. Da Funksignale Wände durchdringen können, kann man mittels Verfahren wie „Time of Origin“ oder durch das Messen der Signalstärke die Position einer Person bestimmen. In der Praxis hat sich jedoch gezeigt, dass diese Systeme sehr Fehleranfällig sind. Ein Hauptgrund hierfür ist, dass Wände und andere Hindernisse in Gebäuden die Signale beeinflussen und somit die Positionsberechnung verfälschen können [HVB01].
- *Ultraschallverfahren:* Im Gegensatz zu den Funk-basierten Systemen beruhen die Ultraschallverfahren auf Ultraschallsignalen. Da Schallsignale wesentlich langsamer weitergeleitet werden als Funk-Signale, ist eine genauere Positionsbestimmung möglich. Im Projekt „ActiveBat“ wurde z. B. eine Genauigkeit von bis zu 10 Zentimeter erreicht [WJ97].
- *Visuelle Verfahren:* Bei visuellen Verfahren wird versucht, anhand der Auswertung von Videobildern die Position von Personen oder Objekten zu bestimmen. Um die Positionserkennung zu erleichtern, werden dabei häufig Etiketten verwendet, die ein bestimmtes Muster haben und leicht wiedererkannt werden können [SMR+97].

Netzwerkgestützte Positionierung

Bei der netzwerkgestützten Positionierung kann man zwischen den Mobilfunkbasierten und den Funknetzbasierten Verfahren unterscheiden [Rot05].

- *Mobilfunkbasierte Verfahren:* Ein Beispiel für diese Art von Verfahren stellen Positionierungssysteme für das GSM-Netz dar. Da GSM-Netze Zellenbasiert sind, eignen sich die in 2.1.2.2 beschriebenen Verfahren besonders gut und werden in der Praxis auch verwendet.

Ericsson hat beispielsweise das „Mobile Positioning System“ (MPS) entwickelt [eri], das im Wesentlichen die oben genannten Verfahren nutzt.

- *Funknetzbasierte Verfahren:* Die bedeutendsten Verfahren basieren auf WLAN-Funknetzen. Hierbei wird für die Positionsbestimmung eine vorhandene WLAN-Infrastruktur verwendet. Die Position wird anhand der Signalstärke verschiedener WLAN-Basisstationen ermittelt. Die genaue Funktionsweise kann z. B. in [BP99] und [CCKM01] nachgelesen werden.

Keines der angeführten Verfahren zur Positionsbestimmung ist vollkommen [Rot05]. Je nach Anforderung der Applikation muss man deshalb die Vor- und Nachteile der einzelnen Verfahren abwägen. Es macht z. B. keinen Sinn, GPS einzusetzen, wenn man eine Anwendung entwickelt, die innerhalb von Gebäuden funktionieren soll. Auf der anderen Seite gehört die GPS-Technologie zu den erprobtesten Positionierungsverfahren überhaupt und hat sich für die Nutzung im Freien bewährt. In manchen Fällen mag eine Kombination mehrerer Verfahren sinvoll sein, insbesondere in Fällen, in denen die Anforderungen an die Applikation eine Indoor- und Outdoor-Positionierung erfordern.

2.1.3 Anforderungen

Ortsbezogene Dienste sollten eine Reihe von Anforderungen erfüllen, damit sie sich durchsetzen und genutzt werden können. [TVM+03] fasst einige wichtige Anforderungen zusammen. Ausgehend von den funktionalen Anforderungen werden sie unter den Oberbegriffen „Usability“, „Zuverlässigkeit“, „Privacy“ und „Infrastruktur“ zusammengefasst.

Funktionale Anforderungen

Die funktionalen Anforderungen an ortsbezogene Dienste haben sich in verschiedenen Benutzerstudien [HH02, KLP+01, Kaa03, Rei03] herauskristallisiert. Zu ihnen gehören die Möglichkeiten, die eigene Position festzustellen, nach ortsbezogenen Informationen zu suchen und Navigationsunterstützung beim Aufsuchen von Orten zu erhalten. Zudem wünschen viele Nutzer eine Personalisierung, so dass die verfügbaren Informationen auf die eigenen Bedürfnisse angepasst sind. Sie erwarten außerdem, dass Informationen innerhalb einer angemessenen Zeit möglichst performant zur Verfügung stehen. Eine weitere Anforderung ist die Darstellung alternativer Ansichten, beispielsweise eine Kartenansicht für einen Überblick und die Navigation sowie eine Detailansicht für die Darstellung von Zusatzinformationen.

Usability

Als Endgeräte ortsbezogener Anwendungen werden meist mobile Computer mit eingeschränkten Kapazitäten verwendet. Geringe Bandbreiten, große Schwankungen in der Bandbreite und die mögliche unvorhersehbare Trennung der Netzwerkverbindung sind einige Probleme, die berücksichtigt werden müssen. Hinzu kommen die Eigenschaften der mobilen Geräte selbst. Kleine Bildschirme, eingeschränkte Eingabemöglichkeiten, eine geringe Rechenleistung, eingeschränkte Laufzeiten und geringe Speicherkapazitäten sind einige Beispiele hierfür [Sat01]. In [TVM+03] werden einige Vorschläge gemacht, um diese Probleme zu beseitigen. Es wird empfohlen, die Netzwerkverbindung selten zu verwenden und so wenig wie möglich Daten zu übertragen. Ein

Offline-Modus sollte als Fallback-Lösung zur Verfügung stehen. Das Benutzerinterface sollte sehr einfach und benutzerfreundlich gestaltet werden.

Zuverlässigkeit
Ortsbezogene Dienste unterstützen die Nutzer und tragen zu Entscheidungen bei. Falsche Informationen können daher falsche Entscheidungen zur Folge haben. Dies kann zu Zeitverlust und zur Verärgerung der Nutzer führen. Eine Fehlerhafte Information über Abfahrtszeiten eines Zugs etwa könnte dazu führen, dass der Nutzer seinen Zug verpasst. Die Zuverlässigkeit der Software und der zugrundeliegenden Daten ist deshalb eine wichtige Anforderung an ortsbezogene Dienste.

Privacy
Ortsbezogene Informationen können sehr sensible Daten darstellen, die die Nutzer nur eingeschränkt preisgeben wollen [Lan05]. Bei der Entwicklung ortsbezogener Dienste müssen daher die Privacy-Bedenken der Benutzer ernsthaft berücksichtigt werden. Dieses sehr heikle Thema wird in Abschnitt 2.1.7 gesondert diskutiert.

Infrastruktur
In [TVM+03] werden einige Anforderungen aufgezählt, die die Positionierungsinfrastruktur betreffen. So wird gefordert, dass eine möglichst genaue Positionierung gewährleistet wird. Des Weiteren sollte das Gebiet, in dem man lokalisiert werden kann, möglichst groß sein. Die Positionierung muss schnell erfolgen, und es muss möglich sein, eine Vielzahl von Personen oder Objekten parallel zu lokalisieren. Gleichzeitig sollte der Aufwand für die Positionierung möglichst gering sein.

2.1.4 Klassifikation

Ortsbezogene Dienste lassen sich nach verschiedenen Merkmalen klassifizieren. Die Unterscheidungsmerkmale helfen dabei, ortsbezogene Applikationen einzuordnen und zu beschreiben. Einige ausgewählte Merkmale werden nachstehend diskutiert.

Reaktive/Proaktive Dienste
Die Unterscheidung zwischen reaktiven und proaktiven Diensten ist in der Literatur am häufigsten zu finden [Küp05, Rot05, Spi04]. Reaktive Dienste (auch „Pull-Dienste“ genannt) werden durch den Benutzer bewusst und explizit aufgerufen. Eine Anwendung, die auf Anforderung des Benutzers alle Restaurants in der Nähe auflistet, wäre beispielsweise solch ein Dienst. Abbildung 2.2 zeigt schematisch einen reaktiven Dienst.

Proaktive Dienste (Push-Dienste) dagegen erfordern keine explizite Anforderung durch den Benutzer. Sie werden automatisch ausgeführt, sobald ein vorher definierter Ort betreten wird (vgl. Abbildung 2.3). Ein Beispiel hierfür wäre ein mobiles Informationssystem, das Touristen beim Betreten bestimmter Sehenswürdigkeiten mit relevanten Informationen versorgt.

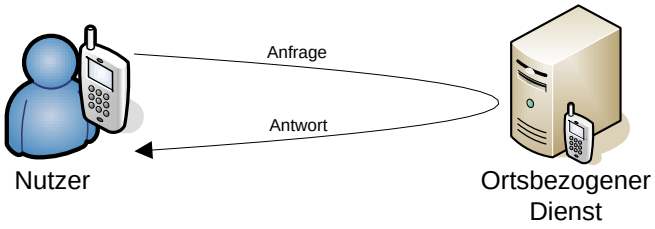


Abbildung 2.2: Reaktiver Dienst

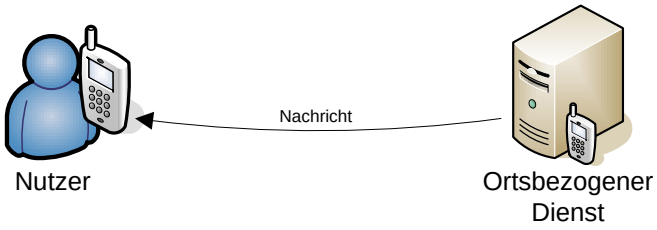


Abbildung 2.3: Proaktiver Dienst

Im Gegensatz zu reaktiven Diensten muss bei proaktiven Diensten die Fortbewegung des Benutzers permanent beobachtet werden, um das Eintreten von Ortsereignissen feststellen und darauf reagieren zu können.

Personenbezogene/Gerätebezogene Dienste

Eine weitere Klassifikation ortsbezogener Dienste wird in [Spi04] vorgenommen. Nach dieser sind personenbezogene Dienste all jene Dienste, bei denen die Position der Person, die die Applikation nutzt, im Mittelpunkt steht. Ein Beispiel ist eine Friend-Finder Applikation, von der man benachrichtigt wird, wenn sich Freunde in der Nähe aufhalten.

Gerätebezogene Dienste dagegen können sich nicht nur auf Personen beziehen, sondern auch auf andere Objekte wie z. B. Autos. Hierbei ist die Person, die die Applikation nutzt, nicht diejenige, deren Position bestimmt wird. Eine Applikation, die die Position eines Autos nach einem Diebstahl verfolgt, wäre ein Beispiel für einen gerätebezogenen Dienst.

Genauigkeit der Positionsbestimmung

Laut [Rot05] und [Spi04] spielen Genauigkeitsanforderungen für ortsbezogene Dienste eine wichtige Rolle für die Klassifikation. Während bei einigen Diensten eine geringe Genauigkeit der Position ausreichen kann, sind für andere Dienste sehr exakte Positionsinformationen vonnöten. Beispielsweise kann für eine Person, die einen Supermarkt in der Nähe sucht, die Postleitzahl als Positionsinformation ausreichen, obwohl dies eine relativ ungenaue Positionierung zur Folge hat. Ein PKW-Navigationssystem dagegen benötigt relativ exakte Positionsangaben, um den Benutzer effizient zum gewünschten Ziel führen zu können.

Indoor-/Outdoor-Systeme
Ortsbezogene Dienste kann man danach unterscheiden, ob sie für die Nutzung innerhalb oder außerhalb von Gebäuden konzipiert wurden [Han06]. Dies ist deshalb wichtig, weil sich einige Verfahren zur Positionsbestimmung nur für die eine oder andere Art von Systemen eignen. So liefert beispielsweise die Positionierung über GPS zwar außerhalb von Gebäuden relativ genaue Daten, ist aber innerhalb von Gebäuden unbrauchbar. Weitere Details hierzu sind in Abschnitt 2.1.2 zu finden.

Vertraulichkeit der Position
Es gibt ortsbezogene Dienste, bei denen die Position für die Bearbeitung des Dienstes nicht an Dritte weitergegeben werden muss. Klassisches Beispiel hierfür sind GPS-Navigationssysteme. Einige ortsbezogene Dienste erfordern jedoch die Weitergabe der Positionsinformationen [Rot05]. Die bereits oben genannte Friend-Finder Applikation wäre ein Beispiel hierfür. Bei dieser Art von Diensten müssen gesetzliche Richtlinien und persönliche Vorbehalte und Vorgaben der Nutzer berücksichtigt werden. Eine ausführlichere Darstellung zu dieser Problematik findet sich in Abschnitt 2.1.7.

Zustandsbehaftete/Zustandslose Dienste
Diese aus technischer Sicht wichtige Klassifikation ortsbezogener Dienste findet man in [Jac04]. Dienste, bei denen ein Zustand über mehrere Anfragen hinweg gespeichert werden muss, werden dort als zustandsbehaftete Dienste bezeichnet. Bei zustandslosen Diensten dagegen ist eine Speicherung des Zustands nicht notwendig.

Die beschriebenen Klassifikationsmerkmale stellen keine vollständige Auflistung dar. Je nach Blickwinkel findet man in der Literatur verschiedene weitere Merkmale (vgl. z. B. [Jac04, Rot05]). Sie werden hier nicht weiter ausgeführt, weil sie für den Inhalt der vorliegenden Arbeit eine untergeordnete Rolle spielen werden.

2.1.5 Anwendungsszenarien

Applikationen aus dem Bereich der ortsbezogenen Dienste lassen sich thematisch zu Gruppen zusammenfassen. So existieren verschiedene Anwendungsszenarien, die sich jeweils durch einige gemeinsame Eigenschaften auszeichnen. Bei der Entwicklung neuer Anwendungen macht es daher Sinn zu untersuchen, um welchen Anwendungstyp es sich dabei handelt. So ist ein Vergleich möglich. Außerdem lassen sich Erfahrungen, die bei der Entwicklung ähnlicher Anwendungen gemacht wurden, für die vorliegende Arbeit nutzen. Laut [Rot05] gibt es in der Literatur keine einheitliche Kategorisierung. Abbildung 2.4 zeigt daher eine Zusammenfassung verschiedener Anwendungsszenarien für ortsbezogene Dienste, die sich aus den Beschreibungen mehrerer Autoren [Rot05, Küp05, Spi04, Jac04, Jun05, MS05] zusammensetzt.

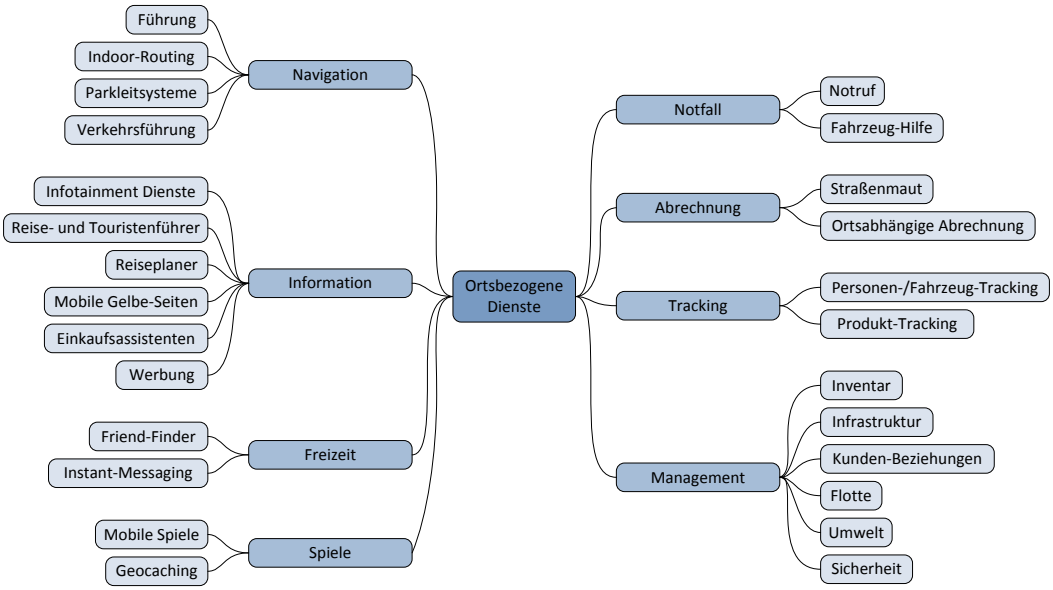


Abbildung 2.4: Szenarien für ortsbezogene Dienste

Navigationsdienste
Navigationsdienste dienen der Führung von Personen oder Fahrzeugen von einem Ort zu einem anderen [SVMY04]. Sie kommen heutzutage vor allem in der Fahrzeugnavigation zum Einsatz. Bekannte Autonavigationssysteme sind z. B. die Produkte von *TomTom*[1] und *Navigon*[2]. Speziell für Fußgänger konzipierte Navigationssysteme sind derzeit noch Untersuchungsobjekte der Forschung [TS06, BS06, MK06]. Die einhellige Meinung der Autoren ist hierbei, dass sich Fußgänger an Landmarken orientieren. Deshalb sind PKW-Navigationssysteme für die Nutzung durch Fußgänger nur eingeschränkt geeignet [MS07]. Weitere Beispiele für Navigationssysteme sind Parkleitsysteme und Verkehrsführungssysteme.

Informationsdienste
Diese Dienste versorgen die Nutzer mit ortsbasierten Informationen jeglicher Art [RM03]. Dies können u. a. Informationen über Orte von Interesse wie z. B. Restaurants oder Tankstellen sein. Eine typische Anfrage an solch ein System wäre z. B.: *„Zeige mir alle chinesischen Restaurants in der Nähe an“*. Auch Touristenführer und Einkaufsassistenten lassen sich in diese Kategorie einordnen. Eine weitere Art von Informationsdiensten sind ortsbasierte Werbedienste. Laut [Küp05] sind Informationsdienste die am weitesten verbreiteten Applikationen in der Gruppe der ortsbezogenen Dienste.

Freizeit & Spiele
Ortsbezogene Dienste ermöglichen vielfältige Szenarien im Freizeit- und Spielebereich. Friend-Finder-Applikationen etwa benachrichtigen die Nutzer, wenn sich Freunde oder Personen mit

[1] TomTom: http://www.tomtom.com
[2] Navigon: http://www.navigon.com

ähnlichen Interessen in der Nähe aufhalten [SC04]. Bei ortsbasierten mobilen Spielen vermischen sich physikalische und virtuelle Welten [JW06]. Bei dem Spiel „Capture the Flag“ [CSLT06] z. B. müssen Spieler eine virtuelle Flagge des Gegners in einer realen Umgebung finden und damit die gegnerische Basis erobern. Eine Reihe weiterer ortsbezogener Spiele findet man in [MCMN05].

Notfalldienste
In Notfallsituationen können Menschen oft ihren Aufenthaltsort nicht exakt beschreiben. Dies kann daran liegen, dass sie sich örtlich nicht auskennen oder sie körperlich dazu nicht in der Lage sind. In solchen Fällen helfen Notfallsysteme, die die Position der hilfebedürftigen Personen automatisch an die Rettungsdienste übermitteln [Spi04]. Hierzu gibt es z. B. eine Richtlinie der US Federal Communications Commission, die von allen Mobilfunknetzbetreibern in den USA fordert, die Position der Anrufer bei Notrufen zu übermitteln [DRW02].

Abrechnungsdienste
Abrechnungsdienste sind dazu in der Lage, abhängig vom Ort eines Benutzers die Nutzung eines Dienstes festzustellen und diese in Rechnung zu stellen. Ein Beispiel sind Mautsysteme, wie z. B. „Toll Collect“ [tol], die automatisch die gefahrene Strecke von Fahrzeugen auf bestimmten Straßen feststellen können [Küp05]. Alternative Systeme, wie die Abrechnung über Mautstationen und mittels Vignetten, werden dadurch überflüssig.

Tracking- & Managementdienste
Bei Tracking-Diensten verfolgt ein potentieller stationärer Benutzer die Position eines anderen Benutzers oder mobilen Objekts [Rot05]. In Unternehmen werden solche Dienste häufig eingesetzt, um Mitarbeiter oder Fahrzeuge zu orten. So können z. B. Kundenbesuche von Servicemitarbeitern effizienter geplant werden. Des Weiteren können Unternehmen zu jeder Zeit in Erfahrung bringen wo sich ihre Produkte befinden [RM03]. Eine weitere Anwendung ist bekannt aus der Nachverfolgbarkeit von Paketsendungen. So ist es beim Einsatz entsprechender Tracking-Systeme möglich, den aktuellen Auslieferungsstand verschickter Pakete zu ermitteln.

2.1.6 Architektur und Middleware

Mit der Entwicklung verschiedenster Applikationen aus dem Bereich der ortsbezogenen Dienste sind Vorschläge für Architekturen entstanden. Auch wenn die meisten Applikationen eine individuelle Architekturanpassung erfordern, gibt es wiederkehrende Komponenten, die in allen ortsbezogenen Diensten zu finden sind. Abbildung 2.5 und die folgende Auflistung zeigen die häufigsten Architektur-Komponenten [Jac04, MS05]:

- *Client:* Hierbei handelt es sich um Endgeräte der Benutzer. Mit ihnen lassen sich die ortsbezogenen Dienste nutzen. Dies sind in der Regel mobile Geräte, wie z. B. Mobiltelefone, PDAs und Navigationsgeräte.

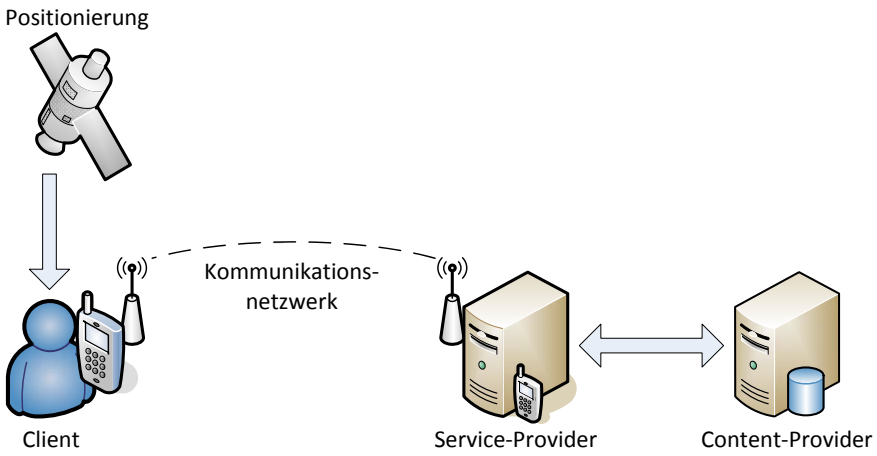


Abbildung 2.5: Architekturkomponenten ortsbezogener Dienste (vgl. [Jac04])

- *Positionierung:* Diese Komponente ist dafür zuständig, die Position des Nutzers zu bestimmen. Hierbei können die in Abschnitt 2.1.2 diskutierten Verfahren eingesetzt werden.
- *Kommunikationsnetzwerk:* Das Kommunikationsnetzwerk sorgt für eine Datenverbindung zwischen den Dienstkonsumenten und den Anbietern.
- *Service-Provider:* Beim Service-Provider handelt es sich um den Anbieter des ortsbezogenen Dienstes. Je nach Art der Applikation können z. B. berechnete Routen, Orte von Interesse oder andere Dienste vermittelt werden.
- *Content-Provider:* Der Content-Provider hält die Basisdaten bereit, die wiederum der Service-Provider in Anspruch nehmen kann. Dies können z. B. geographische Datenbanken sein.

Middleware

Eine Middleware stellt Schnittstellen, Protokolle und Infrastrukturdienste bereit, die von einer Vielzahl von Applikationen benötigt werden [Küp05]. Häufig benötigte Dienste werden gekapselt und standardisiert zur Verfügung gestellt. Eine Middleware für ortsbezogene Dienste sollte folgende Aufgaben beherrschen [Jac04]:

- Sie sollte die Positionierungstechnologie verwalten, potentiell mehrere Positionierungstechnologien ermöglichen und die konkret eingesetzte Technik transparent kapseln.
- Sie muss unterbrochene Netzwerkverbindungen und Schwankungen in der Netzwerkqualität managen und vor dem Nutzer verbergen.
- Sie muss eine hohe Anzahl von Nutzer und große Datenmengen verwalten können.
- Sie muss unterschiedliche Inhalte und Dienste bewältigen können und verschiedene Netzwerk- und Anwendungsschicht-Protokolle beherrschen.
- Sie muss eine hohe Verfügbarkeit garantieren.

- Sie sollte Funktionen zur Abrechnung von Diensten bereitstellen.
- Sie sollte Security- und Privacy-Funktionen zur Verfügung stellen.

Mit dem „OpenGIS Location Service“ (OpenLS) hat das „Open Geospacial Consortium“ (OGC) eine Spezifikation für eine Middleware für ortsbezogene Dienste veröffentlicht [Ope]. Das OGC ist eine 1994 gegründete gemeinnützige Organisation, die sich zum Ziel gesetzt hat, die Entwicklung von ortsbezogener Informationsverarbeitung auf Basis allgemeingültiger Standards zum Zweck der Interoperabilität festzulegen. Die Kernkomponente der Spezifikation stellt der „GeoMobility Server“ dar. Er implementiert im Wesentlichen die oben genannten Anforderungen. Details hierzu können in [Ope] und [Küp05] nachgelesen werden.

2.1.7 Privacy

Der Begriff „Privacy“ wird im Deutschen häufig mit „Privatsphäre“ oder „Datenschutz“ übersetzt. Laut [Wes70] ist Privacy das Anrecht von Personen, Gruppen und Institutionen, selbst darüber zu bestimmen wann, wie und in welchem Ausmaß Informationen über sie an Andere weitergegeben werden.

Risiken

Menschen sehen es als eine Bedrohung an, wenn Informationen über sie selbst ohne ihr Wissen an die Öffentlichkeit gelangen. An dieser Stelle spielt die Angst vor der Veröffentlichung intimer Details ohne die Einwilligung des Betroffenen eine wichtige Rolle [Gar03]. Einen potentiellen Arbeitgeber etwa könnten Informationen über Krankheiten dazu bewegen, einen Bewerber nicht einzustellen. Durch fehlerhafte Informationen könnte man irrtümlicher Weise in das Visier staatlicher Strafverfolgungsorgane gelangen. Diese zwei Beispiele zeigen, dass die Ängste der Menschen vor dem Missbrauch persönlicher Daten durchaus berechtigt sind.

Gesetzgebung

In Deutschland regelt das Bundesdatenschutzgesetz (BDSG) die Bestimmungen für den Datenschutz [BDS]. In § 1 des BDSG wird das Recht auf informationelle Selbstbestimmung festgelegt. Dieses Recht räumt jedem Bürger ein, selbst über die Weitergabe und Verwendung persönlicher Daten zu entscheiden. Die Verarbeitung personenbezogener Daten ist laut § 4 BDSG nur dann zulässig, wenn der Betroffene einwilligt, oder ein gesetzlicher Erlaubnistatbestand vorliegt. Weiterhin regelt § 3a BDSG, dass personenbezogene Daten nur in dem tatsächlich notwendigen Umfang verarbeitet werden dürfen. § 28 BDSG legt außerdem fest, dass personenbezogene Daten nur für den Zweck verarbeitet werden dürfen, für den sie rechtmäßig erhoben wurden.

Privacy in ortsbezogenen Diensten

Ein ungenügender Schutz der persönlichen Daten kann zu Akzeptanzproblemen bei den Nutzern von ortsbezogenen Diensten führen [Küp05]. Außerdem erfordern die gesetzlichen Regelungen entsprechende Maßnahmen zum Schutz der Privatsphäre.

Die Brisanz des sog. „Location-Privacy“ liegt vor allem darin, dass man über die Kenntnis des Aufenthaltsorts einer Person auf viele weitere persönliche Daten schließen kann [Lan05]. Zum Beispiel ist es denkbar, über die Analyse von häufigen Aufenthaltspunkten einer Person an Informationen über Hobbies, Freunde oder politische Einstellungen zu gelangen.

Privacy-Konzepte in ortsbezogenen Diensten
In der Literatur werden folgende drei Ansätze für den Schutz von persönlichen Daten in ortsbezogenen Diensten diskutiert [Küp05].

- *Sichere Kommunikation*
- *Privacy-Regeln*
- *Anonymisierung*

Sichere Kommunikation Bei diesem Ansatz geht es um die Absicherung der Kommunikationsleitung zwischen dem Dienstanbieter und dem Dienstkonsumenten. Dabei wünscht man die Gewährleistung von Authentifikation, Integrität und Vertraulichkeit. Vertraulichkeit bedeutet, dass auf die übertragenen Informationen nicht von Dritten zugegriffen werden kann. Die Integrität soll gewährleisten, dass der Empfänger genau die Informationen erhält, die der Sender abschickt. Authentifikation schließlich bedeutet, dass die Teilnehmer der Kommunikation tatsächlich diejenigen sind, als die sie sich ausgeben. Die genannten Mechanismen können z. B. über eine verschlüsselte Verbindung und gängige Authentifikationstechniken[3] sichergestellt werden [Eck04].

Privacy-Regeln Über eine sichere Kommunikationsleitung kann gewährleistet werden, dass keine Unbefugten auf die übertragenen Informationen zugreifen oder sie verändern. Jedoch wird nicht geregelt, in welchem Umfang einzelne Dienstanbieter auf die Ortsdaten zugreifen dürfen [Küp05]. An dieser Stelle kommen die Privacy-Regeln ins Spiel. Sie sollen regeln, wer, wann, in welchem Umfang auf welche Ortsdaten zugreifen darf.

Im einfachsten Fall könnte man den Nutzer eines ortsbezogenen Dienstes bei jeder Dienstanfrage um eine Bestätigung bitten [MFD03]. Dadurch würde der Nutzer des Dienstes jedoch zu häufig gestört werden. Aus diesem Grund versucht man durch die Einführung von Privacy-Regeln den Bestätigungsprozess zu automatisieren. Ein Beispiel für ein Konzept, das auf Privacy-Regeln basiert, kann z. B. in [MFD03] nachgeschlagen werden.

Anonymisierung Privacy-Regeln sind nur dann sicher, wenn alle Teilnehmer eines ortsbezogenen Dienstes vertrauenswürdig sind [Küp05]. Es kann nicht ausgeschlossen werden, dass einer der Beteiligten Informationen an unbefugte Dritte weitergibt. Aus diesem Grund gibt es eine Reihe von Arbeiten, die sich mit der Anonymisierung von Ortsinformationen beschäftigen. In [Cue02] wird dabei zwischen der „Abstraktion vom Identifikator“ und der „Abstraktion vom Inhalt“ unterschieden. Bei der Abstraktion vom Identifikator wird die tatsächliche ID eines Teilnehmers

[3] Gängige Authentifikationstechniken sind z. B. das Challenge-response-Verfahren oder die Authentifikation durch Benutzerzertifikate (vgl. [Eck04]).

durch eine Anonyme ID ersetzt. Mit Abstraktion vom Inhalt meint man die Reduzierung der Genauigkeit von Zeit- und Ortsinformationen.

In [BS04] wird das Konzept der sog. „Mix Zones“ vorgestellt. Es handelt sich um einen Ansatz der Anonymisierung und fällt in die Kategorie der Abstraktion vom Identifikator. Die Grundidee besteht darin, Orte in „Application Zones“ und „Mix Zones“ zu unterteilen. Die Nutzung von ortsbezogenen Diensten findet nur in den Application Zones statt, nicht aber in den Mix Zones. Jedes Mal, wenn eine Person eine Mix Zone betritt, erhält sie von einer Middleware ein neues Pseudonym. Alle Personen in solch einer Mix Zone sind sozusagen „gemixt“. Sie können nicht mehr ohne weiteres verfolgt werden, da sich die Nutzer bei jeder Applikation mit einem anderen Pseudonym anmelden. Die Verwendung von Pseudonymen führt außerdem dazu, dass die Nutzer von den Dienstanbietern nicht identifiziert werden können. Aus Privacy-Sicht stellt dies ebenfalls einen Vorteil dar. Weitere Details zum Konzept der Mix Zones sind in [BS04] und [BS03] zu finden.

2.2 Ubiquitous Computing

> *„The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.“*

Mit diesem Satz beginnt der legendäre Aufsatz *„The Computer for the 21st Century“* [Wei91] von Mark Weiser aus dem Jahre 1991. Weiser beschreibt dort seine Vision der allgegenwärtigen Computer, die zwar überall zu finden sein sollten, aber trotzdem nicht als solche wahrgenommen würden. Er leitete damals einen neuen Zweig in den Computerwissenschaften ein, den man heute unter den Namen „Ubiquitous Computing“ und „Pervasive Computing“ kennt. Diese beiden Begriffe werden von den meisten Autoren synonym verwendet [AGIS05, Mat01]. Laut [HMNS03] wurde der Begriff „Pervasive Computing“ von der Industrie geprägt und steht für den kurzfristigen kommerziellen Einsatz von „Ubiquitous Computing“-Technologien.

Weiser war der Auffassung, dass zukünftige Prozessoren, Speicherbausteine und Sensoren sehr klein, günstig und energieeffizient sein würden. Deshalb könne man sie in Alltagsgegenstände einbauen, um so einen Mehrwert zu erzielen. Während die damals verfügbaren Technologien eine befriedigende Umsetzung noch nicht ermöglichten, stehen heutzutage eine Reihe von technologischen Mitteln zur Verfügung, um die Vision des Ubiquitous Computing zu realisieren [Sat01]. Eines dieser Mittel stellt z. B. die RFID-Technologie dar, mit der bereits heute Anwendungen in der Lagerwirtschaft erheblich vereinfacht werden können [HBHS06].

Auf den folgenden Seiten werden zunächst die Anforderungen an Ubiquitous Computing-Systeme erläutert (Abschnitt 2.2.1). In Abschnitt 2.2.2 werden Anwendungsszenarien ubiquitärer Systeme beschrieben. Anschließend folgt in Abschnitt 2.2.3 eine Diskussion über die Technologien des Ubiquitous Computing. Abschnitt 2.2.4 diskutiert schließlich mögliche Auswirkungen des Ubiquitous Computing auf den Menschen, sein soziales Verhalten und die Umwelt.

2.2.1 Anforderungen

Das Ubiquitous Computing kann als ein bedeutender evolutionärer Schritt einer Reihe von Entwicklungen seit Mitte der 70er-Jahre des 20. Jahrhunderts verstanden werden [Sat01]. Die zwei vorangehenden Schritte sind die verteilten Systeme und das Mobile Computing. Aus diesem Grund übernimmt das Ubiquitous Computing die technischen Anforderungen und Probleme dieser beiden Vorgängertechnologien. Es kommen jedoch weitere spezifische Anforderungen hinzu. Abbildung 2.6 zeigt eine Übersicht über die Anforderungen an die jeweiligen Technologien.

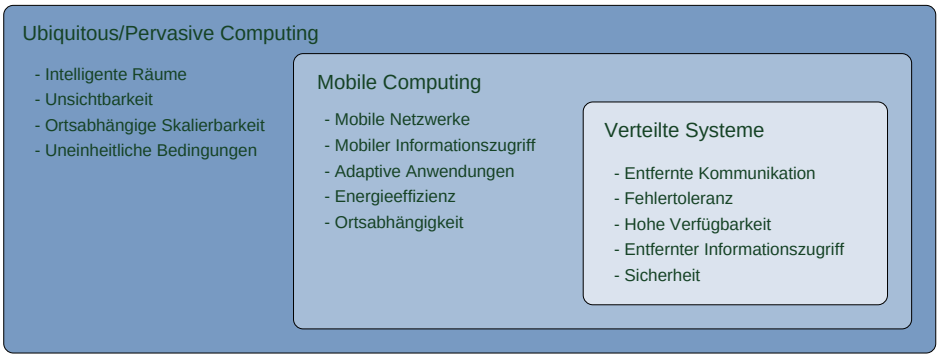


Abbildung 2.6: Anforderungen an das Ubiquitous Computing, das Mobile Computing und an verteilte Systeme (vgl. [Sat01])

2.2.1.1 Verteilte Systeme

Das Fachgebiet der Verteilten Systeme entstand Mitte der 70er-Jahre des 20. Jahrhunderts mit dem Aufkommen lokaler Netzwerke. Die Forschung in dem Bereich setzt sich mit Problemen auseinander, die aus der Vernetzung mehrerer Rechner entstehen [TvS03]. Für das Gebiet des Ubiquitous Computing spielen diese Probleme und die daraus resultierenden Anforderungen eine wichtige Rolle. Dabei handelt es sich um folgende Aufgabenstellungen, für die inzwischen erprobte Lösungen entwickelt wurden.

- *Entfernte Kommunikation:* Hierbei geht es um Themen wie den Entwurf von Protokollschichten, die Durchführung von Remote Procedure Calls und den Umgang mit Timeouts.
- *Fehlertoleranz:* Themen wie atomare Transaktionen, verteilte und geschachtelte Transaktionen und das Zwei-Phasen-Commit sind Bestandteil dieses Bereichs.
- *Hohe Verfügbarkeit:* Optimistische und pessimistische Replikation und optimistische Wiederherstellung sind Hauptschlagworte.
- *Entfernter Informationszugriff:* Hier werden verteilte Datei- und Datenbanksysteme und die daraus resultierenden Probleme und Verfahren behandelt.
- *Sicherheit:* Verfahren zur Gewährleistung von Sicherheit und Privacy, wie z. B. Verschlüsselung und Authentifikation sind hier Schlüsselthemen.

Dies ist lediglich eine kurze Auflistung der Kernpunkte verteilter Systeme. Ausführliche Informationen zu diesem Themenbereich findet man z. B. in [TvS03].

2.2.1.2 Mobile Computing

Die nächste Generation in der Vorgeschichte des Ubiquitous Computing stellt das Mobile Computing dar. Mit dem Aufkommen mobiler Kommunikationstechnologien und mobiler Geräte Anfang der 90er-Jahre des 20. Jahrhunderts entstanden eine Reihe neuer Probleme. Diese resultieren im Wesentlichen aus neuen Anforderungen, die die drahtlose Kommunikation mit sich bringt [Rot05]. Drahtlose Netze können große Qualitäts-Schwankungen haben. Die Robustheit mobiler Komponenten kann eingeschränkt sein. Mobile Endgeräte sind in ihrer Leistungsfähigkeit begrenzt und haben limitierte Energiekapazitäten. Diese Defizite erfordern im Bereich des Mobile Computing neue Lösungen [Rot05], die im Folgenden kurz zusammengefasst werden:

- *Mobile Netzwerke:* Hierzu gehören Mobile IP, Ad-hoc-Protokolle und Technologien, die die Leistung von TCP in drahtlosen Netzwerken steigern.
- *Mobiler Informationszugriff:* Der Umgang mit Verbindungsabbrüchen und daraus resultierende Probleme sind hier Kernaspekte.
- *Adaptive Anwendungen:* Hier werden Themen, wie z. B. die Anpassung der Datenübertragung an die verfügbare Bandbreite und die Kapazität der Endgeräte, behandelt.
- *Energieeffizienz:* Hier geht es um Verfahren zur energiesparenden Verarbeitung von Informationen.
- *Ortsabhängigkeit:* Die Bestimmung und Nutzung des Standorts für die Verarbeitung von Informationen, wie sie in Abschnitt 2.1 diskutiert werden, sind die Kernthemen.

Weitere Informationen zum Thema *Mobile Computing* sind in [Rot05] zu finden.

2.2.1.3 Ubiquitous Computing

Ubiquitäre Computer sind allgegenwärtig vorhanden. Sie sind integriert in die Objekte des Alltags und die Umwelt des Menschen und sollen für den Nutzer unsichtbar sein. Da Mobilität den Alltag des Menschen prägt, kann das Thema Ubiquitous Computing als eine Weiterentwicklung des Mobile Computing gesehen werden. Wenn Computer sich unauffällig in den Alltag integrieren sollen, ist dies ohne die Unterstützung von Mobilität nicht vorstellbar. In [Sat01] werden vier zusätzliche Anforderungen an Applikationen im Bereich des *Ubiquitous Computing* aufgelistet, die Nutzung intelligenter Räume, Unsichtbarkeit, ortsabhängige Skalierbarkeit und der Umgang mit uneinheitlichen Rahmenbedingungen.

Intelligente Räume Ein Raum kann beispielsweise ein abgeschlossener Bereich, wie z. B. ein Konferenzzimmer, sein. Intelligent wird er durch die Integration eingebetteter Computerinfrastruktur, wie Sensoren und die zugehörige „intelligente“ Software. Ein einfaches Beispiel ist die automatische Einstellung der Beheizung und der Lichtverhältnisse eines Raums, abhängig von den elektronisch gespeicherten Vorgaben der Nutzer des Raums.

Unsichtbarkeit Das Idealziel des Ubiquitous Computing wäre ein komplettes Eliminieren der benutzten Computer aus dem Bewusstsein der Nutzer. In der Praxis bedeutet dies, den Benutzer möglichst wenig zu stören. Allerdings ist es nicht einfach, die Erwartungen der Benutzer angemessen zu erfüllen, ohne direkt beim Nutzer nachzufragen. Dieser Konflikt stellt eines der Hauptprobleme des Ubiquitous Computing dar.

Ortsabhängige Skalierbarkeit Wenn man von Skalierbarkeit in Bezug auf klassische Applikationen, wie z. B. Web-Applikationen, spricht, spielt die räumliche Distanz zwischen Client und Server kaum eine Rolle. Anders sieht es jedoch beim Ubiquitous Computing aus. Je weiter sich ein Benutzer z. B. von einem Ubiquitous Computing-Server entfernt, um so mehr muss die Interaktion zwischen dem Benutzer und dem Server abnehmen. Anderenfalls würden sowohl der Nutzer als auch der Server unnötigerweise durch zu viele Anfragen stark belastet.

Uneinheitliche Rahmenbedingungen Ubiquitous Computing-Technologien werden sich nicht an allen Orten gleich schnell durchsetzen [Sat01]. Aus diesem Grund wird es Orte geben, die sehr stark mit dieser Technologie durchsetzt sind, und Orte, bei denen das weniger der Fall ist. Ubiquitous Computing-Clients müssen mit diesem Problem umgehen können, indem sie z. B. versuchen, die fehlenden Kapazitäten vor dem Nutzer zu verbergen. Auf eine nicht vorhandene Netzwerkverbindung etwa könnte man durch das Anbieten einer Offline-Funktion reagieren.

2.2.2 Anwendungsszenarien

Das Ubiquitous Computing verfolgt das Ziel, intelligente Computer in den Alltag von Menschen zu integrieren. Daher ist es nicht verwunderlich, dass sich die Entwicklungen in Forschung und Wirtschaft auf viele unterschiedliche Bereiche des täglichen Lebens beziehen. Es ist davon auszugehen, dass sich das Ubiquitous Computing nicht in allen Bereichen gleich schnell und stark durchsetzt. So kann man bereits heute einen gewissen Zusammenhang zwischen wirtschaftlichen Interessen und dem Implementierungsfortschritt von Ubiquitous Computing-Systemen beobachten [Sch06a]. Je vielversprechender die wirtschaftliche Ausbeute ist, desto höher ist in der Regel der derzeitige Implementierungsgrad. Weitere Treiber sind erhoffte medizinische sowie umwelt- und sicherheitspolitische Fortschritte. Nachfolgend werden einige der derzeit am meisten verbreiteten Anwendungen des Ubiquitous Computing vorgestellt.

Logistik und Produktion

Ein sehr vielversprechendes Anwendungsfeld für Ubiquitous Computing-Systeme ist die Steuerung und Überwachung des Warenflusses, sowohl innerbetrieblich als auch unternehmensübergreifend [WS06]. Grundlage hierfür sind Identifikationssysteme, bei denen Datenträger an Produkten angebracht werden und mittels eines Lesegeräts ausgelesen werden können. Die herkömmliche Variante, dies mittels Barcodes zu bewerkstelligen, wird zunehmend durch die RFID-Technik abgelöst. Eine Reihe von Vorteilen der RFID-Technik macht diesen Schritt verständlich [Fin06]: RFID-Systeme verlangen keine direkte Sichtverbindung, sind weniger empfindlich gegenüber Verschmutzungen, und die gespeicherten Daten lassen sich ändern. Demgegenüber stehen die vergleichsweise hohen, aber stetig sinkenden, Kosten von RFID-Chips. RFID-Systeme ermöglichen die automatische Verfolgung des Warenlaufs. So können Unternehmen sehr viel einfacher, präziser und schneller den Standort ihrer Produkte ermitteln. Die Vorteile, die hierdurch für die Logistik entstehen, liegen auf der Hand [HBHS06]: Teure Lagerkapazitäten für Sicherheitsreserven und Zwischenlagerung können vermieden werden. Die Herkunft von Produkten lässt sich zurückverfolgen, was insbesondere bei Lebensmitteln und sicherheitsrelevanten Bauteilen von z. B. Fahr- und Flugzeugen von hoher Relevanz sein kann bzw. teilweise gesetzlich vorgeschrieben ist [Pit06]. Inventuren in Supermärkten und anderen Ladengeschäften lassen sich automatisiert und effizient durchführen. Für die Produktion von Waren gibt es ebenfalls Vorteile: Steuerungsfunktionen von Anlagen können dezentralisiert werden, wenn jedes Produkt seinen nächsten Produktionsschritt selbst kennt [Sch06a]. Hierdurch erhofft man sich, weniger fehleranfällige Fertigungsprozesse zu etablieren. Die Fälschungssicherheit von Produkten kann durch die Integration kryptographischer Funktionen in die Chips erhöht werden. Anwendungsbeispiele, gerade für die Bereiche Logistik und Produktion, lassen sich fast beliebig fortsetzen. Die vorgestellten Szenarien sollten jedoch an dieser Stelle reichen, um ein Gefühl für die enormem Möglichkeiten des Ubiquitous Computing in diesen Anwendungsbereichen zu vermitteln.

Autoverkehr

Bereits heute existieren eine Reihe von Assistenzsystemen in Fahrzeugen. Sie unterstützen den Fahrer, nehmen ihm teilweise Entscheidungen ab und tragen so zur Fahrsicherheit und zum Fahrkomfort bei. Beispiele sind Regensensoren, die die Scheibenwaschanlage automatisch einschalten, Systeme zur automatischen Geschwindigkeitsregelung und elektronische Stabilitätsprogramme [Sch06a]. Weiter reichende Systeme beziehen die Umgebung des Fahrzeugs stärker mit ein. Durch den Einsatz von Bildverarbeitungstechnologien etwa ist eine Kollisionswarnung bzw.-vermeidung vorstellbar und findet teilweise bereits heute Anwendung [MM02]. In zukünftigen Entwicklungen werden Fahrzeuge untereinander Informationen austauschen. Ein Fahrzeug wird dabei als ein Teil eines Gesamtsystems betrachtet. Durch die gegenseitige Versorgung von Informationen können Gefahrensituationen frühzeitig erkannt werden. Beispielsweise können voranfahrende Fahrzeuge nachfolgende Verkehrsteilnehmer über einen Stau informieren und so eine rechtzeitige Reaktion auf die neue Situation fördern. Eine vielversprechende Initiative ist das „Car 2 Car Communication Consortium“ [C2C], ein Zusammenschluss verschiedener Automobilhersteller, um genau solche Systeme zu entwickeln und die Kompatibilität unter den verschiedenen Herstellen zu gewährleisten.

Identifikationssysteme
Elektronische Identifikationssysteme werden von Firmen und staatlichen Institutionen eingesetzt, um die Identität von Personen zu überprüfen. So soll insbesondere der Zutritt zu bestimmten Bereichen kontrolliert und der Identitätsmissbrauch unterbunden werden [Sch06a]. Ein Beispiel für solche Systeme sind elektronische Reisepässe. Sie sollen die Überprüfung der Identität bei Grenzkontrollen erleichtern und dise sowohl schneller als auch sicherer machen. So werden in Deutschland bereits seit November 2005 Pässe mit RFID-Chips ausgegeben, auf denen biometrische Merkmale gespeichert werden [BMI]. Das Bundesministerium für Gesundheit ist aktuell dabei, die sog. elektronische Gesundheitskarte einzuführen. Auf dieser sollen neben einer Versicherungs-ID wichtige medizinische Informationen gespeichert werden [bmg]. Ausgesprochenes Ziel ist die bestehenden IT-Systeme in Praxen, Apotheken und Krankenhäusern in Deutschland in das System einzubinden, um so eine effektivere und effizientere Kommunikation zwischen den Beteiligten zu ermöglichen. Ein weiteres Einsatzgebiet sind elektronische Mitarbeiterkarten, die viele Unternehmen einsetzen, um z. B. die Anwesenheit ihrer Mitarbeiter zu ermitteln und den Zugang zu den unternehmenseigenen Räumen zu sichern [Sch06a].

Intelligentes Haus
Die Vision des intelligenten Hauses sieht vor, dass Gegenstände im Haus und das Haus selbst mit „unsichtbaren“ Technologien ausgestattet werden. Diese kommunizieren miteinander, mit dem Ziel, den Bewohnern ein komfortableres und sichereres Wohnen zu ermöglichen [BLP06]. Das intelligente Haus soll sich automatisch und unauffällig an die Bedürfnisse der Bewohner anpassen und dabei die Bewohner nicht stören. Die Steuerung der Haustechnik, wie Heizung, Klimaanlage und Beleuchtung, ist hier nur ein erster Schritt. Defekte Geräte, die automatisch eine Fehleranalyse an die Werkstatt schicken können, und Kühlschränke, die in der Lage sind, automatisch eine Einkaufsliste zu erstellen, sind weitere mögliche Bestandteile des intelligenten Hauses. Eine Reihe weiterer möglicher Szenarien und eine Übersicht über die Forschung im Bereich „Ubiquitous Homes“ findet man in [MR03].

Medizintechnik
Mit der Integration von Ubiquitous-Computing-Technologien in die Medizin hofft man, vor allem älteren und chronisch kranken Menschen helfen zu können [Sch06a]. Ziel ist es, den Komfort und die Lebensqualität von Patienten zu steigern und gleichzeitig eine medizinische Versorgung zu gewährleisten. Für kranke Menschen soll ein Alltag in gewohnter Umgebung bei gleichzeitiger ärztlicher Aufsicht ermöglicht werden. So entwickelte beispielsweise die südafrikanische Firma *SIMpill* eine Pillendose, die eine SMS an den behandelnden Arzt schickt, wenn sie für einen längeren Zeitraum nicht gebraucht wird [sim]. Das Unternehmen *VivoMetrics* bietet das *LifeShirt* an [viv]. Mit über 40 integrierten Sensoren kann es physiologische Werte, wie z. B. Blutdruck, Herzfrequenz und Sauerstoffverbrauch, messen und diese Daten zur Analyse an einen Arzt weiterschicken. Dies sind nur zwei Beispiele für eine ganze Reihe möglicher Ubiquitous Computing-Anwendungen im Bereich der Medizintechnik. Weitere Informationen und eine Übersicht über das Thema „Ubiquitous Healthcare“ findet man in [Kun06].

Neben den hier genannten Anwendungsfeldern konzentriert sich die *Ubiquitous Computing*-Forschung auf viele weitere Felder des täglichen Lebens. Zu nennen sind hier beispielsweise

die Bereiche Umwelttechnik, Kommunikation, Notfallhilfe, Militär, Innere Sicherheit, Freizeit, Spiele und elektronischer Handel. Ausführliche Informationen zu diesen Themen findet man z. B. in [SW06]. Einen weiteren Bereich stellt das Thema „Ubiquitous Annotations“ dar. Es spielt in dieser Arbeit eine besondere Rolle und wird deshalb in einem eigenen Unterkapitel (2.3) diskutiert.

2.2.3 Technologien

Ein wichtiger Treiber für die Entwicklung des Ubiquitous Computing sind IT-Komponenten, die immer kleiner, günstiger, energiesparender und „intelligenter“ werden [AGIS05]. Die Integration dieser Technologien in Alltagsgegenstände macht die Realisierung der Ubiquitous Computing-Vision erst möglich [Mat05]. Dabei werden selten alle Technologien in einer Anwendung gemeinsam verwendet. Vielmehr nutzen Anwendungen in der Regel nur einen Teil dieser Technologien. An dieser Stelle werden einige „Enabling Technologies“ vorgestellt, die besonders vielversprechend im Bezug auf das Ubiquitous Computing sind.

Neue Materialien und Ausgabemedien

Das Grundmaterial der Halbleiterindustrie ist bis heute Silizium geblieben. So bestehen auch heutige Mikroprozessoren aus diesem Material. In den vergangenen Jahren gab es jedoch Bestrebungen, nach weiteren Materialien zu forschen, um sie für den Einsatz in Computern nutzen zu können [Mat05].

Ein Beispiel hierfür sind Kunststoffe, aus denen sich dünne und biegsame Displays herstellen lassen. Diese sog. lichtemittierenden Polymere machen es möglich, kleine Geräte mit großen Displays auszustatten. So entwickelt z. B. Philips einrollbare Displays auf Basis dieser Technologie [pv2].

Ein weiteres relativ neues Medium ist das elektronische Papier. Es soll die Vorteile von klassischem Papier mit denen von elektronischen Displays in einem Medium vereinen. Im Endstadium der Entwicklung soll es wie Papier beschrieben, gelesen, gebogen und behandelt werden können. Die darzustellenden Informationen werden jedoch elektronisch gespeichert.

Die hier genannten Beispiele für neue Ausgabemedien spielen in der Welt des Ubiquitous Computing eine wichtige Rolle. Geräte, die immer intelligenter, kleiner und unauffälliger sein sollen, müssen sich nahtlos in die Umwelt des Menschen integrieren. Dies gilt auch für die Darstellung von Informationen. Je weniger sie den Menschen an Computer erinnern, und je leichter sie sich in Gegenstände des Alltags integrieren lassen, desto größer ist die Chance, dass sie sich durchsetzen [Mat05].

Sensoren

Sensoren kann man als die „Sinnesorgane“ intelligenter Geräte betrachten. Sie sind dazu da, ihre Umwelt zu „fühlen“ und die ermittelten Informationen in digitaler Form weiterzugeben [Sch06b]. Dabei gibt es eine Vielzahl unterschiedlicher Sensoren. Neben klassischen Sensoren für die Messung von z. B. Licht, Beschleunigung, Temperatur, Feuchtigkeit und Druck können auch Flüssigkeiten und Gase analysiert werden. Außerdem werden heute Kameras vielfach als Sensoren eingesetzt. So können z. B. durch die Analyse von Fotos Objekte erkannt werden.

Kommunikation
In Zukunft werden viele Alltagsgegenstände, die mit eingebetteten Prozessoren und Sensoren ausgerüstet sind, miteinander kommunizieren [Sch06b]. Sie werden z. B. Informationen über den Aufenthaltsort dieser Alltagsgegenstände oder über andere Sensorwerte an andere Gegenstände übertragen. In diesem Zusammenhang spricht man vom „Internet der Dinge“ [GKC04]. Im klassischen Internet kommunizieren Menschen mit Menschen, Menschen mit Maschinen und Maschinen mit anderen Maschinen. Mit dem Internet der Dinge verfolgt man die Vision, dass auch Alltagsgegenstände miteinander kommunizieren und Informationen austauschen. Hierzu sind in den vergangenen Jahren viele Technologien entwickelt worden, die eine solche Kommunikation ermöglichen. Hierbei handelt es sich insbesondere um kabellose Übertragungstechniken, wie z. B. WLAN, Bluetooth, ZigBee oder die RFID-Technik [Leh03]. Ein wesentliches Designkriterium für diese Verfahren ist eine möglichst hohe Energiesparsamkeit und damit eine möglichst lange autonome Verfügbarkeit [GKC04].

Lokalisierung
Kapitel 2.1 widmete sich den ortsbezogenen Diensten. Diese spielen auch für das Ubiquitous Computing eine wichtige Rolle. Viele Objekte sind mobil und können daher ihren Aufenthaltsort ändern. Daher gehört es zu den Aufgaben vieler Anwendungen, den Ort intelligenter Alltagsgegenstände zu bestimmen bzw. zu kennen [Mat05]. Die Verfahren, die zur Bestimmung des Orts verwendet werden können, wurden bereits in Abschnitt 2.1.2 diskutiert und müssen daher an dieser Stelle nicht erneut erläutert werden.

Energie
Die Gewährleistung ausreichender Energieversorgung ist ein wesentliches Problem von Anwendungen aus dem Bereich des Ubiquitous Computing [Sch06b]. Eine netzgebundene Stromversorgung ist aufgrund der Mobilität der Geräte in vielen Fällen nicht praktikabel. So existieren im Wesentlichen zwei Ansätze, diesem Problem zu begegnen [Mat05]: die Entwicklung neuer langlebigerer Energiequellen und der Einsatz von Techniken zum Energiesparen. Ersteres wird beispielweise durch die Entwicklung von Brennstoffzellen versucht, die eine wesentlich höhere Energiedichte haben als Batterien. Allerdings besteht hier derzeit noch das Problem, dass sich Brennstoffzellen nicht beliebig verkleinern lassen. Energiesparfunktionen sind vertraut aus Funktechnologien, wie z. B. Bluetooth, die speziell für mobile Geräte entwickelt wurden [Leh03]. So verfügt Bluetooth über spezielle Stromsparmodi, die für einen geringeren Energieverbrauch sorgen, wenn keine Daten zu übertragen sind. Komplett ohne eigene Energiequelle kommen dagegen passive RFID-Chips aus, die das Prinzip der magnetischen Induktion ausnutzen, um ihre Energie aus den Funkwellen der RFID-Lesegeräte zu gewinnen [Fin06].

Die hier beschriebenen Technologien stellen lediglich eine Auswahl verfügbarer Technologien dar. Eine Gesamtauflistung würde den Rahmen dieser Arbeit sprengen. Weitere Forschung wird z. B. im Bereich der Mensch-Maschine-Interaktion betrieben. Hierbei wird erforscht, wie zukünftig Menschen mit den intelligenten Dingen des Alltags interagieren und kommunizieren können [Sch06b]. Sicherheitstechnologien, die den sicheren Umgang mit Ubiquitous Computing-Technologien ermöglichen, sind ein anderes wichtiges Thema der Forschung. Auch Middleware-Technologien für Ubiquitous Computing-Systeme werden in der Forschung diskutiert.

2.2.4 Auswirkungen auf den Menschen

Der vorangegangene Abschnitt hat gezeigt, dass das Ubiquitous Computing eine Vielzahl neuer nützlicher Anwendungen verspricht. Jedoch darf nicht außer Acht gelassen werden, dass die Integration unsichtbarer, miteinander kommunizierender, Computer in die Alltagswelt viele Risiken und gesellschaftliche Auswirkungen mit sich bringt [LM03].

In der Geschichte der Entwicklung von Informations- und Kommunikationssystemen wurde immer wieder über mögliche Auswirkungen auf die Gesellschaft diskutiert. Dies gilt vor allem auch für das mobile Computing, das hier ja als Vorläufer des Ubiquitous Computing identifiziert wurde. Allerdings gibt es einen entscheidenden Unterschied zwischen dem Ubiquitous Computing und den Vorgängertechnologien: Während bisher die Geräte immer als Computer im weitesten Sinne erkannt werden konnten, ist dies im Ubiquitous Computing nicht mehr möglich [Hil07]. Daten werden nunmehr gespeichert und übertragen, ohne dass die Nutzer etwas davon mitbekommen. Nicht mehr nur das Handy oder das Notebook sind hierfür verantwortlich, sondern unverdächtige Gegenstände, wie Bücher, Tische, Stifte oder Kühlschränke. Neue Risiken und Auswirkungen auf die Nutzer kommen hier zum Tragen.

In einer umfassenden Studie wurden in [HSK04] mögliche menschliche, soziale und umweltbezogene Auswirkungen des Ubiquitous Computing untersucht. [Hil07] fasst diese unter vier Stichpunkten zusammen:

- *Stress:* Für die Auslösung von Stress kann es laut [Hil07] mehrere Gründe geben. Hierzu gehören die schlechte Benutzbarkeit der Systeme, die Störung und Ablenkung der Aufmerksamkeit und das Gefühl des Überwachtwerdens. Weitere Ursachen können ein möglicher krimineller Missbrauch der Technologien sowie steigende Anforderungen an die Produktivität des Einzelnen sein.
- *Unfreiwilligkeit:* Wenn Ubiquitous Computing-Technologien sich soweit durchsetzen, dass sie die klassischen Anwendungen vollständig ersetzen, könnten Menschen das Gefühl entwickeln, der Technologie ausgeliefert zu sein. Aus diesem Grund ist es wichtig, dass auch in Zukunft alternative Verfahren bestehen bleiben. Wenn etwa die Funktion eines Türschlosses ausschließlich von der Verfügbarkeit von Online-Ressourcen abhängig ist, entsteht eine zu starke Abhängigkeit. Die Nicht-Verfügbarkeit dieser Ressourcen würde in solchen Fällen zur Unbenutzbarkeit der jeweiligen Objekte führen [LM03].
- *Verursacherprinzip:* Bereits heute lassen sich die wahren Ursachen von Schäden, die durch das Zusammenwirken mehrerer Komponenten von Computersystemen in komplexen Netzwerken entstehen, nur schwer ermitteln. Es ist davon auszugehen, dass die mathematische und juristische Beherrschbarkeit von Anwendungen des Ubiquitous Computing aufgrund der technischen Komplexität weiter abnehmen wird. Die Folge könnte sein, dass ein wachsender Anteil des Alltagslebens von unbeherrschbaren Technologien dominiert wird.
- *Ökologische Nachhaltigkeit:* Aufgrund des zu erwartenden steigenden Bedarfs an Rohstoffen könnte der Verbrauch dieser stark ansteigen. Die Entsorgung von Millionen winziger Komponenten als Elektronikabfall könnte zunehmen. Wenn keine adäquaten Regelungen

zum Umgang mit diesen Problemen entwickelt werden, ist zu befürchten, dass wertvolle Rohstoffe verloren gehen, und Schadstoffe in die Umwelt gelangen [KWE+05].

Dem Schutz der Privatsphäre fällt auch im Ubiquitous Computing eine besondere Rolle zu [Mac06]. Hier werden eine Reihe von Fragen aufgeworfen, die bis heute nicht abschließend beantwortet werden können. Wie kann der Privacy-Schutz in ubiquitären Umgebungen gewährleistet werden? Wie kann das Gebot der Datensparsamkeit [BDS] in einer Welt, in der alles und jeder Daten sammelt, noch sichergestellt werden? Wie kann sich der Einzelne vor einer Überwachung durch staatliche Stellen schützen? Wie kann man die Abhängigkeit von Ubiquitous Computing-Technologien minimieren? Was muss getan werden, um das Vertrauen und damit die Akzeptanz der Nutzer zu steigern? Diese und weitere Fragen stehen im Mittelpunkt verschiedener Ausarbeitungen [FL07, Hil07, Lan07, Pit06, Roß07], auf die der interessierte Leser an dieser Stelle verwiesen wird.

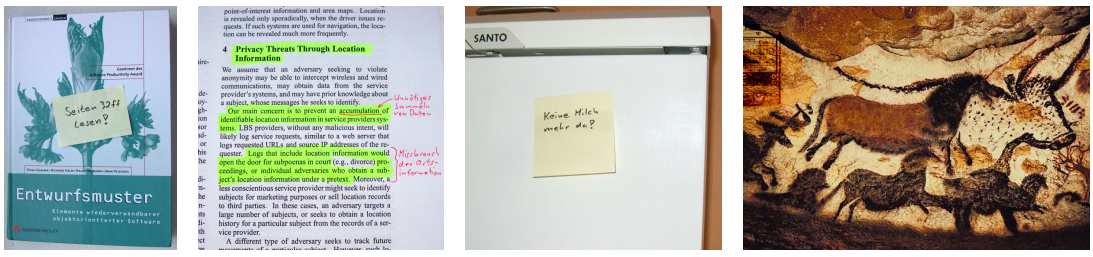
2.3 Ubiquitäre Annotationen

Bereits in Abschnitt 2.2.2 wurden die Anwendungsszenarien für das Ubiquitous Computing aufgezählt und erläutert. Dabei wurden ubiquitäre Annotationssysteme als ein Anwendungsbereich identifiziert. Dieser wird auf den folgenden Seiten näher erörtert.

Zunächst wird in Abschnitt 2.3.1 der Begriff der „Annotation“ näher beschrieben. Anschließend folgt in Abschnitt 2.3.2 eine Betrachtung der Besonderheiten von ubiquitären Annotationen. Zum Schluss werden in Abschnitt 2.3.3 die Anforderungen an ubiquitäre Annotationssysteme diskutiert.

2.3.1 Annotationen

Schlägt man im Brockhaus nach, findet man für den Begriff „Annotation“ die Erklärung „Aufzeichnung, Vermerk, Anmerkung“. Dabei gibt es verschiedene Möglichkeiten und Gründe dafür, Dinge aufzuzeichnen, zu vermerken oder Anmerkungen zu machen [Han06]. Anhand von vier Beispielen sollen diese veranschaulicht werden (vgl. Abb. 2.7).



(a) Annotiertes Buch (b) Annotierter Text (c) Annotierter Kühlschrank (d) Höhlenmalerei

Abbildung 2.7: Beispiele für Annotationen

In Abbildung 2.7(a) ist die Annotation eines Buches mit einer Notiz erkennbar. Auf der Notiz hat der Leser vermerkt, dass er bestimmte Seiten im Buch lesen möchte. Er nutzt die Annotation in diesem Fall zur Erinnerung an eine Tätigkeit, die er später durchführen will.

Abbildung 2.7(b) zeigt einen annotierten Text. Der Leser hat sich Notizen gemacht und bestimmte Textpassagen markiert. Diese Annotationen dienen hier zur Kommentierung und Zusammenfassung des Textes. Man kann sie außerdem als Personalisierung des Textes betrachten.

Im dritten Beispiel (Abb. 2.7(c)) ist ein annotierter Kühlschrank abgebildet. Auf einem Notizzettel wird darauf hingewiesen, dass es keine Milch mehr im Kühlschrank gibt. Die Annotation hat den Zweck, Mitbewohner auf diesen Umstand hinzuweisen. Sie dient als Kommunikationsmittel und u. U. als Aufforderung an die Mitbewohner, Milch einzukaufen.

Annotationen sind keine neue Erfindung, sondern werden bereits seit langer Zeit vom Menschen genutzt. Dies soll Abbildung 2.7(d) verdeutlichen. Dort sieht man Tiere als Malerei an der Wand einer Höhle. Menschen nutzten solche Annotationen, um ihre Mitmenschen vor gefährlichen Tieren zu warnen oder sie auf mögliche Nahrungsquellen hinzuweisen.

Anhand dieser Beispiele sollte deutlich geworden sein, dass Menschen Annotationen häufig und für verschiedenste Zwecke im Alltag nutzen. Mit ubiquitären Annotationen versucht man nun, diesen selbstverständlichen und intuitiven Umgang mit Annotationen für das Ubiquitous Computing zu nutzen. Das Ziel ist die Verwendung einer wohlbekannten Metapher aus der Alltagswelt, um sich der Vision des Ubiquitous Computing anzunähern [Han06].

2.3.2 Ubiquitäre Annotationen

Ubiquitäre Annotationssysteme erlauben Nutzern physikalische Orte und Personen mit digitalen Informationen zu annotieren [APV+06]. Schon vor der Zeit des *Ubiquitous Computing* gab es Bestrebungen, Objekte zu annotieren bzw. miteinander zu verknüpfen. So stellte 1945 Vannevar Bush den sog. Memex vor [Bus45].

Abbildung 2.8: Der Memex

Abbildung 2.8 zeigt dieses fiktive Gerät in Form eines Schreibtischs. Es sollte in der Lage sein, Dokumente auf Mikrofilmen zu speichern und diese auf zwei Bildschirmen anzuzeigen. Auf einer transparenten Fläche auf der linken Seite sollte der Nutzer Notizen machen können. Ferner sollte es möglich sein, in Dokumenten zu blättern und Verknüpfungen zwischen diesen zu erstellen. So sollte ein schnelles Wechseln von miteinander verknüpften Dokumenten möglich sein. Dieses Prinzip ist heute in Form von Hyperlink aus dem Internet bekannt.

Der Memex wurde zwar nie gebaut, aber diente in den folgenden Jahren als Vision für die Versuche, digitale Annotations- und Hypertextsysteme zu realisieren. Neben dem heutigen Internet entstanden viele Systeme zur digitalen Annotation digitaler Objekte, wie z. B. *Microcosm* [DCKH94] und *Chimera* [ATJ00]. Ein aktuelles Beispiel ist z. B. das Add-On *Fleck* für den Web-Browser *Firefox*. Mit ihm lassen sich gewöhnliche Webseiten digital annotieren (Abbildung 2.9).

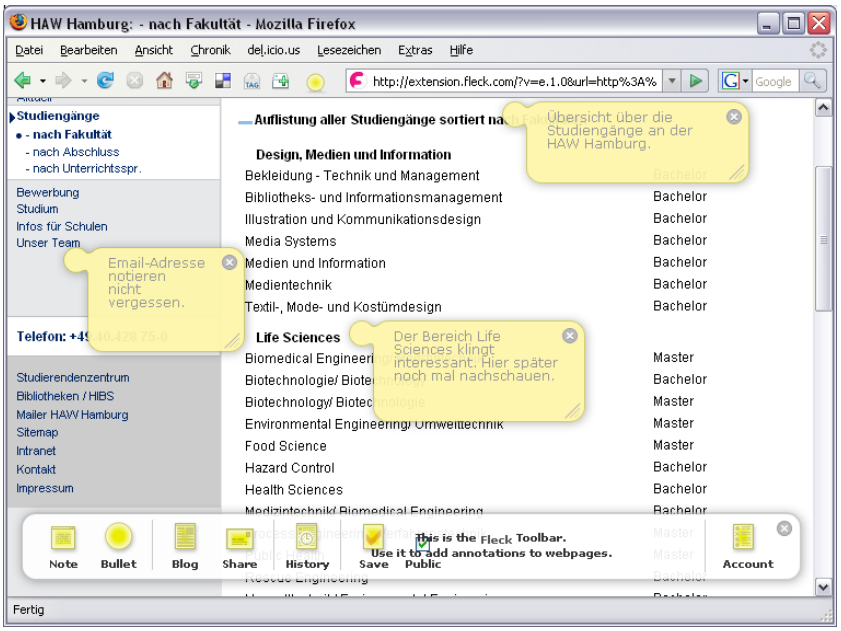


Abbildung 2.9: Eine annotierte Webseite mit dem Firefox Add-On *Fleck*

Mit dem Zeitalter des Ubiquitous Computing entstand schließlich die Idee, auch reale Objekte digital zu Annotieren [APV+06]. Eine Reihe von Forschungsarbeiten beschäftigen sich mit diesem Thema [AAH+97, Pas97, EPS+01, MS00, PBC+01, BCGH03]. Im Projekt *Websigns* beispielsweise wurde eine PDA-basierte Software entwickelt [PBC+01]. Sie ermöglicht, Informationen über zuvor annotierte Gebäude anzuzeigen (Abbildung 2.10).

Die oben genannten Projekte befassen sich allesamt mit der Frage, auf welche Art und Weise reale Objekte annotiert werden können, machen Vorschläge zur Realisierung und stellen eine prototypische Implementierung vor. Im folgenden Abschnitt 2.3.3 werden einige Ergebnisse, die sich aus den genannten Arbeiten ergeben, zusammengefasst.

Abbildung 2.10: *Websigns*: Annotierte Gebäude

2.3.3 Anforderungen

In [Han06] werden wesentliche Anforderungen an ubiquitäre Annotationssysteme aufgezählt. Diese lassen sich z. B. in [APV+06] wiederfinden. Es handelt sich hierbei um folgende Merkmale:

- Identifizierung von Objekten
- Präsentation von Annotationen
- Erstellung und Bearbeitung von Annotationen

Der folgende Abschnitt wird sich mit den oben genannten Merkmalen auseinandersetzen.

2.3.3.1 Identifizierung von Objekten

Bei der physikalischen Annotation von Objekten ist es meist möglich, die Annotation direkt am Objekt selbst anzubringen. Beispielsweise würde man eine Post-It-Notiz einfach auf ein Buch kleben, um dieses zu annotieren. Diese Form der Annotation hat den großen Vorteil, dass das annotierte Objekt selbst in der Annotation nicht beschrieben werden muss. Die Post-It-Notiz auf dem Buch etwa muss keinen Verweis auf den Namen des Buches enthalten, da durch das direkte Anbringen der Annotation auf das Buch bereits ein impliziter Verweis vorhanden ist.

Leider ist diese Art der Annotation bei digitalen Systemen nur selten möglich [Han06], da Objekte des täglichen Lebens in der Regel nicht über die Möglichkeit verfügen, digitale Informationen zu speichern. Die zu annotierenden Objekte, wie z. B. Bücher, müssten über einen digitalen Speicher verfügen und Displays besitzen, damit die traditionelle Art der Annotation eins zu eins auf die digitale Annotation übertragen werden kann.

Aufgrund der genannten Problematik versucht man, bei ubiquitären Annotationssystemen einen anderen Weg zu gehen. Statt die Objekte direkt zu annotieren verleiht man den Objekten Identitäten und verknüpft die Annotationen mit diesen Identitäten. Beispielsweise könnte man eine Annotation auf ein Buch auf die ISBN-Nummer des Buches verweisen lassen.

In bereits durchgeführten Projekten [PBC+01, HBC+04, CCD+04, Roh05] im Bereich „Ubiquitous Annotations“ findet man zwei unterschiedliche Typen von Identifikationstechnologien, die eingesetzt werden: Verfahren die auf Positionsbestimmung basieren und Tag/ID-basierte Verfahren. Die meisten Projekte, die den Weg der Positionierungsbestimmung wählen, machen dies über GPS (vgl. [HBC+04, PBC+01]). Aber auch andere bekannte Positionierungsverfahren, wie z. B. Positionierung über WLAN, GSM, Bluetooth oder Ultraschall, sind grundsätzlich einsetzbar. Bei den Tag/ID-basierten Verfahren kommen primär RFID- oder Barcode-Tags zum Einsatz. Beim RFID-Verfahren werden die Objekte mit RFID-Tags versehen. Diese können über ein RFID-Lesegerät per Funk erfasst werden [GKOE03]. Bei den Barcode-Verfahren werden die Objekte hauptsächlich mit 2D-Barcodes beklebt. Diese können über eine Kamera abfografiert und über entsprechende Bildverarbeitungsverfahren dekodiert werden [Roh05]. Die Mechanismen und Algorithmen zur Dekodierung von Barcodes werden u. a. in [OHH04] und [KT05] diskutiert.

2D-Barcodes haben sich gegenüber eindimensionalen Barcodes durchgesetzt, weil sie wesentlich mehr Informationen speichern können. Darüber hinaus sind sie meist so konzipiert, dass sich beschädigte Teile des Codes wiederherstellen lassen. Dies ist bei einigen Codes sogar bei bis zu 50% Beschädigung noch möglich [KT05]. Auf der anderen Seite sind die Algorithmen zur Erkennung der eindimensionalen Barcodes meist einfacher und performanter. Abbildung 2.11 zeigt drei der am häufigsten verwendeten 2D-Barcodes. Sie Unterscheiden sich in ihrem Aussehen, ihrer Größe, der maximalen Speicherkapazität und der maximalen Fehlertoleranz. Während die Codes links und in der Mitte die URL *http://www.haw-hamburg.de* enthalten, speichert der rechts abgebildete Barcode die Zahl *666654988052*.

(a) QR-Code (b) Datamatrix (c) Shotcode

Abbildung 2.11: 2D-Barcodes - drei Beispiele

2.3.3.2 Präsentation von Annotationen

Die Identifizierbarkeit von Objekten spielt für ubiquitäre Annotation eine wichtige Rolle. Eine weitere wichtige Designentscheidung muss man jedoch treffen, wenn festgelegt werden soll, in welcher Form die Annotationen den Nutzern anzuzeigen sind.

In [Mac98] wird hierzu festgestellt, dass man die Präsentation auf drei unterschiedliche Arten durchführen kann:

- Anreicherung des Benutzers,
- Anreicherung des annotierten Objekts,
- Anreicherung der Umwelt, die den Benutzer und das Objekt umgibt.

Die Anreicherung des Benutzers kann z. B. mit einem Head-Mounted-Display (HMD) realisiert werden. Die Annotationen würden in dem Fall auf dem HMD angezeigt werden, sobald ein Nutzer das Objekt betrachtet. Die Anreicherung des annotierten Objekts wäre beispielsweise über eine Integration von Displays in die Objekte denkbar. Die dritte Möglichkeit, die Anreicherung der Umwelt, könnte über Videoprojektoren oder öffentliche Displays realisiert werden, die die Annotationen in der Nähe der Objekte anzeigen.

Diese Klassifikation stammt ursprünglich aus der Welt des Virtual Reality. Sie berücksichtigt deshalb nicht, dass man Annotationen u. U. nicht nur in direkter Nähe des annotierten Objekts betrachten möchte. Vorstellbar ist etwa, die Annotationen über das Internet lesen zu können.

In [Han06] wird daher für ubiquitäre Annotationen eine weitere Klassifikation vorgeschlagen. Demnach kann das Betrachten von Annotationen *on-location*, d. h. direkt am Objekt selbst, oder *off-location* (z. B. über das Internet) geschehen. Außerdem kann die Annotation *attached*, d. h. direkt am annotierten Objekt angebracht, oder *detached*, d. h. vom Objekt losgelöst, sein.

2.3.3.3 Erstellung und Bearbeitung von Annotationen

Neben der Präsentation von Annotationen spielt die Möglichkeit, Annotationen zu erstellen und zu bearbeiten, eine zentrale Rolle. Von ubiquitären Annotationen kann nur dann die Rede sein, wenn die Annotationen nicht nur präsentiert werden, sondern prinzipiell von jedermann erstellt und bearbeitet werden können. Gerade frühe Entwicklungen in diesem Bereich lassen oft nur einen lesenden Zugriff auf Annotationen zu und verwehren dem Nutzer die Möglichkeit, Annotationen zu erstellen und zu bearbeiten [PBC+01]. Die Annotationen werden bei solchen Systemen typischer Weise von Autoren erstellt und in das System eingepflegt.

Ein wichtiger Grund dafür, dass das Erstellen und Bearbeiten von Annotationen nicht in allen Applikationen möglich ist, mag die Tatsache sein, dass es nicht immer einfach ist, die Präsentation und das Erstellen bzw. Bearbeiten in einer Applikation bzw. in einem Endgerät zu kombinieren. Der Umstand, dass es sich bei den Geräten häufig um mobile Geräte mit eingeschränkten Hard- und Softwarekapazitäten handelt, trägt hierzu wesentlich bei.

Für das Erstellen bzw. Bearbeiten von Annotationen kann prinzipiell die gleiche Klassifikation verwendet werden wie für die Präsentation [Han06]. Das bedeutet, dass zwischen *on-* und *off-location* und zwischen *attached* und *detached* unterschieden wird.

Zusammenfassend sei betont, dass Annotationen der *many-to-many*-Semantik folgen sollten und nicht der *one-to-many*-Semantik [CCD+04]. Das heißt, dass prinzipiell die Möglichkeit bestehen sollte, dass Annotationen von jedermann erstellt, bearbeitet und betrachtet werden können.

3 Verwandte Forschungsarbeiten

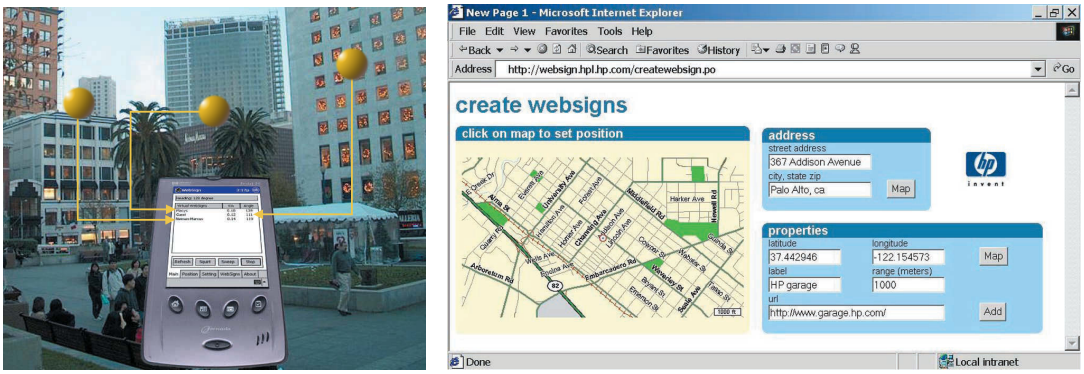
In diesem Kapitel wird eine Reihe von Arbeiten vorgestellt, die sich mit den Themen „Ortsbezogene Dienste“ oder „Ubiquitäre Annotationen“ auseinandersetzen. Dem Leser soll so ein Einblick in konkrete Entwicklungen und Forschungsschwerpunkte ermöglicht werden. Außerdem ermöglicht dieses Vorgehen eine Abgrenzung und Spezifizierung dieser Arbeit im nachfolgenden Kapitel 4. Im Anschluss an die Projektbeschreibungen werden die wesentlichen Merkmale des jeweiligen Projekts bezüglich ortsbezogener Dienste und ubiquitärer Annotationen tabellarisch zusammengefasst.

3.1 Websigns

Das *Websigns*-Projekt wurde 2001 im Rahmen des *HP CoolTown*-Forschungsprojekts in den HP-Laboratories in Palo Alto, Kalifornien, durchgeführt [PBC+01].

Die Anwendung ermöglicht, physikalische Orte mit digitalen Informationen anzureichern. Beispielsweise soll man auf seinem PDA Informationen zu einem Gebäude erhalten können, sobald man in die Nähe des Gebäudes kommt.

Das zentrale Objekt stellen hierbei die sog. *Websigns* dar. Dabei handelt es sich um virtuelle Markierungen von Objekten der realen Umgebung. In Abbildung 3.1(a) sind drei *Websigns* mit jeweils einer gelben Kugel gekennzeichnet. Die Kugeln dienen hier der Veranschaulichung und sind in der Realität nicht sichtbar.



(a) Webisgns (virtuelle Markierungen) (b) Webseite zur Erstellung von Annotationen

Abbildung 3.1: *Websigns* (vgl. [PBC+01])

Als Endgerät dient ein PDA mit Internetzugang, einem GPS-Empfänger und einem digitalen Kompass. Die Software lädt auf Anforderung des Benutzers alle verfügbaren *Websigns*, die sich in der Nähe des Benutzers befinden, von einem Server über das Internet auf das Gerät und cached diese.

Es gibt zwei unterschiedliche Arten von *Websigns*: „Virtuelle Beacons“ und „Virtuelle Tags“. Die Nutzer sehen die virtuellen Beacons nur dann, wenn sie sich in ihrer Nähe befinden und mit dem PDA auf diese zeigen. Ein virtueller Tag dagegen wird automatisch angezeigt, sobald ein Nutzer in die Nähe des Tags kommt.

Über eine Webseite können die *Websigns* erstellt und bearbeitet werden (vgl. Abb. 3.1(b)). Das Erstellen von *Websigns* ist nur durch bestimmte Nutzer mit Administrationsrechten möglich. Eine Navigation ist nicht vorgesehen, eine Kartenansicht auf dem mobilen Gerät wird ebenfalls nicht zur Verfügung gestellt.

Tabelle 3.1 fasst die wesentlichen Merkmale der *Websigns*-Anwendung zusammen.

Ortsbezogene Dienste:	*Positionierungstechnik:* *Indoor / Outdoor:* *Anwendungsszenario:* *Push- / Pull-Dienst:* *Navigation möglich:*	GPS Outdoor Information Push und Pull nein
Ubiquitäre Annotationen:	*Identifizierung:* *Präsentation:* *Erstellen / Bearbeiten:*	Positionierung (GPS) on-location / detached (über PDA) off-location / detached (über Web-Browser)

Tabelle 3.1: *Websigns:* Merkmale bezüglich ortsbezogener Dienste und ubiquitärer Annotationen

3.2 LoL@

Das Projekt *LoL@* (Local Location Assistant) ist 2002 am Forschungszentrum Telekommunikation in Wien entstanden. Es handelt sich um einen elektronischen Touristenführer für die Innenstadt von Wien [Gün02].

Die Anwendung wurde konzipiert für ein Mobiltelefon. Die Realisierung des Prototyps erfolgte jedoch aus praktischen Gründen auf einem Notebook (vgl. Abb. 3.2).

Mit LoL@ lassen sich verschiedene vordefinierte Touren durch die Innenstadt Wiens durchführen. Dabei wird man von Punkt zu Punkt navigiert. Hierzu existiert eine Kartenansicht. Zu Orten von Interesse gibt es Informationen in Textform.

Der Nutzer kann sich vor Beginn der Tour eine Route aussuchen und diese anschließend virtuell auf dem mobilen Gerät betrachten („Hotel Room Scenario“). Die Route kann auch real abgelaufen werden („Walk through the City Scenario“). Anschließend kann sich der Nutzer die abgelaufene Route auf dem mobilen Gerät ansehen.

Abbildung 3.2: LoL@ - Local Location Assistant (vgl. [Gün02])

Die Positionierung erfolgt über GPS. Zusätzlich kann der Nutzer das Erreichen von Orten von Interesse manuell bestätigen. Die Anwendung ist auf die Wiener Innenstadt beschränkt. Touren an anderen Orten sind bei diesem System nicht möglich.

Ortsbezogene Dienste:	*Positionierungstechnik:* *Indoor / Outdoor:* *Anwendungsszenario:* *Push- / Pull-Dienst:* *Navigation möglich:*	GPS Outdoor Information, Navigation Push ja
Ubiquitäre Annotationen:	*Identifizierung:* *Präsentation:* *Erstellen / Bearbeiten:*	Positionierung (GPS) on- und off-location / detached (über Handy) on- und off-location / detached (über Handy)

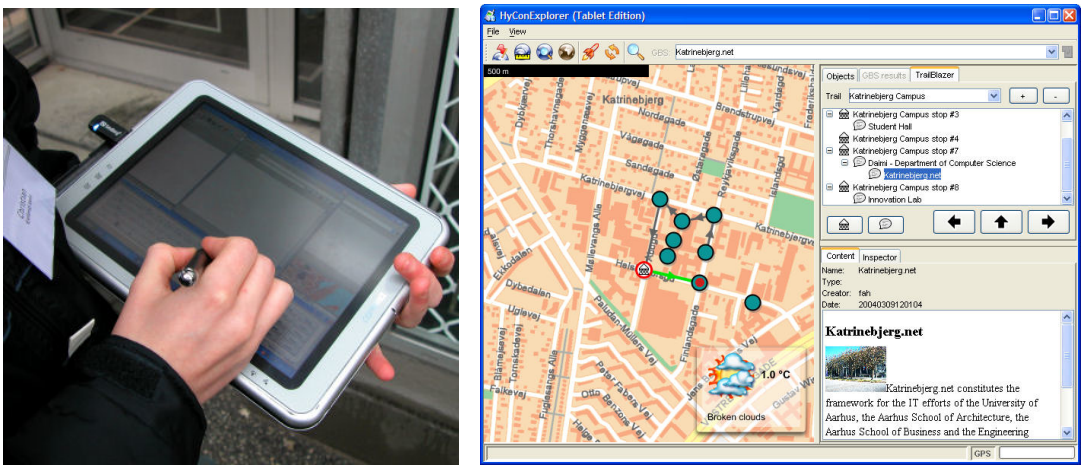
Tabelle 3.2: *LoL@:* Merkmale bezüglich ortsbezogener Dienste und ubiquitärer Annotationen

3.3 HyCon Explorer

Der *HyCon Explorer* [HBC+04] ist 2004 im Rahmen des *ContextIT*-Projekts des *Danish National Centre of IT Research* an der *University of Aarhus* entstanden [HBC+04].

Mit dem *HyCon Explorer* können Nutzer Orte annotieren. Es handelt sich um eine Java-Applikation, die auf einem Tablet-PC betrieben wird (vgl. Abb. 3.3(a)). Außerdem ist eine weitere Version verfügbar, die auf einigen *Nokia*-Handys mit einem *Symbian* Betriebssystem läuft. Es wird eine Internetverbindung über GPRS oder WLAN vorausgesetzt. Mittels einer Kamera ist die Aufnahme von Fotos und Videos möglich, über ein Mikrofon können Audioclips aufgezeichnet werden. Ein GPS-Empfänger dient zur Positionsbestimmung.

Der *HyCon-Explorer* bietet die Möglichkeit, ortsbasierte Annotationen zu erstellen. Nutzer können die Orte, an denen sie sich aufhalten, annotieren. Andere Nutzer, die die gleichen Orte besuchen, können die Annotationen lesen und kommentieren. Abbildung 3.3(b) zeigt einen Screens-



(a) Bedienung mit einem Tablet-PC (b) Ein Screenshot der Applikation

Abbildung 3.3: *HyCon-Explorer (vgl. [HBC+ 04])*

hot des *HyCon-Explorers*. Im linken Bereich sehen Nutzer ihre aktuelle Position und die Position bereits vorhandener Annotationen auf einer Landkarte. Eine Liste der Annotationen der näheren Umgebung ist im oberen rechten Bereich sichtbar. Unten rechts können Nutzer den Inhalt der Annotationen lesen und betrachten.

Ortsbezogene Dienste:	*Positionierungstechnik:* *Indoor / Outdoor:* *Anwendungsszenario:* *Push- / Pull-Dienst:* *Navigation möglich:*	GPS Outdoor Information Push und Pull nein
Ubiquitäre Annotationen:	*Identifizierung:* *Präsentation:* *Erstellen / Bearbeiten:*	Positionierung (GPS) on- und off-location / detached (über Tablet-PC oder Handy) on- und off-location / detached (über Tablet-PC oder Handy)

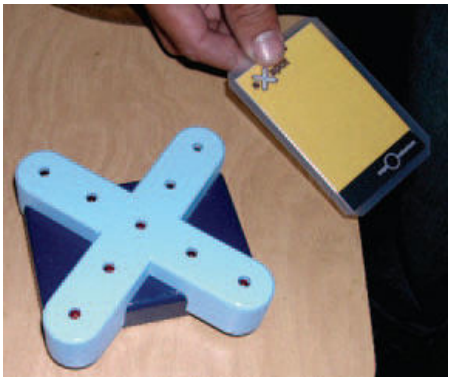
Tabelle 3.3: *HyCon Explorer:* Merkmale bezüglich ortsbezogener Dienste und ubiquitärer Annotationen

3.4 eXspot

Das *eXspot*-Projekt wurde in den Jahren 2003 bis 2005 in Zusammenarbeit der *University of Washington*, den *Intel Labs Seattle* und dem *Exploratorium* entwickelt [HF05]. Beim *Exploratorium*[1] handelt es sich um ein Wissenschaftsmuseum in San Francisco.

[1] Exploratorium: http://www.exploratorium.edu

Die *eXspot*-Applikation funktioniert folgendermaßen: Jeder Besucher des Exploratoriums erhält beim Eintritt eine Karte mit einem RFID-Tag, der eine eindeutige ID speichert. Die Ausstellungsstücke im Museum sind mit RFID-Lesegeräten ausgestattet. Abbildung 3.4(a) zeigt ein solches Lesegerät (links) und eine RFID-Karte (rechts).



(a) RFID Lesegerät (links) und RFID-Karte (rechts)

(b) Screenshot der eXspot Webseite

Abbildung 3.4: *eXspot* (vgl. [HF05])

Besucher können sich mit ihren RFID-Karten bei den Ausstellungsstücken, für die sie sich besonders interessieren, registrieren. Die Registrierung wird über WLAN an einen Server übertragen. Dieser wiederum generiert im Hintergrund eine personalisierte Webseite für den Besucher. Auf diese Weise werden Informationen über alle getaggten Exponate auf der Webseite verlinkt.

Die Besucher können ihre RFID-Karten auch aktiv einsetzen, um Aktionen an bestimmten Exponaten auszulösen. Als Beispiel hierfür sei das Ausstellungsstück „Heat Camera“ genannt. Besucher können dort sich selbst von einer Wärmebildkamera aufzeichnen lassen und das so entstehende Wärmebild auf einem Monitor beobachten. Sie haben die Möglichkeit, mit ihrer RFID-Karte eine Fotokamera zu aktivieren. Das dabei gemachte Foto wird automatisch in die personalisierte Webseite des Besuchers integriert.

Nach dem Besuch des Museums können sich die Besucher mit ihrer ID auf der Internetseite des Museums anmelden. Dort kann man den zuvor durchgeführten Museumsbesuch noch ein Mal virtuell nachstellen. Es werden primär die Exponate angezeigt, bei denen sich der Besucher zuvor im Museum registriert hat. Abbildung 3.4(b) zeigt einen Screenshot der Webseite mit einem Wärmebild eines Besuchers und weiterführenden Informationen.

Ortsbezogene Dienste:	*Positionierungstechnik:* *Indoor / Outdoor:* *Anwendungsszenario:* *Push- / Pull-Dienst:* *Navigation möglich:*	RFID Indoor Information Pull nein
Ubiquitäre Annotationen:	*Identifizierung:* *Präsentation:* *Erstellen / Bearbeiten:*	Tag/ID-basiert (RFID) off-location / detached (über Webseite) on-location / detached (über RFID-Karte)

Tabelle 3.4: *eXspot:* Merkmale bezüglich ortsbezogener Dienste und ubiquitärer Annotationen

3.5 Semapedia

Als letztes Projekt soll hier *Semapedia*[2] vorgestellt werden. Nutzer von *Semapedia* können Objekte der Realität mit 2D-Barcodes versehen, um sie mit digitalen Informationen zu verknüpfen.

Auf der Webseite von *Semapedia* können die Nutzer zu beliebigen Wikipedia-Einträgen 2D-Barcodes erzeugen und ausdrucken. Diese codieren die URL zum entsprechenden Wikipedia-Artikel, z. B. zum Eintrag über die HAW-Hamburg (vgl. Abb. 3.5(a)). Den erzeugten Barcode kann man nun ausdrucken und an einer geeigneten Stelle an der HAW anbringen. Andere Nutzer können ihr mobiles Gerät (z. B. PDA oder Handy) nutzen, um über die *Semapedia*-Software den Barcode abzufotografieren. Die Software erkennt die codierte URL und leitet den Nutzer auf die zugehörige Wikipedia-Seite weiter (vgl. Abb. 3.5(b)).

(a) 2D-Barcode

(b) Wikipedia-Artikel über die HAW, optimiert für mobile Geräte

Abbildung 3.5: *Semapedia*

[2] Semapedia: http://www.semapedia.org

Ortsbezogene Dienste:	*Positionierungstechnik:* *Indoor / Outdoor:* *Anwendungsszenario:* *Push- / Pull-Dienst:* *Navigation möglich:*	2D-Barcodes Indoor und Outdoor Information Pull nein
Ubiquitäre Annotationen:	*Identifizierung:* *Präsentation:* *Erstellen / Bearbeiten:*	Tag/ID-basiert (2D-Barcodes) on-location / detached (über Handy oder PDA) off-location / detached (über Webseite)

Tabelle 3.5: *Semapedia:* Merkmale bezüglich ortsbezogener Dienste und ubiquitärer Annotationen

Neben den oben vorgestellten Projekten existiert eine Reihe weiterer Arbeiten, die sich mit ähnlichen Themen befassen. Eine Übersicht ist z. B. in [KB03] zu finden.

4 Systemdesign

Ein Teil der vorliegenden Arbeit besteht in der prototypischen Entwicklung eines Anwendungssystems. In Kapitel 4 werden Designüberlegungen bzgl.. des Systems diskutiert. Zunächst wird in Abschnitt 4.1 ein Ausgangsszenario beschrieben. Die folgenden Abschnitte und Entwürfe basieren auf diesem Szenario. Abschnitt 4.2 erörtert die Anforderungen an das Anwendungssystem. In den darauf folgenden drei Abschnitten 4.3, 4.4 und 4.5 werden Architekturdetails der Komponenten des Systems vorgestellt.

4.1 Ausgangsszenario

Im Rahmen dieser Arbeit sollen verschiedene Ansätze aus den Forschungsarbeiten, die in Kapitel 3 vorgestellt wurden, in einem neuen Anwendungssystem erprobt werden. Dabei sollen Technologien aus den Bereichen „Ortsbezogene Dienste“ und „Ubiquitäre Annotationen“ miteinander verknüpft und so in einen neuen Zusammenhang gebracht werden. Als Szenario für die Arbeit dient ein Stadtrundgang für Touristen, der durch mobile Geräte unterstützt wird.

Folgende Merkmale sollen als Kernstück der Anwendung dienen:

- Die Software soll einen Stadtrundgang von der Planung über die Durchführung bis zur Rekapitulation des Erlebten unterstützen.
- Dem Touristen sollen Informationen für Orte von Interesse (OvIs) im Freien, aber auch für Objekte innerhalb von Gebäuden zur Verfügung stehen.
- Der Tourist soll eigene Annotationen in Text- und Bildform hinterlegen und diese nach dem Stadtrundgang betrachten können.

Diese Merkmale beruhen auf der Beobachtung, dass aktuelle Anwendungssyteme für Stadtrundgänge meist sehr statisch einen Rundgang vorgeben [KB03]. Der Benutzer selbst hat kaum Möglichkeiten, die Route seines Rundgangs zu beeinflussen und selbst Aktionen durchzuführen. Die hier entwickelte Anwendung soll dem Nutzer mehr Freiheiten einräumen und so die Durchführung individueller Stadtrundgänge ermöglichen.

Anhand eines beispielhaften Szenarios wird nachfolgend der Funktionsumfang der zu entwickelnden Anwendung umrissen.

Marco besucht die Stadt Hamburg zum ersten Mal und möchte sie besser kennenlernen. Um einen Stadtrundgang zu planen, begibt er sich auf eine Webseite, auf der er zunächst einen Rundgang für Hamburg auswählt. Anschließend hat er die Möglichkeit, nach bestimmten Orten von Interesse zu suchen. Er kann sich über die Orte informieren, und sie auf Wunsch zu seiner

persönlichen Route hinzufügen. Marco führt diesen Vorgang so lange durch, bis er alle Orte, die er besuchen möchte, ausgewählt hat.

Hiernach installiert Marco eine Software für den Stadtrundgang auf seinem PDA. Er startet die Software und lädt sich die zuvor erstellte Route herunter. Nun kann der Rundgang beginnen. Marco erhält Navigationsanweisungen, die ihn zum ersten Ziel, den Landungsbrücken, führen. Zusätzlich sieht er seine Position und die Position des Ziels auf einer Karte. Sobald er das Ziel erreicht, werden ihm entsprechende Hintergrundinformationen angezeigt. Marco sieht sich außerdem historische Bilder der Landungsbrücken auf dem PDA an. Während des gesamten Stadtrundgangs kann Marco mit dem PDA Fotos und Textnotizen machen. Diese werden, verknüpft mit der aktuellen Position, gespeichert. Auch die gesamte Wegstrecke, die Marco abläuft, wird festgehalten. Anschließend lotst ihn die Software zum Museumsschiff Rickmer Rickmers. Auch hier erhält Marco Informationen und beschließt daraufhin, das Schiff zu besichtigen. Auf und unter Deck werden verschiedene Exponate ausgestellt. Marco kann sich zu diesen Exponaten ebenfalls Hintergrundinformationen auf seinem PDA ansehen. Nach der Schiffsbesichtigung besucht Marco einige weitere Sehenswürdigkeiten in der Umgebung und beendet seinen Stadtrundgang.

Zu Hause angekommen, lädt Marco die soeben abgelaufene Route von seinem PDA wieder hoch. Auf der schon zu Beginn besuchten Webseite kann er sich nun die eigene Hamburg-Tour ansehen. Auf einer Karte werden die zurückgelegte Strecke und die besuchten Orte angezeigt. Die Informationen über die Orte kann Marco ebenfalls abrufen. Außerdem findet er die Fotos und Notizen, die er während der Tour gemacht hat, an den entsprechenden Stellen auf der Karte wieder. Er kann die Strecke virtuell erneut ablaufen und seine Eindrücke mit seinen Freunden teilen. Wenn Marco möchte, können sich andere Nutzer der Webseite Marcos Tour ebenfalls ansehen. Dort haben Sie die Möglichkeit, die gleiche Tour herunterzuladen und abzulaufen.

Dieses Szenario gibt einen typischen Ablauf der Nutzung des Anwendungssystems wieder. Es beginnt mit einer Tour-Planung, setzt sich mit der Durchführung der Tour fort und endet mit einer Rekapitulation.

4.2 Anforderungen

Dieser Abschnitt beschäftigt sich mit den Anforderungen an das zu entwickelnde Anwendungssystem. Dabei wird zwischen fachlichen und technischen Anforderungen unterschieden. Die Umsetzung der Anforderungen wird im später folgenden Abschnitt 6.1 diskutiert.

4.2.1 Fachliche Anforderungen

Die fachlichen Anforderungen legen die Funktionen fest, die das Anwendungssystem bereitstellen soll. Sie lassen sich im Wesentlichen aus dem obigen Szenario ableiten. Im Mittelpunkt des Szenarios steht ein Tourist, der einen Stadtrundgang macht. Die Software unterstützt ihn bei diesem Vorhaben. Dabei führt der Tourist eine Reihe von Aktionen durch. Aus diesen lassen sich zunächst drei Hauptanwendungsfälle identifizieren. Diese sind im UML-Anwendungsfalldiagramm (vgl. [Oes06]) in Abbildung 4.1 zu sehen.

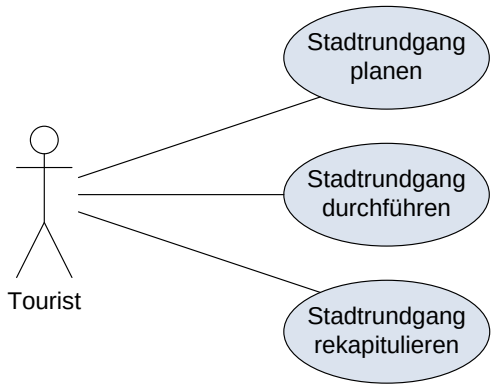


Abbildung 4.1: Die Hauptanwendungsfälle der Anwendung

Die drei Hauptanwendungsfälle „Stadtrundgang planen“, „Stadtrundgang durchführen“ und „Stadtrundgang rekapitulieren“ beinhalten weitere Unteranwendungsfälle. Diese gehen weiter ins Detail und beschreiben gleichzeitig die Anforderungen an die Software. Es folgt eine Erläuterung der Anwendungsfälle und Anforderungen.

Planung eines Stadtrundgangs (vgl. Abb. 4.2)

Akteur: Tourist

Beschreibung: Der Tourist plant auf einer Webseite einen Stadtrundgang. Hierzu wählt er zunächst den gewünschten Ort des Rundgangs aus. Er erhält eine Karten-Übersicht über alle OvIs, die zu dem Ort gehören. Er kann nach OvIs suchen und sich über sie informieren. Die OvIs, die er während des Stadtrundgangs besuchen möchte, fügt er seiner persönlichen Route hinzu.

Auslöser: Tourist besucht die Webseite zur Planung des Stadtrundgangs.

Ergebnisse: Der Stadtrundgang ist geplant.

Vorbedingungen: Der gewünschte Ort und die zugehörigen OvIs müssen im System existieren.

Nachbedingungen: Der Tourist kann den Stadtrundgang beginnen.

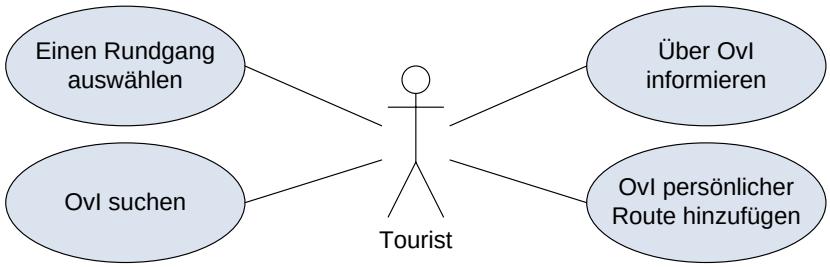


Abbildung 4.2: Unteranwendungsfälle: *Stadtrundgang planen*

Durchführung eines Stadtrundgangs (vgl. Abb. 4.3)

Akteur: Tourist

Beschreibung: Der Tourist nutzt die Software für den Stadtrundgang, um sich zu den OvIs navigieren zu lassen. Er sieht seine eigene Position und die Position der OvIs auf einer Karte. Bei Erreichen eines OvIs werden ihm automatisch die zugehörigen Informationen angezeigt. Dies können Texte und Bilder sein. In oder bei einem OvI kann es weitere Objekte geben, zu denen ebenfalls Informationen abgerufen werden können. Ein Beispiel wäre ein Museum als OvI und die Exponate im Museum als zusätzliche Objekte. Der Tourist kann zu jedem Zeitpunkt Annotationen machen. Das heißt, er kann mit dem PDA Fotos machen oder Textannotationen schreiben. Diese werden, verknüpft mit der aktuellen Position des Touristen, abgespeichert. Auch die Wegstrecke, die der Tourist läuft, wird gespeichert.

Auslöser: Der Tourist startet das Programm für den Stadtrundgang auf seinem PDA.

Ergebnisse: Der Stadtrundgang ist durchgeführt.

Vorbedingungen: Die Software für den Stadtrundgang muss auf dem PDA installiert sein. Der Stadtrundgang muss geplant sein. Er muss auf den PDA hochgeladen worden sein.

Nachbedingungen: Die Nachbereitung des Stadtrundgangs kann beginnen.

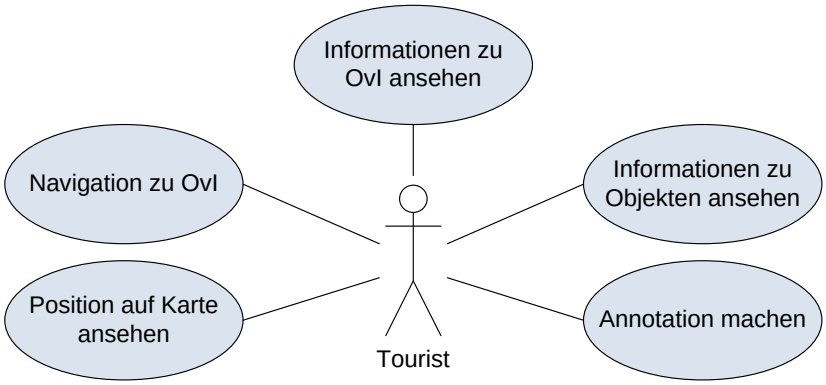


Abbildung 4.3: Unteranwendungsfälle: *Stadtrundgang durchführen*

Rekapitulation eines Stadtrundgangs (vgl. Abb. 4.4)

Akteur: Tourist

Beschreibung: Zur Rekapitulation des Stadtrundgangs steht dem Touristen auf einer Webseite eine Kartenansicht zur Verfügung. Dort ist die zurückgelegte Strecke eingezeichnet. Auch die besuchten OvIs und Objekte sind auf der Karte zu sehen. Der Tourist kann die entsprechenden Informationen abrufen. Die Annotationen, die der Tourist während des Stadtrundgangs gemacht hat, sind ebenfalls verfügbar. Der Tourist kann sie betrachten und bearbeiten. Er kann alle Daten zur Nutzung durch andere Touristen freigeben.

Auslöser: Der Tourist besucht die Webseite zur Rekapitulation des Stadtrundgangs.

Ergebnisse: Die Rekapitulation ist abgeschlossen.

Vorbedingungen: Ein Stadtrundgang wurde durchgeführt und vom PDA heruntergeladen.

Nachbedingungen: Andere Touristen können den gleichen Stadtrundgang planen und durchführen, wenn der Tourist das erlaubt.

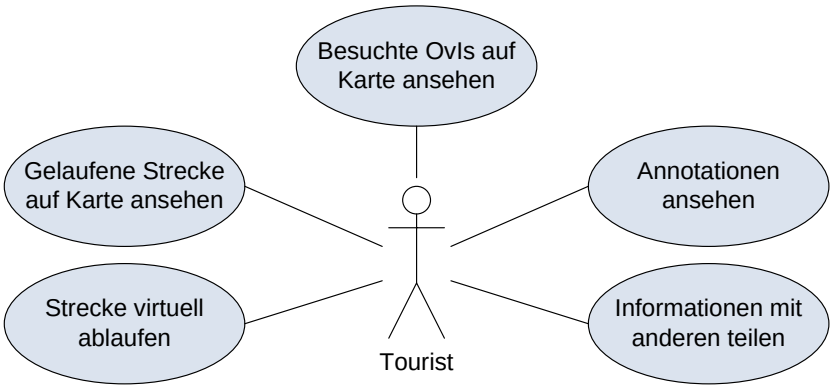


Abbildung 4.4: Unteranwendungsfälle: *Stadtrundgang rekapitulieren*

Verwaltung (vgl. Abb. 4.5)

Akteur: Nutzer mit erweiterten Benutzerrechten

Beschreibung: Zusätzlich zu den bisherigen Anwendungsfällen sind eine Reihe von Verwaltungsaufgaben durchzuführen. So muss es möglich sein, neue Karten, Touren, OvIs und Objekte anzulegen bzw. diese zu bearbeiten. Dies kann durch einen Benutzer mit erweiterten Benutzerrechten ausgeführt werden.

Auslöser: Der Nutzer besucht die Webseite zur Verwaltung.

Ergebnisse: Die Verwaltung ist abgeschlossen.

Vorbedingungen: - keine -

Nachbedingungen: Die erstellten Karten, Touren, OvIs und Objekte können von Touristen zur Planung von Stadtrundgängen genutzt werden.

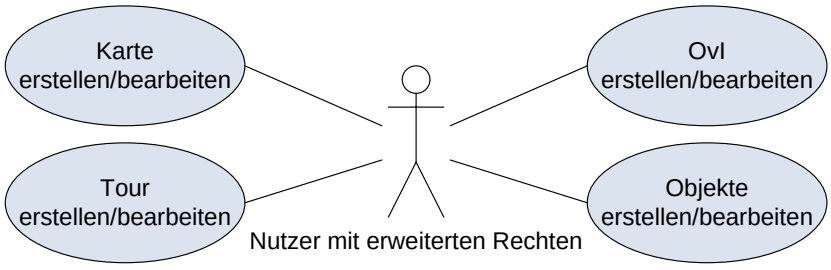


Abbildung 4.5: Anwendungsfälle für die Verwaltung

4.2.2 Technische Anforderungen

Die technischen Anforderungen beschreiben die nicht-fachlichen Rahmenbedingungen, die die Anwendung erfüllen muss. Sie sind teilweise Voraussetzung für die Realisierbarkeit der fachlichen Anforderungen. Um beispielsweise die fachliche Anforderung „*Navigation*“ realisieren zu können, müssen technische Voraussetzungen, wie z. B. eine Positionierung, gewährleistet werden. Deshalb lassen sich die technischen Anforderungen teilweise aus dem Ausgangsszenario und den fachlichen Anforderungen herleiten. Sie sind aber auch Folge der in Kapitel 2 diskutierten Anforderungen an ortsbezogene Dienste und ubiquitäre Annotationen.

Positionierung
Während des Stadtrundgangs soll der Tourist zu den OvIs navigieren können. Voraussetzung für eine Navigation ist eine Positionierung (vgl. [Küp05]). Die Anwendung muss deshalb eine Komponente zur Positionierung enthalten. Dabei ist zu berücksichtigen, dass die Positionierung, laut Szenario, in einer Outdoor-Umgebung möglich sein muss. Die Genauigkeit muss mindestens so gut sein, dass das Auffinden von einzelnen Sehenswürdigkeiten möglich ist. Des Weiteren müssen auch Objekte gefunden werden können, die sich innerhalb von geschlossenen Räumen befinden. Deshalb muss auch eine Möglichkeit gefunden werden, solche Objekte zu identifizieren.

Kartenansicht
Für die Planung und Rekapitulation des Stadtrundgangs wird eine Kartenansicht gefordert. Außerdem soll dem Touristen auf einem mobilen Gerät eine Karte zur besseren Orientierung zur Verfügung stehen. Deshalb ist zu untersuchen, welche Technologien sich für die Realisierung der jeweiligen Darstellungen eignen.

Annotationen
Laut Kapitel 2.3.3 gibt es verschiedene Möglichkeiten, Annotationen zu identifizieren, zu präsentieren und zu erstellen bzw. zu bearbeiten. Es ist zu überlegen, welche der theoretischen Möglichkeiten in dieser Anwendung am ehesten einsetzbar sind.

Benutzbarkeit
Bei der Entwicklung von mobilen und ubiquitären Anwendungen müssen besondere Rahmenbedingungen berücksichtigt werden (vgl. Kapitel 2.2.1). So kann beispielsweise nicht zu jeder Zeit eine Online-Verbindung vorausgesetzt werden. Auch die Bandbreitenkapazitäten bei der Datenübertragung sind häufig eingeschränkt und nicht konstant. Es gibt Einschränkungen bezüglich der Ein- und Ausgabe-Geräte, wie z. B. Tastatur und Display. Für den mobilen Teil der Anwendung sind diese Einschränkungen zu berücksichtigen.

Persistenz
Die bei der Verwaltung erstellten Daten müssen persistent gespeichert werden. Gleiches gilt für die persönlichen Routen der Touristen. Es ist eine geeignete Form der persistenten Speicherung und ein geeignetes Datenformat zu finden. Während des Stadtrundgangs müssen Daten auf dem mobilen Gerät festgehalten werden. Die zurückgelegte Wegstrecke und neue Annotationen müssen ebenfalls gespeichert werden. Außerdem kann nicht ausgeschlossen werden, dass der Tourist seinen Stadtrundgang unterbricht und zu einem späteren Zeitpunkt fortsetzt. Folglich muss auch auf dem mobilen Gerät eine Datenspeicherung möglich sein.

Kommunikation
Geplante Stadtrundgänge müssen auf das mobile Gerät hochgeladen, durchgeführte Touren wieder heruntergeladen werden. Deshalb ist eine geeignete Form der Datenübertragung zwischen dem stationären System und der mobilen Anwendung zu entwickeln. Des Weiteren muss ein Format für die zu übertragenen Daten konzipiert werden.

Privacy
In Kapitel 2.1.7 wurde auf die besonderen Privacy-Risiken von ortsbezogenen Diensten eingegangen. Werden keine Maßnahmen zum Schutz der persönlichen Daten ergriffen, kann es zu großen Akzeptanzproblemen seitens der Nutzer kommen. Daher gehören geeignete Konzepte zum Schutz der Privatsphäre, wie z. B. die Anonymisierung der Daten, zu den Anforderungen an die Anwendung.

4.3 Anwendungsarchitektur

Das Gesamtsystem besteht aus zwei Teilsystemen: einem stationären System für die Planung und Rekapitulation und einem mobilen System für die Durchführung des Stadtrundgangs. Diese Unterteilung ergibt sich aus dem Ausgangsszenario, beschrieben in Abschnitt 4.1. Der Benutzer-Zugriff auf das stationäre System erfolgt über ein Web-Interface. Die beiden Teilsysteme spiegeln sich in der Gesamtarchitektur wieder und stellen zwei Hauptkomponenten dar. Zwischen ihnen findet eine Kommunikation zwecks Synchronisierung statt. Eine direkte Peer-to-Peer-Kommunikation zwischen mehreren mobilen Geräten wäre möglich, um während eines Stadtrundgangs Informationen auszutauschen. Solch eine Verbindung ist aber weder im Szenario noch in den Anforderungen angeführt. Deshalb findet in der Anwendungsarchitektur diese Art der Kommunikation keine Verwendung. Nachfolgend werden die beiden Hauptkomponenten konzeptionell, d. h. auf einer hohen Abstraktionsebene, vorgestellt. Detailliertere Erläuterungen zu den Komponenten folgen in Abschnitt 4.4 und Kapitel 5.

4.3.1 Stationäres System

Das stationäre System basiert auf der klassischen Drei-Schichten-Architektur (vgl. [DH07]). Sie besteht aus der *Persistenzschicht*, der *Anwendungsschicht* und der *Präsentationsschicht* und wird in Abbildung 4.6 gezeigt. Diese drei Schichten spiegeln laut [DH07] die grundsätzlichen

Aufgaben von Softwaresystemen wieder und haben sich in der Softwareentwicklung bewährt. Die Schichten haben voneinander klar abgegrenzte Aufgaben und Verantwortlichkeiten und sind lose aneinander gekoppelt. Dadurch wird die Architektur übersichtlicher und leichter nachvollziehbar.

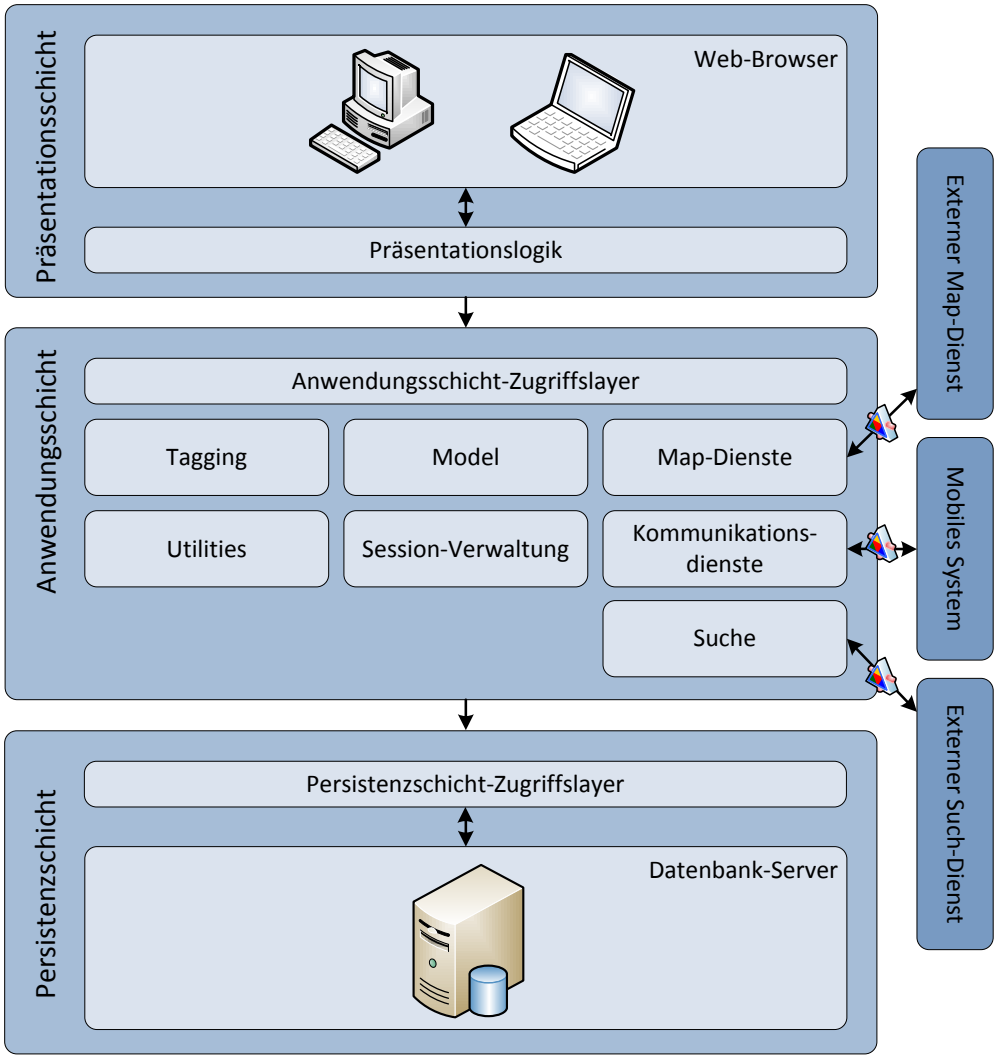


Abbildung 4.6: Stationäres System: Konzeptionelle Architektur-Sicht

Persistenzschicht

Die *Persistenzschicht* ist die unterste Schicht und sorgt für die dauerhafte Speicherung aller benötigten Daten. Dabei erfolgt die Speicherung auf einem *Datenbank-Server*. Der Zugriff aus der *Anwendungsschicht* erfolgt nicht direkt auf die Datenbank, sondern über den *Persistenzschicht-Zugriffslayer*, der als Fassade (vgl. [GHJV01]) dient.

Anwendungsschicht
Die *Anwendungsschicht* liegt eine Ebene über der *Persistenzschicht*. Hier befindet sich die eigentliche Anwendungslogik. Das *Model* beinhaltet die Model-Objekte des objektorientierten Datenmodells. Die Komponente *Session-Verwaltung* ist für die An- und Abmeldung der Benutzer und die Verwaltung der Benutzerrechte zuständig. Die *Tagging*-Komponente sorgt dafür, dass Objekte mit Identifikatoren versehen werden können, so dass sie in der Anwendung z. B. während eines Stadtrundgangs wiedererkannt werden können. Die Komponente *Map-Dienste* stellt Funktionen zur Verfügung, die das Anzeigen und Manipulieren von Karten möglich machen. Für diese Funktionen kommen externe Dienste zum Einsatz. Die *Kommunikationsdienste* stellen die Schnittelle zum mobilen System zur Verfügung. Hierüber können Daten ausgetauscht, d. h. hoch- und heruntergeladen, werden. Über die Komponente *Suche* kann z. B. nach OvIs und Objekten gesucht werden. Auch hierfür sind externe Dienste vorgesehen. Die *Utilities*-Komponente, beinhaltet Hilfsfunktionen, die von den anderen Komponenten genutzt werden. Beispielsweise stellt sie Funktionen zum Ausdrucken von Dokumenten zur Verfügung. Der *Anwendungsschicht-Zugriffslayer* fungiert als Fassade für die *Anwendungsschicht*. Sämtliche Zugriffe der *Präsentationsschicht* auf die *Anwendungsschicht* erfolgen über diese Komponente.

Präsentationsschicht
Die *Präsentationsschicht* stellt die Schnittstelle zum Benutzer dar. Da es sich um eine Web-Applikation handelt, erfolgt der Zugriff über einen Web-Browser. Der Benutzer kann sämtliche Eingaben vornehmen, auf Daten zugreifen und diese manipulieren. Die eigentliche Logik zur Darstellung der gewünschten Webseiten befindet sich in der Komponente *Präsentationslogik*.

4.3.2 Mobiles System

Auch die Architektur des mobilen Systems ist in drei Schichten untergliedert (vgl. Abbildung 4.7). Im Unterschied zum stationären System handelt es sich um eine Standalone-Anwendung, die auf einem mobilen Gerät installiert werden muss.

Persistenzschicht
Diese Schicht übernimmt die persistente Speicherung der Daten des Stadtrundgangs. Als Speichermedium werden XML-Dateien verwendet.

Anwendungsschicht
Die *Anwendungsschicht* beinhaltet eine Reihe von Komponenten mit unterschiedlichen Aufgaben. Das *Model* legt die Objekte fest, die zur Laufzeit des Programms benötigt werden. Die Komponente *Positionierung* ist für die Bestimmung der Position des Nutzers zuständig. Daneben stellt es z. B. Funktionen zur Distanzberechnung und Navigation zur Verfügung. Die *Tagging*-Komponente ermöglicht das Erkennen von Objekten anhand eines Identifikators. Es stellt das Pendant zur *Tagging*-Komponente des stationären Systems dar. Der *Tour-Manager* verwaltet einen Stadtrundgang. Er sorgt z. B. für die Speicherung der zurückgelegten Strecke und legt fest, welche Orte als nächste zu besuchen sind. Der *Annotation-Manager* ist für die Erstellung von

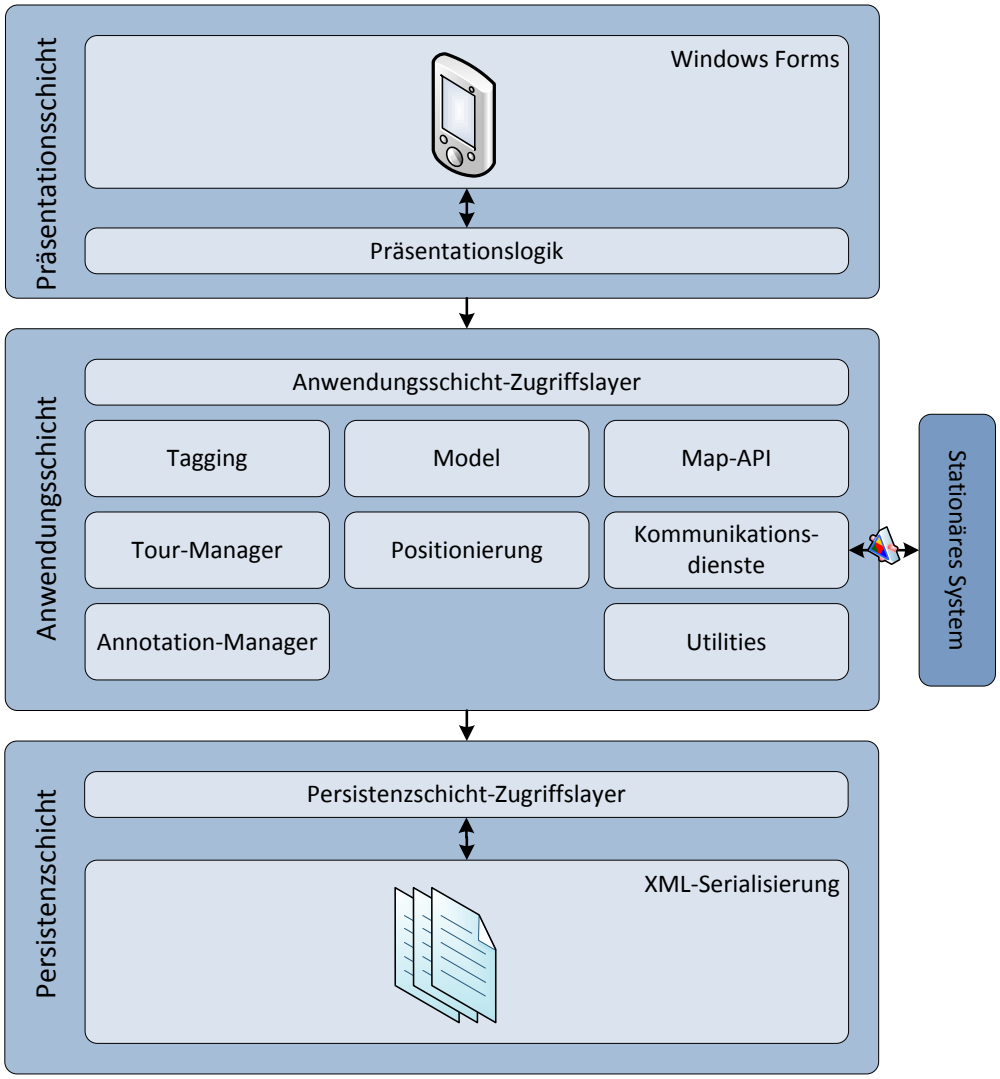


Abbildung 4.7: Mobiles System: Konzeptionelle Architektur-Sicht

Annotationen zuständig. Für die Realisierung einer Kartenansicht wird die Komponente *Map-API* verwendet. Die *Kommunikationsdienste* ermöglichen den Datenaustausch mit dem stationären System. Die Komponente *Utilities* stellt Hilfsfunktionen zur Verfügung.

Präsentationsschicht

Die Interaktion mit dem Benutzer wird über ein User-Interface, das auf Ein- bzw. Ausgabeformularen basiert, bewerkstelligt. Die Eingaben des Nutzers bestimmen den Programmfluss.

4.4 Architektur des stationären Systems

Im vorangegangenen Abschnitt wurden die beiden Teilsysteme „Stationäres System“ und „Mobiles System“ in abstrakten Architekturübersichten vorgestellt. In diesem und dem nächsten Abschnitt folgen nun detailliertere Erläuterungen zu einzelnen Komponenten innerhalb der Schichten, die aus Architektursicht interessant sind. Weitere implementierungsspezifische Details werden in Kapitel 5 erläutert.

4.4.1 Persistenzschicht

4.4.1.1 Datenbank-Server

Es ist zu erwarten, dass in der Anwendung große Datenmengen gespeichert werden. Die Anzahl der Maps, Touren, OvIs und Objekte wird mit der Laufzeit der Anwendung stark wachsen. Außerdem sind Anfragen verschiedenster Art auf die Daten zu erwarten. Typische Anfragen sind z. B. *„Gib mir alle Touren für die Stadt Hamburg“.* oder *„Zeige alle OvIs für die Tour »Museen in Hamburg«“*. Eine Datenbank ist für genau diese Art von Anforderungen geeignet (vgl. [Sta05]). Aus diesem Grund wird für die persistente Speicherung der Daten eine relationale Datenbank verwendet. Das zugehörige Datenbankschema muss so aufgebaut sein, dass das stationäre System die in Abschnitt 4.2 diskutierten Anforderungen erfüllen kann. Abbildung 4.8 zeigt die Tabellen des entwickelten Datenbankschemas.

Die einzelnen Entitäten werden nachfolgend beschrieben.

Tour
Die *Tour*-Entität ist die zentrale Tabelle, in der vorbereitete Stadtrundgänge gespeichert werden. Neben einer ID (*id*) hat jede *Tour* einen Namen *(name)*, eine Beschreibung *(description)* und eine Karte *(map)*. Außerdem gehören zu einer *Tour* keine bis mehrere OvIs *(Poi)*.

Map
Zu jeder *Tour* gehört eine *Map*-Entität. Hierbei handelt es sich um eine Karte, die später auf das mobile Gerät geladen und beim Stadtrundgang angezeigt wird. Eine *Map* kann von mehreren *Touren* genutzt werden. Wie auch schon im Fall der *Tour*, hat jede Map eine ID *(id)*, einen Namen *(name)* und eine Beschreibung *(description)*. Beim *image*-Attribut handelt es sich um ein Binary Large Object (BLOB). Hier wird eine Karte in einem Bildformat (z. B. jpeg oder png) gespeichert. Das Attribut *meters_in_pixel* gibt an, wie viele Meter in der Realität ein Pixel im Bild enthält. Dieses und die vier folgenden Attribute werden für die Darstellung der Karte auf dem mobilen Gerät benötigt. Die Attribute *latitude_top* und *latitude_bottom* definieren die obere und untere Begrenzung der Karte in Breitengraden. Die Attribute *longitude_left* und *longitude_right* stellen die linke und rechte Grenze der Karte in Längengraden dar.

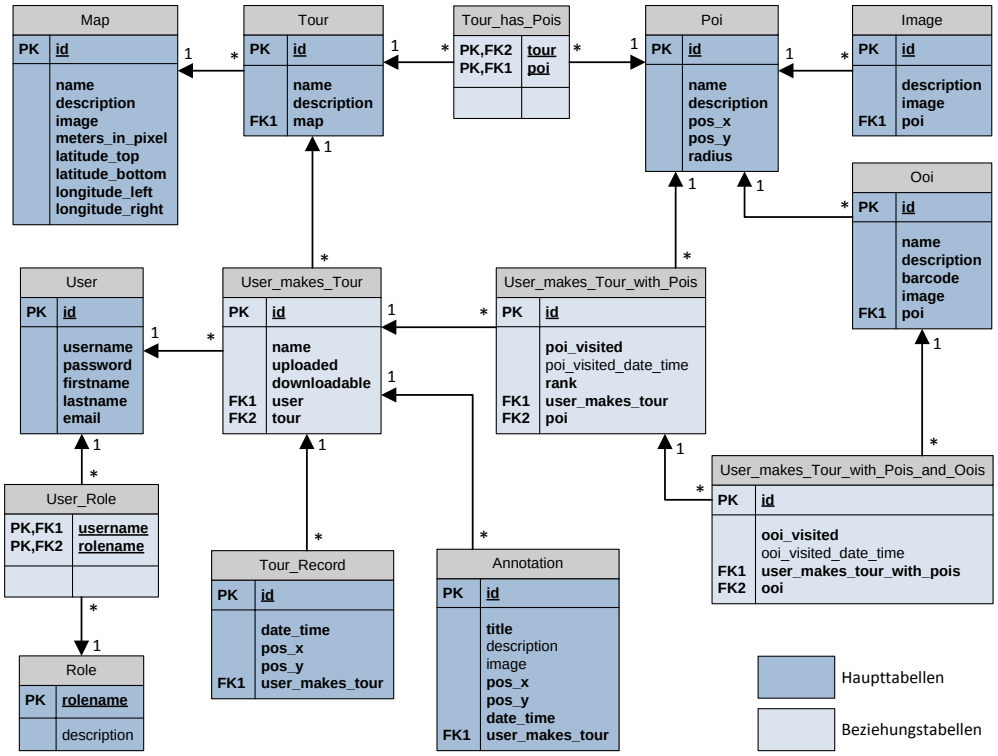


Abbildung 4.8: Tabellen und Relationen des Datenbankschemas

Poi

Ein Eintrag in der Tabelle *Poi* (Point of Interest) speichert einen OvI. OvIs sind Orte, die in einem Stadtrundgang besucht werden können. Jede *Poi* hat eine ID *(id)*, einen Namen *(name)* und eine Beschreibung *(description)*, daneben eine Position als x- und y-Koordinate (*pos_x* und *pos_y*). Das *radius*-Attribut definiert einen Umkreis um die Position. Zu jeder *Poi* gehören keine bis mehrere Bilder *(Image)* und ebenfalls keine bis mehrere Objekte *(Ooi)*. Die Beziehung zwischen den Entitäten *Tour* und *Poi* wird über die Relation *Tour_has_Poi* realisiert.

Image

In der *Image*-Entität werden Bilder zu OvIs gespeichert. Dabei hat jedes Bild eine ID *(id)* und eine Beschreibung *(description)*. Das *image*-Attribut speichert das Bild als BLOB. Eine *Image*-Entität gehört immer zu einer *Poi*.

Ooi

Den Anforderungen nach sollen sich in und bei OvIs weitere Objekte befinden können. Diese werden in der Entität *Ooi* (Object of Interest) gespeichert. Eine *Ooi*-Entität besteht u. a. aus den Attributen ID *(id)*, Name *(name)* und Beschreibung *(description)*. Das Attribut *barcode* speichert das Bild eines zweidimensionalen Barcodes. Dieser wird später genutzt, um die Objekte zu identifizieren. Das *image*-Attribut speichert ein Bild des Objekts. Jede *Ooi*-Entität gehört zu genau einer *Poi*-Entität.

User
In der Tabelle *User* werden die Benutzer des Systems gespeichert. Neben einer ID *(id)* hat jeder Benutzer einen Benutzernamen *username*, ein Passwort *(password)*, einen Vornamen *(firstname)*, einen Nachnamen *(lastname)* und eine E-Mail-Adresse *(email)*. Mit dem Benutzernamen und dem Passwort kann sich der Benutzer am stationären System anmelden. Auch für das Herunter- und Hochladen von Stadtrundgängen am mobilen Gerät werden diese Daten verwendet.

Role
Jeder Nutzer hat eine oder mehrere Rollen *(Role)*. Über die Rollen werden die Zugriffsrechte der Benutzer festgelegt. Jede *Role*-Entität hat einen eindeutigen Rollennamen *(rolename)* und eine Beschreibung *(description)*. Typische Rollen sind z. B. *Tourist* und *Tourist mit zusätzlichen Rechten für die Verwaltung*. Diese sind bereits aus den Anforderungen in Abschnitt 4.2 bekannt. Die Tabelle *user_role* speichert die Beziehung zwischen Benutzern und Rollen.

User_makes_Tour
Wenn ein Benutzer *(User)* einen Stadtrundgang *(Tour)* plant, wird hierfür ein Datensatz in der Beziehungstabelle *User_makes_Tour* angelegt, die die Tabellen *User* und *Tour* verknüpft. Der Benutzer kann der persönlichen Tour einen eigenen Namen *(name)* geben. Bei den Attributen *uploaded* und *downloadable* handelt es sich um Boolean-Flags. Im *uploaded*-Attribut wird gespeichert, ob der Benutzer diese Tour schon auf den mobilen Gerät hochgeladen hat. Über das Attribut *downloadable* kann gesteuert werden, ob die Tour für das Herunterladen auf das mobile Gerät freigegeben werden soll oder nicht. Zu jeder *User_makes_Tour*-Relation gehören keine bis mehrere *Tour_Record*-Entitäten und keine bis mehrere *Annotation*-Entitäten.

Tour_Record
Während der Durchführung eines Stadtrundgangs wird die zurückgelegte Strecke auf dem mobilen Gerät gespeichert. Hierzu wird in bestimmten zeitlichen und räumlichen Abständen die Position des Benutzers gespeichert. Nach dem Hochladen des Stadtrundgangs auf das stationäre System werden diese Positionen in der Tabelle *Tour_Record* abgelegt. Jeder *Tour_Record*-Datensatz hat eine ID *(id)*. Das *date_time*-Attribut speichert das Datum und die Zeit der Position. Die Attribute *pos_x* und *pos_y* speichern die Koordinaten, an denen sich der Benutzer zu der Zeit *date_time* befand.

Annotation
In der Tabelle *Annotation* werden die Annotationen gespeichert, die der Benutzer während des Stadtrundgangs gemacht hat. Jede Annotation hat eine ID *(id)*, einen Titel *(title)*, eine Beschreibung *(description)* und ein Bild *(image)*. Die Attribute *description* und *image* sind optional. Das Bild stellt ein Foto dar, das der Benutzer beim Stadtrundgang gemacht hat. In den Attributen *pos_x* und *pos_y* wird die Position gespeichert, an der die Annotation gemacht wurde. Das Attribut *date_time* speichert das Datum und die Uhrzeit der Annotationserstellung.

User_makes_Tour_with_Pois
Bei der Planung eines Stadtrundgangs hat der Benutzer die Möglichkeit, aus allen verfügbaren OvIs der jeweiligen Tour die für ihn interessanten auszuwählen. Diese werden über die Relation *User_makes_Tour_with_Pois* gespeichert. Das Boolean-Attribut *poi_visited* speichert, ob der Benutzer während des Stadtrundgangs den jeweiligen OvI besucht hat. Das optionale Attribut *poi_visited_date_time* speichert den Zeitpunkt des Besuchs. Über das *rank*-Attribut kann die Reihenfolge der zu besuchenden OvIs festgelegt werden.

User_makes_Tour_with_Pois_and_Oois
Diese Relation speichert, welche Objekte *(Ooi)* ein Benutzer während des Stadtrundgangs besucht hat *(ooi_visited)*. Außerdem wird der Zeitpunkt des Besuchs *(ooi_visited_date_time)* festgehalten.

4.4.1.2 Persistenzschicht-Zugriffslayer

Der *Persistenzschicht-Zugriffslayer* wurde eingeführt, um die *Persistenzschicht* von der *Anwendungsschicht* sauber zu trennen. Sämtliche Zugriffe der *Anwendungsschicht* erfolgen nicht direkt auf die Datenbank, sondern über diesen zusätzlichen Layer. Hiermit wird der Idee der lose gekoppelten Schichten und Komponenten (vgl. [Som07]) nachgekommen. Die *Anwendungsschicht* kann unabhängig vom *Datenbank-Server* entwickelt werden. Mögliche Änderungen der Datenbank wirken sich nicht unmittelbar auf die *Anwendungsschicht* aus. Die Wiederverwendbarkeit, Erweiterbarkeit und Wartbarkeit der Anwendung bzw. einzelner Komponenten wird vereinfacht (vgl. [DH07]).

In der *Anwendungsschicht* wird ein objektorientiertes Datenmodell verwendet. Im *Persistenzschicht-Zugriffslayer* wird deshalb auch gleichzeitig ein Objekt-Relationales Mapping (vgl. [Sta05]) umgesetzt. Dabei werden die Tabellen der Datenbank in ein objektorientiertes Modell übertragen. Die notwendigen Objekte sind Teil der Komponente *Model*; sie werden im nächsten Abschnitt behandelt.

4.4.2 Anwendungsschicht

4.4.2.1 Model

Die *Model*-Komponente stellt die Abbildung des Datenbankschemas als Objekte eines objektorientierten Datenmodells dar. Die Daten aus der Datenbank werden zur Verarbeitung in diese Objekte geladen. Nach der Verarbeitung werden sie zur persistenten Speicherung wieder in die Datenbank geschrieben. Im UML-Klassendiagramm (vgl. [Oes06]) in Abbildung 4.9 ist zu sehen, dass das Objektmodell dem Datenmodell aus Abschnitt 4.4.1.1 stark ähnelt. Lediglich die Relationen ohne zusätzliche Attribute (*User_Role*, *Tour_has_Pois*) entfallen im Objektmodell.

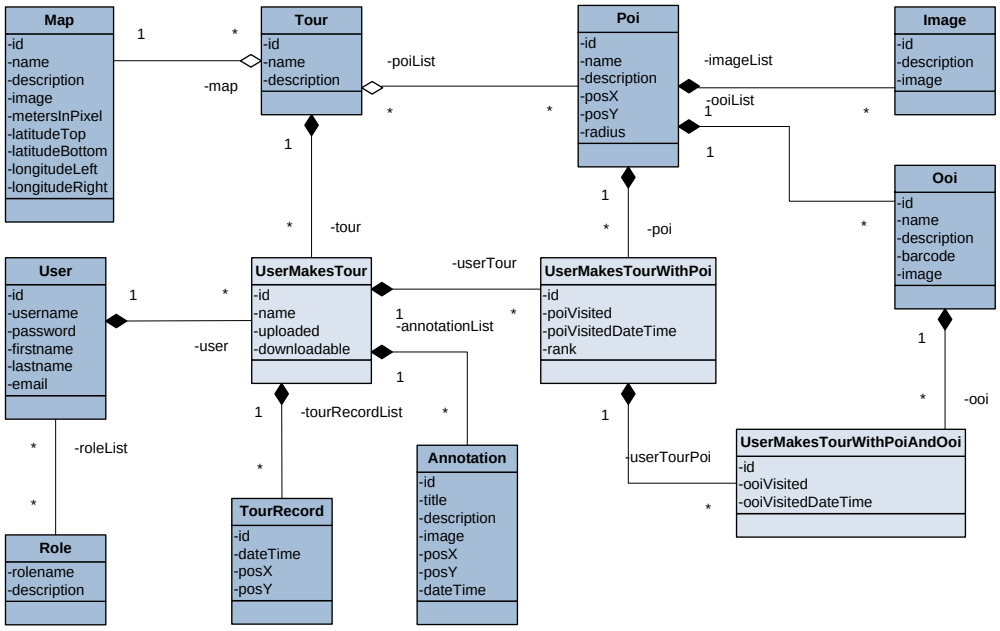


Abbildung 4.9: UML-Klassendiagramm der *Model*-Komponente des stationären Systems

4.4.2.2 Kommunikationsdienste

Diese Komponente definiert und implementiert die Schnittstelle für die Kommunikation zwischen dem stationären und dem mobilen System. Realisiert wird die Kommunikation mittels Webservices (vgl. [Bur07]). Dabei fungiert das stationäre System als Webservice-Anbieter. Das mobile System hat die Rolle des Webservice-Konsumenten. Listing 4.1 zeigt einen Codeausschnitt der Webservice-Schnittstelle.

```
1  interface ITourguideWebService {
2      public String login(String username, String password) throws Exception;
3
4      public Tour[] downloadTours(String username, String password) throws
           Exception;
5
6      public boolean uploadTour(String username, String password, Tour tour)
           throws Exception;
7  }
```

Listing 4.1: Schnittstelle für die Kommunikation mit dem mobilen System

Das Interface, das die Schnittstelle definiert, heißt `ITourguideWebService`. Über die Methode `login` ist eine Anmeldung beim Webservice möglich. Hierfür müssen der Benutzername und ein Passwort als Argumente übergeben werden. Die Methode `downloadTours` ermöglicht das Herunterladen von zuvor erstellten Stadtrundgängen vom stationären auf das mobile System. Auch hier werden Benutzername und Passwort als Argumente übergeben. Die Methode liefert ein Array des Typs `Tour` zurück, das bereits aus der *Model*-Komponente bekannt ist. Um durchgeführte Stadtrundgänge vom mobilen Gerät wieder zurück auf das stationäre System zu laden, wird die Methode `uploadTour` verwendet. Neben einem Benutzernamen und

einem Passwort wird als Argument die hochzuladende Tour übergeben. Alle Methoden werfen eine Exception aus, falls es bei der Übertragung der Daten ein Problem gibt.

4.4.3 Präsentationsschicht

Beim stationären System handelt es sich um eine Web-Applikation. Hieraus ergibt sich, dass die Bedienung des Anwendungssystems über Webseiten mittels eines Web-Browsers erfolgt.

Bei der Realisierung der Präsentationsschicht werden Gestaltungsgrundsätze bzgl. der Ergonomie berücksichtigt. Diese sind in der Norm *DIN EN ISO 9241-110:2006 (Ergonomics of human-system interaction - Part 110: Dialogue principles)*[1] festgehalten und fordern folgende Maßnahmen:

- **Aufgabenangemessenheit:** Es sollen nur Informationen angezeigt werden, die für die Erledigung der aktuellen Aufgabe notwendig sind. Zusätzliche Informationen können auf Wunsch zur Verfügung gestellt werden.
- **Selbstbeschreibungsfähigkeit:** Die Dialoge sollen möglichst intuitiv und ohne zusätzliche Beschreibung verstanden und benutzt werden können.
- **Steuerbarkeit:** Der Benutzer soll möglichst frei entscheiden können, in welcher Reihenfolge er Aufgaben erledigt.
- **Erwartungskonformität:** Das System soll sich so verhalten, wie es ein typischer Nutzer aus ähnlichen Anwendungen gewohnt ist.
- **Fehlertoleranz:** Das System soll potentielle Eingabefehler erkennen und den Nutzer bei der Korrektur unterstützen.
- **Individualisierbarkeit:** Der Nutzer soll die Benutzeroberfläche nach seinen eigenen Bedürfnissen und Fähigkeiten anpassen können.
- **Lernförderlichkeit:** Das Anwendungssystem sollte den Nutzer in den ersten Lernphasen unterstützen und in die Bedienung einführen.

Bei der Entwicklung von Benutzerschnittstellen ist die Berücksichtigung der späteren Nutzergruppe von hoher Bedeutung. In diesem Fall wird davon ausgegangen, dass der typische Nutzer des Anwendungssystems den Umgang mit Webseiten bereits gewohnt ist. Dadurch kann ein Grundverständnis der Nutzer für den Umgang mit Webseiten vorausgesetzt werden.

4.5 Architektur des mobilen Systems

In diesem Abschnitt werden Architekturdetails des mobilen Systems diskutiert. Ähnlich wie in Abschnitt 4.4 werden gezielt Komponenten näher beleuchtet, die aus Architekturgesichtspunkten besonders von Interesse sind.

[1] DIN EN ISO 9241-110:2006: http://www.iso.org

4.5.1 Persistenzschicht

Für die Speicherung von Daten auf dem mobilen System gelten andere Anforderungen als beim stationären System. Die Daten auf dem mobilen Gerät werden in der Regel nicht langfristig gespeichert. Es handelt sich eher um eine Zwischenspeicherung der Daten. Damit wird bezweckt, dass Daten, die sich während eines Stadtrundgangs im Arbeitsspeicher befinden, nicht verloren gehen, wenn das mobile Gerät zwischenzeitlich ausgeschaltet wird. Dies kann z. B. passieren, wenn die Akkukapazität des mobilen Geräts aufgebraucht ist, oder der Benutzer manuell den Stadtrundgang unterbricht. Die Speicherung dient also in erster Linie der Gewährleistung der Robustheit der Anwendung. Des Weiteren werden relativ wenige Daten zu speichern sein, und Datenabfragen sind nicht zu erwarten. Aus diesen Gründen würde der Einsatz einer Datenbank auf dem mobilen Gerät einen zu großen und unnötigen Overhead bedeuten. Die begrenzten Speicher- und Rechenresourcen auf mobilen Geräten sind ein weiterer Grund, keine Datenbank einzusetzen.

Aufgrund dieser Überlegungen ist ein leichtgewichtigerer Ansatz gefragt. Eine relativ einfache Möglichkeit stellt die Speicherung in Form von XML-Dateien dar ([Von07]). Da nicht zu erwarten ist, dass die Dateien übermäßig groß werden, hält sich der Speicher- und Rechenaufwand in Grenzen. Zudem bieten moderne Programmiersprachen bereits Schnittstellen, um XML-Dateien aus Objekten zu generieren. Der umgekehrte Weg, aus XML-Dateien wieder Objekte zu erzeugen, stellt meist ebenfalls kein Problem dar. Diese Überlegungen führten zu der Entscheidung, zur Speicherung der Daten XML-Dateien zu verwenden. Listing 4.2 zeigt beispielhaft einen Ausschnitt aus einer solchen XML-Datei. Die zugehörige XML-Schema-Datei ist im Anhang (A.1) zu finden.

```
<TGModel>
  <Tours>
    <Tour>
      <Id>4028818a168163a50116824b21be0003</Id>
      <Name>Sample Tour</Name>
      <Description>A Sample Tour</Description>
      <Map>
        <Id>4028818a168163a5011682481105002</Id>
        <Name>Sample Map</Name>
        <Description>A Sample Map</Description>
        <ImageFilename>samplemap.jpg</ImageFilename>
        <MetersPerPixel>2.8571428571</MetersPerPixel>
        <LatitudeTop>53.5768115784221</LatitudeTop>
        <LatitudeBottom>53.5513231456644</LatitudeBottom>
        <LongitudeLeft>9.91657733917237</LongitudeLeft>
        <LongitudeRight>9.98095035552979</LongitudeRight>
      </Map>
      <PoiList>
        <Poi>
          <Id>4028818a168163a50116824fa3fb0004</Id>
          <Name>Sample POI 1</Name>
          <Description>A Sample POI</Description>
          <PosX>53.564406</PosX>
          <PosY>9.949334</PosY>
          <Visited>true</Visited>
          <VisitedDateTime>2007-12-01T15:47:10+01:00</VisitedDateTime>
```

```
27            <Radius>15</Radius>
28            <ImageList>
29              <Image>
30                <Id>4028818a16830cda011683163b190001</Id>
31                <FileName>sampleimage.jpg</FileName>
32                <Description>A Sample Image</Description>
33              </Image>
34            </ImageList>
35            <OoiList />
36          </Poi>
37          <!-- Here more Pois -->
38        </PoiList>
39        <AnnotationList>
40          <Annotation>
41            <Title>Sample Annotation</Title>
42            <ImageFileName>sampleannotation.jpg</ImageFileName>
43            <PosX>53.564425</PosX>
44            <PosY>9.949665</PosY>
45            <DateTime>2007-12-01T15:48:50+01:00</DateTime>
46          </Annotation>
47          <!-- Here more Annotations -->
48        </AnnotationList>
49        <TourRecorder>
50          <TourRecordList>
51            <TourRecord>
52              <DateTime>2007-12-01T15:47:13+01:00</DateTime>
53              <PosX>53.56434666666668</PosX>
54              <PosY>9.9492933333333333</PosY>
55            </TourRecord>
56            <!-- Here more Tour Records -->
57          </TourRecordList>
58        </TourRecorder>
59      </Tour>
60      <!-- Here more Tours -->
61    </Tours>
62    <CurrentTour />
63    <NavigationOn>true</NavigationOn>
64    <NotifyAlreadyVisitedPois>false</NotifyAlreadyVisitedPois>
65    <User />
66  </TGModel>
```

Listing 4.2: XML-Datei zur Speicherung von Daten auf dem mobilen System

Die XML-Datei sollte im Wesentlichen selbsterklärend sein. Die Model-Elemente, die bereits aus Abschnitt 4.4.2.1 bekannt sind, lassen sich hier wiederfinden. Ein *TGModel* (Tour Guide Model) besteht aus mehreren Touren *(Tours)*. Eine *Tour* wiederum enthält mehrere OvIs *(PoiList)*, Annotationen *(AnnotationList)* und einen Tour-Recorder *(TourRecorder)* mit einer Liste von Tour-Records *(TourRecordList)*.

4.5.2 Anwendungsschicht

4.5.2.1 Model

Die Model-Komponente stellt die Objektdarstellung des in Listing 4.2 beschriebenen XML-Dokuments dar. Alle Elemente des XML-Dokuments lassen sich im Model wiederfinden (vgl. Abbildung 4.10).

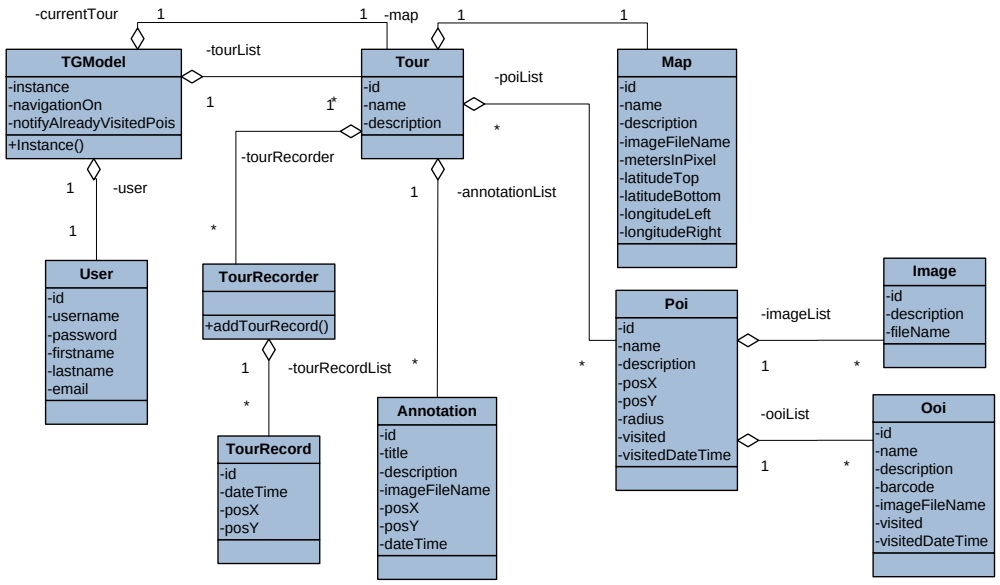


Abbildung 4.10: UML-Klassendiagramm der *Model*-Komponente des mobilen Systems

Die meisten Klassen und Attribute entsprechen den Klassen und Attributen der *Model*-Komponente des stationären Systems aus Abschnitt 4.4.2.1. Deshalb sollen hier nur die Unterschiede näher erläutert werden.

Die *TGModel*-Klasse stellt den zentralen Zugangspunkt der *Model*-Komponente dar. In der Client-Applikation macht nur eine Instanz des Models Sinn. Deshalb handelt es sich bei der *TGModel*-Klasse um einen Singleton im Sinne des Singleton-Design-Patterns (vgl. [GHJV01]). Über die statische *Instance()*-Methode wird die Singleton-Instanz aufgerufen. In der booleschen Instanzvariablen *navigationOn* wird gespeichert, ob der Nutzer eine Navigation zum nächsten Ziel wünscht, oder nicht. Über die Instanzvariable *notifyAlreadyVisitedPois* kann festgelegt bzw. abgefragt werden, ob eine Benachrichtigung stattfinden soll, wenn OvIs erreicht werden, die bereits zuvor besucht wurden. Die *TGModel*-Klasse hat neben einer Liste von Touren *(tourList)* auch eine Instanzvariable *currentTour*, die den aktuell ausgewählten Stadtrundgang speichert. Die *Tour*-Klasse wiederum hat einen *TourRecorder*, der für die Speicherung der zurückgelegten Strecke zuständig ist.

4.5.2.2 Kommunikation

Die Kommunikationskomponente enthält einen Webservice-Client, der über den in Abschnitt 4.4.2.2 definierten Webservice mit dem stationären System kommuniziert. Seine Hauptaufga-

be besteht darin, vor und nach einem Stadtrundgang Daten mit dem stationären System zu synchronisieren.

Auf eine Kommunikation während des Stadtrundgangs wurde verzichtet. Sie ist einerseits nicht zwingend notwendig, andererseits ist die Anwendung dadurch von der Kommunikation unabhängig. Mögliche Probleme bezüglich der Robustheit bei schwankender Qualität einer Datenverbindung werden dadurch ausgeschlossen.

4.5.2.3 Positionierung

Diese Komponente ist für die Bestimmung der Position des Nutzers während eines Stadtrundgangs zuständig. Ein wichtiges Kriterium hierbei ist, dass von der konkret benutzten Positionierungstechnik, wie z. B. GPS, zu abstrahieren ist ([Küp05]). Auch wenn in dieser Applikation zunächst GPS als primäre Positionierungstechnik genutzt wird, ist durch die Abstraktion die Nutzung einer anderen Technologie ohne großen Aufwand möglich. Dadurch ist die Applikation nicht auf die GPS-Positionierung beschränkt.

Zu diesem Zweck wurde das Interface *IPositioningLayer* entwickelt (vgl.. Abbildung 4.11). Es definiert alle Methoden, die die Positionierungskomponente anbietet. Hierzu gehören z. B. Methoden zum Starten und Anhalten der Positionierung (*startPositioning()*, *stopPositioning()*) und eine Methode zur Ermittlung der Position (*getPosition()*). Für jede konkrete Positionierungstechnologie muss es eine konkrete Implementierung der Schnittstelle geben.

Die Positionierungskomponente ist auch für die Navigation zuständig. Aus diesem Grund stellt das Interface *IPositioningLayer* auch Methoden wie *calculateCourse()* und *calculateDistance()* zur Verfügung.

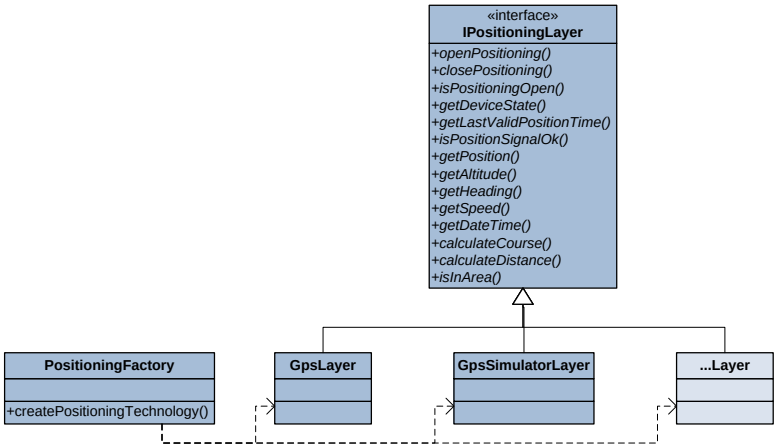


Abbildung 4.11: UML-Klassendiagramm der Komponente *Positionierung* des mobilen Systems

Als erster Schritt sind die konkreten Klassen *GpsLayer* und *GpsSimulatorLayer* vorgesehen. Dabei implementiert die Klasse *GpsLayer* die Positionierung mittels GPS. Die Klasse *GpsSimulatorLayer* dagegen simuliert einen GPS-Empfänger. Dieser wird während der Entwicklung zu Testzwecken genutzt.

Es kann nicht davon ausgegangen werden, dass zur Entwicklungszeit bereits bekannt ist, welche konkrete Technologie zur Positionierung genutzt werden soll. Deshalb erfolgt die Festlegung erst zur Laufzeit durch Konfiguration. Zur Erstellung einer konkreten Instanz wird die Klasse *PositioningFactory* verwendet. Dabei handelt es sich um die Implementierung des *Factory*-Patterns (vgl. [GHJV01]). Die Klasse *PositioningFactory* liest die Konfiguration ein und erzeugt über die Methode *createPositioningTechnology()* eine Instanz der gewünschten Positionierungstechnik.

Die imaginäre Klasse *...Layer* in Abbildung 4.11 soll andeuten, dass weitere Positionierungstechnologien an dieser Stelle eingebaut werden können.

4.5.2.4 Tagging

Während des Stadtrundgangs sollen, laut Anforderungen, innerhalb von Gebäuden Objekte erkannt, und Informationen zu diesen Objekten angezeigt werden. Im Grundlagenkapitel wurden verschiedene Möglichkeiten diskutiert, die Objektidentifizierung zu realisieren. Zwei relativ einfach und kostengünstig realisierbare Ansätze sind die Identifizierung mittels Barcodes und mittels der RFID-Technologie [MBGH07]. Beide haben ihre Vor- und Nachteile (vgl. [Han06]). Ein wesentlicher Vorteil der Barcode-Technik besteht darin, dass sie bereits heute von vielen mobilen Geräten, wie z. B. PDAs und Handys, ohne zusätzliche Hardwarekomponenten nutzbar ist. Die wichtigste Voraussetzung ist eine eingebaute Kamera, um Barcodes abfotographieren zu können. Ähnlich wie bei der Komponente *Positionierung* sieht das Architekturdesign vor, dass die konkret verwendete Tagging-Technologie individuell konfigurierbar ist. Der Entwickler kann demnach selbst entscheiden, welche der verfügbaren Technologien er verwenden möchte.

4.5.2.5 Tour-Manager

Der *Tour-Manager* ist für die Verwaltung eines Stadtrundgangs zuständig. Den zentralen Zugriffspunkt stellt die Singleton-Klasse *TourManager* da (vgl. Abbildung 4.12).

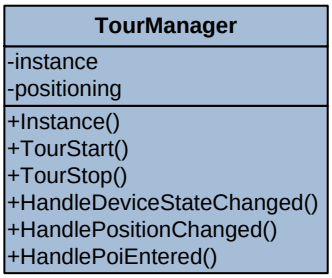


Abbildung 4.12: UML-Darstellung der *TourManager*-Klasse des mobilen Systems

Über die Methode *Instance()* wird auf die Singleton-Instanz der Klasse *TourManager* zugegriffen. Die Methode *TourStart()* wird aufgerufen, wenn der Benutzer den Stadtrundgang beginnt. An dieser Stelle wird die Positionierung eingeschaltet. Beendet oder unterbricht der Benutzer den Stadtrundgang, führt dies zum Aufruf der Methode *TourStop()*. Die Positionierung wird ausgeschaltet. Zusätzlich wird das Model *TGModel* gespeichert.

Die drei Methoden *HandleDeviceStateChanged()*, *HandlePositionChanged()* und *HandlePoiEntered()* werden automatisch bei bestimmten Ereignissen aufgerufen. Ein Aufruf kann dabei nur erfolgen, wenn die Tour gestartet ist. Bei einer Änderung des Zustands der Positionierungskomponente folgt ein Aufruf der Methode *HandleDeviceStateChanged()*. Auf diese Weise wird dem *Tour-Manager* z. B. mitgeteilt, dass die Positionierung gestartet oder gestoppt wurde. Die Methode *HandlePositionChanged()* wird gefeuert, wenn sich die Position des Nutzers ändert. So kann der *Tour-Manager* z. B. dafür sorgen, dass die Navigationshinweise aktualisiert werden. Außerdem erfolgt hier ein Aufruf des *Tour-Recorders*, damit dieser die aktuelle Position zur zurückgelegte Wegstrecke hinzufügen kann. Die Methode *HandlePoiEntered()* wird gefeuert, wenn ein OvI erreicht wurde. Der *Tour-Manager* kann dann z. B. die Anzeige der entsprechenden Informationen initiieren.

4.5.3 Präsentationsschicht

In Abschnitt 4.4.3 wurden bereits die Gestaltungsgrundsätze für Benutzeroberflächen thematisiert. Da es sich bei diesem Teilsystem um eine mobile Anwendung handelt, sind spezielle Gegebenheiten mobiler Anwendungen zu berücksichtigen [Sat01].

In der Regel verfügen mobile Geräte über kleine Displays. Dies führt dazu, dass vergleichsweise wenige Informationen dargestellt werden können. Die angezeigten Inhalte werden deshalb auf das Wesentliche beschränkt. Bei langen Texten kann es oftmals Sinn machen, diese als Sprachausgabe vorlesen zu lassen [BC03].

Mobile Geräte haben darüber hinaus die Einschränkung, dass sie über begrenzte Rechen-, Speicher- und Energiekapazitäten verfügen. Dennoch darf die Anwendung zu keinem Zeitpunkt in einen undefinierten Zustand kommen und eine für den Benutzer unerwartete Reaktion zeigen. Bei einem geringen Akkustand werden deshalb alle Informationen im Speicher des mobilen Geräts persistent festgeschrieben, so dass nach dem Wiederaufladen des Akkus der entsprechende Zustand wieder hergestellt werden kann.

4.6 Bewertung der Architektur

An dieser Stelle soll erläutert werden warum die vorgestellte Architektur die in Abschnitt 4.2 aufgestellten Anforderungen erfüllt.

Für die Umsetzung der technischen Anforderungen existieren Komponenten, die genau die jeweiligen Anforderungen realisieren sollen. Beispielsweise wird die Anforderung der Positionierung in der Komponente *Positionierung* des mobilen Systems erfüllt. Die geforderte Kartenansicht wird in der Komponente *Map-Dienste* bzw. *Map-API* des stationären bzw. mobilen Systems umgesetzt. Für die Realisierung der Annotationen wurde die Komponente *Annotation-Manager* des mobilen Systems eingeführt. Um die Persistenz der Daten des stationären und mobilen Systems zu gewährleisten existiert jeweils die Persistenzschicht in den Architekturen. Die Komponente *Kommunikationsdienste* im stationären und mobilen System ermöglicht den geforderte Datenaustausch zwischen beiden Teilsystemen. Die Anforderungen bezüglich der Benutzbarkeit wurden in den Abschnitten 4.4.3 und 4.5.3 thematisiert und müssen bei der Implementierung gesondert

berücksichtigt werden. Um z. B. mögliche Probleme im Zusammenhang mit einer dauerhaften Online-Verbindung zu vermeiden, wurde das System so konzipiert, dass eine solche dauerhafte Verbindung nicht notwendig ist. Diese Ausführungen zeigen, dass die vorgestellte Architektur die zuvor definierten technischen Anforderungen tatsächlich umsetzt.

In Abschnitt 4.2.2 wurde bereits geschildert, dass die technischen Anforderungen eine notwendige Voraussetzung für die Umsetzung der fachlichen Anforderungen sind. Aus diesem Grund wurden die technischen Anforderungen so gewählt, dass sie die Realisierung der fachlichen Anforderungen gewährleisten. Der Nachweis darüber, dass die vorgestellte Architektur tatsächlich auch die fachlichen Anforderungen erfüllt, wird mit der prototypischen Implementierung des Anwendungssystems erbracht. Die Implementierung, die das Hauptthema des nachfolgenden Kapitels 5 darstellt, basiert schließlich auf dem Architekturdesign.

5 Systemrealisierung

Das folgende Kapitel behandelt die Entwicklung eines Prototyps des Anwendungssystems. In Abschnitt 5.1 wird zunächst kurz die Vorgehensmethodik der Entwicklung beschrieben. Anschließend folgt in Abschnitt 5.2 ein Bericht über den aktuellen Entwicklungsstand der Anwendung. Hiernach werden in den Abschnitten 5.3, 5.4 und 5.5 die verwendeten Hardware- und Software-Komponenten vorgestellt. In Abschnitt 5.6 wird kurz die Durchführung von Software-Tests während der Entwicklung beschrieben. Abschnitt 5.7 befasst sich schließlich mit einer Reihe von Implementierungsdetails.

5.1 Vorgehen

Die Entwicklung des Anwendungssystems erfolgte in mehreren Phasen, die aufeinander aufbauen und sich gegenseitig beeinflussen (vgl. Abbildung 5.1).

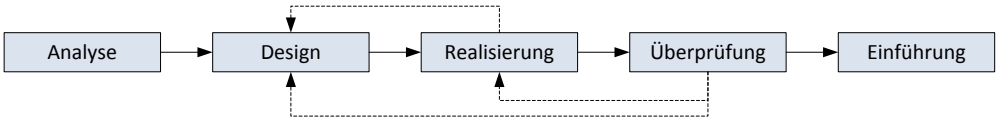


Abbildung 5.1: Vorgehensmethodik bei der Entwicklung

Der Entwicklungsprozess begann mit der Untersuchung und Erhebung der Anforderungen *(Analyse)*. Darauf aufbauend wurde die Architektur des Systems erstellt *(Design)* und anschließend die Anwendung implementiert *(Realisierung)*. Nach einer Überprüfung der zuvor aufgestellten Anforderungen *(Überprüfung)* wurde das System eingeführt *(Einführung)*.

Die einzelnen Entwicklungsphasen wurden jedoch nicht ausschließlich sequenziell bearbeitet, wie es das Wasserfallmodell (vgl. [FK04]) vorsieht. Vielmehr hatten einige der Phasen wiederum Auswirkungen auf die vorangegangenen Phasen. Probleme, die z. B. erst bei der Realisierung festgestellt wurden, hatten Änderungen auf das Systemdesign zur Folge. Wurden bei der Überprüfung Mängel bei der Erfüllung der Anforderungen festgestellt, mussten entsprechende Änderungen des Designs und der Realisierung durchgeführt werden. Diese Rückkopplungen werden in Abbildung 5.1 als gestrichelte Pfeile dargestellt. Das Vorgehen kann insgesamt als eine leicht abgewandelte Form des evolutionären Prototypings (vgl. [FK04]) bezeichnet werden.

5.2 Stand der Implementierung

In der vorliegenden Arbeit sollte nicht nur der Entwurf, sondern auch die prototypische Implementierung des Systems behandelt werden. Hierbei war das Ziel, möglichst viele der Anforderungen bereits im ersten Prototypen zu realisieren. Die Entwicklungsarbeiten nahmen ca. fünf Wochen in Anspruch. Entstanden ist eine Software für Stadtrundgänge mit dem Namen *„HAW Tour Guide“*, die alle drei Phasen Planung, Durchführung und Rekapitulation umfasst. Auch die geforderten Funktionen für die Verwaltung wurden realisiert. Der Programmcode befindet sich auf der beigefügten CD im Anhang (A.2). Anhand von Screenshots der drei Programm-Phasen und der Verwaltung wird der aktuelle Stand der Entwicklung beschrieben.

5.2.1 Planung eines Stadtrundgangs

Zur Tour-Planung gelangt man mit einem Klick auf die Web-Links *Neue Tour planen* oder *Geplante Touren* (Abbildung 5.2).

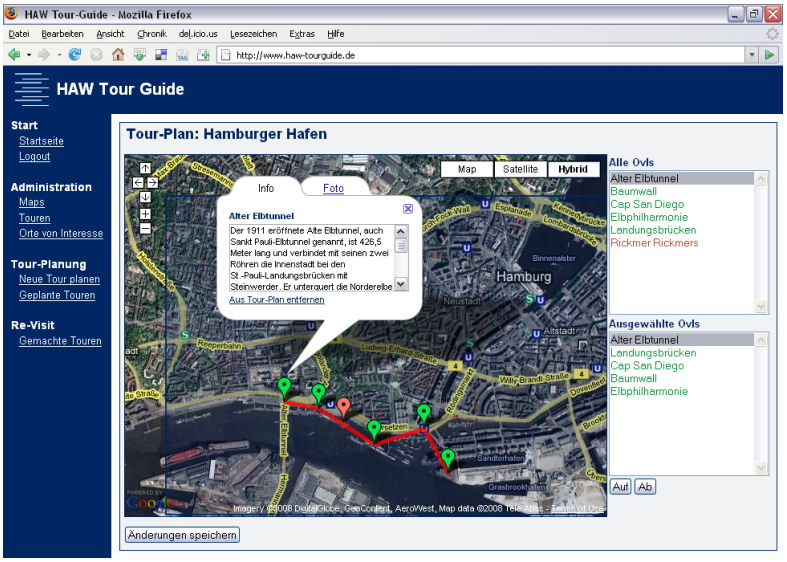


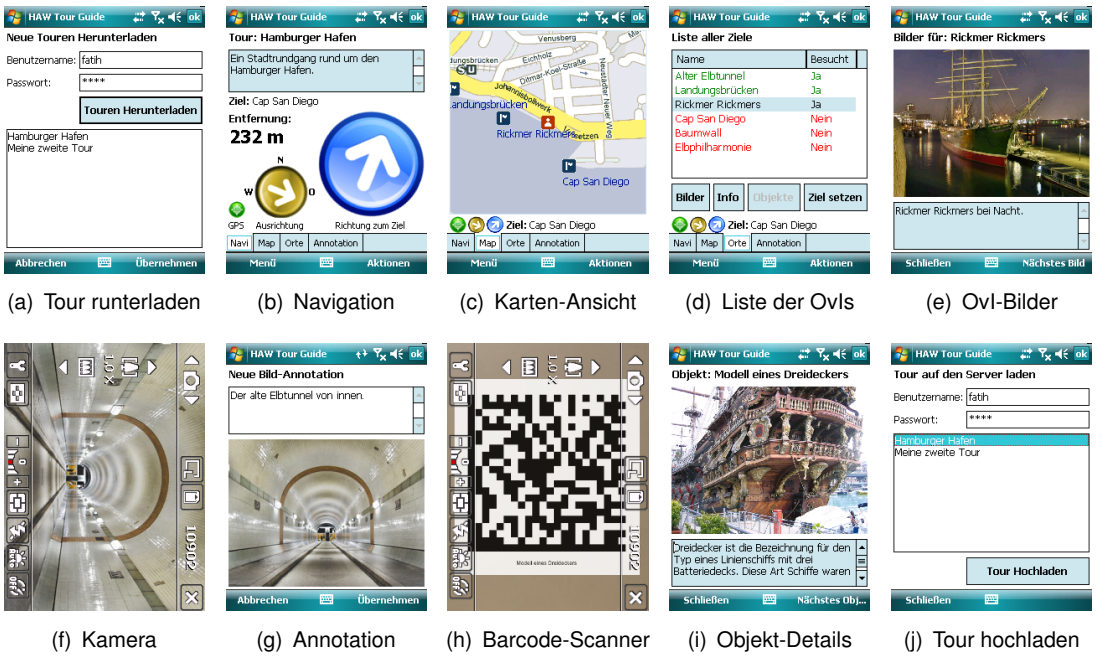
Abbildung 5.2: Screenshot: Planung eines Stadtrundgangs

Auf der Webseite sieht man eine Karte mit eingezeichneten Markern, die die OvIs repräsentieren. Rechts befinden sich zwei Listen. Die obere Liste zeigt alle OvIs, die zur Tour *Hamburger Hafen* gehören. Die untere Liste zeigt alle OvIs, die der Tourist als Ziele ausgewählt hat.

Klickt man auf einen der Marker oder auf einen der Einträge in den Listen, erhält man Informationen zu dem jeweiligen OvI. Man hat außerdem die Möglichkeit, den OvI zur persönlichen Tour hinzuzufügen (grüne Marker) bzw. von dort wieder zu entfernen (rote Marker). Weiterhin hat man die Möglichkeit, mit den Buttons *Auf* und *Ab* die Reihenfolge der, zu besuchenden, OvIs zu ändern.

5.2.2 Durchführung eines Stadtrundgangs

In Abbildung 5.3 sind Screenshot zur Durchführung des Stadtrundgangs zu sehen.



(a) Tour runterladen (b) Navigation (c) Karten-Ansicht (d) Liste der OvIs (e) OvI-Bilder

(f) Kamera (g) Annotation (h) Barcode-Scanner (i) Objekt-Details (j) Tour hochladen

Abbildung 5.3: Screenshots: Durchführung eines Stadtrundgangs

Bevor der Stadtrundgang beginnen kann, lädt der Tourist die vorbereitete Tour auf seinen PDA (Abbildung 5.3(a)). Nachdem er die Tour gestartet, hat findet eine Navigation zu den OvIs statt (Abbildung 5.3(b)). In der aktuellen Implementierung werden dem Benutzer lediglich die Richtung und die Entfernung zum Ziel angezeigt. Eine Beschreibung des Weges anhand von Straßennamen oder Landmarken findet nicht statt. Dies sollte in zukünftigen Weiterentwicklungen realisiert werden. Auf einer Karte kann sich der Tourist seine eigene Position und die Positionen der OvIs ansehen (Abbildung 5.3(c)). Eine Listenübersicht über alle OvIs ist ebenfalls möglich (Abbildung 5.3(d)). Beim Erreichen eines OvI werden dem Touristen relevante Informationen angezeigt (Abbildung 5.3(e)). Weiterhin kann der Tourist eigene Annotationen machen (Abbildungen 5.3(f), 5.3(g)). Befindet er sich in oder bei einem OvI, kann er Objekte identifizieren, indem er vorhandene Barcodes abfotographiert (Abbildung 5.3(h)). Anschließend werden ihm Informationen zu dem Objekt angezeigt (Abbildung 5.3(i)). Nach Beendigung des Stadtrundgangs kann die Tour wieder auf den Server hochgeladen werden (Abbildung 5.3(j)).

5.2.3 Rekapitulation eines Stadtrundgangs

Abbildung 5.4 zeigt einen Screenshot der Rekapitulation des Stadtrundgangs. Die rote Linie symbolisiert die tatsächlich zurückgelegte Strecke. Die grünen Marker stellen die besuchten OvIs dar. Bei einem Klick auf einen OvI-Marker werden Informationen zu diesem angezeigt. Die Ansicht

der besuchten Objekte muss in einer Folgeversion noch implementiert werden. Die Annotationen, die der Tourist selbst gemacht hat, werden als gelbe Marker dargestellt. Klickt man auf die gelben Marker, wird die Annotation angezeigt. Mit den Buttons am unteren Rand der Seite kann man zwischen den Markern hin- und herschalten. Die Möglichkeit der Freigabe des eigenen Stadtrundgangs für andere Nutzer ist derzeit noch nicht implementiert.

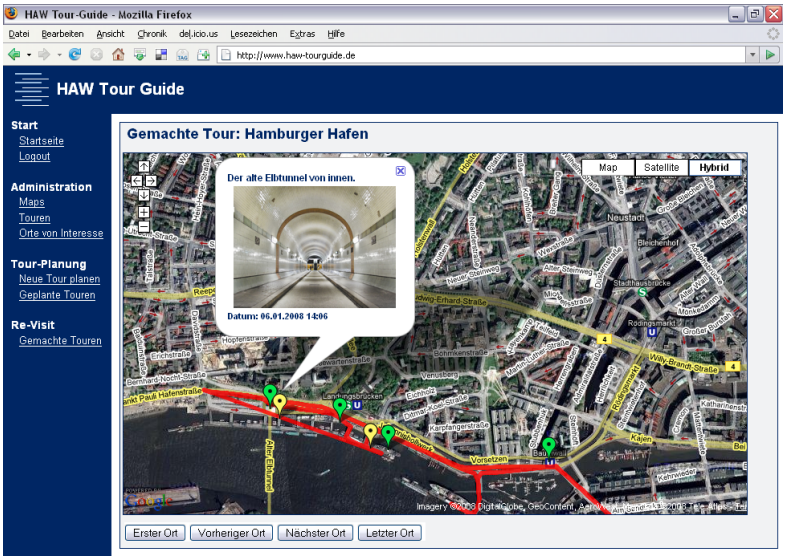


Abbildung 5.4: Screenshot: Rekapitulation eines Standrundgangs

5.2.4 Verwaltung

In diesem Programmteil können Karten, Touren, OvIs und Objekte erstellt und bearbeitet werden. Abbildung 5.5 zeigt beispielhaft einen Screenshot für die Bearbeitung einer Tour.

5.3 Entwicklungs-Hardware

In diesem Abschnitt wird die Hardware aufgezählt, die für die Entwicklung der Anwendung eingesetzt wurde.

- **Samsung X10 1400:** Auf diesem Notebook wurde die gesamte Applikation entwickelt. Der Rechner dient gleichzeitig als Test-Server. Als Betriebssystem wird *Windows XP* eingesetzt (vgl. Abbildung 5.6(a)).
- **HTC TyTN:** Dieser PDA dient als mobiles Endgerät für den Test der *Mobile Client*-Anwendung (vgl. Abbildung 5.6(b)). Beim Betriebssystem handelt es sich um *Windows Mobile 6*. Laut Microsoft sollten alle PDAs, die *Windows Mobile 5* oder *Windows Mobile 6* als Betriebssystem benutzen und über eine Kamerafunktion verfügen, einsetzbar sein. In der

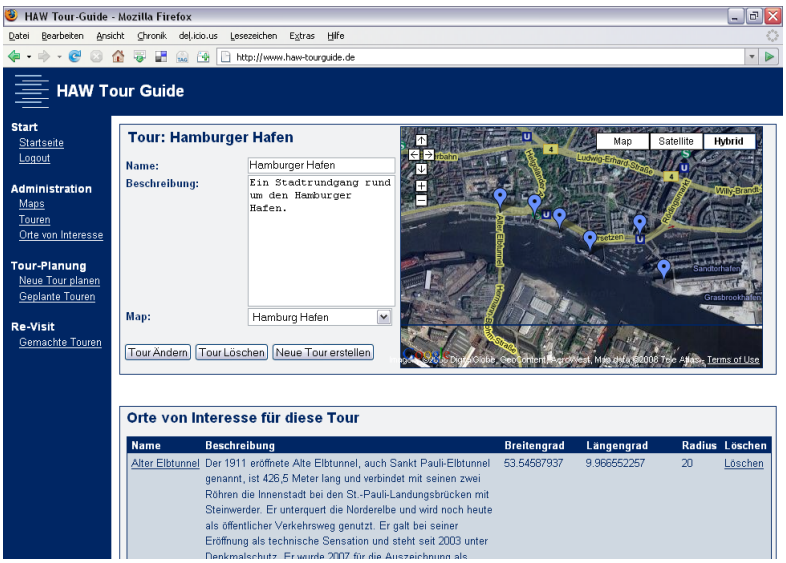


Abbildung 5.5: Screenshot: Verwaltung - Bearbeitung einer Tour

Praxis zeigt sich jedoch, dass die Microsoft-Spezifikation nicht von allen Herstellern eingehalten wird. So konnte z. B. die Kamerafunktion des PDA-Modells *E-TEN glofiish M700* nicht über die Microsoft API angesprochen werden.

- **Holux GPSlim 236 GPS-Empfänger:** Dieser GPS-Empfänger wird per Bluetooth mit dem PDA gekoppelt. Hierüber werden die GPS-Satellitensignale empfangen. Verfügt der verwendete PDA über einen integrierten GPS-Empfänger, ist kein zusätzliches GPS-Empfangsgerät notwendig (vgl. Abbildung 5.6(c)).

(a) Samsung X10 1400 (b) HTC TyTN (c) Holux GPSlim 236

Abbildung 5.6: Entwicklungs-Hardware

5.4 Entwicklungs-Software

Neben der genannten Hardware sind bei der Entwicklung verschiedene Software-Produkte zum Einsatz gekommen.

- **Microsoft Visual Studio 2005:** Für die Entwicklung des *Mobile Clients* in der Programmiersprache C# wurde Microsofts Entwicklungsumgebung *Visual Studio 2005* verwendet. Sie ermöglicht neben der Entwicklung von Software für *Windows Mobile*-Geräte auch das Testen der entwickelten Software direkt auf dem mobilen Gerät.
- **EMIC Location and Mapping Framework Controls:** Diese Komponente wird in *Visual Studio 2005* eingebunden und ermöglicht die Nutzung von speziellen GUI-Elementen für die Darstellung von Karten.
- **Eclipse SDK 3.4.0:** Die Server-Anwendung wurde mit der Programmiersprache Java entwickelt. Als Entwicklungsumgebung wurde das *Eclipse SDK* in der Version 3.4.0 eingesetzt.
- **Eclipse Web Tools Platform 3.0 M3:** Dieses *Eclipse*-Plugin vereinfacht und unterstützt die Entwicklung von Web-Applikationen, die auf Java basieren.
- **Hibernate Tools for Eclipse 3.2.0:** Bei dieser Komponente handelt es sich ebenfalls um ein *Eclipse*-Plugin. Es ermöglicht u. a., per Reverse-Engineering Java-Klassen aus einem Datenbankschema zu erzeugen. Dabei werden zusätzlich Klassen für das objektrelationale Mapping generiert.
- **MySQL 5.0.45:** Als Datenbankserver des Testsystems ist die Datenbank *MySQL* in der Version 5.0.45 zum Einsatz gekommen. Sie kann prinzipiell durch jede andere relationale Datenbank ersetzt werden, die von *Hibernate* unterstützt wird [Hib].
- **Apache Tomcat 5.5 Webserver:** Als Webserver für die *Server*-Anwendung wurde *Apache Tomcat 5.5* verwendet.

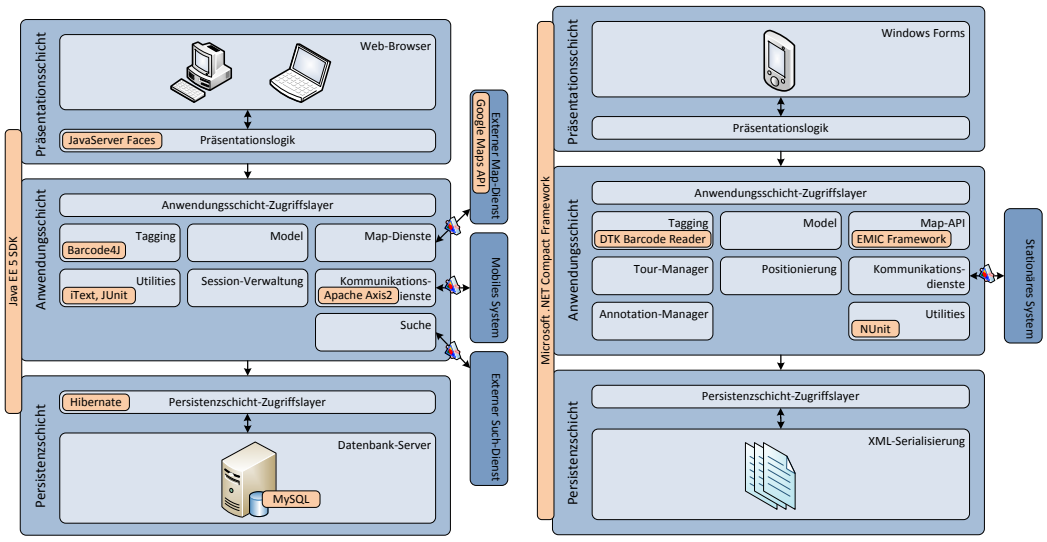
5.5 Externe Software-Komponenten

Die entwickelte Software nutzt externe Software-Komponenten von Drittanbietern. Die folgende Liste zeigt die eingesetzten Komponenten und beschreibt ihren Einsatzzweck.

- **Microsoft .NET Compact Framework 2.0:** Das *Compact Framework* von *Microsoft* stellt eine Bibliothek für die Programmierung von mobile Geräten mit *Windows Mobile*-Betriebssystem zur Verfügung. Es wurde bei der Entwicklung des *Mobile Clients* eingesetzt.
- **DTK Barcode Reader SDK 3.6.48 (Testversion):** Hierbei handelt es sich um eine Bibliothek für das *Compact Framework* zum Einlesen von ein- und zweidimensionalen Barcodes.
- **EMIC Location and Mapping Framework:** Diese Microsoft-Bibliothek ermöglicht das Anzeigen von Karten u. a. auf mobilen Geräten. Sie wird in der *Map*-Komponente des *Mobile Clients* eingesetzt.
- **NUnit 2.4.6:** Die *NUnit*-Bibliothek ermöglicht sog. Unit-Tests (vgl. [HT05]) in .NET-basierten Anwendungen.

- **Java EE 5 SDK:** Die Java-Bibliothek dient als Grundlage für die Entwicklung der *Server*-Komponente. Neben Standalone-Applikationen ermöglicht sie auch die Entwicklung von Web-Applikationen.
- **JavaServer Faces 1.2:** Die *JavaServer Faces*-Bibliothek implementiert das MVC-Paradigma (vgl. [GHJV01]) für Java-Web-Applikationen. Sie wird für die Entwicklung des User-Interfaces des *Servers* verwendet.
- **Hibernate 3.2.0:** Das *Hibernate*-Framework implementiert das objektrelationale Mapping für Java-Anwendungen. Es kommt in der *Persistentschicht* des *Servers* zum Einsatz.
- **Barcode4J 2.0:** *Barcode4J* ist eine Java-Bibliothek zum Erstellen von ein- und zweidimensionalen Barcodes.
- **iText 2.0.7:** *iText* ist eine Java-Bibliothek zum Erstellen von PDF-Dateien.
- **JUnit 4.4:** Bei der *JUnit*-Bibliothek handelt es sich um eine Unit-Test Implementierung für Java-Anwendungen.
- **Apache Axis2 1.3:** Diese Komponente ist eine sog. *SOAP-Engine*. Mit ihr können Java-Webservices erstellt und betrieben werden. Verwendet wird sie in der *Kommunikationskomponente* des *Servers*.
- **Google Maps API:** Die *Google Maps API* ist eine JavaScript-Bibliothek zum Einbinden von Karten auf Webseiten. Sie wird in der Komponente *Map-Dienste* des Servers verwendet.

Die externen Software-Komponenten lassen sich den verschiedenen Architektur-Modulen aus Abschnitt 4.3 zuordnen. Abbildung 5.7 zeigt die Architekturübersichten des stationären bzw. des mobilen Systems mit den zugehörigen Software-Komponenten in der Implementierung.



(a) Stationäres System (b) Mobiles System

Abbildung 5.7: Software-Komponenten

5.6 Tests

Zur Qualitätssicherung wurden während der Entwicklung der Anwendung sog. Modultests - auch Unit-Test genannt - (vgl. [HT05]) eingesetzt. Dabei wurden die Tests sowohl bei der *Server*-Anwendung als auch bei der *Mobile Client*-Anwendung durchgeführt.

Für die Tests der *Server*-Anwendung wird die *JUnit*-Bibliothek verwendet. Sie ermöglicht Unit-Tests für Java-Anwendungen. Das Pendant für .NET-Anwendungen stellt die *NUnit*-Bibliothek dar. Sie wird für den *Mobile Client* genutzt.

Um Unit-Tests auch für die Durchführung eines Stadtrundgangs inklusive Positionierung durchführen zu können, wurde in der *Positionierungskomponente* des *Mobile Clients* die Klasse *GpsSimulatorLayer* entwickelt (vgl. Abschnitt 4.5.2.3). Sie simuliert einen GPS-Empfänger und erleichtert somit den Testvorgang.

5.7 Details der Implementierung

In diesem Abschnitt werden Implementierungsdetails einiger *Server* und *Mobile-Client*-Komponenten beschrieben. Die Details sollen dem besseren Verständnis der Funktionsweise der Komponenten und der Anwendung insgesamt dienen.

5.7.1 Implementierung des Servers

5.7.1.1 Map-Dienste

Für die Darstellung der Karte in der *Server*-Applikation wird der externe Webdienst *Google Maps API*[1] verwendet. Mit der *Google Maps API* können die *Google Maps*-Funktionen in eigene Web-Applikationen über *JavaScript*[2] eingebettet werden. Der Codeausschnitt in Listing 5.1 zeigt ein kurzes Beispiel hierfür.

```
1  <script type="text/javascript">
2    // Google Maps-API-Funktionen einbinden
3    google.load("maps", "2");
4
5    // Initialisierung der Map
6    function initialize() {
7      // Map erstellen und an das HTML-Element mit der id='map' binden
8      var map = new GMap2(document.getElementById("map"));
9
10     // Zoom- und Navigationscontrols anzeigen
11     map.addControl(new GSmallMapControl());
12
13     // Karte-/Satellit-/Hybrid-Ansicht Controls anzeigen
14     map.addControl(new GMapTypeControl());
15
```

[1] Google Maps API: http://www.google.de/apis/maps/

[2] JavaScript: http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Reference

```
16      // Map auf Position (53.552639/10.00671) zentrieren mit Zoomlevel 14
17      map.setCenter(new google.maps.LatLng(53.552639, 10.00671), 14);
18    }
19    // Beim Laden der Seite die Methode 'initialize' aufrufen
20    google.setOnLoadCallback(initialize);
21  </script>
22
23  // Map an dieses Element binden
24  <div id="map" style="width:450px;height:300px;"></div>
```

Listing 5.1: Quellcode: Map-Dienste (Auszug)

Zunächst wird in Zeile 3 die *Google Maps API* in die Webseite eingebunden. In der Funktion *initialize()* (Zeilen 6 bis 18) wird eine Instanz einer Karte erstellt und an das HTML-Element mit der ID *map* gebunden. Das HTML-Element befindet sich in Zeile 24 und stellt die Stelle auf der HTML-Seite dar, an der die Karte angezeigt werden soll. Der Ausdruck *google.setOnLoad-Callback(initialize)* in Zeile 20 bewirkt, dass die Funktion *initialize()* beim Aufruf der Seite ausgeführt wird. Weitere Details zur *Google Maps API* findet man in der *Google Maps API*-Referenz [Goob].

5.7.1.2 Barcode-Generator

In Abschnitt 2.3.3.1 wurden verschiedene Typen zweidimensionaler Barcodes beschrieben. Für diese Implementierung wurde der *Datamatrix*-Code verwendet. Für ihn existiert die freie Java-Bibliothek *Barcode4J*[3], mit der *Datamatrix*-Codes generiert werden können. Listing 5.2 zeigt einen Codeausschnitt der Barcode-Generierung.

```
1   public byte[] generateDatamatrix(String code) {
2     // DataMatrixBean-Intanz erstellen
3     DataMatrixBean bean = new DataMatrixBean();
4
5     // Auflösung setzen
6     final int dpi = 300;
7
8     // In diesen Stream wird der Barcode geschrieben
9     ByteArrayOutputStream out = new ByteArrayOutputStream();
10
11    // Canvas-Provider für PNG-Ausgabe einstellen
12    BitmapCanvasProvider canvas = new BitmapCanvasProvider(out, "image/png",
          dpi, BufferedImage.TYPE_BYTE_BINARY, false, 0);
13
14    // Barcode generieren
15    bean.generateBarcode(canvas, code);
16
17    // Code in byte-Array speichern
18    byte[] barcode = out.toByteArray();
19    return barcode;
20  }
```

Listing 5.2: Quellcode: Barcode-Generator (Auszug)

[3] Barcode4J: http://barcode4j.sourceforge.net/

Die Barcode-Generierung erfolgt über die Methode *generateBarcode* der Klasse *DataMatrixBean* (Zeile 15). Der Methode wird ein *BitmapCanvasProvider*-Objekt (*canvas*) und ein String (*code*) übergeben. Der generierte Barcode wird später genau diesen String enthalten. Der *BitmapCanvasProvider* wird in Zeile 12 erstellt. Er definiert eine Reihe von Einstellungen, die den Barcode betreffen. Unter anderem wird dem Konstruktor ein *ByteArrayOutputStream*-Objekt (*out*) übergeben. Nach der Generierung (Zeile 15) kann der Barcode als Bild aus dem *ByteArrayOutputStream* gelesen und in ein byte-Array geschrieben werden (Zeile 18).

5.7.2 Implementierung des Mobile Clients

5.7.2.1 Tagging

Die Tagging-Komponente wird verwendet, um Objekte während des Stadtrundgangs identifizieren zu können. In Abschnitt 4.5.2.4 wurde bereits erläutert, dass es verschiedene Möglichkeiten gibt, das Tagging umzusetzen. Die Verwendung von RFID-Tags und die Nutzung von zweidimensionalen Barcodes sind zwei populäre Verfahren.

Einer Gartner-Studie[4] nach verfügen bereits heute über 60% aller verkauften Handys über eine Kamera. Demgegenüber sind derzeit kaum Geräte auf dem Markt, die über einen RFID-Leser verfügen. Damit haben die meisten mobilen Geräte die Möglichkeit, über die Kamera Barcodes zu lesen, können jedoch mit RFID-Tags nicht arbeiten. Aufgrund dieser Überlegungen werden für die Identifizierung von Objekten in dieser Applikation zweidimensionale Barcodes verwendet.

Die Tagging-Komponente ist dafür zuständig, Barcodes einzulesen und zu erkennen. Der Benutzer macht ein Foto des Barcodes mit der eingebauten Kamera des mobilen Geräts. Anschließend wird aus dem Barcode softwareseitig eine ID ermittelt. Diese identifiziert das Objekt, und die zugehörigen Informationen werden dem Benutzer angezeigt.

Das Lesen des Barcodes erfolgt in zwei Schritten. Zunächst wird über eine interne Kamera des mobilen Geräts ein Foto des Barcodes gemacht. Das *Microsoft .NET Compact Framework 2.0*[5] bietet hierfür eine Schnittstelle an, so dass keine weitere Software-Komponente für diesen Schritt nötig ist. Der Programmcode für das Fotographieren ist in den Zeilen 2 bis 9 in Listing 5.3 zu erkennen.

```
1  private void ScanBarcode() {
2    // Kamera-Dialog öffnen um einen Barcode abzufotographieren
3    CameraCaptureDialog cameraCapture = new CameraCaptureDialog();
4    cameraCapture.Mode = CameraCaptureMode.Still;
5    cameraCapture.StillQuality = CameraCaptureStillQuality.High;
6    cameraCapture.Resolution = new Size(640, 480);
7
8    // Kamrera-Screen aufrufen und prüfen, ob das Fotographieren erfolgreich
         war
9    if (DialogResult.OK == cameraCapture.ShowDialog()) {
10     // Dateiname der Bilddatei holen
11     string barcodeFileName = cameraCapture.FileName;
12
```

[4] Gartner: http://www.gartner.com

[5] Microsoft .NET Compact Framework: http://msdn.microsoft.com/smartclient/understanding/netcf/

```
13      // Barcode erkennen (DTK Barcode Reader SDK)
14      BarcodeReader barcodeReader = new BarcodeReader();
15      barcodeReader.ReadDataMatrix = true;
16      Barcode[] barcodes = barcodeReader.Read(barcodeFileName);
17
18      // Wenn der Barcode erkannt wurde
19      if (barcodes.Length > 0) {
20        // Informationen für zugehöriges Objekt anzeigen
21      }
22    }
23  }
```

Listing 5.3: Quellcode: Barcode-Reader (Auszug)

Im zweiten Schritt erfolgt die eigentliche Barcode-Erkennung (Zeilen 13 bis 16). Hierfür wird die externe Komponente *DTK Barcode Reader SDK*[6] eingesetzt. In Zeile 15 wird festgelegt, dass Datamatrix-Codes erkannt werden sollen. Danach wird die Barcodeerkennung gestartet. Verläuft dieser Vorgang erfolgreich, werden anschließend die Informationen zum entsprechenden Objekt angezeigt (Zeilen 19 bis 21).

5.7.2.2 Tour-Manager

In Abschnitt 4.5.2.5 wurde bereits erörtert, dass der *Tour-Manager* u. a. für das Aufzeichnen der zurückgelegten Strecke zuständig ist. Das Design sah vor, dass der *Tour-Recorder* immer dann die aktuelle Position zur Historie der zurückgelegten Strecke hinzufügt, wenn sich die Position des Nutzers ändert. Aus praktischen Gründen weicht die Implementierung vom Anwendungsdesign ein wenig ab.

Bei realen Messungen hat sich herausgestellt, dass der GPS-Empfänger ca. jede Sekunde eine Änderung der Position signalisiert, auch wenn diese nur unwesentlich ausfällt. Bei einem Stadtrundgang, der eine Stunde dauert, werden demnach 3600 (= 60 x 60) Einträge gespeichert. Dies führte zur Überlastung des Speichers auf dem mobilen Gerät. Aus diesem Grund wurde die Speicherung der zurückgelegten Wegstrecke modifiziert. Es wird nicht jede Positionsänderung gespeichert, sondern nur Positionen, die einen Mindestabstand (*DISTANCE_THRESHOLD*) zur Vorgängerposition aufweisen (vgl. Listing 5.4).

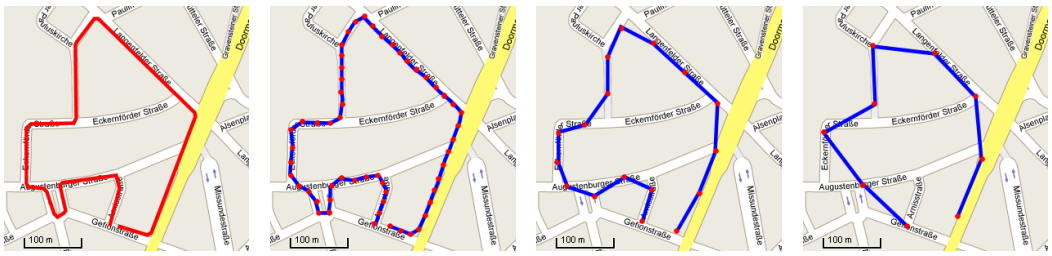
```
1   public void AddTourRecord(TourRecord newTourRecord) {
2
3     // Ermittlung des Vorgänger-Tour-Records
4     TourRecord lastTourRecord = this.GetLastTourRecord();
5
6     // Distanz zwischen dem Vorgänger-Tour-Record und dem neuen Tour-Record
7     double distance =
8         TourManager.Instance.Positioning
9               .CalculateDistance(lastTourRecord.PosX,
10                                 lastTourRecord.PosY,
11                                 newTourRecord.PosX,
12                                 newTourRecord.PosY);
13
```

[6] DTK Barcode Reader SDK: http://www.dtksoft.com/barreader.php

```
14    // Grenzwertprüfung
15    if (distance >= DISTANCE_THRESHOLD) {
16      tourRecordList.Add(newTourRecord);
17    }
18  }
```

Listing 5.4: Quellcode: Speicherung der Wegstrecke (Auszug)

Bei diesem Ansatz kann das Problem auftreten, dass die gespeicherte Wegstrecke zu ungenau wird, weil wichtige Positionen ausgelassen werden. Dies ist besonders dann der Fall, wenn der *DISTANCE_THRESHOLD* zu groß gewählt wird, und sich die ausgelassenen Positionen in Kurven befinden. Aus diesem Grund musste ein Kompromiss gefunden werden, zwischen einer möglichst geringen Anzahl gespeicherter Positionen und einer möglichst genauen Speicherung der zurückgelegten Wegstrecke. Durch experimentelle Versuche mit unterschiedlichen Werten für *DISTANCE_THRESHOLD* wurde der Wert *20* (= 20 Meter) als ein guter Kompromiss ermittelt. Hierbei wurden bei einer Tour von einer Stunde Länge durchschnittlich 180 Positionen gespeichert. Die gespeicherte Strecke wies dabei nur unwesentliche Abweichungen von der zurückgelegten Strecke auf.



(a) Zurückgelegte Strecke (b) Mind.-Abstand = 20m (c) Mind.-Abstand = 50m (d) Mind.-Abstand = 100m

Abbildung 5.8: Darstellung der Wegstrecke bei verschiedenen Mindestabständen

Abbildung 5.8 vergleicht eine real zurückgelegte Strecke mit drei Aufzeichnungen zu den Mindestabständen 20, 50 und 100 Meter. Es ist gut zu erkennen, dass die Anzahl der gespeicherten Positionen (rote Punkte) mit der Höhe des Mindestabstands abnimmt. Gleichzeitig sinkt aber auch die Genauigkeit im Vergleich zur real zurückgelegten Strecke.

6 Resümee

In diesem abschließenden Kapitel wird eine zusammenfassende Darstellung der bisherigen Ergebnisse erfolgen. Darüber hinaus wird eine Vorschau auf mögliche zukünftige Weiterentwicklungen gegeben. In Abschnitt 6.1 werden zunächst die Anforderungen an das entwickelte Anwendungssystem evaluiert. Anschließend folgt eine Zusammenfassung und Bewertung der vorliegenden Arbeit in Abschnitt 6.2. Zum Schluss werden in Abschnitt 6.3 Möglichkeiten für weitere Untersuchungen und Entwicklungen skizziert.

6.1 Evaluierung der Anforderungen

In diesem Abschnitt wird die Umsetzung der fachlichen sowie technischen Anforderungen evaluiert. Der größte Teil der Anforderungen wurde bereits im ersten Prototypen umgesetzt. Lediglich einige wenige Anforderungen konnten in der Implementierung derzeit noch nicht berücksichtigt werden. Details hierzu folgen in den Abschnitten 6.1.1 und 6.1.2.

6.1.1 Evaluierung der fachliche Anforderungen

Planung eines Stadtrundgangs
Die fachlichen Anforderungen dieser Phase sind fast vollständig in der Implementierung erfüllt worden. Wie bereits in Abschnitt 5.2.1 erörtert wurde können Stadtrundgänge geplant werden. Der Nutzer hat die Möglichkeit, Rundgänge auszuwählen, sich über OvIs zu informieren und diese der persönlichen Route hinzuzufügen. Eine Anforderung, die derzeit noch nicht implementiert wurde, ist die Möglichkeit, nach OvIs zu suchen. Außerdem ist es im aktuellen Zustand der Anwendung nicht möglich, die ausgewählten OvIs automatisch nach der kürzesten bzw. schnellsten Route zu sortieren.

Durchführung eines Stadtrundgangs
Die fachlichen Anforderungen an diesen Teil der Anwendung sind vollständig erfüllt worden (vgl. Abschnitt 5.2.2). Stadrundgänge können unter Zuhilfenahme von mobilen Geräten durchgeführt werden. Der Nutzer kann seine Position auf einer Karte sehen, er wird zu den gewünschten OvIs geführt und kann sich Informationen über die OvIs ansehen. Ebenso können Informationen über Objekte in und bei OvIs abgerufen und Annotationen in Bild- und Textform gemacht werden.

Rekapitulation eines Stadtrundgangs
Im aktuellen Prototypen fehlt die Möglichkeit der Freigabe des eigenen Stadtrundgangs für andere Nutzer. Mit Ausnahme dieser Funktion, sind die fachlichen Anforderungen für die Rekapitulation erfüllt worden (vgl. Abschnitt 5.2.3). In einer Kartenansicht kann die zuvor real zurückgelegte Strecke virtuell abgelaufen werden. Dabei werden neben der Strecke auch die besuchten OvIs und erstellten Annotationen angezeigt.

Verwaltung
In Abschnitt 5.2.4 wurde bereits erörtert, dass Karten, Touren, OvIs und Objekte erstellt und bearbeitet werden können. Damit sind alle fachlichen Anforderungen bezüglich der Verwaltung der Daten umgesetzt worden.

6.1.2 Evaluierung der technischen Anforderungen

Positionierung
Hier wurde verlangt, dass eine Positionierung im Freien möglich sein sollte. Weiterhin sollten auch innerhalb von Gebäuden Objekte identifiziert werden können.

Umgesetzt wurden diese Anforderungen in den Komponenten *Positionierung* und *Tagging* des mobilen Teilsystems. In Praxistests konnte beobachtet werden, dass die Genauigkeit der GPS-Positionierung stark vom verwendeten GPS-Empfangsgerät abhängig ist. Mit dem GPS-Empfänger *Holux GPSlim 236* konnte eine Genauigkeit von ca. zehn bis fünfzehn Metern erreicht werden. Diese war für das Auffinden der OvIs ausreichend. Probleme entstanden dagegen bei der Aufzeichnung der zurückgelegten Wegstrecke. Die GPS-Signale wiesen in einzelnen Fällen kurzzeitig starke Abweichungen von der tatsächlichen Position auf. Dies führte dazu, dass in der Aufzeichnung der zurückgelegten Wegstrecke an manchen Stellen große Unterbrechungen der Streckenführung vorhanden waren. In Zukunft könnte dieses Problem dadurch gelöst werden, dass über einen Interpolationsalgorithmus die aufgezeichnete Strecke korrigiert wird.

Die Anforderung der Objektidentifizierung wurde über die Barcode-Erkennung realisiert und damit ebenfalls erfüllt. Die verwendete externe Komponente *DTK Barcode Reader SDK* erwies sich dabei als sehr zuverlässig.

Kartenansicht
Die Forderung lautete, dem Nutzer sowohl auf dem stationären als auch auf dem mobilen Teilsystem eine Kartenansicht zur Verfügung zu stellen. Diese Anforderung wurde, wie bereits in den Abschnitten 5.2 und 5.5 erläutert, umgesetzt.

Annotationen
Annotationen werden über die Position, an der sie gemacht wurden, identifiziert. Das Identifikationsverfahren basiert demnach auf der Positionsbestimmung (vgl. Abschnitt 2.3.3.1). Die Präsentation erfolgt während des Stadtrundgangs *on-location* und *detached* auf dem mobilen Gerät und bei der Rekapitulation *off-location* und ebenfalls *detached* auf der Webseite. Das Erstellen der

Annotation ist nur *on-location* und *detached* auf dem mobilen Gerät möglich. Für das Bearbeiten der Annotationen gilt das gleiche wie für die Präsentation.

Benutzbarkeit
Um eine Abhängigkeit von der Verfügbarkeit einer mobilen Datenverbindung zu vermeiden, wurde das Design so konzipiert, dass Datenverbindungen während des Stadtrundgangs nicht erforderlich sind.

Bei der Entwicklung des mobilen Geräts wurde stets darauf geachtet, eine möglichst einfache Bedienung zu ermöglichen. So kann die Anwendung fast komplett über die Navigationstasten des mobilen Geräts bedient werden. Die Verwendung eines Eingabestifts ist nur notwendig, wenn Texte geschrieben werden sollen. Dies ist deshalb wichtig, weil es relativ unergonomisch wäre, wenn der Nutzer während der Fortbewegung einen Eingabestift nutzen müsste.

Zur Verbesserung der Ergnomie wäre es sinnvoll, die mobile Anwendung in einer zukünftigen Version mit einer Sprachausgabe auszustatten. Das Lesen langer Texte auf dem mobilen Gerät hat sich als eher unpraktikabel herausgestellt.

Persistenz
Die Anforderung, eine persistente Speicherung aller Daten, sowohl auf dem stationären als auch auf dem mobilen Teilsystem, zu gewährleisten, wurde umgesetzt. Die entsprechenden Details wurden in den Abschnitten 4.4.1 und 4.5.1 beschrieben.

Kommunikation
Die Kommunikation zwischen dem stationären System und der mobilen Anwendung wird, wie in Abschnitt 4.4.2.2 beschrieben, über Webservices realisiert. In diesem Abschnitt wurden auch eine Schnittstelle und ein entsprechendes Datenmodell für die Übertragung der Daten vorgestellt.

Privacy
Das Systemdesign sieht vor, dass Daten eines Stadtrundgangs anderen Nutzern nur dann zur Verfügung gestellt werden, wenn der Nutzer diese explizit freischaltet. Insofern hat jeder Nutzer die Möglichkeit, selbst über die Verwendung der eigenen Daten zu entscheiden. Ein Problem besteht jedoch darin, dass der Betreiber des Anwendungssystems prinzipiell Zugriff auf die Daten aller Nutzer hat. Er könnte somit Missbrauch treiben, indem er sensible Informationen der Nutzer rechtswidrig verwertet. Dieses grundsätzliche Privacy-Problem bleibt bestehen. Letztendlich hängt die Gewährleistung der Privatsphäre, wie so häufig, vom Vertrauen zwischen dem Anbieter und den Nutzern und der Verlässlichkeit des Anbieters ab [BS03]. Eine gewisse Anonymität kann der Nutzer dadurch erreichen, dass er dem Anbieter seine persönlichen Daten nicht preisgibt. Jedoch ist nicht auszuschließen, dass aufgrund der gespeicherten Informationen aus den Stadtrundgängen auf die Identität des Nutzers geschlossen werden kann.

6.2 Zusammenfassung & Bewertung

In dieser Arbeit wurden die Themenbereiche „Ortsbezogene Dienste“ und „Ubiquitäre Annotationen“ untersucht. Die Grundlagen dieser Bereiche wurden erarbeitet und die wichtigsten Eigenschaften, Technologien und Anwendungsbereiche identifiziert. Dabei wurde festgestellt, dass eine große Zahl unterschiedlicher Szenarien und Anwendungen für beide Bereiche möglich sind. Einige konkrete Arbeiten wurden in Kapitel 3 vorgestellt.

Anschließend wurde ein neues Anwendungssystem entworfen und prototypisch implementiert. Der Fokus hierbei lag darauf, Technologien aus den beiden Grundlagenbereichen miteinander zu verknüpfen. Ziel war es herauszufinden, ob diese Verknüpfung einen Mehrwert für die Nutzer solcher Systeme bietet. Als Beispielszenario wurde ein Stadtrundgang gewählt.

Dieser enthält tatsächlich Elemente aus beiden Bereichen. Dem Nutzer werden beim Erreichen von OvIs Informationen ortsbezogen zur Verfügung gestellt. Dies ist ein wesentlicher Aspekt ortsbezogener Dienste. Der Themenbereich der ubiquitären Annotationen wird dadurch abgedeckt, dass Nutzer die Möglichkeit haben, Bild- und Textannotationen an beliebigen Orten zu erstellen und diese mit den Orten zu verknüpfen.

Erste Erfahrungen mit dem entwickelten Prototypen lassen tatsächlich einen Mehrwert dieser Kombination für die Nutzer erahnen. Die Nutzer werden als aktive Teilnehmer stärker in die Anwendung einbezogen. Sie sind nicht länger reine Informationskonsumenten, sondern leisten selbst einen Beitrag, indem sie Informationen sammeln und anderen Nutzern zur Verfügung stellen. Die Annotationen, die während des Stadtrundgangs gemacht werden können, tragen zur Interaktivität und zur Individualisierung bei. Das Erlebnis „Stadtrundgang“ endet nicht mit der Durchführung, sondern setzt sich auch danach fort.

Leider war es im Rahmen dieser Arbeit nicht mehr möglich, umfangreiche Benutzerstudien durchzuführen. Interessant wäre vor allem gewesen, zu erfahren, inwieweit die Informationen, die von den Nutzern zusammengetragen werden, tatsächlich einen Mehrwert für andere Nutzer bieten können. In diesem Zusammenhang wäre auch zu untersuchen gewesen, wie mit einer möglichen bewussten Manipulation von Daten durch Nutzer umgegangen werden kann. Diese Studien sollten Thema einer zukünftigen Untersuchung sein.

Ebenfalls ging es in dieser Arbeit darum, die Kombination von Outdoor- und Indoor-Szenarien zu testen. Das Anwendungssystem ermöglicht daher die Durchführung von Stadtrundgängen im Freien und gleichzeitig die Erkennung von Objekten innerhalb von Gebäuden, wie z. B. in Museen. Der Vorteil dieser Kombination besteht darin, dass die Touristen mit nur einem System ihren kompletten Stadtrundgang durchführen können. Dabei werden sie nicht nur von Ort zu Ort geführt, sondern können auch in und bei bestimmten OvIs zusätzliche Detailinformationen über interessante Objekte abrufen. Dies führt dazu, dass auch in der Rekapitulation viele detaillierte und personalisierte Informationen über den eigenen Stadtrundgang abgerufen werden können. In ersten Praxistests hat sich auch dieses Feature bewährt.

6.3 Ausblick

Der entwickelte Prototyp befindet sich in einem einsetzbaren Zustand. Dies ermöglichte bereits erste Erprobungen, die Positives versprechen. Jedoch sind für die Zukunft weitere Entwicklungen und Erweiterungen vorstellbar und notwendig. Die folgenden Ausführungen skizzieren einige mögliche Ansätze. Dabei wird zwischen einem fachlichen und einem technischen Ausblick unterschieden.

6.3.1 Fachlicher Ausblick

Das Ausgangsszenario in Abschnitt 4.1 sah keine direkte Kommunikation zwischen zwei Teilnehmern während eines Stadtrundganges vor. Dennoch sind sicherlich Situationen vorstellbar, in denen solch eine Funktionalität Sinn machen würde. Die Nutzer könnten beispielsweise schon während des Stadtrundgangs Annotationen austauschen und sich gegenseitig über OvIs informieren. In einer zukünftigen Weiterentwicklung sollten daher die Möglichkeiten solcher Direktverbindungen zwischen zwei oder mehr Nutzern evaluiert und bei Bedarf entworfen und realisiert werden. Eine stärkere Kollaboration von Touristen vor und nach einem Stadtrundgang sollte ebenfalls in Betracht gezogen werden.

Zugunsten der Robustheit der Anwendung wurde auf die Möglichkeit einer Daten-Verbindung während eines Stadtrundrundganges verzichtet (vgl. Abschnitt 4.5.2.2). Auch diese Entscheidung könnte im Zusammenhang mit einer Weiterentwicklung des Anwendungssystems überprüft werden. Dabei ist ein möglicher Mehrwert einer Daten-Verbindung zu untersuchen. Es ist insbesondere vorstellbar, das eine dauerhafte Daten-Verbindung zusätzliche Szenarien und Interaktionsmöglichkeiten zulässt.

In der vorliegenden Arbeit waren aus zeitlichen Gründen keine umfassenden Benutzerstudien inklusive einer anschließenden Auswertung möglich. Diese könnten aber wichtige Hinweise auf den Nutzen des entwickelten Anwendungssystems geben. Des Weiteren könnten zusätzliche Anliegen der Nutzer identifiziert werden, so dass sich eine mögliche Weiterentwicklung an den Wünschen der Nutzer orientiert. Aus diesen Gründen wären umfassende Benutzerstudien eine sinnvolle Möglichkeit, den Ansatz und die Ziele dieser Arbeit fortzuführen.

Ebenfalls aus zeitlichen Gründen konnten keine Usability-Tests durchgeführt werden. Diese sollten nachgeholt werden, um die Benutzbarkeit des Anwendungssystems zu evaluieren. Des Weiteren können die Usability-Tests wichtige Informationen über den Umsetzungsgrad der Gestaltungsgrundsätze für Benutzeroberflächen (vgl. Abschnitt 4.4.3) liefern.

Weitere Überlegungen könnten sich mit der Frage auseinandersetzen, wie und von wem die angebotenen Inhalte produziert werden. Das derzeitige Modell sieht vor, dass die Nutzer einen Teil der Inhalte selbst erstellen. Hier ist zu untersuchen, wie man die Korrektheit und Zuverlässigkeit der Informationen sicherstellen kann. Zusätzlich könnten Schnittstellen angeboten werden, um z. B. Gemeinden und Museen das Einstellen von eigenen Informationen zu ermöglichen.

6.3.2 Technischer Ausblick

In der derzeitigen Implementierung wird GPS als Positionierungstechnologie verwendet. Das Design des Anwendungssystems lässt jedoch prinzipiell jede beliebige Technologie zu (vgl. Abschnitt 4.5.2.3). Zu überprüfen wäre, ob nicht weitere Technologien zur Positionsbestimmung integriert werden könnten und sollten. Aktuelle Forschungsergebnisse versprechen z. B. über die Positionierung per WLAN eine genauere Ortsbestimmung in Ballungsräumen und innerhalb von Gebäuden [YAF06].

Für die Implementierung der Identifizierung von Objekten wurden zweidimensionale Barcodes verwendet. Der Hauptgrund hierfür lag darin, dass derzeit ein hoher Anteil mobiler Geräte auf dem Markt in der Lage ist, die Barcode-Technologie zu nutzen (vgl. Abschnitt 5.7.2.1). Demgegenüber ist die RFID-Technik in Consumer-Produkten derzeit kaum verfügbar. Diese Situation könnte sich jedoch in den folgenden Jahren ändern. In so einem Fall wäre die Entwicklung einer RFID-Komponente in Erwägung zu ziehen.

Das Thema „Navigation“ gehörte nicht zu den Hauptthemen dieser Arbeit. Dennoch wurde im entwickelten Anwendungssystem eine rudimentäre Navigationsfunktion implementiert. Sie zeigt, wie in Abschnitt 5.2.2 beschrieben, die Richtung und Entfernung zum nächsten Ziel an. Einzelne Straßen oder Landmarken werden nicht angezeigt. Eine mögliche Weiterentwicklung könnte sich daher mit der Erweiterung der Navigationsfunktionen befassen. In der Forschung ist insbesondere das Thema „Fußgängernavigation“ stark in der Diskussion [TS06, BS06, MK06]. In diesem Bereich sind in der nächsten Zeit neue Forschungsergebnisse zu erwarten.

In Abschnitt 6.1 wurde bereits dargelegt, dass das Lesen langer Texte auf mobilen Geräten sehr unpraktikabel ist. Dies wird in verschiedenen Usability-Studien bestätigt [BC03]. Als Lösung wird oftmals eine Sprachausgabe als Alternative vorgeschlagen. Aus technischen Gründen konnte dies im aktuellen Prototypen jedoch nicht realisiert werden. Für eine zukünftige Weiterentwicklung sollte dieser Aspekt erneut aufgegriffen werden.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Literaturverzeichnis

[AAH+97] Gregory D. Abowd, Christopher G. Atkeson, Jason Hong, Sue Long, Rob Kooper, and Mike Pinkerton. Cyberguide: A mobile context-aware tour guide. 1997.

[AGIS05] Frank Adelstein, Sandeep K. S. Gupta, Golden G. Richard III, and Loren Schwiebert. *Fundamentals of Mobile and Pervasive Computing*. McGraw-Hill, 2005.

[Alb07] Tom Alby. *Web 2.0*. Hanser Fachbuchverlag, 2007.

[APV+06] Heikki Ailisto, Lauri Pohjanheimo, Pasi Välkkynen, Esko Strömmer, Timo Tuomisto, and Ilkka Korhonen. Bridging the physical and virtual worlds by local connectivity-based physical selection. *Personal and Ubiquitous Computing*, 10(6):333–344, 2006.

[ATJ00] Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead Jr. Chimera: hypermedia for heterogeneous software development enviroments. *ACM Trans. Inf. Syst*, 18(3):211–245, 2000.

[BC03] Christian Bornträger and Keith Cheverst. Social and Technical Pitfalls Designing a Tourist Guide System. Technical report, Lancaster University, 2003.

[BCGH03] Niels Olof Bouvin, Bent G. Christensen, Kaj Grønbæk, and Frank Allan Hansen. Hycon: a framework for context-aware mobile hypermedia. *The New Review of Hypermedia and Multimedia*, 9(1):59–88, 2003.

[BDS] Bundesdatenschutzgesetz (BDSG). http://www.gesetze-im-internet.de/bdsg_1990 - Abruf am 05.02.2008.

[BLP06] P Bull, R Limb, and R Payne. Pervasive Home Environments. In Alan Steventon and Steve Wright, editors, *Intelligent Spaces - The Application of Pervasive ICT*. Springer-Verlag, 2006.

[bmg] Bundesministerium für Gesundheit - Die Gesundheitskarte. http://www.die-gesundheitskarte.de - Abruf am 05.02.2008.

[BMI] Bundesministerium des Inneren - Elektronischer Reisepass. http://www.bmi.bund.de/nn_1082274/Internet/Navigation/DE/Themen/PaesseUndAusweise/ElektronischerReisepass/elektronischer__reisepass__node.html__nnn=true - Abruf am 05.02.2008.

[BP99] Paramvir Bahl and Venkata N. Padmanabhan. User Location and Tracking in an In-Building Radio Network. Technical Report MSR-TR-99-12, Microsoft Research (MSR), February 1999.

[BS03] Alastair R. Beresford and Frank Stajano. Location Privacy in Pervasive Computing. *Pervasive Computing, IEEE*, 2:46–55, 2003.

[BS04] Alastair Beresford and Frank Stajano. Mix Zones: User Privacy in Location-aware Services. In *Proceedings of the Second IEEE Annual Conference On Pervasive Computing and Communications Workshops*, pages 127–131. IEEE, 2004.

[BS06] Ashweeni Kumar Beeharee and Anthony Steed. A natural wayfinding exploiting photos in pedestrian navigation systems. In Marko Nieminen and Mika Röykkee, editors, *Mobile HCI*, pages 81–88. ACM, 2006.

[Bur07] Herbert Burbiel. *SOA & Webservices in der Praxis*. Franzis, 2007.

[Bus45] V. Bush. As We May Think. *The Atlantic Monthly*, 176:101–108, 1945.

[C2C] Car 2 Car Communication Consortium. http://www.car-to-car.org - Abruf am 05.02.2008.

[CCD+04] Scott Carter, Elizabeth Churchill, Laurent Denoue, Jonathan Helfman, and Les Nelson. Digital graffiti: public annotation of multimedia content. In *Proceedings of ACM CHI 2004 Conference on Human Factors in Computing Systems*, volume 2 of *Late breaking result papers*, pages 1207–1210, 2004.

[CCKM01] Paul Castro, Patrick Chiu, Ted Kremenek, and Richard Muntz. A Probabilistic Room Location Service for Wireless Networked Environments. *Lecture Notes in Computer Science*, 2201, 2001.

[CSLT06] Adrian David Cheok, Anuroop Sreekumar, Cao Lei, and Le Nam Thang. Capture the flag: mixed-reality social gaming with smart phones. *IEEE Pervasive Computing*, 5(2):62–69, 2006.

[Cue02] Jorge R. Cuellar. *Geographic Location in the Internet*, chapter Location Information Privacy, pages 179–208. Kluwer Academic Publishers, 2002.

[DA00] Anind K. Dey and Gregory D. Abowd. Towards a Better Understanding of Context and Context-Awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness, as part of the 2000 Conference on Human Factors in Computing Systems (CHI 2000)*, April 2000.

[DCKH94] Davis, Hugh C., Simon Knight, and Wendy Hall. Light Hypermedia Link Services: A Study of Third Party Application Integration. In *Proceedings of the ECHT'94 European Conference on Hypermedia Technologies*, Papers, pages 41–50, 1994.

[DH07] Jürgen Dunkel and Andreas Holitschke. *Softwarearchitektur für die Praxis*. Springer Verlag, 2007.

[DRW02] Diep Dao, Chris Rizos, and Jinling Wang. Location-based services: technical and business issues. *GPS Solutions*, 6(3):169–178, December 2002.

[Eck04] Claudia Eckert. *IT-Sicherheit: Konzepte, Verfahren, Protokolle*. Oldenbourg Wissenschaftsverlag GmbH, München, 2004.

[EPS+01] Fredrik Espinoza, Per Persson, Anna Sandin, Hanna Nyström, Elenor Cacciatore, and Markus Bylund. GeoNotes: Social and Navigational Aspects of Location-Based Information Systems. Technical Report T2001-08, Swedish Institute of Computer Science, June 2, 2001.

[eri] Ericsson Mobile Positioning System. http://www.ericsson.com/products/hp/Ericsson_Mobile_Positioning_System__MPS__8_0_pos.shtml - Abruf am 05.02.2008.

[Fin06] Klaus Finkenzeller. *RFID-Handbuch*. Hanser Fachbuchverlag, 2006.

[FK04] Peter Forbrig and Immo Kerner. *Lehr- und Übungsbuch Softwareentwicklung*. Hanser Fachbuchverlag, 2004.

[FL07] Michael Friedewald and Ralf Lindner. Datenschutz, Privatsphäre und Identität in intelligenten Umgebungen. In Friedemann Mattern, editor, *Die Informatisierung des Alltags - Leben in smarten Umgebungen*, pages 207–232. Springer-Verlag, 2007.

[Gar03] Hansjürgen Garstka. Informationelle Selbstbestimmung und Datenschutz: Das Recht auf Privatsphäre. In Christiane Schulzki-Haddouti, editor, *Bürgerrechte im Netz*, volume 382, pages 48–70. Bundeszentrale für politische Bildung, 2003.

[GHJV01] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*. Addison-Wesley, 2001.

[GKC04] Neil Gershenfeld, Raffi Krikorian, and Danny Cohen. The Internet of Things. *Scientific American*, 291(4):46–51, 2004.

[GKOE03] Kaj Gronbaek, Jannie Friis Kristensen, Peter Orbaek, and Mette Agger Eriksen. Physical hypermedia: organising collections of mixed physical and digital material. In *Hypertext*, pages 10–19. ACM, 2003.

[Gün02] Günther Pospischil and Martina Umlauft and Elke Michlmayr. Designing LoL@, a Mobile Tourist Guide for UMTS. *Lecture Notes in Computer Science*, 2411:140–154, 2002.

[gooa] Google Geocoding. http://www.google.com/apis/maps/documentation/services.html#Geocoding - 05.02.2008.

[Goob] Google Maps API. http://www.google.de/apis/maps - Abruf am 05.02.2008.

[Han06] Frank Allan Hansen. Ubiquitous annotation systems: technologies and challenges. In Uffe Kock Wiil, Peter J. Nürnberg, and Jessica Rubart, editors, *Hypertext*, pages 121–132. ACM, 2006.

[HBC+04] Frank Allan Hansen, Niels Olof Bouvin, Bent G. Christensen, Kaj Grønbæk, Torben Bach Pedersen, and Jevgenij Gagach. Integrating the web and the world: contextual trails on the move. In *Hypertext*, pages 98–107. ACM, 2004.

[HBHS06] Gregor Hackenbroich, Christof Bornhövd, Stephan Haller, and Joachim Schaper. Optimizing Business Processes by Automatic Data Acquisition: RFID Technology and Beyond. In George Roussos, editor, *Ubiquitous and Pervasive Commerce - New Frontiers for Electronic Business*. Springer Verlag, 2006.

[Hel06] Udo Helmbrecht. *Pervasive Computing: Entwicklungen und Auswirkungen*. Bundesamt für Sicherheit in der Informationstechnik, 2006.

[HF05] Sherry Hsi and Holly Fait. RFID enhances visitors' museum experience at the Exploratorium. *Commun. ACM*, 48(9):60–65, 2005.

[HH02] Fabian Hermann and Frank Heidmann. User Requirement Analysis and Interface Conception for a Mobile, Location-Based Fair Guide. In Fabio Paternò, editor, *Mobile HCI*, volume 2411 of *Lecture Notes in Computer Science*, pages 388–392. Springer, 2002.

[Hib] Hibernate - Unterstütze Datenbanken. http://www.hibernate.org/80.html - Abruf am 05.02.2008.

[Hil07] Lorenz M. Hilty. Risiken und Nebenwirkungen der Informatisierung des Alltags. In Friedemann Mattern, editor, *Die Informatisierung des Alltags - Leben in smarten Umgebungen*, pages 187–206. Springer-Verlag, 2007.

[HMNS03] U. Hansmann, I. Merk, M. S. Nicklous, and T. Stober. *Pervasive Computing Handbook*. Springer-Verlag, New York, 2 edition, 2003.

[HSK04] Lorenz M. Hilty, Claudia Som, and Andreas Köhler. Assessing the Human, Social, and Environmental Risks of Pervasive Computing. In *Journal of Human and Ecological Risk Assessment*, volume 10, pages 853–874. The Association for Environmental Health and Sciences, oct 2004.

[HT05] Andrew Hunt and David Thomas. *Pragmatisch Programmieren: Unit-Tests mit JUnit*. Hanser Fachbuchverlag, 2005.

[HVB01] Jerey Hightower, Chris Vakili, and Gaetano Borriello. Design and Calibration of the SpotON Ad-Hoc Location Sensing System, August 27 2001.

[Jac04] Hans-Arno Jacobsen. Middleware for Location-Based Services. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 83–114. Morgan Kaufmann, 2004.

[Jun05] Iris A. Junglas. An Experimental Investigation of Location-Based Services. In *HICSS*. IEEE Computer Society, 2005.

[JW06] Kalle Jegers and Mikael Wiberg. Pervasive gaming in the everyday world. *IEEE Pervasive Computing*, 5(1):78–85, 2006.

[Kaa03] Eija Kaasinen. User needs for location-aware mobile services. *Personal and Ubiquitous Computing*, 7(1):70–79, 2003.

[KB03] Christian Kray and Jörg Baus. A survey of mobile guides. In *Mobile HCI*, 2003.

[KLP+01] Eija Kaasinen, Juha Luoma, Merja Penttinen, Tuula Petäkoski-Hult, and Rolf Södergård. Basics of Human-Centered Design in Personal Navigation. Technical report, NAVI programme report, 2001.

[Küp05] Axel Küpper. *Location-Based Services: Fundamentals and Operation*. John Wiley & Sons Ltd., West Sussex, 2005.

[KT05] H. Kato and K. T. Tan. 2D barcodes for mobile phones. In *2nd International Conference on Mobile Technology, Applications and Systems*, 2005.

[Kun06] Christophe Kunze. Ubiquitous Healthcare: Anwendung ubiquitärer Informationstechnologien im Telemonitoring. Master's thesis, Universität Fridericiana Karlsruhe, 2006.

[KWE+05] P. Krauchi, P. A. Wager, M. Eugster, G. Grossmann, and L. Hilty. End-of-life impacts of pervasive computing. *IEEE Technology and Society Magazine*, 24(1):45–53, 2005.

[Lan05] Marc Langheinrich. Personal Privacy in Ubiquitous Computing: Tools and System Support. Dissertation, Swiss Federal Institute of Technology Zurich, 2005.

[Lan07] Marc Langheinrich. Gibt es in einer total informatisierten Welt noch eine Privatsphäre? In Friedemann Mattern, editor, *Die Informatisierung des Alltags - Leben in smarten Umgebungen*, pages 233–264. Springer-Verlag, 2007.

[Leh03] Franz Lehner. *Mobile und drahtlose Informationssysteme - Technologien, Anwendungen, Märkte*. Springer-Verlag, 2003.

[LM03] Marc Langheinrich and Friedemann Mattern. Digitalisierung des Alltags - Was ist Pervasive Computing? In *Aus Politk und Zeitgeschichte*, volume 42. Bundeszentrale für politische Bildung, 2003.

[Mac98] Wendy E. Mackay. Augmented reality: linking real and virtual worlds: a new paradigm for interacting with computers. In Tiziana Catarci, Maria Francesca Costabile, Giuseppe Santucci, and Laura Tarantino, editors, *AVI*, pages 13–21. ACM Press, 1998.

[Mac06] Allan Maclean. Sozioökonomische Voraussetzungen und Auswirkungen des Pervasive Computing. In *Pervasive Computing: Entwicklungen und Auswirkungen*. Bundesamt für Sicherheit in der Informationstechnik, 2006.

[Mat01] Friedemann Mattern. Pervasive / Ubiquitous Computing. *Informatik-Spektrum*, 24(3):145–147, June 2001.

[Mat05] Friedmann Mattern. Die technische Basis für das Internet der Dinge. In Elgar Fleisch and Friedmann Mattern, editors, *Das Internet der Dinge - Ubiquitous Computing und RFID in der Praxis*. Springer-Verlag, 2005.

[MBGH07] Kaj Mäkelä, Sara Belt, Dan Greenblatt, and Jonna Häkkilä. Mobile interaction with visual and RFID tags: a field study on user perceptions. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, 2007.

[MCMN05] Carsten Magerkurth, Adrian David Cheok, Regan L. Mandryk, and Trond Nilsen. Pervasive games: bringing computer entertainment back to the real world. *Computers in Entertainment*, 3(3):4, 2005.

[MFD03] Ginger Myles, Adrian Friday, and Nigel Davies. Preserving Privacy in Environments with Location-Based Applications. *Pervasive Computing, IEEE*, 2:56–64, 2003.

[MK06] Yoji Miyazaki and Toshiyuki Kamiya. Pedestrian Navigation System for Mobile Phones Using Panoramic Landscape Images. In *SAINT*, pages 102–108. IEEE Computer Society, 2006.

[MM02] Yoshiki Miichi and Susumu Masuda. Mitsubishi's ASV-2 Passenger Car Obtained Japan's Land, Infrastructure and Transport Ministerial Approval - Testing on Public Roads Prior to Commercialization. Technical Report 14, Mitsubishi Motors, 2002.

[MR03] Sven Meyer and Andry Rakotonirainy. A Survey of Research on Context-Aware Homes. In Chris Johnson, Paul Montague, and Chris Steketee, editors, *Workshop on Wearable, Invisible, Context-Aware, Ambient, Pervasive and Ubiquitous Computing*, volume 21 of *CRPIT*, pages 159–168, Adelaide, Australia, 2003. ACS.

[MS00] Natalia Marmasse and Chris Schmandt. Location-Aware Information Delivery with ComMotion. In Peter J. Thomas and Hans-Werner Gellersen, editors, *HUC*, volume 1927 of *Lecture Notes in Computer Science*, pages 157–171. Springer, 2000.

[MS05] Dilip Mohapatra and S. B. Suma. Survey of location based wireless services. In *IEEE International Conference on Personal Wireless Communications*, pages 358–362, 2005.

[MS07] A. Millonig and K. Schechtner. Developing Landmark-Based Pedestrian-Navigation Systems. *IEEE Trans. Intelligent Transportation Systems*, 8(1):43–49, March 2007.

[Oes06] Bernd Oesterreich. *Analyse und Design mit UML 2.1 - Objektorientierte Softwareentwicklung*. Oldenbourg Wissenschaftsverlag, 2006.

[OHH04] Eisaku Ohbuchi, Hiroshi Hanaizumi, and Lim Ah Hock. Barcode Readers using the Camera Device in Mobile Phones. In *CW*, pages 260–265. IEEE Computer Society, 2004.

[Ope] OpenGIS Location Service (OpenLS). http://www.opengeospatial.org/standards/olscore - Abruf am 05.02.2008.

[Pas97] Jason Pascoe. The Stick-e Note Architecture: Extending the Interface beyond the User. In *Intelligent User Interfaces*, pages 261–264, 1997.

[PBC+01] Salil Pradhan, Cyril Brignone, Jun-Hong Cui, Alan McReynolds, and Mark T. Smith. Websigns: Hyperlinking Physical Locations to the Web. *IEEE Computer*, 34(8):42–48, 2001.

[Pit06] Olli Pitkänen. Legal Challenges to Ubiquitous Commerce. In George Roussos, editor, *Ubiquitous and Pervasive Commerce - New Frontiers for Electronic Business*. Springer Verlag, 2006.

[pv2] Polymer Vision. http://www.polymervision.com - Abruf am 05.02.2008.

[Rei03] Tumasch Reichenbacher. Mobile Cartography - Adaptive Visualisation of Geographic Information on Mobile Devices. Master's thesis, Technische Universität München, 2003.

[RM03] Bharat Rao and Louis Minakakis. Evolution of mobile location-based services. *Commun. ACM*, 46(12):61–65, 2003.

[Roß07] Alexander Roßnagel. Informationelle Selbstbestimmung in der Welt des Ubiquitous Computing. In Friedemann Mattern, editor, *Die Informatisierung des Alltags - Leben in smarten Umgebungen*, pages 265–290. Springer-Verlag, 2007.

[Roh05] Michael Rohs. Visual Code Widgets for Marker-Based Interaction. In *ICDCS Workshops*, pages 506–513. IEEE Computer Society, 2005.

[Rot04] Jörg Roth. Data Collection. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 175–205. Morgan Kaufmann, 2004.

[Rot05] Jörg Roth. *Mobile Computing: Grundlagen, Technik, Konzepte*. dpunkt.verlag GmbH, Heidelberg, 2005.

[Rou06] George Roussos. *Ubiquitous and Pervasive Commerce: New Frontiers for Electronic Business*. Springer Verlag, 2006.

[Sat01] M. Satyanarayanan. Pervasive Computing: Vision and Challenges, May 29 2001.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen. There is more to context than location. *Computers & Graphics*, 23(6):893–901, 1999.

[SC04] Mark Strassman and Clay Collier. Case Study: The Development of the Find Friends Application. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 27–40. Morgan Kaufmann, 2004.

[Sch06a] Jochen Schiller. Die Anwendungen des Pervasive Computing. In *Pervasive Computing: Entwicklungen und Auswirkungen*. Bundesamt für Sicherheit in der Informationstechnik, 2006.

[Sch06b] Albrecht Schmidt. Die Technologie des Pervasive Computing. In *Pervasive Computing: Entwicklungen und Auswirkungen*. Bundesamt für Sicherheit in der Informationstechnik, 2006.

[sim] SIMpill. http://www.simpill.com - Abruf am 05.02.2008.

[SMR+97] Thad Starner, Steve Mann, Bradley J. Rhodes, Jeffrey Levine, Jennifer Healey, Dana Kirsch, Rosalind W. Picard, and Alex Pentland. Augmented Reality Through Wearable Computing. *Presence*, 6(4):386–398, 1997.

[Som07] Ian Sommerville. *Software Engineering*. Pearson Studium, 2007.

[Spi04] Sarah Spiekermann. General Aspects of Location Based Services. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 9–26. Morgan Kaufmann, 2004.

[Sta05] Josef L. Staud. *Datenmodellierung und Datenbankentwurf - Ein Vergleich aktueller Methoden*. Springer-Verlag, 2005.

[SV04] Jochen H. Schiller and Agnès Voisard. Introduction. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 1–8. Morgan Kaufmann, 2004.

[SVMY04] Shashi Shekhar, Ranga Raju Vatsavai, Xiaobin Ma, and Jin Soung Yoo. Navigation Systems: A Spatial Database Perspective. In Jochen H. Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 41–82. Morgan Kaufmann, 2004.

[SW06] Alan Steventon and Steve Wright, editors. *Intelligent Spaces - The Application of Pervasive ICT*. Springer-Verlag, 2006.

[tol] Toll Collect. http://www.toll-collect.de - Abruf am 05.02.2008.

[TS06] Manfred Tscheligi and Reinhard Sefelin. Mobile navigation support for pedestrians: can it work and does it pay off? *Interactions*, 13(4):31–33, 2006.

[TVM+03] Aphrodite Tsalgatidou, Jari Veijalainen, Jouni Markkula, Artem Katasonov, and Stathes Hadjiefthymiades. Mobile E-Commerce and Location-Based Services: Technology and Requirements. In Kirsi Virrantaus and Håvard Tveite, editors, *ScanGIS*, pages 1–14. Department of Surveying, Helsinki University of Technology, 2003.

[TvS03] Andrew Tanenbaum and Marten van Steen. *Verteilte Systeme - Grundlagen und Paradigmen*. Pearson Studium, 2003.

[viv] VivoMetrics. http://www.vivometrics.com - Abruf am 05.02.2008.

[VMG+01] Kirsi Virrantaus, Jouni Markkula, Artem Garmash, Vagan Y. Terziyan, Jari Veijalainen, Artem Katasonov, and Henry Tirri. Developing GIS-Supported Location-Based Services. In *WISE (2)*, pages 66–75, 2001.

[Von07] Helmut Vonhoegen. *Einstieg in XML*. Galileo Press, 2007.

[Wei91] Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, 1991.

[Wes70] Alan Furman Westin. *Privacy and Freedom*. Atheneum, New York, 1970.

[WHFG92] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The Active Badge Location System. *ACM Transactions on Information Systems*, 10(1):91–102, January 1992.

[WJ97] Andy Ward and Alan Jones. A New Location Technique for the Active Office, November 26 1997.

[WS06] Steve Wright and Alan Steventon. A Vision of Intelligent Spaces. In Alan Steventon and Steve Wright, editors, *Intelligent Spaces - The Application of Pervasive ICT*. Springer-Verlag, 2006.

[YAF06] Ansar-Ul-Haque Yasar, M. A. Ansari, and Sherjeel Farooqui. Low cost solution for location determination of mobile nodes in a wireless local area network. In Hiroshi Ishii, Newton Lee, Stéphane Natkin, and Katsuhide Tsushima, editors, *Advances in Computer Entertainment Technology*, page 75. ACM, 2006.

Glossar

API
Abkürzung für Application Programming Interface. Eine Schnittstelle, die von einem Softwaresystem anderen Programmen zur Anbindung an das System zur Verfügung gestellt wird.

Context Awareness
Bezeichnet das Verhalten von Anwendungsprogrammen, die Informationen über ihren Kontext, d. h. ihre Umgebung und ihren Zustand, benutzen, um ihr Verhalten darauf abzustimmen.

Entwurfsmuster
(engl. *Design Pattern*) Beschreibt eine bewährte Lösungs-Schablone für ein Entwurfsproblem in der Softwareentwicklung.

Geocoding
Ein Vorgang, bei dem Informationen, wie z. B. Texte, Bilder und Videos, mit geographischen Koordinaten verknüpft werden. Der Begriff wird auch für Dienste verwendet, die zu einem gesuchten Objekt die geographische Position ermitteln können.

GPRS
Abkürzung für *General Packet Radio Service*. Ein paketorientierter Datenübertragungsdienst, der im Bereich des Mobilfunks eingesetzt wird. Er baut auf dem *GSM*-Mobilfunknetz auf.

GPS
Abkürzung für *Global Positioning System*. Ein satellitengestütztes Positionierungssystem zur Bestimmung der geographischen Position von Objekten auf der Erde.

GSM
Abkürzung für *Global System for Mobile Communications*. Ein Standard für digitale Mobilfunknetze. Wird hauptsächlich für Telefonie, Datenübertragung und die Übertragung von Kurzmitteilungen (SMS) genutzt.

GUI
Abkürzung für *Graphical User Interface*. Eine Softwarekomponente, die dem Benutzer eines Computers die Interaktion mit der Maschine über grafische Symbole erlaubt.

Ortsbezogener Dienst
(engl. *Location Based Services*) Anwendungssystem, das die Position eines Nutzers oder Geräts kennt und sie verwendet, um Dienste ortsabhängig anzubieten.

OvI
Abkürzung für *Ort von Interesse*. Ein Begriff aus dem Bereich der Navigationssysteme und Routenplaner. Bezeichnet Adressen und Orte, die in einem Anwendungsszenario potentiell von Interesse für die Nutzer sein können.

PDA

Abkürzung für *Personal Digital Assistant*. Ein kompakter tragbarer Computer, der häufig für die persönliche Kalender-, Adress- und Aufgabenverwaltung benutzt wird, in der Regel aber auch andere Programme ausführen kann.

Positioning

Ein Positionierungsverfahren, bei dem die Position vom Benutzer selbst bestimmt wird. Es handelt sich um das Gegenstück zum *Tracking*.

RFID

Abkürzung für *Radio Frequency Identification*. Ein Verfahren zur automatischen Identifizierung von Objekten. Die Objekte werden mit elektronischen Tags, den sog. RFID-Tags, versehen. Diese können von entsprechenden Lesegeräten über Funk ausgelesen werden.

SDK

Abkürzung für *Software Development Kit*. Eine Sammlung von Programmen und Dokumentationen zu einer bestimmten Software, die es Software-Entwicklern ermöglichen soll, eigene darauf basierende Anwendungen zu erstellen.

SOAP

Ein Netzwerkprotokoll, mit dessen Hilfe Daten zwischen Systemen ausgetauscht und entfernte Methodenaufrufe durchgeführt werden können. Nutzt *XML* zur Repräsentation der Daten. Wird von *Webservices* zur Datenübertragung verwendet.

Symbian OS

Ein proprietäres Betriebssystem für Smartphones und *PDAs*, das von der Firma Symbian angeboten wird.

Tracking

Ein Positionierungsverfahren, bei dem die Position eines Nutzers von einem externen Netzwerk bestimmt wird. Es handelt sich um das Gegenstück zum *Positioning*.

Ubiquitäre Annotationen

(engl. *Ubiquitous Annotations*) Forschungszweig der Informatik im Bereich des *Ubiquitous Computing*. Überträgt die aus dem Alltag bekannte Metapher der Annotation in die Welt der Informationsverarbeitung. Befasst sich mit der Verknüpfung digitaler Informationen mit Objekten der Realwelt.

Ubiquitous Computing

Forschungszweig der Informatik, der sich mit der Allgegenwärtigkeit der Informationsverarbeitung im Alltag von Menschen befasst.

UML

Abkürzung für *Unified Modeling Language*. Eine standardisierte Sprache für die Modellierung von Software-Systemen.

UMTS

Abkürzung für *Universal Mobile Telecommunications System*. Ein Mobilfunkstandard der sog. dritten Generation (3G), mit dem deutlich höhere Datenübertragungsraten als mit dem *GSM*-Standard möglich sind.

Unit-Test

Ein Test-Verfahren zur Verifikation der Korrektheit von Modulen einer Software. Ermöglicht automatisierte und reproduzierbare Tests parallel zur entwickelten Software.

Webservice

Software-Dienste, die für die Kommunikation zwischen Computern konzipiert und über verschiedene Plattformen hinweg interoperabel sind.

XML

Abkürzung für *Extended Markup Language*. Eine Auszeichnungssprache, die textbasierte Dokumente beschreibt und die darin enthaltenen Informationen strukturiert.

XML-Schema

Eine Sprache zur Beschreibung eines XML-Typsystems. Ein XML-Typsystem spezifiziert, welche Elemente eine XML-Datei haben darf, und in welcher Reihenfolge die einzelnen Elemente aufgeführt sein dürfen.

A Anhang

A.1 XML-Schema der XML-Datei für die Persistenzschicht des mobilen Systems

In Abschnitt 4.5.1 wurde eine XML-Datei gezeigt, in der die Speicherung der Daten auf dem mobilen System stattfindet. Listing A.1 zeigt die zugehörige XML-Schema-Datei.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="TGModel">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Tours">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Tour">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Id" type="xs:string" />
                    <xs:element name="Name" type="xs:string" />
                    <xs:element name="Description" type="xs:string" />
                    <xs:element name="Map">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="Id" type="xs:string" />
                          <xs:element name="Name" type="xs:string" />
                          <xs:element name="Description" type="xs:string" />
                          <xs:element name="ImageFilename" type="xs:string" /
                              >
                          <xs:element name="MetersPerPixel" type="xs:decimal"
                               />
                          <xs:element name="LatitudeTop" type="xs:decimal" />
                          <xs:element name="LatitudeBottom" type="xs:decimal"
                               />
                          <xs:element name="LongitudeLeft" type="xs:decimal"
                              />
                          <xs:element name="LongitudeRight" type="xs:decimal"
                               />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                    <xs:element name="PoiList">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="Poi">
```

```
34                              <xs:complexType>
35                                <xs:sequence>
36                                  <xs:element name="Id" type="xs:string" />
37                                  <xs:element name="Name" type="xs:string" />
38                                  <xs:element name="Description" type="
                                        xs:string" />
39                                  <xs:element name="PosX" type="xs:decimal" />
40                                  <xs:element name="PosY" type="xs:decimal" />
41                                  <xs:element name="Visited" type="xs:boolean"
                                        />
42                                  <xs:element name="VisitedDateTime" type="
                                        xs:dateTime" />
43                                  <xs:element name="Radius" type="
                                        xs:unsignedByte" />
44                                  <xs:element name="ImageList">
45                                    <xs:complexType>
46                                      <xs:sequence>
47                                        <xs:element name="Image">
48                                          <xs:complexType>
49                                            <xs:sequence>
50                                              <xs:element name="Id" type="
                                                    xs:string" />
51                                              <xs:element name="FileName" type=
                                                    "xs:string" />
52                                              <xs:element name="Description"
                                                    type="xs:string" />
53                                            </xs:sequence>
54                                          </xs:complexType>
55                                        </xs:element>
56                                      </xs:sequence>
57                                    </xs:complexType>
58                                  </xs:element>
59                                  <xs:element name="OoiList" />
60                                </xs:sequence>
61                              </xs:complexType>
62                            </xs:element>
63                          </xs:sequence>
64                        </xs:complexType>
65                      </xs:element>
66                      <xs:element name="AnnotationList">
67                        <xs:complexType>
68                          <xs:sequence>
69                            <xs:element name="Annotation">
70                              <xs:complexType>
71                                <xs:sequence>
72                                  <xs:element name="Title" type="xs:string" />
73                                  <xs:element name="ImageFileName" type="
                                        xs:string" />
74                                  <xs:element name="PosX" type="xs:decimal" />
75                                  <xs:element name="PosY" type="xs:decimal" />
76                                  <xs:element name="DateTime" type="xs:dateTime
                                        " />
77                                </xs:sequence>
78                              </xs:complexType>
79                            </xs:element>
```

```
80                        </xs:sequence>
81                      </xs:complexType>
82                    </xs:element>
83                    <xs:element name="TourRecorder">
84                      <xs:complexType>
85                        <xs:sequence>
86                          <xs:element name="TourRecordList">
87                            <xs:complexType>
88                              <xs:sequence>
89                                <xs:element name="TourRecord">
90                                  <xs:complexType>
91                                    <xs:sequence>
92                                      <xs:element name="DateTime" type="
                                            xs:dateTime" />
93                                      <xs:element name="PosX" type="
                                            xs:decimal" />
94                                      <xs:element name="PosY" type="
                                            xs:decimal" />
95                                    </xs:sequence>
96                                  </xs:complexType>
97                                </xs:element>
98                              </xs:sequence>
99                            </xs:complexType>
100                         </xs:element>
101                       </xs:sequence>
102                     </xs:complexType>
103                   </xs:element>
104                 </xs:sequence>
105               </xs:complexType>
106             </xs:element>
107           </xs:sequence>
108         </xs:complexType>
109       </xs:element>
110       <xs:element name="CurrentTour" />
111       <xs:element name="NavigationOn" type="xs:boolean" />
112       <xs:element name="NotifyAlreadyVisitedPois" type="xs:boolean" />
113       <xs:element name="User" />
114     </xs:sequence>
115   </xs:complexType>
116 </xs:element>
117 </xs:schema>
```

Listing A.1: XML-Schema der XML-Datei für die Persistenzschicht des mobilen Systems

A.2 Inhalt der CD-ROM

Dieser Arbeit ist eine CD-ROM beigefügt. Sie beinhaltet eine PDF-Version dieses Dokuments und den gesamten Programm-Quellcode des stationären und mobilen Systems. Die CD verfügt über eine Autostart-Funktion. Nach dem Einlegen der CD ins CD-ROM-Laufwerk öffnet sich ein Web-Browser mit einer Übersichtsseite. Sollte dies einmal nicht der Fall sein, kann die Seite

manuell gestartet werden. Hierzu reicht das Öffnen der Datei *start.html* im Wurzelverzeichnis der CD-ROM.

Versicherung über die 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, 08. Februar 2008
Ort, Datum

Unterschrift