UNIVERSITÄT
KOBLENZ · LANDAU
FB4: Institut für Informatik

Location Provider

Diplomarbeit

zur Erlangung des Grades eines
Diplom-Informatikers
im Studiengang Informatik mit Anwendungsfach Wirtschaftsinformatik

vorgelegt von

Tobias Hebel

Erstgutachter: Prof. Dr. J. Felix Hampe, Institut für Informatik, FB4
Zweitgutachter: Dipl. Inform. Stefan Stein, Institut für Informatik, FB4

Koblenz, im November 2010

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

	Ja	Nein
Mit der Einstellung der Arbeit in die Bibliothek bin ich einverstanden	☐	☐
Der Veröffentlichung dieser Arbeit im Internet stimme ich zu	☐	☐

.....

(Ort, Datum) (Unterschrift)

Inhaltsverzeichnis

Kapitel 1

Überblick

Bei der Inanspruchnahme mobiler Datendienste kommt der aktuellen Position des Nutzers ein hoher Stellenwert zu. Denn mit dem Wissen über den genauen Aufenthaltsort des Dienstkonsumenten ist der Anbieter in der Lage, dem Nutzer Informationen und Dienstleistungen individueller und gezielter anbieten zu können. Die daraus entstehende höhere Dienstqualität bedeutet für den Anbieter einen wirtschaftlicher Vorteil und kann zu einem Alleinstellungsmerkmal gegenüber den Wettbewerbern ausgebaut werden.

Die Nutzung eines solchen Datendienstes ist mit einem aktuellen Smartphone möglich. Diese Geräteklasse eignet sich aufgrund der günstigen Hardwarevoraussetzungen und der hohen Verbreitung und Akzeptanz in der Zielgruppe besonders gut als Plattform für Dienstanbieter.

Die vorliegende Arbeit befasst sich mit der Entwicklung einer Verortungskomponente für mobile Endgeräte zur Nutzung mobiler ortsbezogener Datendienste. Sie ermittelt Positionsschätzungen für Nutzer solcher Geräte nicht nur mithilfe des GPS-Systems, sondern zusätzlich auf Basis der Daten, die die Geräte über die integrierten Sensoren aus ihrer unmittelbaren Umgebung empfangen.

Zusätzlich wird ein Communityansatz vorgestellt, der es ermöglicht, auch ohne vorherige Kartographierung in bis dahin unbekannten Arealen eine Positionsschätzung zu erhalten.

1.1 Motivation

Ortsbezogene Dienste sind Dienste, die über ein Datennetz genutzt werden und die zur Bereitstellung einer Funktionalität Nutzen aus der geographischen Position des Anwenders ziehen. Um im weiteren Verlauf der Ausarbeitung einzelne Teilaspekte einfacher erklären zu können, werden im folgenden einige Szenarien für die Nutzung eines mobilen Dienstes mit Ortsbezug dargestellt.

Dabei wird qualitativ auf Unterscheidungskriterien wie die Verortungsgenauigkeit oder die entstehenden Kosten eingegangen. Eine umfangreichere Betrachtung dieser Kriterien erfolgt in Kapitel 2.4.

1.1.1 Szenario 1 - mGeoWiki

Beim Projekt mGeoWiki [MGWI08] handelt es sich um ein Client-Server-System, mit dessen Hilfe ein Wiki von einer mobilen Plattform aus mit georeferenzierten Daten gefüllt oder schon bestehende Einträge editiert werden können. Im Mittelpunkt steht dabei die ubiquitäre Nutzung des Systems von mobilen Endgeräten wie Smartphones aus. Durch den für Wikis üblichen Communityansatz liegt die redaktionelle Aufbereitung der Artikel in der Verantwortung der Anwender selbst. Über geeignete Verortungsverfahren bestimmt der mGeoWiki-Client die Position jedes Nutzers und reichert den geschriebenen Artikel mit Geoinformationen an.

Durch die hohe Flexibilität des System sind verschiedenste Anwendungsgebiete denkbar, die unterschiedliche Anforderungen an die Verortungsgenauigkeit stellen.

1.1.2 Szenario 2 - Ausstellung

Aus aktuellem Anlass wird hier das Beispiel der Bundesgartenschau eingeführt. Die Gartenbauausstellung, die im Jahr 2011 in Koblenz statt finden wird, bietet umfangreiche Einsatzgebiete für ortsbezogene Dienste. So wird es viele Exponate geben, die für den Besucher einer Erklärung bedürfen. Vorstellbar wäre die Möglichkeit einer individuellen Führung anhand eines mobilen Endgeräts, die nur solche Exponate umfasst, die der Besucher vorher ausgewählt hat oder die seinen bekannten Interessen entsprechen.

Um die passenden Informationen zu dem Exponat darstellen zu können, dass der Besucher gerade betrachtet, muss das mobile Endgerät in der Lage sein, seine Position in einem definierten Genauigkeitsrahmen zu ermitteln. Dieser Bereich ist für ein Blumenarrangement in einem großen Beet erheblich viel größer, als für eine einzelnes Exemplar einer exotischen Pflanze, zu der Informationen dargestellt werden sollen. Man kann jedoch von Genauigkeitsanforderungen von unter fünf Metern ausgehen. Neben der Genauigkeit spielt bei diesem Anwendungsfall auch die Rechtzeitigkeit der Verortung eine große Rolle. Der Besucher der Ausstellung erwartet die Informationen, wenn er vor dem Exponat steht. Eine Verortung, die nicht in Echtzeit stattfindet, ist für den Besucher uninteressant.

Weiterhin muss die Positionsbestimmung in verschiedenen räumlichen Situationen möglich sein, wie im Inneren von Gebäuden oder in dichter Vegetation.

Verfügt der Betreiber der Ausstellung über ein solches System, wäre er in der Lage, jedem Besucher die Möglichkeit einzuräumen, sich eine individuelle Führung zusammenzustellen. Dies erhöht die Kundenzufriedenheit, da jedem Besucher nur noch die Exponate vorgeführt werden, die ihn interessieren und spart weiterhin Personal, da das mobile Endgerät alle relevanten Informationen bereit hält und diese gegebenenfalls mit eingehenderen Informationen aus Internetquellen verknüpfen kann.

Durch die für jeden Besucher optimierte Führung ließe sich weiterhin die mittlere Verweildauer in der Ausstellung pro Kunde verkürzen, was zu einer Optimierung der Auslastung der Veranstaltung beiträgt.

1.1.3 Szenario 3 - Wettervorhersage

Ein weiteres Anwendungsgebiet für Verortung sind Internetdienste, die Informationen mit Bezug zum Aufenthaltsort des Nutzers darstellen und dabei mit einer sehr geringen Verortungsgenauigkeit auskommen. Ein einfaches Beispiel hierfür ist ein Dienst, der dem Nutzer das aktuelle Wetter oder eine Wettervorhersage für den aktuellen Aufenthaltsort anzeigt. Ein Beispiel für einen solchen Dienst ist das Wetter-Widget der Smartphonebenutzeroberfläche HTC Sense (vergl. [HTC10]).

Für diesen Dienst reicht eine Verortungsgenauigkeit von mehreren Kilometern, da Wetterphänomene meist großflächig auftreten. Da es sich weiterhin um

einen Massendienst handelt, liegt es im Interesse des Betreibers, die Nutzungskosten so niedrig wie möglich zu halten und den Dienst so vielen Nutzern wie möglich zur Verfügung zu stellen. Für den Betreiber liegt also die Notwendigkeit einer möglichst umfangreichen Unterstützung verschiedener Verortungstechnologien auf der Hand. Insbesondere bieten sich hier auch solche Technologien an, die nur vergleichsweise geringe Verortungsgenauigkeiten erreichen, jedoch weit verbreitete Hardware benutzen.

1.1.4 Szenario 4 - Dienstintegration

Das letzte Beispiel ist eine Architektur bestehend aus mehreren Diensten, wie sie im Projekt Aloqa zusammengefasst sind (vergl. [ALOQ]). Denkbar ist eine Kombination aus einem Angebot, über das ein Restaurant in der Nähe des eigenen Standortes gefunden werden kann. Das Restaurant wird dabei über einen Kriterienkatalog, den der Nutzer vorher festgelegt hat und der die eigenen Vorlieben beschreibt, möglichst passend ausgewählt. Dem Nutzer wird weiterhin sofort eine Speisekarte mit Onlinebuchung zur Verfügung gestellt und eine Platzreservierung ermöglicht. Im nächsten Schritt berechnet ein weiterer Dienst den kürzesten Weg zur Gaststätte und führt den Nutzer interaktiv dort hin.
Nach dem Essen bezahlt der Gast online seine Rechnung über einen Bezahldienst und bewertet das Restaurant in einem Portal, so dass mit steigender Teilnehmerzahl immer genauere Empfehlungen möglich werden.

Der Anreiz zur Teilnahme an einem solchen System ist für den Gaststättenbesitzer die Möglichkeit, einen neuen Benutzerkreis anzusprechen und somit seinen Kundenstamm zu erweitern. Die Abwicklung der monetären Transaktion online spart zudem Personal ein. Der Finanzdienstleister hat ähnliche Beweggründe. Bargeldlose Transaktionen sparen Automaten und Personal. Außerdem ist eine Provision für jede Transaktion denkbar.

Die Bewertungsportale, die in diesem Beispiel vorkommen, verdienen Ihr Geld über personalisierte Werbung oder mit Provisionen für erfolgreiche Vermittlungen von Gästen.

Dieses Beispiel verdeutlicht, dass nicht alle Dienste in einer solchen Architektur gleich genaue Positionsdaten des Nutzers benötigen. Eventuell existieren sogar

solche, die diese Daten überhaupt nicht benötigen, wie in diesem Beispiel das Bewertungsportal, das der Benutzer nach dem Besuch nutzt. Eventuell möchte der Benutzer aus mangelndem Vertrauen heraus sogar verhindern, dass bestimmte Dienste die eigene Position überhaupt oder zumindest zu genau kennen. Hierfür wird die Möglichkeit benötigt, die Ermittlung der eigenen Position unterdrücken zu können.

Für die oben genannten Anwendungsfälle existieren technologische Insellösungen, die jeweils einen Teil der aufgezählten Anforderungen erfüllen. Die meisten sind jedoch aufwändig in der Wartung, da in bisher unbekannten Gebieten vor der ersten Begehung durch Nutzer zuerst alle Referenzstationen kartographiert werden müssen. Ein Ansatz, diesen Aufwand zu minimieren wäre es, die Kartographierung in einem Communityprojekt zu organisieren und die Nutzer selbst diese Aufgabe übernehmen zu lassen. Ein weiterer wichtiger Ansatzpunkt dieser Arbeit ist die Kontrolle, die der Nutzer über seine persönlichen Daten hat. In den wenigsten Fällen kann er kontrollieren, an wen genau seine Position weitergegeben wird und welche Daten zu deren Ermittlung herangezogen werden.
Diese Arbeit schlägt ein Verortungssystem vor, das bezüglich der verwendeten Referenzstationen lernfähig ist, eine modulare Auswahl der verwendeten Verortungstechnologien erlaubt und dem Nutzer Kontrolle darüber gibt, welche Daten weitergegeben werden sollen und welche Daten überhaupt zur Verortung verwendet werden sollen.

1.2 Thematische Einordnung

Für ortsbezogene Dienste, wie sie in den Beispielszenarien in Kapitel 1.1 vorgestellt wurden, muss der Dienstanbieter Informationen über den Standort des Nutzers haben. Anhand dieser Informationen kann er den bereit gestellten Dienst mit ortsbezogenen Informationen anreichern. Dadurch erhält der Nutzer eine individuell auf seine Anforderungen angepasste Dienstleistung. Da es sich bei der anforderungsgerechten Versorgung des Nutzers mit Informationen um einen Wett-

bewerbsvorteil handelt, liegt eine möglichst genaue Kenntnis über dessen Aufenthaltsort im Interesse des Dienstanbieters.

Dagegen ist aus der Sicht des Nutzers die Datenminimalität ein wichtiges Bedürfnis. Das bedeutet, dass er nur die personenbezogenen Daten preisgeben muss, die zur Erbringung des Dienstes absolut notwendig sind. Weiterhin muss der Benutzer die direkte Kontrolle darüber haben, welche personenbezogenen Daten an welche Instanz weitergegeben werden dürfen.

Unter diesen Voraussetzungen, einerseits der Notwendigkeit, möglichst genaue Standortdaten eines Nutzers zu ermitteln andererseits Datenminimalität zu wahren und dem Nutzer die Kontrolle über die Weitergabe seiner personenbezogenen Informationen zu geben, wird in dieser Arbeit eine Verortungskomponente entwickelt. Diese bildet mit ihren Komponenten mehrere Instanzen der in Kapitel 2.1 näher beschriebenen Dienstarchitektur zur Realisierung kontextbezogener Dienste nach [STEI10].

1.3 Fragestellung

Wie aus den vorhergehenden Kapiteln hervorgeht, besteht ein großer Bedarf an Verortungstechnologien, um Gewinn versprechende Geschäftsfelder im Bereich der ortsbezogenen mobilen Dienstleistungen abdecken zu können.

Ein spezieller Fokus liegt auf der Einbindung von Mobiltelefonen und PDAs[1] in bestehende und neue Geschäftskonzepte. Denn diese kleinen Geräte sind in einem höheren Grad mobil als es bei Netbooks und Laptops der Fall ist. Weiterhin ist der Verbraucher inzwischen an die Verwendung der Geräte im Alltag gewöhnt und empfindet die ständige Präsenz der Geräte aus eigenem Antrieb als Notwendigkeit. Dadurch verfügen Mobiltelefone und PDAs aufgrund des persönlichen Interesses des Verbrauchers über einen hohen Grad der Personenbindung, was sie für Dienstanbieter interessant macht.

In dieser Arbeit wird ein Verortngssystem für Mobiltelefone und PDAs entwickelt werden, das so viele von den Geräten messbare Signale wie möglich zur Positionsbestimmung verwendet. Da die genannten Geräte in ihrer technischen Leistungsfähigkeit stark eingeschränkt sind, ist zu untersuchen, ob die für eine

[1] Personal Digital Assistant

Verortung notwendigen Berechnungen auf ein leistungsfähigeres System ausgelagert werden können.

Die Schätzung der Position eines mobilen Systems erfordert Referenzpunkte. Diesen müssen in jedem bisher unbekannten Gebiet mühsam kartographiert werden, bevor dort eine Verortung möglich ist. Wie das Beispiel [FAZ2010] zeigt, ist dieses Vorgehen aufwändig und birgt rechtliche Risiken. Es ist zu untersuchen, ob es andere Möglichkeiten der Kartographierung gibt. Ein Communityansatz erscheint in diesem Zusammenhang als vielversprechend.

Weiterhin ist zu prüfen, ob die anhand der registrierten Sensordaten ermittelten Positionsschätzungen eine ausreichende Genauigkeit aufweisen. Sollte dies nicht der Fall sein, können probabilistische Filter verwendet werden, um eine genauere Verortung zu erreichen. In diesem Fall wäre ein entsprechender Filter zu implementieren und zu überprüfen, ob dadurch eine Verbesserung der Schätzung erreicht werden kann.

Um die Software möglichst flexibel einsetzen zu können, sollte die Implementierung über Schnittstellen verfügen, mit deren Hilfe das System in bestehende und neue Komponenten integriert werden kann, die eine Positionsschätzung benötigen. Welche Technologien sich hier anbieten, wird ebenfalls in den folgenden Kapiteln erarbeitet werden.

1.4 Vorgehensweise und weiterer Aufbau der Arbeit

Um die in Kapitel 1.3 definierten Fragen zu untersuchen, wird in dieser Ausarbeitung ein Prototyp entwickelt, an dem die formulierten Fragestellungen anhand von Versuchen überprüft werden können. Dazu ist folgendes Vorgehen notwendig:

Das Kapitel 2 wird eingehend die zur Beantwortung der Fragestellung notwendigen Grundlagen beleuchten. Dort werden verschiedene Verortungsverfahren, Signalquellen, Störeinflüsse, aber auch Datenformate vorgestellt, die bei der Realisierung des Prototyps eine Rolle spielen. Weiterhin enthält das Kapitel theoretische Erklärungen der Funktionsweise von Filtermechanismen, die die Positionsschätzung verbessern können. Das Unterkapitel 2.10 beschreibt Sicherheits-

konzepte und Sicherheitsmaßnahmen, die die Übertragung persönlicher Daten im Rahmen dieser Ausarbeitung schützen sollen. Da keine Verortungstechnologie ohne Koordinatensysteme auskommt, enthält Kapitel 2 auch die Beschreibung der in Deutschland weit verbreiteten Systeme.

Kapitel 3 erstellt nach einer Ist-Analyse anhand der vorher vorgestellten Grundlagen basierend auf der Fragestellung ein Soll-Konzept für die Implementierung. Hierbei werden sowohl die Verortungstechnologien, die im Prototyp verwendet werden sollen, ausgewählt als auch die Softwarearchitektur und das Kommunikationsprotokoll.

Nachdem die Grundlagen erklärt und die richtigen Verfahren ausgewählt sind, befasst sich das Kapitel 4 mit der Implementierung des Prototypen. Dabei werden als Erstes sowohl der Client als auch der Server mit ihren jeweiligen Modulen ausführlich beschrieben. Das Eventsystem, über das die Module sowohl von Client als auch von Server Nachrichten austauschen, wird ebenfalls einzeln betrachtet. Weiterhin enthält das Kapitel Beschreibungen der verwendeten Kommunikations– und Sicherheitstechnologien und schließt mit der Erklärung des Verfahrens, wie mit unbekannten Referenzstationen umgegangen wird.

Die oben definierten Fragestellungen werden an einem Prototypen exemplarisch untersucht. Hierzu enthält Kapitel 5 Beschreibungen der unternommenen Versuche, mit deren Hilfe einzelne Komponenten und das Gesamtsystem evaluiert wurden und die unternommenen Anstrengungen, um die definierten Fragen zu beantworten.

Das letzte Kapitel dieser Ausarbeitung enthält mit dem Fazit eine Zusammenfassung der unternommenen Anstrengungen zur Beantwortung der im Kapitel Fragestellung definierten Fragen und die Ergebnisse der Arbeit. Ein Ausblick, in dem verschiedene Weiterentwicklungsmöglichkeiten vorgeschlagen werden, rundet die Arbeit ab.

Kapitel 2

Grundlagen

In diesem Kapitel werden die theoretischen und technischen Grundlagen dieser Arbeit betrachtet. Dazu enthält Kapitel 2.1 eine Beschreibung der in [STEI10] entwickelten LBS-Architektur, zu deren Instanzen auch der in dieser Arbeit entwickelte Location-Provider und seine Bestandteile gehören. Um im Folgenden die Verortungstechniken besser beschreiben zu können, führt Kapitel 2.2 eine Taxonomie ein, die die einzelnen Technologien in ein Schema einordnet. Abschnitt 2.3 beschreibt die verschiedenen Möglichkeiten, aus der Anwesenheit von Signalen eine Positionsschätzung zu erzeugen.

In den im vorherigen Kapitel aufgezeigten Szenarien wurden verschiedene Qualitätskriterien für Verortungslösungen angesprochen. Diese werden in Kapitel 2.4 eingehend beschrieben.

Wenn die Grundlagen erklärt sind, wendet sich die Arbeit den verfügbaren Signalquellen zu. In diesem Abschnitt werden die in der Realität verfügbaren und mit handelsüblicher und verbreiteter Hardware messbaren Signale mit ihren Vor- und Nachteilen beschrieben.

Nachdem Kapitel 2.6 die Störquellen beschrieben hat, die bei Positionsschätzungen auftreten können, erklärt der folgende Abschnitt Möglichkeiten, die Genauigkeit einer Positionsschätzung zu verbessern. Hier werden sowohl technologiespezifische Verfahren angeführt, also solche, die wie dGPS speziell für eine Verortungstechnologie entwickelt wurden, als auch generelle Verfahren, die auf eine Positionsschätzung aus einer beliebigen Quelle angewendet werden können.

Anschließend beschreibt Kapitel 2.8 eine Auswahl von Koordinatensystemen, die zur Verwendung im Prototypen in Frage kommen. Der darauf folgende Abschnitt enthält Erklärungen zu einigen Datenformaten, die bei der Erstellung des Prototypen von Bedeutung sind.

Der letzte Abschnitt des Kapitel befasst sich mit Sicherheitsanforderungen und Mitteln, um diese Anforderungen zu erfüllen.

2.1 LBS-Architektur

Das in der vorliegenden Arbeit konzeptionierte und implementierte System bildet eine Instanz der in [STEI10] entwickelten Architektur zur Realisierung ortsbezogener Dienste. Abbildung 2.1 zeigt eine Übersicht über die Instanzen der Architektur und deren Zusammenhang. Ziel der Dienstarchitektur ist es, dem Benutzer

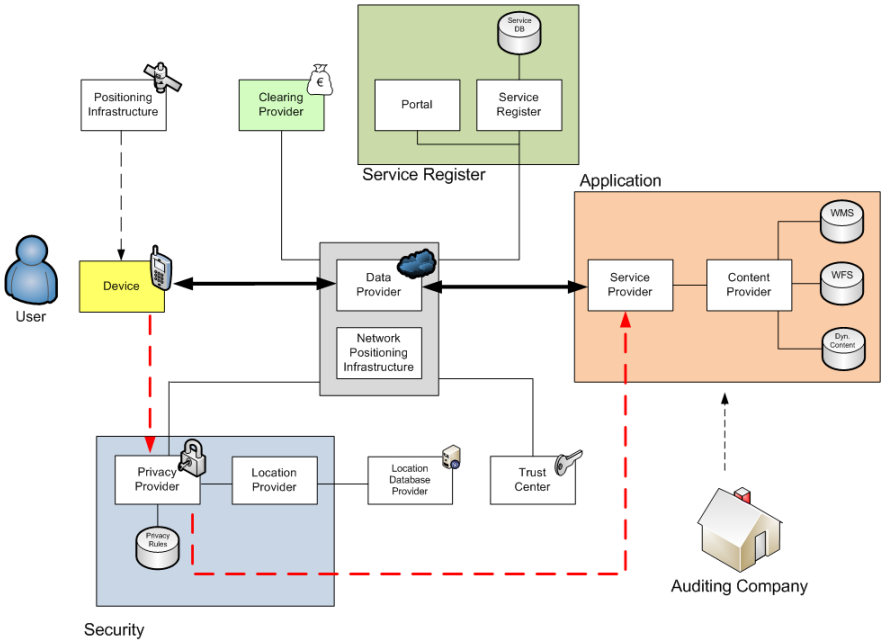

Abbildung 2.1: Die in [STEI08] entworfene Metaarchitektur, deren Lokalisationskomponente diese Arbeit bildet

die Inanspruchnahme verschiedenster mobiler Dienste zu ermöglichen und ihm gleichzeitig die größtmögliche Kontrolle über seine personenbezogenen Daten wie seinen Standort, seine persönlichen Präferenzen oder seine Kontodaten zu geben. Dazu besteht die Architektur aus verschiedenen, Instanz genannten Diensten, die

explizit von unterschiedlichen Dienstanbietern betrieben werden können. Jede Instanz kann mehrmals in verschiedenen Ausprägungen, beispielsweise von verschiedenen Anbietern existieren, so dass der Benutzer den Dienstanbieter wählen kann, dessen Leistungen ihm am ehesten zusagen. Im Folgenden werden die einzelnen Instanzen oder Instanzgruppen kurz dargestellt:

Das fakultativ mobile Endgerät (**Device**) hat in der Architektur mehrere Aufgaben. Zum Einen bildet es die Schnittstelle, über die der Benutzer Dienste konsumiert. Zum Anderen ist es mit Sensoren ausgestattet, die Kontextinformationen aus der Umgebung registrieren. Anhand dieser Kontextinformationen, bei denen es sich um Funksignale von Datenübertragungsnetzen oder GPS-Signale handeln kann, wird die Position des Nutzers bestimmt.

In den meisten Fällen handelt es sich beim Endgerät um mobile Systeme wie Smartphones, PDAs oder Netbooks. Aber auch stationäre Systeme können die kontextbezogenen Dienste nutzen, die über die Architektur angeboten werden.

Der **Data Provider** bildet das zentrale Bindeglied zwischen den Instanzen der Architektur, über den die Datenübertragung abgewickelt wird. Es handelt sich dabei zum Beispiel um einen Mobilfunkanbieter oder Internetprovider, der über ein gemeinsames Netzwerk mit den anderen Instanzen der Architektur in Verbindung steht. Da die Kommunikation über ein Netz stattfindet, das unter fremder administrativer Kontrolle steht, ist es erforderlich, dass die Daten verschlüsselt übertragen werden, um ihre Vertraulichkeit zu gewährleisten.

Beim **Privacy Provider** handelt es sich um die Kerninstanz der Architektur, da über ihn jede Kommunikation zwischen dem Endgerät und den am Dienstleistungsprozess beteiligten Instanzen abläuft. Der Privacy Provider stellt sicher, dass jede Instanz nur die Informationen erhält, die für die Bereitstellung ihres Dienstangebots notwendig sind und sorgt so für Datenminimalität. Welche Informationen weitergegeben werden dürfen, legt der Benutzer durch Regelsätze fest, die permanent gespeichert werden. Außerdem kann der Privacy Provider weitere Kontextinformationen zu Benutzern vorhalten, die ebenfalls basierend auf Regelsätzen an berechtigte Instanzen weitergegeben werden können.

Da der Privacy Provider bei jeder Kommunikation zwischen dem Benutzer und dem Dienst vermittelt, ist es in der Architektur auch möglich, Dienste anonym zu verwenden. Sind für die Nutzung eines Dienstes monetäre Transaktionen notwendig, leitet der Privacy Provider diese ebenfalls notfalls anonym ein.

Die Dienstbereitstellung erfolgt durch eine Menge von Instanzen des Typs **Service Provider**. Entsprechend des Bedarfs an zur Diensterbringung benötigten Daten und der Regelsätze, die der Benutzer im Privacy Provider definiert hat, stellt dieser dem Dienst Kontextinformationen über den Benutzer zur Verfügung. Anhand dieser individuellen Daten findet die Leistungserbringung des Dienstes statt, die über im Internet übliche Technologien zur Datenübertragung und Darstellung abgewickelt wird.

Bei den Diensten kann es sich um Angebote für den Massenmarkt wie beispielsweise kontextsensitive Stadtführungen, aber auch um Angebote handeln, die auf spezielle Personengruppen zugeschnitten sind.

Handelt es sich bei dem vom Service Provider bereitgestellten Angebot um einen komplexeren Dienst, muss der Anbieter eventuell Daten von einer weiteren Instanz, dem **Content Provider** beschaffen. Während die Kernkompetenz des Service Providers in der Darstellung und Verteilung der Inhalte besteht, betrifft die des Content Providers die Erstellung der Inhalte. Durch die Konzentration auf die jeweilige Kernkompetenz ist es den beiden Instanzen möglich, dem Kunden die jeweils bestmögliche Leistung zu bieten.

Ein Beispiel für die Aufteilung in Content- und Service Provider ist ein Nachrichtenportal im Internet wie Google News[1]. Während sich der Portalbetreiber, in diesem Fall Google, auf die Darstellung der Daten und deren Vermarktung konzentriert, ist die Erstellung der Artikel und deren journalistische Aufarbeitung Aufgabe einer Presseagentur.

Das **Service Portal** ermöglicht die Suche nach Diensten in der Architektur. Da der Benutzer in vielen Fällen gerade Dienste in seiner Umgebung finden möchte, kann das Service Portal Kontextinformationen des Anwenders nutzen,

[1] http://news.google.de/

um solche Dienste vorrangig anzuzeigen. Das Service Portal ist eine reine Darstellungskomponente für die Visualisierung der Daten, die von den Servern des Service Registers zur Verfügung gestellt werden.

Mit Hilfe des **Service Registers** kann ermittelt werden, welche Dienste in einem bestimmten geographischen Umfeld angeboten werden. Dazu besteht das Register aus verteilten Servern, die eine geographische Hierarchie abbilden.

Sucht ein Benutzer einen Dienst, richtet sein Endgerät oder ein Service Portal eine Anfrage an einen der Server des Service Register. Damit eine solche Anfrage erfolgreich ist, muss jeder Dienst von dem Server gelistet werden, der für seine geographische Ausbreitung zuständig ist. Anfragen enthalten im Fall eine ortsbezogenen Suche Angaben über den geographischen Bereich, in dem nach einem Dienst gesucht werden soll. Zusätzlich sind auch weitere Suchattribute möglich.

Der **Clearing Provider** wickelt alle monetären Transaktionen zwischen Instanzen der Architektur ab. Unter der Annahme, dass ein personalisierter Dienst kostenpflichtig angeboten wird und dass die Nutzungsentgelte im Centbereich liegen, ist zu erwarten, dass es bei der Dienstnutzung zu vielen Transaktionen mit niedrigen Beträgen kommen muss. Da die Transaktionskosten bei solchen Micropayment Transaktionen die Kosten der Dienstnutzung übersteigen würden, muss der Clearing Provider die Möglichkeit bieten, Zahlungen über einen bestimmten Zeitraum oder bis zu einem Schwellenbetrag zu aggregieren. Erst beim Erreichen der gesetzten Frist oder des definierten Maximalbetrags wird eine Zahlung ausgelöst.

Um zu jedem Zeitpunkt Gewissheit über die Identität des Transaktionspartners zu haben, werden Zertifikate und eine Public Key Infrastructure (vergl. Kapitel 2.10.5). Die Zertifikate werden von einem **Trust Center** ausgestellt und vor jeder Kommunikation von den beteiligten Instanzen verifiziert. Dazu betreibt das Trust Center eine Datenbank, die Informationen über die Gültigkeit der ausgegebenen Zertifikate enthält.

Die Architektur erlaubt es jeder Person oder Institution, Dienste anzubieten. Diese Dienste können vom Ausland aus operieren und unterstehen somit einer anderen Jurisdiktion als der Benutzer. Dies stellt gerade beim Datenschutz ein Problem dar, da die Gesetzgebung anderer Länder oftmals einen sehr unkontrollierten Umgang mit personenbezogenen Daten seitens der Dienstanbieter ermöglicht.

Um einheitliche Standards beim Umgang mit personenbezogenen Daten und bei der Erbringung der bestellten Dienstleistung zu gewährleisten, ist es möglich, einen Dienst durch ein unabhängiges **Auditing Unternehmen** nach vorher definierten Kriterien zertifizieren zu lassen. Eine solche Zertifizierung ist eine vertrauensbildende Maßnahme und kann somit auch zu einem Popularitätsgewinn für den Dienst führen.

Der **Location Provider**, der in dieser Arbeit entwickelt wird, ermittelt die Position des Benutzers. Er verwendet dazu Sensordaten, die vom Endgerät zur Verfügung gestellt werden. Diese stammen aus verschiedenen Quellen. Netzwerke, deren ursprüngliche Funktion die Verortung mobiler Systeme ist, werden als **Positioning Infrastructure** bezeichnet. Hierzu zählt das GPS-System. Andere Technologien zur Verortung, die zum Beispiel auf den vorhandenen Datenübertragungsnetzen basieren werden in der Architektur als **Network Positioning Infrastructure** bezeichnet. Um anhand der Network Positioning Infrastructure eine Positionsbestimmung erreichen zu können, sind weitere Daten wie die geographische Zuordnung bestimmter Funkzellen notwendig. Diese Daten hält der **Location Database Provider** in einer Datenbank vor.

Zur Abgrenzung der hier vorgestellten Dienstarchitektur von der Softwarearchitektur des Prototyps wird in dieser Arbeit die Dienstarchitektur nach [STEI10] als Metaarchitektur bezeichnet.

2.2 Taxonomie

Aufgrund der Vielzahl verschiedener Verortungslösungen muss diese Arbeit auf der im Folgenden vorgestellten Taxonomie basieren und die eingeführten Begriffe

verwenden. Als Taxonomie wird hier die Ordnung der in dieser Arbeit relevanten Verortungstechnologien anhand der nachfolgend vorgestellten Kriterien verstanden.

Nach [ROTH2005] können Verortungslösungen wie in der Grafik 2.2 dargestellt in drei grundsätzliche Bereiche unterteilt werden. Satellitengestützte Navi-

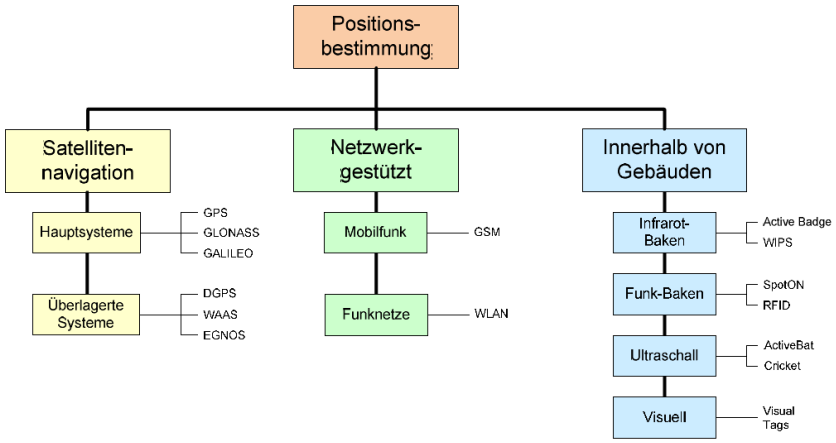

Abbildung 2.2: Lokalisierungstechnologien (nach [ROTH2005])

gationslösungen erhalten ihre Signale von Satelliten. Als Beispiel für eine solche Technologie wird in Kapitel 2.5.1 das GPS-System eingeführt. Kapitel 2.7 behandelt dGPS und NTRIP als überlagernde Systeme. Überlagernd bedeutet, dass die Hauptsignale weiterhin von einem Navigationssatelliten stammen, aber zusätzlich über andere Kanäle verfügbar gemachte Korrekturdaten zur Verbesserung der Positionsschätzung verwendet werden. Bei Satellitengestützter Navigation handelt es sich um terminalbasierte Positionsschätzungen. Das bedeutet, dass die Schätzung der Position im Endgerät stattfindet.

Netzwerkgestützte Technologien zur Verortung mobiler Systeme ermitteln die Position anhand von Daten, die in einem Netzwerk vorliegen. Als Beispiele dienen hier WLAN in Kapitel 2.5.2 und GSM in Kapitel 2.5.5. Zu unterscheiden sind hier terminalbasierte Navigation und netzwerkbasierte Navigation. Bei der netzwerkbasierten Navigation ermittelt ein spezieller Dienst innerhalb des Netzwerks die Position der verbundenen Clients. Aufgrund des direkten Zugriffs auf alle verfügbaren Informationen über den jeweiligen Client und die genauen Posi-

tionen der Referenzstationen ist eine netzwerkbasierte Verortung im Bereich der netzwerkgestützten Navigation präziser.

Innerhalb von Gebäuden lassen sich verschiedene weitere Technologien identifizieren, die für eine Verortung mobiler Systeme verwendet werden können. Hierzu zählen die in Kapitel 2.5.3 beschriebenen Nachbereichsfunksysteme Bluetooth und ZigBEE, aber auch visuelle Markierungen, wie sie in Kapitel 2.5.8 vorgestellt werden.

2.3 Verortungstechniken

Bei der Bestimmung des Aufenthaltsorts eines Objekts im dreidimensionalen Raum gibt es verschiedene Möglichkeiten, eine Positionsschätzung zu ermitteln. Alle hier vorgestellten Verfahren bedienen sich dabei vorher vermessener Referenzpunkte, deren Positionen hinreichend genau bekannt sind und von denen aus die Position des Objekts im Raum berechnet werden kann. Wie viele solcher Referenzpunkte nötig sind, hängt vom jeweiligen Verfahren ab – es sind jedoch zumeist mindestens drei. Denn eine Verortung findet zumeist in einem dreidimensionalen Raum anhand der drei Größen $\binom{x}{y}{z}$ statt. Abstrakt spricht man von einem Vektorraum der Dimension 3. Es sind also drei Basisvektoren notwendig, um jeden Punkt im Raum eindeutig beschreiben zu können.

Die hier präsentierten Formeln stammen aus [NEBE97], die Einteilung der Technologien aus [LANG06] und [ROTH2005] und können dort in wesentlich ausführlicherer Form eingesehen werden.

2.3.1 Proximation

Die Proximation (von lat.: proximus, -a, -um = der, die, das Nächste) ist ein sehr einfaches Verfahren zur Positionsschätzung. Hier schließt der Positionsschätzer aus der Wahrnehmbarkeit eines Referenzpunktes, dass dieser sich nahe zur eigenen Position befinden muss. Aus dem Wissen um die Position des Referenzpunktes, welches zum Beispiel aus einer Karte oder Datenbank bezogen werden kann, leitet der Schätzer seine Position ab. Üblicherweise gibt er die Position des Referenzpunktes als seine eigene an und berechnet anhand einer Entfernungs-

schätzung, die auf der gemessenen Signalabschwächung oder dem Wissen über die Reichweite des Signals basieren könnte, eine mittlere Abweichung der Positionsschätzung zur wahren Position.

Befindet sich das zu verortende Objekt in „Sichtweite“ mehrerer Referenzpunkte, kann ein zu allen Referenzpunkten äquidistanter Punkt als eigene Position angenommen werden.

2.3.2 Lateration

Bei der Lateration (von lat.: latitudo, -inis, f: Länge, Strecke) wird die Entfernung zu einem oder mehreren Bezugspunkten, deren Position bekannt ist, zur eigenen Verortung verwendet. Ist der Abstand d_1 zwischen einem Objekt und einem Punkt M_1 bekannt, so befindet sich der Beobachter auf der Oberfläche der Kugel mit dem Radius $r = d_1$, die durch

$$(\vec{x} - \overrightarrow{m_1})^2 = r^2$$

beschrieben wird. Ist ein weiterer Punkt M_2 und dessen Abstand zum Objekt d_2 bekannt, befindet sich das Objekt auf dem Schnittkreis, den die beiden Kugeloberflächen bilden. Eine dritte Abstandsmessung d_3 zu einem weiteren Referenzpunkt M_3 schränkt die Position des Objektes auf zwei Punkte ein, an denen sich jeweils alle drei Kugeloberflächen schneiden.

Mit diesen Angaben ist eine Positionsschätzung prinzipiell möglich, da einer der beiden Punkte aus Plausibilitätsgründen ausgeschlossen werden kann. Das GPS System beispielsweise verwendet Lateration zur Verortung. Von den beiden auf die gerade beschriebene Weise ermittelten Punkten liegt einer auf oder in der Nähe der Erdoberfläche, während sich der zweite tief im Weltraum befindet. Eine vierte Messung d_4 zu einem Referenzpunkt M_4 gibt im Zweifelsfall Gewissheit.Abbildung 2.3 zeigt die beim GPS System verwendete Lateration, bei der Satelliten als Referenzpunkte verwendet werden.

Die bei der Lateration benötigte Entfernung d wird elektronisch meist anhand einer Signallaufzeit berechnet. Hierzu wird entweder aktiv ein Signal, zum Beispiel eine elektromagnetische Welle ausgesandt, die vom Referenzpunkt reflektiert wird, oder der Referenzpunkt sendet periodisch ein Erkennungssignal.

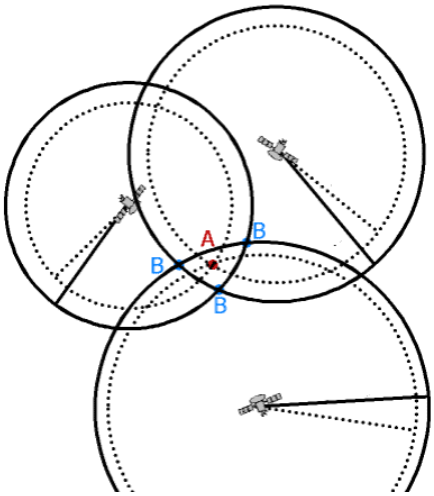

Abbildung 2.3: Lateration mit drei Referenzpunkten am Beispiel der Positionsbestimmung mit GPS (Quelle [NMEA2])

Technisch umsetzbar ist die Lateration, indem ein Signal verwendet wird, das sich in Abhängigkeit von der Zeit ändert. Dies erfordert eine genaue Synchronisation zwischen Sender und Empfänger. Weiterhin dürfen sich mehrere gleichzeitige Signale bei Überlagerung nicht gegenseitig unkenntlich machen. GPS verwendet hierzu Code Division Multiplexing (siehe Kapitel 2.5.1).

2.3.3 Angulation

Die Angulation (von lat.: angulus, -i, m = Winkel) ist eine historisch sehr bedeutsame Navigationsform. Indem man die Winkel zwischen Geraden durch verschiedene Himmelskörper und den eigenen Standort misst, bestimmt man die eigene Position, ohne auf terrestrische Referenzpunkte zurückgreifen zu müssen. Da viele Himmelskörper scheinbar ortsfest immer an der gleichen Position am Himmel erscheinen und von den meisten Orten aus sichtbar sind, genügt bei der Astronavigation eine vergleichsweise geringe Datenmenge für eine Positionsschätzung.

Doch auch bei der Verortung mit terrestrischen Referenzpunkten kann die Angulation eingesetzt werden. Geht man davon aus, dass sich der eigene Aufenthaltsort auf dem Erdboden befindet, ist eine einfache zeichnerische Positionsschätzung möglich. Hierzu wird mit einem Kompass die relative Richtung zweier Referenzpunkte vom eigenen Standort aus bestimmt. Die eigene Position liegt

nun am Schnittpunkt der Geraden durch jeweils einen der Referenzpunkte im jeweils gemessenen Winkel.

Aus historischer Sicht war es ohne elektronische Mittel einfacher, mit einem optischen Instrument einen Winkel zu bestimmen, als eine Entfernung. Im Gegensatz dazu braucht man, um einen Winkel elektronisch zu bestimmen eine rotierende Antenne. Trifft ein Signal auf, ermittelt man die Richtung, in die die Antenne zum Zeitpunkt des Empfangs gerichtet war und berechnet daraus den Winkel, aus dem das Signal auftraf.

Ein neuerer Ansatz zur Winkelbestimmung ist der Einsatz vieler kleiner Signalrezeptoren als Array. Trifft ein Signal auf, berechnet das Array anhand der Laufzeitunterschiede der elektromagnetischen Wellen, aus denen das Signal besteht und deren Interferenzen die Richtung der Signalquelle.

2.3.4 Koppelnavigation

Die englischsprachige Bezeichnung für Koppelnavigation lautet „dead reckoning“ (etwa leere Berechnung). Dabei handelt es sich im Gegensatz zu den bis jetzt vorgestellten Techniken nicht um ein Verfahren, das durch eine einzelne Messung eine Positionsschätzung erstellt, sondern um eine kontinuierliche Reihe diskreter Messungen, die in ihrer Gesamtheit eine Positionsschätzung ergeben. Der englische Begriff „dead“ weist auf eine weitere Besonderheit der Koppelnavigation hin: Außer einem Ausgangspunkt sind hier keine weiteren Referenzpunkte nötig, sondern die Positionsschätzung wird ausschließlich anhand lokal verfügbarer Werte generiert.

Gemessen wird üblicherweise Kurs, Geschwindigkeit und die vergangene Zeit. In den Intervallen zwischen den Messungen geht man von gleichbleibenden Werten aus. Diese kann man dann rechnerisch oder zeichnerisch ausgehend von einem Startpunkt, dessen Koordinaten bekannt sind, aneinander reihen, um die eigene Position zu ermitteln.

Die Koppelnavigation ist vergleichsweise ungenau, da sich Messfehler oder Verfahrensungenauigkeiten aus einem Schritt in alle weiteren Schritte übertragen. Eine solche Ungenauigkeit ist die Geschwindigkeitsmessung relativ zum umgebenden Medium. Bewegt sich ein Flugzeug durch das Medium Luft, kann an Bord

des Flugzeugs zwar ein Kurs und eine Geschwindigkeit bestimmt werden, diese sind jedoch nur relativ zum umgebenden Luftstrom gültig. Die absoluten Werte für Kurs und Geschwindigkeit können ohne externe Infrastruktur im Flugzeug nicht ermitteln werden.

2.3.5 Fingerprinting

Das Fingerprinting ist im Vergleich zu den bisher vorgestellten Verfahren das aufwändigste. Denn um eine möglichst präzise Positionsschätzung zu erhalten, ist es notwendig, das Gebiet, in dem später verortet werden soll, in einem vorhergehenden Schritt sehr genau zu kartografieren. Hierzu werden an bestimmten, regelmäßig über die Karte verteilten und eventuell zusätzlich an besonders wichtigen Punkten Referenzmessungen aller zu empfangenden Signale durchgeführt.

Existieren in einem Kartenabschnitt beispielsweise drei Signalquellen, S_1, S_2 und S_3, so bestimmt man an allen relevanten Punkten den Vektor $\binom{s_1}{s_2}{s_3}$, bestehend aus den jeweils messbaren Signalstärken der einzelnen Quellen am jeweiligen Punkt. Diese Daten werden mit der Position des Messpunktes in einer Datenbank hinterlegt.

Ein Client kann nun seinerseits eine Messung vornehmen und die erhaltenen Werte mit denen aus der Datenbank vergleichen. Um eine Positionsschätzung zu erhalten, ermittelt der Client den Referenzpunkt, dessen Messwerte zu den selbst ermittelten die geringste Differenz aufweisen und nimmt per Proximation dessen Position als die eigene an.

Eine genauere Positionsschätzung als bei der Proximation ist möglich, wenn der Client mehrere Referenzpunkte per Lateration in die Schätzung einbezieht. Zur Ermittlung der Entfernung kann näherungsweise die absolute Differenz der ermittelten zu den gespeicherten Werten ($|\vec{m} - \vec{s}|$) herangezogen werden.

2.4 Qualitätskriterien für Verortungslösungen

Bei der Verortung mobiler Systeme spielen die im Folgenden näher beschriebenen Faktoren Genauigkeit, Zeit, Verfügbarkeit und Kosten eine wichtige Rolle.

Es ist wichtig zu bemerken, dass es sich hierbei um ein System sich gegenseitig beeinflussender Faktoren handelt. Deshalb wird es nie möglich sein, gleichzeitig in jeder Eigenschaft ein Qualitätsmaximum zu erreichen. Möchte man ein präzises Verortungssystem, das sofort nach der Inbetriebnahme eine Positionsschätzung liefert, so fallen dafür mit hoher Wahrscheinlichkeit zusätzliche Kosten an. Erwartet man hingegen ein günstiges System, sind Abstriche in den anderen Eigenschaften, zum Beispiel bei der Genauigkeit oder der Verfügbarkeit hinzunehmen. So könnte der Betreiber eines Satellitennavigationssystems beispielsweise verschiedene Tarifstaffelungen anbieten, die je nach erreichbarer Präzision der Positionsschätzung einen höheren monatlichen Beitrag erfordern.

2.4.1 Genauigkeit

Im Zusammenhang dieser Arbeit ist mit Genauigkeit die Abweichung einer Positionsschätzung zur realen Position gemeint. Bei einer einmaligen Beobachtung wird meist einfach die Differenz $\Delta p = |p_g - p_r|$ zwischen der geschätzten Position p_g und der realen Position p_r angegeben, während bei einer Menge von n Positionsschätzungen eher die Standardabweichung aller Werte von der realen Position interessant ist. Als Standardabweichung σ wird die mittlere Abweichung aller geschätzten Positionen p_{g_i} zur realen Position p_r bezeichnet:

$$\sigma = \sqrt{\sum_{i=0}^{n-1} (p_r - p_{g_i})^2}$$

2.4.2 Zeitfaktor

Unter diesen Oberbegriff fallen gleich mehrere Eigenschaften eines Verortungssystems. Unter Echtzeitfähigkeit versteht man die Eigenschaft eines Systems, unmittelbar auf Veränderungen reagieren zu können und Daten in der Geschwindigkeit zu liefern, in der die Anwendung sie benötigt. Im Kontext des hier entwickelten Systems bedeutet das, dass Positionsschätzungen unmittelbar nach der Messung der Signale zur Verfügung stehen und ständig aktuell sind. Der Versand zum Server, die dort stattfindende Verarbeitung und der Rückversand dürfen

nur Sekundenbruchteile in Anspruch nehmen, da die Informationen sonst zum Zeitpunkt der Nutzung schon veraltet wären.

Eine weitere wichtige Eigenschaft eines Verortungssystems bezüglich des Faktors Zeit ist die Dauer von der Inbetriebnahme bis zur ersten verwertbaren Positionsschätzung. Für diese Zeit hat sich im Umfeld des GPS der englische Begriff „Time To First Fix“ (TTFF)[2] entwickelt. Er bezeichnet die Zeit, meist in Sekunden, die vom Einschalten des GPS-Empfängers bis zur ersten Positionsschätzung vergeht.

2.4.3 Verfügbarkeit

Selbstverständlich ist die generelle Verfügbarkeit eines Dienstes die fundamentale Voraussetzung für seine Nutzung. Allerdings lässt sich der Begriff noch etwas differenzierter betrachten: So kann eine Verortungslösung nur lokal begrenzt verfügbar sein. Ein auf Signalen basierendes System kann nur dort eingesetzt werden, wo diese Signale verfügbar sind. So ist ein System, das WLAN–Signale nutzt, zwar sowohl im Außenbereich als auch in Gebäuden nutzbar, jedoch nur in einem verhältnismäßig kleinen Abstand von höchstens 50 Metern um einen Accesspoint herum. Im Gegensatz dazu ist GPS überall auf der Erde nutzbar, jedoch aufgrund der Schwäche der Signale in Gebäuden nur eingeschränkt.

Verfügbarkeit ist jedoch nicht nur auf den Verwendungsort bezogen ein Faktor, der bei der Entscheidung für ein System beachtet werden muss. Auch die zeitliche Verfügbarkeit einer Lösung trägt zu ihrem Wert bei. Hier ist zu beachten, wie lange der Dienst in Zukunft zur Verfügung stehen wird und wie sicher damit die Investition ist. Dieser Aspekt spielt für die weitere Betrachtung in dieser Arbeit jedoch nur eine untergeordnete Rolle und wird an dieser Stelle deshalb übergangen.

2.4.4 Kosten

Bei der Betrachtung der Kosten einer Verortungslösung muss zwischen einem selbst betriebenen System und einer angemieteten Lösung eines Drittanbieters

[2] Zeit bis zur ersten Positionsschätzung

unterschieden werden. Bei einem selbst betriebenen System müssen die gesamten TCO[3] betrachtet werden. Dazu gehören die Anschaffungs– beziehungsweise Herstellungskosten, die bei der Erstellung der Verortungsinfrastruktur anfallen, genauso wie deren Betriebskosten. Diese beinhalten Energie–, Wartungs– und Reparaturkosten, sodass zu einer Anfangsinvestition eine stetige Reihe von teils unvorhersehbaren Ausgaben hinzu kommt.

Im Gegensatz dazu hat eine nicht selbst betriebene Lösung den Vorteil, dass die Gesamtkosten auf mehrere Dienstnutzer verteilt werden können und der einzelne Nutzer periodisch Kosten in fester Höhe einplanen kann. Im Gegensatz zum selbst betriebenen System ist der Nutzer in diesem Fall jedoch abhängig von den Daten, die der Dienstanbieter zur Verfügung stellt und damit letztendlich dem Wohlwollen des Dienstanbieters ausgeliefert. So entscheidet der Anbieter über die Qualität der einzelnen Faktoren und könnte einem Nutzer sogar im Extremfall wichtige Daten vorenthalten, sofern es seinem eigenen Vorteil dient.

Ein Beispiel für eine solche Abhängigkeit ist die Selective Availability des GPS Systems. Hierbei handelt es sich um die Option, bei Bedarf die Genauigkeit der zivilen Komponente von GPS zu reduzieren. Diese Option hat sich das amerikanische Verteidigungsministerium bis ins Jahr 2000 zur jederzeitigen Anwendung vorbehalten.

2.5 Signalquellen für Positionsschätzungen

In den vorherigen Kapiteln wurden die Techniken behandelt, mit deren Hilfe sich ein mobiles System unter Ausnutzung verschiedener Signale selbst verorten kann. In diesem Abschnitt wird eine nicht erschöpfende Auswahl von Signalquellen vorgestellt, deren Verwendung sich im Prototypen des Location-Providers anbietet. Zur Erinnerung: Die mobilen Geräte, die als Thin Clients verwendet werden, sind Smartphones und PDAs. Sie haben Sensoren für diverse kabellose Kommunikationstechnologien und in den meisten Fällen einen GPS Empfänger. Die Sensoren der Geräte sind mit omnidirektionalen Antennen verbunden, um Signale aus allen Richtungen gleich gut aufnehmen zu können. Vor diesem Hintergrund bieten

[3] Total Costs of Ownership

sich die im folgenden zuerst genannten Technologien vorrangig vor den später erwähnten zur Verwendung an.

Es ist zu bemerken, dass bei den folgenden Betrachtungen der Datenübertragungstechnologien der Fokus nicht auf der Funktion des jeweiligen Verfahrens liegt. Diese können in [TANN03] oder [PETE08] nachgelesen werden. Hier sollen nur die für den Einsatzzweck als Signalquelle in einem Verortungssystem relevanten Eigenschaften betrachtet werden.

Weiterhin gilt für alle hier vorgestellten Verfahren, dass der Positionsschätzer vorab Informationen zum Standpunkt der Signalquellen benötigt. Denn er kann seinen Standpunkt nur relativ zum Standpunkt der Signalquellen schätzen. Je genauer also deren Positionen bekannt sind, desto genauer kann der Schätzer die eigene Position berechnen.

Lässt man die für diese Arbeit weniger brauchbaren Verfahren der visuellen Markierungen und der Odometrie außer Acht, eigenen sich die nachfolgend vorgestellten Signalquellen nur, um eine Positionsschätzung zu erhalten, nicht jedoch um die Blickrichtung oder die Geschwindigkeit zu ermitteln, sofern man die diskreten Messungen zu den jeweiligen Zeitpunkten unabhängig betrachtet. Um diese Bewegungsparameter zu ermitteln, müssen zwei konsekutive Positionsschätzungen verknüpft werden. Im Fall der Bewegungsrichtung $\vec{h}$ würde dies die Differenz $\vec{h} = \vec{p_2} - \vec{p_1}$ bedeuten, wenn p_1 und p_2 zwei auf einander folgende Positionsschätzungen sind. Die Durchschnittsgeschwindigkeit $\vec{v}$ zwischen den Zeitpunkten t_1 und t_2, an denen jeweils eine Position p_1 und p_2 ermittelt wurden, ergibt sich durch $\vec{v} = \frac{|\vec{p_2} - \vec{p_1}|}{t_2 - t_1}$. Bei ausreichend kleinem Δt kann diese Geschwindigkeit als Momentangeschwindigkeit angenommen werden.

2.5.1 GPS

Als erste Signalquelle für Positionsschätzung wird hier ein Satellitennavigationssystem betrachtet. Es ist die einzige der hier vorgestellten Technologien, die ursprünglich zur Verortung mobiler Einheiten entwickelt wurde und ausschließlich diesem Zweck dient. Ein globales Positionierungssystem ist eine überall auf der Welt nutzbare Infrastruktur, mit deren Hilfe entsprechend ausgestattete Objekte ihre Position ermitteln können. Das in diesem Zusammenhang einzige funktions-

fähige System dieser Art stellt das NAVSTAR-GPS[4], das vom Verteidigungsministerim der USA betrieben wird, dar. Es besteht aus drei so genannten Segmenten: Dem Kontrollsegment, das aus den Bodenstationen besteht, dem Weltraumsegment bestehend aus mindestens 24 Satelliten und dem Benutzersegment – dem GPS-Empfänger. Zum Verständnis der Funktion reicht die Betrachtung der beiden Letztgenannten aus.

Das Weltraumsegment besteht aus mindestens 24 Satelliten, von denen jeder mit zwei Cäsiumatomuhren ausgestattet ist, die ein absolut exaktes Zeitsignal garantieren. Jeweils 4 Satelliten bewegen sich auf einer zum Äquator um 55° geneigten Umlaufbahn mit einem mittleren Radius von 26560 Kilometern. Damit umkreisen sie die Erde als Medium Earth Orbiter (MEO) in genau 12 Sternstunden einmal. Es gibt 6 Umlaufbahnen, die in der Äquatorebene um jeweils 60° zu einander versetzt sind. Durch diese Anordnung ist sichergestellt, dass über jedem Ort auf der Erdoberfläche zu jedem Zeitpunkt mindestens vier Satelliten sichtbar sind, ohne dass über den Erdpolen ein überdurchschnittlich hohes und weitestgehend nutzloses Satellitenaufkommen herrscht. Abbildung 2.4 zeigt ein maßstabsgetreues Bild der Umlaufbahnen. Der im Folgenden erklärte Prozess

Abbildung 2.4: Maßstabsgetreue Darstellung der Umlaufbahnen der NAVSTAR-GPS Satelliten (Quelle: [NMEA2])

[4]Navigation Signal Timing and Ranging Global Positioning System, engl. Globales Verortungssystem mit Navigationssignalen durch Zeit- und Entfernungsmessung

der Positionsschätzung mit Hilfe von GPS ist stark simplifiziert dargestellt. Eine weitaus ausführlichere Erklärung steht unter [GPS] zur Verfügung.

Jeder Satellit sendet ein unverwechselbares Signal zur Erde, das aus einer Pseudozufallsfolge besteht. Durch zeitliches Verschieben des empfangenen Signals und einen Abgleich mit den gespeicherten Daten über alle Satelliten kann ein Empfänger den Zeitpunkt ermitteln, an dem das Signal ausgesandt wurde. Seine Datenbank enthält ebenfalls Informationen über die Umlaufbahnen der Satelliten und deren erwarteten Aufenthaltsort zu jedem Zeitpunkt. Anhand der Informationen über den Zeitraum, der zwischen dem Absenden (t_1) und dem Empfangen (t_2) des Signals vergangen ist und der konstanten Ausbreitungsgeschwindigkeit elektromagnetischer Wellen $\overrightarrow{v_{el}}$ berechnet der Empfänger die so genannte Pseudostrecke $s_p = \overrightarrow{v_{el}} * (t_2 - t_1)$, also die Entfernung, die das Signal zurückgelegt hat. Anhand der Pseudostrecke und der Position des Satelliten zum Aussendezeitpunkt t_1 führt der GPS-Empfänger nun eine Lateration durch und ermittelt so seine Position. Für diesen Vorgang sind drei unterschiedliche Signale notwendig, um durch Lateration zwei mögliche Positionen zu ermitteln, von denen eine wiederum ausgeschlossen werden kann, da sie für gewöhnlich im tiefen Weltraum liegt. Ein viertes Signal ist jedoch erforderlich, um die Uhr des Empfängers so einzustellen, dass sie zu den Atomuhren synchron läuft. Für eine dreidimensionale Positionsschätzung durch GPS sind also vier Signale erforderlich.

Die Genauigkeit einer Positionsschätzung beträgt etwa 10 Meter vertikal und 15 Meter horizontal. GPS-Empfang ist im Freien fast überall möglich, in geschlossenen Gebäuden meistens nicht, da hier die Signale zu stark gedämpft werden. Wird ein GPS Empfänger eingeschaltet, lädt er zuerst den Almanach herunter, einen Katalog mit den aktuellen Parametern des GPS-Systems. Dieser Vorgang kann einige Minuten dauern, während deren noch keine Positionsschätzung möglich ist. Wie lange der Download der Daten dauert, hängt davon ab, wie aktuell die im Gerät vorhandenen Daten sind. War der Empfänger längere Zeit nicht in Betrieb, sind neue Daten erforderlich, ist er regelmäßig aktiv, hat er eventuell die aktuellen Daten schon gespeichert. Die Nutzung von GPS ist kostenlos.

2.5.2 Wireless LAN

Drahtlose Netzwerke ermöglichen es mobilen Benutzern, eine bis zu 300 MBit/s schnelle Verbindung zum nächsten Accesspoint aufzubauen und darüber ohne lästige Kabel schnell große Datenmengen auszutauschen. Die durch die Integration in handelsübliche DSL-Router inzwischen am Weitesten verbreitete Technik basiert auf dem IEEE Standard 802.11g. Geräte dieser Klasse senden und empfangen über Kanäle im 2,4GHz Frequenzband. Sie haben eine Reichweite von bis zu 100 Metern, die aufgrund der verwendeten Frequenzen auch in Gebäuden eine gute Durchdringung der Wände und damit eine flächendeckende Ausleuchtung ermöglicht. Da in den meisten Haushalten WLAN nach 802.11g verwendet wird und auch alle auf dem Markt erhältlichen Smartphones WLAN-Funktionen mitbringen, eignet es sich sehr gut zur Unterstützung einer Verortungslösung.

Nebenbei ist zu erwähnen, dass es neben 802.11g auch den Substandard 802.11a gibt. Geräte dieses Typs arbeiten auf einem anderen Frequenzband (5GHz), was zu unterschiedlichen Signaleigenschaften führt. Während 802.11g hauptsächlich zur Errichtung von Personal Area Networks verwendet wird, bei denen es auf eine große Ausleuchtzone ankommt, hat sich 802.11a zur Weitverkehrsvernetzung etabliert. Da hochfrequente Wellen mehr Energie tragen, haben sie eine höhere Reichweite, sodass man mit Richtantennen Strecken von bis zu 10 Kilometern überbrücken kann. Die höhere Frequenz bringt für die Verwendung im häuslichen Bereich jedoch einige Nachteile mit sich. So ist die Durchdringung von Festkörpern im 5 Gigahertzband wesentlich schlechter. Dafür nehmen Reflexionen zu, sodass es zu Mehrwegeeffekten kommen kann.

Aufgrund der technischen Einschränkung, dass die meisten Smartphones nur Chipsätze enthalten, die auf 802.11g basieren, kann diese Betrachtung schnell abgeschlossen und im Folgenden davon ausgehen werden, dass die Bezeichnung WLAN im Rahmen der vorliegenden Arbeit immer den Standard 802.11 b/g meint. Durch die hohe Verbreitung von DSL Routern, die mit WLAN ausgestattet sind, ist in urbanen Regionen immer mindestens ein Accesspoint in Reichweite. Da es sich um Netzwerkgeräte handelt, besitzt jedes eine weltweit eindeutige MAC-Adresse, die als Identifikationsmerkmal des Gerätes dient. Anhand dieses eindeutigen Bezeichners wird eine Datenbank mit den Positionen öffentlich emp-

fangbarer Accesspoints aufgebaut. Ein mobiles System kann dann mit Hilfe dieser Angaben und einer abhängig von der Signalstärke geschätzten Entfernung eine Lateration vornehmen und so seine Position auf etwa 30 Meter genau bestimmen.

2.5.3 Nahbereichsfunk

Unter dieses Stichwort fallen gleich mehrere Datenübertragungsverfahren, die sich durch geringe Reichweite, meist unterhalb von 10 Metern, und eine geringe Leistungsaufnahme auszeichnen. Zudem ist die Hardware in vielen Fällen sehr günstig zu beschaffen.

2.5.3.1 Bluetooth

Bluetooth ist ein Verfahren, das entwickelt wurde, um Peripheriegeräte kabellos mit einem PC zu verbinden. Hierzu ist es notwendig, dass der PC mit einem entsprechenden Bluetooth-Gerät ausgerüstet ist. Damit sich andere Geräte mit dem in diesem Zusammenhang Host genannten PC verbinden können, muss dieser sich als bluetoothfähiges Gerät zu erkennen geben. Dies tut er, indem er regelmäßig aktiv ein Datenpaket aussendet, das die Geräte, die sich eventuell in Reichweite befinden, auf seine Anwesenheit aufmerksam macht.

Die Reichweite von Bluetooth beträgt meist weniger als 10 Meter, kann jedoch in Ausnahmefällen auch bis zu 100 Meter betragen. Um solche Ausleuchtungsradien zu erreichen, sind jedoch Geräte der Klasse 1 notwendig, die mit bis zu 100mW Sendeleistung senden. Die am weitesten verbreiteten Geräte der Klasse 2 kommen hingegen mit höchstens 2,5mW aus.

Durch ihre geringe Reichweite eignen sich die Geräte der Klasse 2 mit 10 Metern Reichweite für eine einfache Positionsverifikation, indem aus der Sichtbarkeit eines Bluetooth-Senders direkt geschlossen wird, dass man sich in einem Umkreis von maximal der Reichweite des Signals um den Sender befinden muss. Im Gegensatz zu einer solchen einfachen Proximation lohnt die Lateration nur, wenn mehrere Sender verfügbar und in Reichweite sind. Dies ist jedoch aufgrund der mittelmäßigen Verbreitung von Bluetooth an fest installierten PCs nicht sehr wahrscheinlich. Da die meisten Bluetooth-Sender in mobilen Geräten verbaut sind, kann man davon ausgehen, dass sie in vielen Fällen deaktiviert sind, um

Energie zu sparen. Diese Tatsache kommt der Lösung des in der Fragestellung definierten Problems nicht ortsfester Referenzstationen entgegen, da inaktive Sender nicht in die Datenbank aufgenommen werden können. Einen Mechanismus, solche fälschlicherweise als Referenzstation markierten, mobilen Sender zu eliminieren, muss jedoch wie in der Fragestellung definiert, implementiert werden.

2.5.3.2 ZigBee

ZigBee ist eine Funktechnik, die zur kostengünstigen Realisierung von Sensornetzwerken, Anlagensteuerung und Warenverfolgung entwickelt wurde. Der offene Standard sieht vor, dass die ZigBee-Geräte einen so geringen Energieverbrauch haben, dass sie selbst im Batteriebetrieb praktisch wartungsfrei an schwer zugänglichen Orten verwendet werden können. Die Geräte bilden untereinander selbständig ein Netzwerk mit einer Baumtopologie oder sogar ein vermaschtes Netz, über das Sensoren geringe Datenmengen wie Messwerte an einen Router übertragen können.

Für eine Verortungslösung ist dieser kostengünstige und flächendeckende Ansatz überaus interessant. Ein System, das sich in einem Gelände bewegt, welches wiederum von einem Sensornetzwerk überspannt wird, kann anhand einer Lateration zwischen mehreren ZigBee-Sendern, die sich im Empfangsbereich befinden, seine Position ermitteln.

2.5.4 RFID

Radio Frequenz Identifikation umfasst eine Vielzahl von Verfahren, um allgemein Objekte maschinenlesbar zu kennzeichnen. Hierzu werden die zu kennzeichnenden Gegenstände mit einem elektromagnetischen Schwingkreis, Tag[5] genannt, versehen. Dieser hat je nach Bauweise verschiedene Fähigkeiten. Er kann ein einfaches Identifikationstoken senden, wie man es von den Diebstahlschutzvorrichtungen der Elektronikmärkte kennt, während umfangreichere Implementierungen über veränderbare Speicher und Rechenkapazität verfügen. Die Energieversorgung erfolgt entweder per Induktion aus dem elektromagnetischen Feld, das das Auslesegerät erzeugt oder bei Chips mit größerem Funktionsumfang über eine eigene

[5]engl. tag = das Etikett

Batterie. Von der Art der Energieversorgung hängt natürlich auch die Reichweite der Tags ab. Je nach Anwendungszweck liegt diese bei wenigen Millimetern bis zu mehreren Metern.

In der Verortungstechnik hat diese variable Reichweite den Vorteil, dass ein Tag sowohl ein Objekt aus nächster Nähe wie auch eine Fläche von einigen Metern Durchmesser eindeutig identifizieren kann. Und dies bei der Verwendung passiver Etiketten ohne eigene Stromversorgung sogar für einen sehr langen Zeitraum, ohne dass Wartungskosten entstehen.

2.5.5 GSM

GSM ist der in Europa verbreitete Mobilfunkstandard, der mit UMTS zur Zeit in der dritten Generation vorliegt. Das Mobilfunknetz besteht aus Zellen, die je nach Verfügbarkeitsanforderung der Nutzer unterschiedliche geographische Ausbreitungen haben. In urbanen Regionen, in denen durch die hohe Bevölkerungsdichte viele Mobilfunkverbindungen auf engem Raum gefragt sind, haben diese Zellen Durchmesser von weniger als einem Kilometer, während auf dem Land, wo die Mobiltelefondichte weit geringer ist, die Zellen bis auf 20 Kilometer Größe wachsen können. In den meisten Fällen bedient eine Basisstation mehrere Zellen, die jeweils für einen Sektor des Ausleuchtungskreises der Basisstation zuständig sind. Mehrere Zellen bilden eine Location Area, die einen weltweit eindeutigen Location Area Identifier (LAI) hat. Innerhalb der Location Area sind auch die Zellen anhand ihrer Cell Identifier eindeutig identifizierbar. Näheres über die GSM Architektur liefert [GSM].

Wie genau sich ein Teilnehmer mit Hilfe seines Mobiltelefons selbst verorten kann, hängt davon ab, wie genau die Daten sind, die er aus dem GSM–Modul des Gerätes auslesen kann. Die Basisstationen vieler Netzbetreiber senden auf bestimmten Kanälen die jeweils eigenen Ortskoordinaten aus. Die Basisstationen des Netzbetreibers O_2 senden beispielsweise auf dem Cell Broadcast Kanal 221 ihre eigene Position im Format $RRRRRRHHHHHH$. Es handelt sich dabei um Koordinaten im Gauß–Krüger Format, das in Kapitel 2.8.2 vorgestellt wird. Allerdings werden sowohl der Rechts– als auch der Hochwert nicht wie üblich siebenstellig, sondern nur sechsstellig notiert. Dies hat den einfachen Grund,

dass die Koordinaten nur auf 10 Meter genau sind und die Einerstelle in der Koordinatendarstellung, die durch Rundung immer 0 wäre, weggelassen wird.

Mit Zugang zu den internen Daten des GSM–Moduls besteht die Möglichkeit, die Parameter LAI der Location Area und CI der Zelle auszulesen, mit der das Modul gerade verbunden ist. Die Location Area beschreibt im GSM Netz eine Zusammenfassung mehrerer Zellen, die von der gleichen Basisstation kontrolliert werden, während der Cell Identifier (CI) die Zelle in einer Location Area bezeichnet. Zusammen identifizieren die beiden Parameter eine bestimmte GSM Zelle weltweit eindeutig und können deshalb für eine Positionsschätzung per Proximation anhand von hinterlegten Koordinaten der GSM–Basisstationen oder sogar mit Hilfe der Koordinaten, die sie selbst versenden, verwendet werden. In den meisten Fällen wird auch dem GSM–Modul nur der LAI bekannt sein. Der aktuelle CI wird erst bei einer Verbindungsaufnahme mit dem Mobilfunknetz übermittelt. Um diesen Parameter zu erhalten, muss es also zu einer Kommunikation zwischen dem mobilem Gerät und dem Netzwerk kommen, etwa bei einem eingehenden oder abgehenden Anruf, einer SMS oder einer Datenverbindung.

Besteht die Möglichkeit, eine Datenübertragung zu provozieren und nicht nur den CI der aktuell verbundenen Basisstation, sondern auch die CIs aller anderen Stationen, die sich in Reichweite befinden auszulesen, ist sogar eine Verortung via Lateration möglich. Um in diesem Fall genauer verorten zu können, ist die Entfernung der ermittelten Basisstationen von entscheidender Bedeutung. Da GSM auf Zeitmultiplexing basiert (Time Division Multiple Access), bei dem jede Station einen festen Zeitschlitz (engl. timeslot, vergl. [TANN03]) zum Senden zur Verfügung hat, muss jeder Sender wissen, wie lange sein Signal bis zur Basisstation unterwegs ist, um den richtigen Zeitpunkt für den Sendebeginn abzupassen. Mit diesem Wissen, das im Timing Advance Parameter hinterlegt ist, kann das Gerät den Sendezeitpunkt so nach vorne verschieben, dass das Signal genau zur richtigen Zeit beim Empfänger an kommt. Kann eben dieser Parameter ausgelesen werden, ist eine präzisere Lateration möglich.

2.5.6 IP-Adresse

Die IP-Adresse ist die weltweit eindeutige Netzwerkadresse eines Rechners im Internet. Die im Fall von IPv4 4 Byte langen Adressen werden dem Teilnehmer vom Internetprovider zugeteilt. Dieser wiederum erhält sie von der Internet Assigned Numbers Authority[6] (IANA).

Um Datenpakete im Internet routen zu können, ist kein Wissen über die geographische Position eines Benutzers nötig. Das Border Gateway Protokoll (BGP) sorgt dank eines intelligenten Routingalgorithmus, dem Pfadvektorrouting (vergl. [PETE08] dafür, dass die Daten ankommen. Allerdings können Informationen über die Position eines Nutzers die Auswahl des günstigsten Pfades unterstützen. Deshalb teilen große Provider ihre Adresskontingente lokal auf, während der Einzugsbereich kleinerer lokaler Anbieter ohnehin geographisch begrenzt ist.

Die hierzu nötigen Informationen sind in grober Form frei zugänglich, sodass ein mobiles System seine Position anhand der ihm zugewiesenen IP Adresse grob bestimmen kann. Zu diesem Thema wurde im Rahmen der Evaluation dieser Arbeit eine Versuchsreihe durchgeführt, die in Kapitel 5.5 dokumentiert ist. Die Versuche ergaben, dass mit den kostenlos zugänglichen Datenbanken eine grobe Verortung im Bereich eines Bundeslandes, beziehungsweise mit einem Radius von etwa 100–150 Kilometern, abhängig vom Internetprovider, möglich ist.

2.5.7 Hauskoordinaten

Jedes in Deutschland von einem Katasteramt erfasste Gebäude besitzt eine eindeutige Objektkoordinate im Gauß-Krüger Format. Diese ist gewöhnlich (vergl. [VEKO08]) der Mitte der Grundrissseite eines Gebäudes zugeordnet, die zu der Straße hin liegt, von der das Gebäude auch seine Hausnummer erhält. Die diversen Ämter für Geobasisdaten halten Datenbanken mit Tupeln, bestehend aus der kompletten Adresse eines Gebäudes und seiner Objektkoordinate vor. Diese Koordinaten sind amtlich beurkundet und bis auf den Millimeter genau. Gibt ein Teilnehmer also eine Adresse als seine momentane Position an, kann davon ausgegangen werden, dass er sich innerhalb des Gebäudes befindet, da ihm sonst komfortablere Techniken der Positionsschätzung wie GPS zur Verfügung stehen

[6] Zentralstelle für die Vergabe von Nummern im Internet

würden. Befindet der Teilnehmer sich in dem Gebäude, dessen Adresse er angegeben hat und dessen geographische Position anhand dieser Adresse ermittelt wurde, kann bei einer durchschnittlichen Hausgrundfläche von unter $100m^2$ von einer Genauigkeit im Bereich von 10 - 15 Metern ausgegangen werden.

In aller Regel sind die Datenbanken der Katasterämter nicht frei zugänglich, sodass die Kostenbetrachtung in diesem Fall eine wichtige Rolle spielt. Denn sofern ein kommerziell tätiges Unternehmen solche Daten erheben würde, wäre dies ein ungeheurer Aufwand, da alleine für Rheinland-Pfalz etwa 1,2 Mio Datensätze anfallen. Hochgerechnet auf die Einwohnerzahl des gesamten Bundesgebietes dürften die Datenbanken etwa 20 Millionen Einträge enthalten. Sofern die Datensätze anonymisiert vorliegen, ist aus datenschutzrechtlicher Sicht hier kein Einwand zu erwarten und die vielfältig verfügbaren Navigationssysteme, die auf Start- und Zieladressen basieren, zeigen, dass ein Markt für solche Daten besteht.

2.5.8 Visuelle Markierungen

Sofern sichergestellt werden kann, dass eine Kamera an einem mobilen System immer in die gleiche Richtung gerichtet ist, zum Beispiel nach oben, bieten visuelle Markierungen eine gute Möglichkeit, Umgebungen gerade innerhalb von Gebäuden zu markieren. Dazu werden an der Decke Markierungen angebracht, die eine nach oben gerichtete Kamera aufzeichnet. Dieses Vorgehen wird in [PELL05] verwendet. Die Autoren verwenden verschiedene infrarot reflektierende Muster. Dies hat den Vorteil, dass die Markierungen bei unterschiedlichen Lichtverhältnissen gleich gut zu detektieren sind. Um die Fehlertoleranz zu erhöhen, verwenden die Autoren Markierungen, die zueinander einen großen Hammingabstand haben. Der Hammingabstand zweier Zeichenketten ist die Anzahl der unterschiedlichen Stellen. Ist diese groß gewählt, kann auch eine falsch erkannte Zeichenkette bis zu einem gewissen Grad noch dem jeweils richtigen Datum zugeordnet werden. Diese Erklärung mit Zeichenketten kann direkt auf visuelle Markierungen übertragen werden, so dass der Hammingabstand zweier Grafiken den Grad der Unterschiedlichkeit darstellt.

Die Verwendung dieses Verfahrens führt dazu, dass ein in Teilen nicht korrekt erkanntes Muster bis zu einem gewissen Grad trotzdem einer Markierung zugeord-

net werden kann. Die eigentliche Lokalisierung erfolgt auch hier durch Proximation. Eine weitere Möglichkeit, Objekte visuell zu markieren ist die Verwendung von Barcodes. Hier bieten sich gerade mehrdimensionale Barcodes an. Während eindimensionale Strichcodes wegen ihrer geringen Kapazität meist nur einzelne Worte oder Zahlencodes speichern können, haben mehrdimensionale Codes genug Kapazität, um die eigenen Koordinaten in kodierter Form enthalten zu können. Damit wäre es dann möglich, beim Client vollkommen auf eine Verortungslogik zu verzichten und nur die Barcodes auszulesen oder diese als Backupsystem an besonders wichtigen Stellen zu nutzen.

Ein weiteres Merkmal dieser Verortungsvariante ist die Möglichkeit, anhand des Musters auf den visuellen Markierungen die Bewegungsrichtung des mobilen Systems zu ermitteln. Hierzu muss nicht nur die Position, sondern auch Informationen über die Anordnung jeder Markierung in einer Datenbank verfügbar sein, damit aus der erkannten Ausrichtung der Markierung auf die Blick- oder Bewegungsrichtung geschlossen werden kann.

2.5.9 Odometrie

Unter Odometrie versteht man die Verortung eines mobilen Systems unter Verwendung seines Vortriebssystems. Da es sich hierbei um eine Koppelnavigation handelt, werden nur Entfernungen und Richtungsänderungen gemessen und in Bezug zu einem Referenzpunkt - dem Startpunkt - gesetzt, um eine Positionsschätzung zu erhalten.

Handelt es sich bei dem mobilen System um ein Fahrzeug, werden die Radumdrehungen gemessen. Anhand ihrer Anzahl kann das System die zurückgelegte Strecke berechnen. Unter der Voraussetzung, dass sich jedes Rad gleich schnell dreht, es also zu keiner Richtungsänderung im Messzeitraum kam, hat das Fahrzeug die Strecke s mit $s = i * 2\pi r$ zurückgelegt, wenn r der Radius des Rades ist, an dem die Messung vorgenommen wurde und i die Anzahl der Umdrehungen. Das Fahren von Kurven erkennt man entweder anhand der unterschiedlich schnellen Raddrehungen oder mit eigenen Lenkwinkelsensoren an jedem Rad.

Auch bei Robotern, die sich mit Hilfe von Extremitäten fortbewegen, ist es möglich, die zurückgelegte Entfernung zu ermitteln. Ist w die Schrittweite des

Roboters, so beträgt die zurückgelegte Strecke $s = w * i$, wenn i die Anzahl der Schritte beschreibt. Das Kurvengehen ist bei einem Roboter schwerer zu erkennen als bei einem Fahrzeug, da es hierfür verschiedene Techniken gibt, deren Implementierung ungleich aufwändiger ist. Jedes Bein müsste mit zusätzlichen Sensoren ausgestattet werden, die die individuelle Weite jedes Schrittes anhand der Gelenkwinkel oder die Auslenkung der Beinstellung aus der Schrittrichtung messen.

Insgesamt ist die Odometrie aufgrund von schwer zu kompensierenden Messungenauigkeiten, beispielsweise bei der Radgeometrie und gerade in Kurven stark fehleranfällig. Für sich alleine eignet sie sich nur bedingt zur Verortung, da jede Position auf der Gesamtheit ihrer Vorgängerpositionen aufbaut und sich Fehler so während des Messprozesses summieren. Als Unterstützung für andere Technologien und als Fehlerkorrekturinstrument für Filterlösungen ist sie jedoch gut geeignet.

In diesem Projekt kann Odometrie nicht eingesetzt werden, da die verwendeten Geräte keine Kontrolle über ihren Vortriebsapparat haben. Sie sind was ihre Bewegung angeht absolut passiv und haben keine Möglichkeit, Messwerte zu sammeln.

2.6 Störquellen für Verortungssignale

In den vorangegangen Kapiteln wurden Verortungstechniken aus rein theoretischer Sicht betrachtet. In der Realität existieren jedoch Rahmenbedingungen, die von den bisher als störungsfrei angenommenen Laborbedingungen abweichen können. Solche Störquellen können beispielsweise durch das Wetter oder eine ungünstige bauliche Lage im Zielgebiet der Verortungslösung hervorgerufen werden. In den nächsten Abschnitten werden die am häufigsten auftretenden Störungen aufgezeigt und kurz erklärt. Im nächsten Kapitel werden dann Maßnahmen vorgestellt, die verursachten Störungen zu erkennen und zu eliminieren.

2.6.1 Atmosphärische Störungen

In der Satellitentechnologie müssen Daten je nach Bahnhöhe des Satelliten über viele Tausend Kilometer (vergleiche Kapitel 2.5.1), GPS Satelliten umkreisen die Erde in einer Höhe von 26560 Kilometern) übertragen werden. Dabei durchqueren die Signale die gesamte Atmosphäre mit ihren verschiedenen zum Teil elektromagnetisch aktiven Schichten und auch einen Teil des Weltraums. Während die Durchquerung des Weltraums für die Signale weitgehend unbeeinflussend verläuft, sind sie in der Erdatmosphäre zum Teil starken atmosphärischen Störungen ausgesetzt. Diese treten hauptsächlich in der Ionosphäre auf, also in einer Höhe von etwa 80 – 300 Kilometern. Diese Atmosphärenschicht besteht aus geladenen Teilchen, die beim durch UV-Strahlen angeregten Zerfall von Gasmolekülen in Ionen entstehen. Diese Ionen interagieren mit den Signalen auf deren Weg von und zu den Satelliten und können die übertragenen Daten verfälschen oder für eine verzögerte Übertragung sorgen.

Gerade GPS, das anhand der Übertragungszeit genau definierter Muster die Entfernung zwischen Satelliten und Empfangsstation ermittelt, reagiert äußerst empfindlich auf solche Störungen. Denn schon Abweichungen von wenigen Metern bei den errechneten Pseudostrecken potenzieren sich zu erheblichen Ungenauigkeiten bei der Positionsschätzung. Eine Möglichkeit, atmosphärische Störungen zu korrigieren stellt dGPS (vergl. Kapitel 2.7.1) dar.

2.6.2 Mehrwegeeffekt

Treten bei der Signalübertragung Reflexionen auf, kann es zum Mehrwegeeffekt kommen. Dabei erreichen die Signale den Empfänger auf (mindestens) zwei Wegen. Einmal auf der direkten Strecke zwischen dem Sender und dem Empfänger – dabei handelt es sich um das richtige Signal und einmal auf dem Umweg über die Reflexion an einem Hindernis. Ein Hindernis kann ein Gebäude, ein Strommast, eine andere Antenne, aber auch eine dichtere Luftschicht oder die ionisierte Atmosphäre selbst sein. Durch die Reflexion wird die Signallaufzeit länger und es entstehen zwei phasenverschobene Signale, die sich im Extremfall gegenseitig überlagern und unkenntlich machen.

Moderne Antennen können solche Signale, die durch Mehrwegeübertragung entstehen, zum Teil erkennen und ausfiltern. Allerdings wird dadurch auch das Originalsignal geschwächt.

2.6.3 Uhrenfehler

GPS ermittelt die Entfernung zwischen Sender und Empfänger anhand der genauen Laufzeit des Signals und der Geschwindigkeit, mit der sich elektromagnetische Wellen ausbreiten. Um die exakte Signallaufzeit berechnen zu können, sind äußerst genaue Uhren notwendig. Jeder Satellit enthält zwei Cäsiumatomuhren, die allerdings häufiger mit der Bodenstation synchronisiert werden müssen. Denn aufgrund der hohen Umlaufgeschwindigkeit und des größeren Abstands zur Erdgravitation der GPS Satelliten kommt es an Bord zu relativistischen Effekten, wobei die Auswirkungen der Lage im Gravitationsfeld die der hohen Geschwindigkeit übersteigen. Deshalb vergeht die Zeit in einem GPS Satelliten ein wenig schneller als auf der Erde. Dem wird entgegengewirkt, indem die Schwingfrequenz der Atomuhren um ein weniges reduziert wird, sodass die Uhren mit irdischen Atomuhren gleich schwingen.

Auch Mobiltelefone bestimmen den Abstand zwischen ihrer Position und dem Sendemast. Denn es hängt von der Entfernung ab, wann Signale, die zu einer bestimmten Zeit vom Mobiltelefon ausgesendet wurden, an der Basisstation ankommen. Dies ist wichtig, da GSM mit Time Division Multiplexing arbeitet. Das bedeutet, dass es ein geteiltes Medium gibt, das alle übertragenden Einheiten gemeinsam nutzen. Damit es nicht zu Überlagerungen kommt, wird jeder Einheit ein Zeitschlitz zugewiesen. Damit dieser Zeitschlitz genau eingehalten werden kann, ist der Zeitpunkt des Eintreffens der Daten beim Empfänger von größter Wichtigkeit. Um den Sendezeitpunkt auf die Entfernung so abstimmen zu können, dass die Daten genau zur richtigen Zeit beim Empfänger ankommen, benötigt jedes Mobiltelefon eine genau mit der Basisstation synchron laufende Uhr.

2.6.4 Falsche Ausgangskoordinaten

Wie in den vorangegangenen Kapiteln vorgestellt basieren die meisten elektronischen Verortungsverfahren auf den bekannten Positionen von Referenzpunkten.

Von diesen Punkten wird dann anhand geeigneter Methoden auf die eigene Position geschlossen. Wenn aber die Ausgangswerte der Berechnung, also die Positionen der Referenzpunkte falsch waren, erhöht das den Fehler bei der Positionsschätzung quadratisch. Basiert ein Verortungssystem auf der Messung von zwei Referenzpunkten, deren Positionen jeweils mit den Abweichungen f_1 und f_2 bekannt sind, muss bei der Verortung mindestens eine Ungenauigkeit von $f_v = f_1 * f_2$ in Kauf genommen werden.

Dieser Störquelle kann technisch fast nicht beigekommen werden. Es ist zwar möglich, die Messung anhand von mehreren Referenzpunkten zu verifizieren, doch deren Positionen könnten ja ebenfalls fehlerhaft sein. Hier hilft nur eine sehr gewissenhafte Kontrolle der Positionen der Referenzpunkte.

2.6.5 Berechnungsfehler

Hat man die korrekten Entfernungen, Winkel oder sonstigen Messwerte bezüglich der verwendeten Referenzpunkte ermittelt, kann es zu einem letzten Fehler kommen. Bei der Kombination der Messwerte, also der Berechnung der Position, kann es zu einem Rechenfehler kommen. Dafür gibt es viele Gründe. Hier sind unter anderem Variablenüberläufe, falsche Typkonvertierungen und ungenaue Einheitenumrechnung zu nennen. Nur eine genaue Kontrolle und formale Verifikation der verwendeten Algorithmen kann Berechnungsfehler ausschließen.

2.7 Verbesserung der Positionsschätzung

Nachdem im vorherigen Kapitel potentielle Fehlerquellen betrachtet wurden, wendet sich dieses Kapitel den Maßnahmen gegen die häufigsten Fehler zu. Als erstes werden hierzu die Techniken betrachtet, die es ermöglichen, speziell die durch GPS ermittelten Positionsschätzungen zu verbessern. Danach werden zwei Techniken vorgestellt, die es ermöglichen, eine Positionsschätzung generell bezüglich ihrer Wahrscheinlichkeit zu bewerten und sie mit Hilfe von probabilistischen Vorhersagen sogar zu präzisieren. Dazu verfolgen die eingesetzten Systeme eine Reihe von Messwerten und versuchen, den jeweils nächsten vorherzusagen. Doch zuerst zu den für GPS relevanten Verfahren.

2.7.1 Differential GPS

Wie ein Global Positioning System funktioniert, wurde in Kapitel 2.5.1 bereits erklärt. Auch, dass es diverse Störquellen gibt, die eine Verortung, speziell beim GPS erschweren können, wurde bereits ausgeführt (siehe Kapitel 2.6ff). Bei der meistverwendeten Verortungslösung, dem GPS, führen die meisten der vorgenannten Störungen zu einer falschen Berechnung der Pseudostrecken zwischen dem jeweiligen Satelliten und dem Empfänger. Anhand dieser Pseudostrecken, also den Entfernungen zwischen den einzelnen Satelliten und dem Empfänger bestimmt dieser über eine Lateration seine Position.

Zur Erinnerung: Ein GPS Empfänger kennt die Position aller Satelliten des GPS Systems und verwendet diese als Referenzpunkte. Anhand des spezifischen Signals, das jeder Satellit permanent aussendet, berechnet der Empfänger die Pseudostrecke, also die Entfernung zum Satelliten und führt dann eine Lateration wie in Kapitel 2.3.2 beschrieben durch. Während der Übertragung des Satellitensignals wirken verschiedene Störeinflüsse auf die Signale ein, sodass sie oft verzerrt oder verzögert vom Empfänger registriert werden. Dieser ermittelt basierend auf den falschen Signalen falsche Entfernungen und somit letztendlich eine falsche Positionsschätzung.

Die Idee, die sich hinter dGPS verbirgt ist es, einen GPS Empfänger an einer festen, genau definierten Position aufzustellen und die Positionsberechnung auf zwei verschiedene Arten durchzuführen. Zum Einen auf die beschriebene Weise, die jeder GPS Empfänger verwendet, zum Anderen zusätzlich in der umgekehrten Richtung. Das bedeutet, dass der Empfänger die Sollpseudostrecken berechnet, die bei idealen Übertragungsverhältnissen zwischen seiner Position und der Position der Satelliten ermittelt werden müssten. Diese berechneten Sollpseudostrecken vergleicht ein mit dem Empfänger verbundener Rechner mit den vorher vom Empfänger ermittelten Istpseudostrecken. Die Differenz, die sich ergibt, wird im Anschluss in einen RTCM[7] Datenstrom kodiert und über eine Sendestation an dGPS fähige Empfänger übertragen. Diese dekodieren die Pseudostreckendifferenzen und addieren sie zu den von ihnen berechneten Entfernungen, was bei

[7] Radio Technical Commission for Maritime Services

professionellen Empfängern die Verortungsgenauigkeit um den Faktor 10 erhöhen kann.

Die übliche Aktualisierungsrate der dGPS Daten liegt für eine einfache dGPS Korrektur bei 3 Sekunden, bei einer hochgenauen dGPS Korrektur im Bereich von 10^{-1} Sekunden.

2.7.2 NTRIP

Networked Transport of RTCM via Internet Protokoll ist ein Verfahren, um RTCM Daten über das Internet zu verteilen. Es wurde vom deutschen Bundesamt für Kartographie und Geodäsie entwickelt und ist seit September 2004 international als Empfohlender Standard (Recommended Standard) freigegeben.

Ntrip[8] basiert auf dem Hyper Text Transfer Protokoll (HTTP) 1.1 und erlaubt nicht nur die Übertragung eines RTCM Datenstroms, sondern auch vieler anderer navigatorischer Informationen. Aufgrund dieser Flexibilität hat das Verfahren schnell Verwendung gefunden. Außerdem ist es oft günstiger, da die Antennen und Dekodierer für über Funk verteilte RTCM Daten teuer sind, während das Internet selbst für mobile Anwendungen immer höhere Datenraten bei niedrigeren Kosten bietet.

Der physikalische Aufbau von Ntrip besteht aus dem Broadcaster, der die Daten erzeugt, dem Server, der die Daten verteilt und dem Client, der die Daten verbraucht. Es ist also nicht zwingend nötig, dass der Broadcaster Informationen über die Anzahl und Art seiner Clients erhält, womit deren Anonymität gewahrt bleibt.

2.7.3 Assisted GPS

Zu den bisher vorgestellten Verfahren unterscheidet sich assisted GPS[9] (aGPS) in zweierlei Hinsicht. Zum Einen ist hier die Art der Datenübertragung zu nennen: Bei dGPS und Ntrip handelt es sich um Echtzeitverfahren. Das bedeutet, dass ein permanenter Datenstrom immer neue Korrekturinformationen zur Verfügung

[8] „Nach Beschluss des Arbeitskreises Raumbezug der Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder (AdV) ist als Schreibweise nicht mehr NTRIP, sondern Ntrip zu verwenden.“

[9] unterstütztes GPS

stellt. Im Gegensatz dazu stellen die Verfahren, die unter dem Begriff aGPS zusammengefasst werden, Einmalübertragungen dar. Diese finden meist vor der Aufnahme der Verortungstätigkeit durch den Empfänger statt.

Der zweite große Unterschied besteht im Ziel des Verfahrens. Während die in einem RTCM Datenstrom eingebetteten Pseudostreckenkorrekturen die Verortungsgenauigkeit permanent verbessern sollen, ist der Zweck der aGPS Technologie hauptsächlich die Verkürzung der Zeit bis zur ersten Positionsschätzung (Time To First Fix - TTFF).

Unter aGPS versteht man die Übertragung von Daten, die den GPS Empfänger bei der initialen Positionsschätzung unterstützen sollen, über ein Kommunikationsnetz. Im Allgemeinen wird aGPS in Smartphones mit integriertem GPS Empfänger eingesetzt, sodass das Mobilfunknetz zur Datenübertragung genutzt werden kann.

Wie im Kapitel 2.5.5 ausgeführt wurde, kann nur anhand von Daten, die im Mobilfunknetz vorliegen, eine Positionsschätzung des Clients durch das Netz ermittelt werden. Diese ist recht grob und eignet sich nicht besonders gut für feingranulare ortsbezogene Dienste. Der GPS Empfänger kann diese Angaben allerdings nutzen, um seine Startzeit zu verkürzen. Wenn das Gerät seine grobe Position kennt, kann es die Menge der an diesem Standort momentan sichtbaren Satelliten vor dem Start bestimmen und seine Suche beim Systemstart entsprechend eingrenzen. Eine weitere Möglichkeit, den Start zu beschleunigen besteht in der Bereitstellung des Almanachs. Dieser enthält aktuelle Bahndaten und Fehlerkorrekturen der Satelliten, die diese in regelmäßigen Abständen neben dem normalen GPS Signal versenden. Wird der Almanach über das Mobilfunknetz bereitgestellt, reicht dem Empfänger für eine Positionsermittlung ein um ca. 30dB schwächeres Signal aus, da nur noch reine Signaldaten empfangen werden müssen. Dadurch bleibt zum Einen eine Positionsschätzung auch unter schlechten Empfangsbedingungen, zum Beispiel zwischen Hochhäusern möglich, wenn das Signal stark abgeschwächt wird. Zum Anderen kann der Empfänger bei guter Signalqualität aber auch in einen Energiesparmodus schalten, was die Akkulaufzeit des Geräts und damit seine Verfügbarkeit erheblich steigert.

2.7.4 Filter

Viele der in den vorhergehenden Kapiteln aufgezählten Störfaktoren lassen sich durch den Einsatz geeigneter Maßnahmen eliminieren. Dies können störunempfindlichere Antennen mit Zusatzschaltungen zur Echoerkennung sein, die die Störeinflüsse des Mehrwegeeffekts vermindern. Denkbar wären auch präzisere Uhren in den Empfängern, eventuell sogar Atomuhren, die mit den GPS Uhren synchron laufen. Dadurch würde die fehleranfällige Ermittlung der Systemzeit vor der eigentlichen Verortung entfallen. Allerdings tritt hier schnell ein alt bekannter Effekt ein: Denn mit steigender Präzisionsanforderung steigt der Bedarf an finanziellen Mitteln mindestens quadratisch, sodass die benötigten Geräte kein Durchsetzungsvermögen auf dem Massenmarkt hätten.

Ein weiteres Argument gegen die Verwendung von teurer Spezialhardware ist, dass die Geräte, die diese Software bedienen soll, handelsübliche Smartphones sind. Die oben genannten Vorschläge, eine Atomuhr oder Spezialantennen zu verwenden, scheiden also auch aus dem Grund aus, dass sie nicht nur preislich, sondern auch technisch und in Bezug auf den Komfort nicht in die Geräte integriert werden können, ohne deren sonstige Verwendungsmöglichkeiten im Alltag extrem einzuschränken.

Deshalb sollen in dieser Arbeit Lösungen verwendet werden, die keine Hardwaremodifikationen erfordern. Dafür bieten sich Softwarefilter an. Stellt man sich das Ergebnis einer Positionsschätzung als ein Datum vor, das aus zwei Komponenten besteht, nämlich der eigentlichen genauen Position und einem statistischen Rauschen, das die Position überlagert und verfälscht, dann setzen die Filtermethoden, die im Folgenden vorgestellt werden sollen, an diesem Rauschen an. Sie versuchen, sowohl das Rauschen als auch die Bewegung des Empfängers vorherzusagen, um so die Verfälschung der gewonnenen Daten eliminieren zu können.

2.7.5 Kalmanfilter

Der Kalmanfilter dient zur Eliminierung des statistischen Rauschens, das üblicherweise reale Messwerte überlagert und damit verfälscht. Außerdem soll er Störungen, die von den Messsystemen selbst verursacht werden, erkennen und eliminieren. Dazu ist es allerdings notwendig, die mathematische Struktur des

Systems zu kennen, da es sich bei dem Filter um einen Satz mathematischer Gleichungen handelt. Auch die Struktur des Rauschens muss bekannt sein.

Weiterhin nimmt der Kalmanfilter für das zu überwachende Messsignal die Markoveigenschaft an. Das bedeutet, dass der aktuelle Zustand des Systems vollständig aus einer endlichen Menge von Vorgängerzuständen abgeleitet werden kann. Im Fall des Kalmanfilters trifft sogar die Markoveigenschaft erster Ordnung zu. Der aktuelle Zustand wird also als Folgezustand genau eines des letzten Vorgängerzustandes betrachtet.

Der Kalmanfilter arbeitet mit Systemzuständen zu bestimmten Zeitpunkten. Ein Systemzustand wird als Vektor X dargestellt, der aus allen relevanten Eigenschaften des Systems besteht. Der Einfachheit halber werden die Zeitpunkte $t_0,\ldots,\ t_n$ durchnummeriert und man schreibt einfach die Nummern $0,\ldots,\ n$. So kann der Zustand X_k des Systems zum Zeitpunkt t_k folgendermaßen beschrieben werden:

$$X_k = F_{k-1}X_{k-1} + B_{k-1}u_{k-1} + w_{k-1}$$

Während X_{k-1} den Vorgängerzustand des Systems beschreibt, ist F_{k-1} die Übergangsmatrix. Sie beschreibt, wie jede Eigenschaft des Systems sich von Vorgängerzustand zum aktuellen Zustand verändert. Die Komponente u stellt den vollständig vorhersagbaren Teil der Störung mit seiner Dynamik B dar, während w eine rein stochastische Größe ist, die, wenn man so will, das vorhersagbare Rauschen verrauscht. Im Allgemeinen handelt es sich hier um eine Normalverteilung um den Mittelwert 0 mit der Standardabweichung Q.

Bei jedem Schritt wird der Zustand X_k des Systems gemessen. Dies ist allerdings nicht direkt möglich, da ja das Messgerät selbst einer Störung unterliegt, dem Messrauschen v_k. Man misst also

$$Z_k = H_k * X_k + v_k$$

Ein Filterschritt des Kalmanfilters besteht darin, zuerst eine Prädiktion durchzuführen. Das bedeutet, das System schließt vom Zustand zum Zeitpunkt $k-1$ auf den Zustand zum Zeitpunkt k. Dazu wird der Zustand X_{k-1} mit Hilfe der Übergangsmatrix in den Zustand X_k überführt. Vereinfacht könnte man sich vorstellen, dass das System von einer bekannten Position anhand von Geschwindig-

keit und Bewegungsrichtung auf die Position bei der nächsten Messung schließt. Diese wird dann noch mit der Unsicherheit überlagert, die sich ebenfalls von Messzeitpunkt zu Messzeitpunkt ändern kann. Zusätzlich wird noch ein generelles statistisches Rauschen integriert.

Im zweiten Schritt ermittelt das System dann den realen Zustand und daraus die Korrekturmatrix, mit der der zuvor geschätzte Zustand korrigiert wird. Außerdem wird die Übergangsmatrix angepasst. Da aber bekannt ist, dass die Messdaten, die die Sensoren liefern ebenfalls mit Störungen überlagert sind, werden auch diese mit berücksichtigt. So ist über einen längeren Zeitraum die Annäherung an den Idealwert in beliebiger Genauigkeit möglich.

Dies wiederum entspricht einer Positionsmessung, anhand deren Ergebnis die vorherige Schätzung überprüft werden kann. Da bekannt ist, dass der Verortungssensor nicht exakt genau ist, kann die Ungenauigkeit in die Messung eingerechnet werden. Mit den Daten der neuen Messung wird dann die Übergangsmatrix, die in diesem Fall die Geschwindigkeit und Bewegungsrichtung abbildet, angepasst.

Der Nachteil des Kalmanfilters ist, dass es sich um einen linearen Filter handelt. Will man eine mehrdimensionale Bewegung abbilden, werden die Übergangsmatrizen sofort unüberschaubar groß, da für jede Bewegungsrichtung eigene Differenzwerte für alle Systemeigenschaften vorgehalten werden müssen. Ein weiterer Nachteil des Kalmanfilters ist im Zusammenhang mit dem vorliegenden System, dass der Algorithmus immer nur ein Ziel verfolgen kann. Aber gerade bei der Verwendung von WLAN Signalen zur Verortung kommt es zu einem starken Springen, das der Filter nur schlecht abbilden kann. Durch den schnellen (angenommenen) Positionswechsel in den Ergebnissen der Verortung werden die dynamischen Störungen enorm hoch veranschlagt, sodass eine Verortung nur schwer möglich ist.

2.7.6 Partikelfilter

Der Partikelfilter dagegen verfolgt einen anderen Ansatz. Er verfolgt nicht einen einzelnen Systemzustand, sondern kann mehrere wahrscheinliche Zustände gleichzeitig beobachten und sobald eine eindeutige Entscheidung für einen Zustand möglich ist, diesen alleine weiter verfolgen. Dies erreicht der Filter, indem die

Partikel nicht für die einzelnen Eigenschaften eines System stehen. Ganz im Gegenteil ist jeder Partikel ein möglicher Zustand des Systems. Deshalb muss jeder Partikel alle Eigenschaften des Systems besitzen, die überwacht werden sollen. Um einen wahrscheinlichen Systemzustand in der Schar der Partikel zu finden, werden diese nach vorher definierten Regeln gewichtet.

In der Initialisierungsphase erzeugt der Filter eine vorher festgelegte Anzahl von Partikeln, die über den gesamten Wertebereich der zu glättenden Sensordaten verteilt sind. Je mehr Partikel erzeugt werden, desto feingranularer lässt sich eine Übergangssituation abbilden. Jeder der Partikel hat eine Position im Wertebereich und eine Gewichtung. Je höher die Gewichtung ist, desto wahrscheinlicher ist es, dass der Partikel während des Zustandsübergangs nicht verworfen wird. Zum Anfang ist die Gewichtung aller Partikel gleich.

Aus der Partikelmenge wird nun eine Stichprobe entnommen. Hierfür gibt es verschiedene Algorithmen, die in Denzler [DENZ03] vorgestellt werden. In der vorliegenden Arbeit wird die sequentielle gewichtete Stichprobenentnahme verwendet. Das bedeutet, dass die Wahrscheinlichkeit, dass ein Partikel in der Stichprobe enthalten ist, äquivalent zu seiner Gewichtung sein muss. Dazu werden die einzelnen Partikel entlang eines eindimensionalen Zahlenstrahls angeordnet, an dem sie soviel Platz einnehmen, wie es ihrer Gewichtung entspricht. Hoch gewichtete Partikel haben also einen großen Bereich des Zahlenstrahls inne, während niedrig gewichtete Partikel nur einen sehr geringen Bereich besetzen. Der Algorithmus erzeugt nun Zufallszahlen und wählt den Partikel, der an der entsprechenden Stelle des Zahlenstrahls positioniert war aus und fügt ihn der Stichprobe hinzu. Da hoch gewichtete Partikel einen großen Bereich des Zahlenstrahls besetzen, ist die Wahrscheinlichkeit, dass ihr Bereich durch die Zufallszahl getroffen wird, äquivalent zur Gewichtung.

Ein anderer möglicher Algorithmus wäre die partitionierte Stichprobenentnahme. Hierbei wird der Zustandsraum in verschiedene Bereiche unterteilt, aus denen im Folgeschritt jeweils einzeln Stichprobenentnahmen durchgeführt werden. Dieser Ansatz scheint für eine Verortungslösung nicht sinnvoll. Zwar bietet er den Vorteil, dass Positionen auch ohne expliziten Schritt der Übergangsfunktion im gesamten Wertebereich probabilistisch verfolgt werden können, dies jedoch

vor dem Nachteil, dass der gesamte Wertebereich während der gesamten Lebenszeit des Filters von einem Grundrauschen überlagert bleibt.

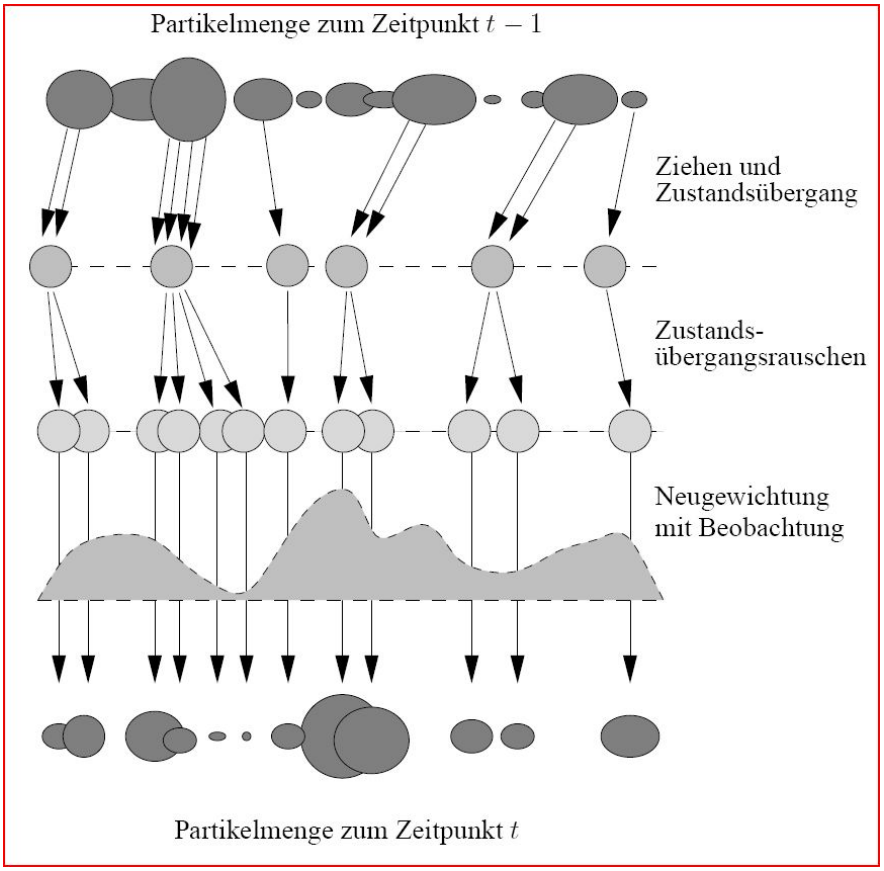

Abbildung 2.5: Schritte eines Partikelfilters (Quelle: [DENZ03])

Im nächsten Schritt findet der Zustandsübergang statt. Hierzu wird eine Zustandsübergangsfunktion benötigt, die auf jeden Partikel angewendet wird. Diese Funktion muss durch geeignete Algorithmen ständig an die Realität angepasst werden, damit die Partikel nicht in völlig realitätsferne Zustände überführt werden. Beim Zustandsübergang werden die Wertigkeiten der Partikel gleichgesetzt. Würde jetzt eine Stichprobenentnahme durchgeführt, wäre die Entnahmewahrscheinlichkeit für alle Partikel gleich.

Nach dem Zustandsübergang haben sich die Werte, die die Partikel darstellen, entsprechend der Übergangsfunktion verändert. Um nun ein Rauschen zu simulieren, werden wahllos Partikel verdoppelt und mit Hilfe einer Normalverteilung deren Werte geringfügig verschoben.

Anschließend kommt es zur Überlagerung der Partikel mit den Sensordaten. Dabei werden die Gewichtungen der Partikel neu justiert. Partikel, die mit einem Sensorwert in besonderem Maße übereinstimmen, werden hoch gewichtet, während Partikel, denen keine Sensordaten entsprechen eine geringe Gewichtung erhalten. Hier wird deutlich, dass es durchaus sein kann, dass bei der Überlagerung mit den Sensordaten mehr als ein Ergebnis eine hohe Wahrscheinlichkeit besitzt. Dies führt dazu, dass die Partikel in den entsprechenden Regionen hoch gewichtet werden. In den weiteren Zyklen des Filters wird es also mehrere Maxima geben, die sich so lange durch ihre hohe Gewichtung fortpflanzen können, bis die Sensordaten für das jeweilige Maximum so schwach werden, dass sie keine höhere Gewichtung mehr zulassen.

Mit den beschriebenen Schritten hat der Filter einen Zyklus hinter sich gebracht und einen neuen Zustand erreicht. Es gibt eine oder mehrere Regionen, in denen sich viele hoch gewichtete Partikel befinden, während in den übrigen Regionen nur wenige, niedrig gewichtete Partikel übrig geblieben sind. Die Regionen mit hoch gewichteten Partikeln stellen für diesen Zyklus den oder die wahrscheinlichsten Zustände des Systems dar. In Grafik 2.5 ist die Gewichtung durch die Größe der Partikel angedeutet.

2.8 Koordinatensystem

Ein Koordinatensystem dient der Angabe der Position eines Punktes im Raum. Es ist ein mathematisches Hilfsmittel, das in allen Anwendungsgebieten mit Ortsbezug Verwendung findet. Die Position eines Punktes wird durch seine Koordinatendarstellung eindeutig bestimmt. Die Anzahl der dazu notwendigen Koordinaten ist direkt gleich der Dimension des Raumes. So genügen in einem zweidimensionalen Koordinatensystem zwei Koordinaten, die im Fall eines kartesichen Koordinatensystems üblicherweise mit x und y bezeichnet werden, während in einem dreidimensionalen Raum noch eine z Koordinate hinzu kommt.

Im Folgenden sollen die beiden dreidimensionalen Koordinatensysteme näher betrachtet werden, die für die Positionsermittlung in unserem Fall eine Rolle spielen.

Während das GPS System auf einem sphärischen Koordinatensystem aufbaut und mit Kugelkoordinaten arbeitet, verwendet das Metasystem, in das diese Arbeit eingebettet ist, das in Deutschland zur Vermessung übliche kartesische Gauß-Krüger Koordinatensystem, das mit rechtwinkligen Koordinaten arbeitet. Die zwischen diesen beiden Systemen nötige Transformation beschreibt [TRGK].

Bei der Angabe von Koordinaten auf dem Globus, ganz besonders bei der Höhe über Normalnull, wie sie in der Vermessung verwendet wird, ist die Form des Globus zu beachten. Denn die Erde besitzt keineswegs eine perfekte Kugelform. Eher besitzt sie eine leicht elliptische Form mit abgeflachten Polen und einem etwas dickeren Äquator, da die Fliehkraft, die durch die Erdrotation auf die Erdmasse wirkt, die Materie nach außen beschleunigt, was zu der Verformung führt. Doch auch ein Ellipsoid ist noch eine angenäherte Form, der der Globus nicht zur Gänze entspricht. Denn zusätzlich zu der globalen elliptischen Verformung besitzt die Erdoberfläche auch lokal starke Unregelmäßigkeiten, die man sich als Dellen und Beulen im Erdmantel vorstellen kann. Um diesen Verformungen Rechnung zu tragen und eine präzise Vermessung zu ermöglichen, gibt es verschiedene mathematische Annäherungen an die Form der Erde. Diese Referenzellipsoide, von denen es knapp 20 wichtige gibt, sind jeweils für ihren Verwendungszweck bestangepasst. Um dem Fakt Rechnung zu tragen, dass es sich bei der Erde nicht um eine Kugel handelt, spricht man in der Geodäsie nicht von der Erdkugel, sondern vom Geoid. Im Folgenden werden die verwendeten Koordinatensysteme kurz beschrieben, um die Unterschiede zwischen ihnen zu verdeutlichen. Weitere Informationen zu den hier beschriebenen Systemen, Formaten und Berechnungen bietet [LANG06].

2.8.1 WGS84

Das World Geodetic System von 1984 (WGS1984) ist das Koordinatensystem, das dem GPS zu Grunde liegt. Da GPS auf dem gesamten Globus möglichst genau sein muss, ist das WGS84 Referenzellipsoid global bestangepasst. Das bedeutet, dass die globale Durchschnittsabweichung zwischen Ellipsoid und Geoid minimal ist. Natürlich führt das vereinzelt zu starken lokalen Abweichungen, die allerdings laut [GPS] in der Summe unbedeutend sind.

Das verwendete Koordinatensystem bei der Verortung mit GPS ist sphärisch und dreidimensional. Der Koordinatenursprung liegt am Schnittpunkt der Erdachse mit der Äquatorebene, also im Mittelpunkt der Erde. Die Koordinaten sind Kugelkoordinaten. Eine Position wird durch einen Tupel aus drei Koordinaten $\binom{\alpha}{\beta}{r}$ bestimmt. Dabei sind α und β Winkel, während r den Abstand zum Ursprung beschreibt. Mit Hilfe dieser Koordinaten lässt sich ein Gitternetz auf der Erdoberfläche errichten. Der Äquator ist der Kreis, der auf einer Ebene durch den Erdmittelpunkt liegt, die senkrecht zur Rotationsachse der Erde steht. Die Breitenkreise sind die Ringe, die parallel zum Äquator verlaufen. Der Breitengrad ist der Winkel zwischen diesen Ringen und dem Äquator bezüglich des Erdmittelpunktes entlang eines Meridians. Ein Meridian oder Längenkreis verläuft senkrecht zu allen Breitenkreisen. Das bedeutet, dass alle Meridiane den Nord- und den Südpol schneiden.

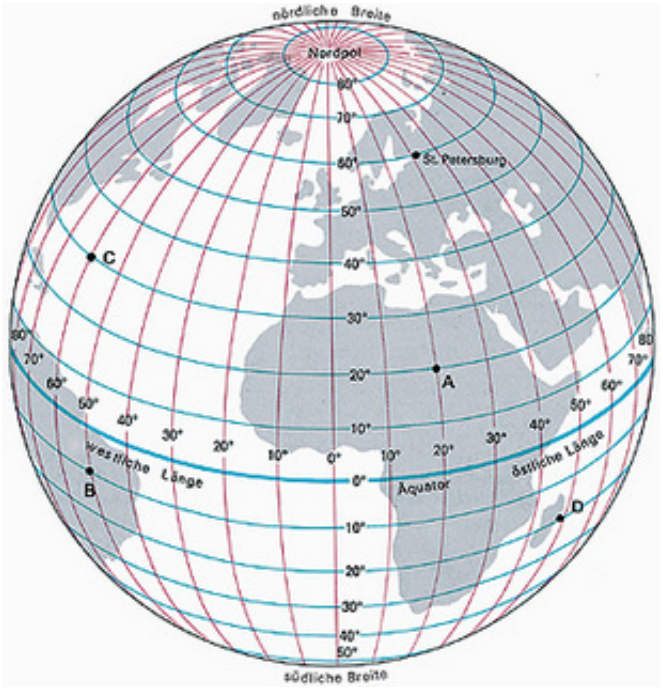

Abbildung 2.6: Längen- und Breitengrade (Quelle: [GPS])

Der Nullpunkt der Breitenkreise ist der Äquator mit 0°, während der Nordpol bei 90° und der Südpol bei -90° liegt. Der Nullmeridian ist definiert als der Längenkreis, der durch die Sternwarte von Greenwich bei London verläuft. Grafik 2.6

zeigt eine schematische Darstellung der Erde mit Breitengraden (horizontal) und Längengraden (vertikal).

Mit der Übereinkunft, dass nur immer eine Position auf der Erdoberfläche bezeichnet werden soll, ist es sogar möglich, den Erdradius als quasi konstant anzusehen und ihn in Positionsdarstellungen wegzulassen. So wäre die Angabe, dass die Spitze des Deutschen Ecks in Koblenz, an dem die Mosel in den Rhein mündet, nach dem amtlichen Stadtplan die Position 7° 36' 22" östlicher Länge, 50° 21' 53" nördlicher Breite hat, ausreichend. Bei dieser Angabe fällt noch eine weitere Besonderheit auf. Die Koordinaten werden nicht dezimal, sondern in einem duodezimalen System notiert. Ein Winkelgrad besteht aus 60 Winkelminuten, die wiederum aus 60 Winkelsekunden bestehen.

Diese Notation und der Umstand, dass mit Winkeln und nicht mit Längen gearbeitet wird, führt zu dem Nachteil des Systems, dass die Angaben nur sehr schlecht menschenlesbar sind. Folgendes Beispiel soll dies verdeutlichen: Das östliche Rheinufer, gegenüber des Deutschen Ecks hat bei gleicher geographischer Breite die Länge 7° 36' 38" Ost. Nun kann man aus der Differenz von 16 Winkelsekunden nicht auf Anhieb die Breite des Rheins herauslesen, die an dieser Stelle etwa 300 Meter beträgt. Dies ermöglicht das Gauß-Krüger Koordinatensystem auf besonders leichte Art.

2.8.2 Gauß-Krüger Koordinaten

Im Gegensatz zum vorher beschriebenen WGS84 handelt es sich beim Gauß-Krüger (abgekürzt: GK) Koordinatensystem um ein kartesisches System. Es besitzt also nur orthogonale Koordinaten. Zusätzlich verwendet es als Maßeinheit den Meter. Das führt dazu, dass eine Positionsangabe in GK-Notation intuitiv verstanden werden kann. Da es sich bei Gauß-Krüger um ein deutsches System handelt, das auch in der amtlichen Vermessung eingesetzt wird, eignet sich das WGS84 Referenzellipsoid nicht zur lokal bestmöglichen Annäherung des Geoiden. Aus diesem Grund gibt es für jedes Land einen lokal bestangepassten Referenzellipsoiden, der für Deutschland der Besselellipsoid ist.

Obwohl die Koordinaten, die sich bei der GK Notation Rechtswert, Hochwert und Höhe nennen intuitiv interpretierbar sind, ist ihre Entstehung nicht ganz tri-

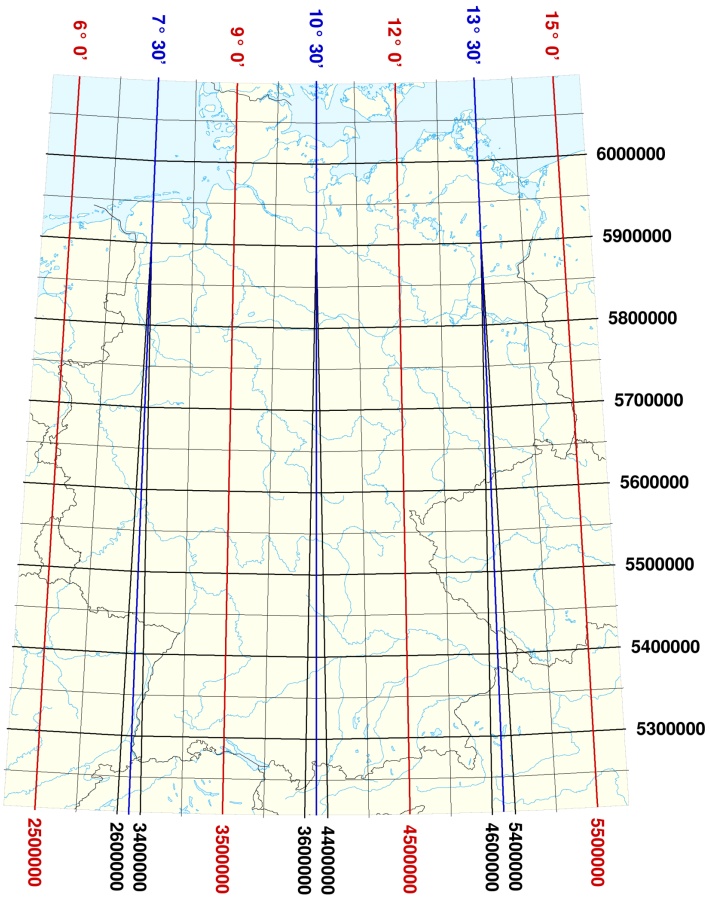

Abbildung 2.7: Gauß-Krüger Koordinaten am Beispiel der Karte von Deutschland (Quelle: [GPS])

vial. Denn eine Kugeloberfläche wie der Erdmantel kann nicht ohne Verzerrungen auf eine flache Ebene ausgebreitet werden. Dieses Problem behebt GK, indem es die Kugeloberfläche nur abschnittsweise in die Ebene überführt. Dazu wird der Äquator als x-Achse und jeder glatt durch 3 teilbare Längengrad als y-Achse verwendet. Der Hochwert stellt also den Abstand zum Äquator dar. Leider verhält es sich beim Rechtswert nicht ganz so einfach. Denn dieser wird immer als Abstand zum nächsten Mittelmeridian angegeben. Da die Angabe in östlicher Richtung orientiert ist, kann der Rechtswert auch negativ werden. Damit dies nicht passiert, addiert man immer 500.000 Meter zum ermittelten Wert. Außerdem stellt

man dem Rechtswert den Index des verwendeten Mittelmeridians voran. Grafik 2.7 zeigt eine Karte von Deutschland mit den jeweiligen Mittelmeridianen. In den GK-Werten der Meridiane am unteren Bildrand sind die Indizes schon integriert.

Das oben genannte Beispiel, das die Spitze des Deutschen Ecks beschreibt, soll auch in diesem Fall wieder verwendet werden. Der Hochwert der Koordinate, die den genannten Punkt beschreibt, also der Abstand zum Äquator beträgt 5.581.938 Meter. Der Rechtswert muss vom 9. Längengrad aus gemessen werden, da die zu verortende Position knapp näher an diesem als am 6. liegt (zur Erinnerung: 7° 36'). Der Abstand beträgt östlich orientiert -99.070 Meter. Da es sich um einen negativen Wert handelt, werden 500.000 Meter addiert (400.930). Anschließend wird der Index des 9. Längengrades voraus gestellt, also die 3. So entsteht der endgültige Rechtswert: 3.400.930. Die Punkte sind nicht unbedingt nötig und dienen hier nur zur Erhöhung der Übersichtlichkeit. Daraus ergeben sich die Koordinaten der Spitze des Deutschen Ecks in Koblenz in Gauß-Krüger Notation: 3.400.930 5.581.938.

2.8.3 Universal Transverse Mercator

Universal Transverse Mercator (UTM) ist der Nachfolger von Gauß-Krüger im Vermessungswesen. Es arbeitet mit einem geringfügig anderen Referenzsystem und Referenzellipsoid. Die Projektion, die UTM verwendet, kann man sich als einen Zylinder vorstellen, der die Erde schneidet, so dass der relevante Kartenbereich mit geringen Verzerrungen auf die Außenhaut des Zylinders projiziert werden kann. Da eine solche Projektion für Objekte, die weit vom Kartenzentrum entfernt liegen, zu starken Verzerrungen führt, wird die Erdoberfläche ähnlich wie beim GK-System in Zonen unterteilt. Im Unterschied zum GK-System sind diese Zonen jedoch 6° breit.
Wurde die Erdoberfläche in die Ebene abgebildet, können Rechts- und Hochwert wie beim GK-System abgelesen werden. Der Abstand zum Äquator bildet den Hochwert, der Abstand zum Mittelmeridian bildet den Rechtswert. Ebenfalls wie bei Gauß-Krüger wird zum Rechtswert eine Konstante von 500.000 Metern addiert, um negative Werte zu vermeiden. Die ermittelten Rechts- und Hochwerte müssen jeweils mit dem einheitenlosen Maßstabsfaktor 0,9996 multipliziert wer-

den, der durch die Projektion auf die Ebene entstanden ist. Zusätzlich muss jedes Koordinatenpaar, um eindeutig zu sein, mit der Zonennummer versehen werden.

2.9 Formate

Die Übertragung von Daten zwischen einem Sender und einem Empfänger kann nur erfolgreich verlaufen, wenn beide Kommunikationspartner nicht nur über den gleichen Zeichenvorrat verfügen, sondern diese Zeichen auch auf die gleiche Weise interpretieren. Um dies zu gewährleisten, sind viele verschiedene Datenformate standardisiert worden. Jedes dieser Formate hat ein spezifisches Anwendungsgebiet, auf dessen Besonderheiten es bestmöglich angepasst ist. Beispielsweise bietet das Base64 Format als Kodier- und Dekodierverfahren den Vorteil, Binärdaten in ein Format umzuwandeln, das einen verringerten Zeichenvorrat zur Darstellung der Daten verwendet. So können nach Base64 kodierte Daten problemlos in Emails eingebettet werden, ohne dass Sonderzeichen, die eventuell in den Binärdaten enthalten waren, zu einem ungewollten Verhalten des POP3 Protokolls führen.

Für diese Arbeit sind zwei Formate besonders wichtig. Zum Einen das NMEA Format, mit dem Daten des GPS Empfängers kodiert sind, zum anderen das XML Format. Dieses findet in der vorliegenden Arbeit Verwendung, um die Daten, die die Clients sammeln und alle weiteren Informationen geordnet an den Server zu übertragen.

2.9.1 NMEA-0183

Das NMEA 0183 Format wurde schon 1983 von der National Marine Electronics Association veröffentlicht. Es definiert sowohl die technische Übertragung der Daten über eine serielle Schnittstelle mit 4800 Baud als auch die Codierung der Daten in ASCII Zeichenketten mit einer Maximallänge von 80 Zeichen.

Auf der technischen Ebene unterscheidet man die Sender, die auch NMEA Talker heißen und die Empfänger, die NMEA Listener genannt werden. In einer Kommunikationseinheit kann es mehrere Listener geben, allerdings nur einen Sender. Gäbe es mehrere, würden sich die gesendeten Daten auf dem gemeinsamen

Medium überlagern, da es bei der seriellen Schnittstelle kein Verfahren der Kollisionsüberwachung gibt. Sollen mehrere Sender in einem Netz betrieben werden, müssen diese physikalisch von einander getrennt werden. Ein Multiplexer, der über jeden seiner Eingänge auf die Daten genau eines NMEA Talkers hört, kann die empfangenen Sätze dann hintereinander auf das geteilte Medium senden.

Die Kodierungsebene von NMEA besteht aus einzelnen Sätzen, die Daten im ASCII Format enthalten. Das hat den enormen Vorteil, dass sie menschenlesbar sind, was wiederum den Aufwand, der zum Parsen und anschließenden Debuggen erforderlich ist, verringert.

Jeder Satz endet mit einem Asterisk und einer hexadezimalen Prüfsumme. Diese wird gebildet, indem alle Zeichen des Datensatzes exklusiv verodert werden. Ein Satz beginnt immer mit dem Dollarzeichen „$“, gefolgt von einer mehrteiligen Satzkennung, zum Beispiel *GPGSA*. Die ersten beiden Buchstaben der Satzkennung beschreiben die Geräteklasse des Senders, der den Satz erzeugt hat. In diesem Fall steht das *GP* für einen GPS Empfänger. Ein *LC* an dieser Stelle würde einen LoranC Empfänger identifizieren.

Die nächsten drei Zeichen beschreiben die Art des Datensatzes. Für diese Anwendung wertet das System die drei Sätze *GPRMC*, *GPGGA* und *GPGSA* aus. Welche Daten diese enthalten, zeigen die Listings unter [NMEA2].

2.9.2 XML

Die Extensible Markup Language[10] ist eine Auszeichnungssprache zur Beschreibung von Objekten. Sie bildet eine Baumstruktur, mit der hierarchisch organisierte Datenstrukturen dargestellt werden können. Die Daten werden dabei in einem menschenlesbaren Textdokument abgelegt, dessen Zeichenformat frei wählbar ist. So können auch Sonderzeichen in XML kodiert werden.

Wie aus Listing 2.1 hervorgeht, besteht die Sprache aus Elementen der Form <Element></Element>, die wiederum andere Elemente oder Attribute enthalten können. Das Listing zeigt, wie ein einfaches XML Dokument zur Beschreibung einer Anschrift aussehen könnte.

[10] engl. erweiterbare Auszeichnungssprache

```
<?xml version="1.0" encoding="UTF-8" ?>
<Anschrift>
        <Vorname>Christian</Vorname>
        <Nachname>Müller</Nachname>
        <Adresse>
                <Straße>Beispielstraße</Straße>
                <Hausnummer>42</Hausnummer>
                <PLZ>12345</PLZ>
                <Ort>Beispielhausen</Ort>
        </Adresse>
</Anschrift>
```

Listing 2.1: Beispiel für ein XML Dokument, das eine Anschrift beschreibt.

Alle XML Dokumente, die eine Anschrift auf diese Weise beschreiben, gehören zu der gleichen Klasse von Dokumenten. Um eine solche Klasse zu beschreiben, gibt es eine XML Metasprache namens XML-Schema. Entspricht eine XML Datei dem zugehörigen Schema, heißt sie gültig. Das Schema, das die oben genannte Datei beschreibt, würde folgendermaßen aussehen:

```
<?xml version="1.0" encoding="utf-8"?>
<xsd:schema version="1.0"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Anschrift">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Vorname" type="xsd:string" />
        <xsd:element name="Nachname" type="xsd:string" />
        <xsd:element name="Adresse">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Straße" type="xsd:string" />
              <xsd:element name="Hausnummer" type="xsd:int" />
              <xsd:element name="PLZ" type="xsd:int" />
              <xsd:element name="Ort" type="xsd:string" />
            </xsd:sequence>
```

```
            </xsd:complexType>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Listing 2.2: XML Schemadefinition für eine Anschrift

Anhand einer solchen XML-Schema Definition können alle Instanzen des vorgenannten Objekts vom Typ Adresse unabhängig von ihrem Inhalt auf syntaktische Richtigkeit überprüft werden. Bei einer maschinellen Verarbeitung der Daten ist dies besonders wichtig, denn ohne eine definierte Syntax, die beiden Kommunikationspartnern bekannt ist, können die eingelesenen Daten keinen semantischen Entitäten zugeordnet werden.

Der Vorteil von XML ist, dass die Sprache sowohl maschinell einfach zu parsen als auch sehr gut menschenlesbar ist. Deshalb werden XML Dialekte in vielen Bereichen eingesetzt. Nur einige Beispiele sind GEDXML in der Ahnenforschung, SOAP für Webservices, oder die Windows Presentation Foundation, deren Markup für das Layout von Windowsprogrammfenstern ebenfalls auf XML basiert. Für diese Anwendung ist einerseits die Verwendung von SOAP als speziellem Dienst, der das XML Format verwendet als auch die generelle Fähigkeit der Sprache, serialisierte Programmobjekte darzustellen, besonders interessant. Weitere Informationen zu XML liefert [VONH09]. An dieser Stelle sollen die Anwendungsgebiete von XML betrachtet werden, die für die vorliegende Arbeit relevant sind.

SOAP[11] ist ein Protokoll zur Realisierung von Webservices. Zum Transport benutzt es den Internetprotokollstapel. Basierend auf den Übertragungsdiensten von TCP/IP werden SOAP Nachrichten in das HTTP(S) Protokoll eingebettet. Es ist also schon auf der Übertragungsebene eine Transportverschlüsselung möglich. SOAP verwendet zur Strukturierung und Darstellung der zu übertragenden Daten XML. Eine SOAP Nachricht besteht aus einem Umschlag (envelope), der einen Header und einen Body enthält und könnte etwa so aussehen:

[11] SOAP stand einmal für Simple Object Access Protokoll, dieses Akronym wird aber heute nicht mehr verwendet.

```
<?xml version="1.0"?>
<soap:Envelope
    xmlns:soap="http://www.w3.org/2001/12/soap-envelope" >
        <soap:Header>
        </soap:Header>

        <soap:Body>
        </soap:Body>
</soap:Envelope>
```

Listing 2.3: Aufbau einer SOAP Nachricht

In dieser Struktur können Daten eingebettet werden, um zum Beispiel eine Anfrage an einen Webservice zu stellen. Diese Anfrage kann auf dem Webservice eine Methode starten und die dafür erforderlichen Parameter enthalten. Nach der Abarbeitung der Methode durch den Webserver sendet dieser den Rückgabewert als HTTP Response ebenfalls eingebettet in eine SOAP Nachricht zurück. Diese Technik erlaubt das Erstellen äußerst flexibler Webdienste und auch deren einfache Verwendung in der Programmierung, denn Programmobjekte lassen sich leicht in ein XML Dokument serialisieren und umgekehrt.

Um einen Webservice ansprechen zu können, muss das aufrufende Programm natürlich gewisse Kenntnisse über den Dienst haben. Hierzu gehören die Namen der aufrufbaren Methoden und deren Menge und Art der Parameter. Zur Beschreibung eines Webservice gibt es deshalb die Web Service Description Language (WSDL). Mit Ihrer Hilfe beschreiben Webservicedienstleister die von ihnen zur Verfügung gestellten Dienste. Dazu enthält ein WSDL Dokument verschiedene Bereiche, in denen die verfügbaren Methoden und deren Parameter, die verwendeten Nachrichtentypen, die Protokolle und so weiter aufgelistet werden. Das .NET Framework bietet mit den beiden Programmen SVCUTIL.EXE und NETCFSVCUTIL.EXE eine sehr einfache Möglichkeit, von einem verfügbaren Webservice eine WSDL zu erstellen und diese sogar noch weiter in eine clientseitige Programmschnittstelle zu überführen, die direkt aus dem Code angesprochen werden kann. Wie ein solcher Aufruf aussehen kann, zeigt Kapitel 4.10.

XML wird in dieser Arbeit ebenfalls zum Speichern von Anwendungseinstellungen des Clientprogramms verwendet. Wie Kapitel 4.2.7 beschreibt, erzeugt der Client beim Beenden eine Konfigurationsdatei, die die aktuellen Einstellungen enthält, indem sie ein Objekt vom Typ ConfigItem über einen XmlSerializer in eine XML Datei serialisiert. Das Kapitel enthält auch ein Beispiel einer solchen Datei.

2.10 Sicherheit

Im Umgang mit personenbezogenen Daten wie den über dieses System ermittelten Standortinformationen kann die Bedeutung der Sicherheit des Systems nicht hoch genug angesetzt werden. Erhält ein Unbefugter Zugriff auf die gesammelten Positionen oder kann er auch nur die Kommunikation zwischen einem Sender und dem Server belauschen, erhält er umfangreiche Informationen über das Bewegungsverhalten des oder der Clients und kann daraus sehr einfach auf deren Bewegungsmuster schließen. Da sich solche Daten auf die vielfältigsten Arten wirtschaftlich nutzen lassen, besteht ein begründetes Interesse, sie so gut wie möglich zu schützen. Zu den möglichen Nutzungsarten könnte ein vergleichsweise harmloser Missbrauch für unerwünschte ortsbezogene Werbung gehören, aber auch verschiedenste kriminelle Vorhaben.

2.10.1 Sicherheitsanforderungen

In [GRIM05] werden die folgenden Punkte als Sicherheitsanforderungen für eine digitale Kommunikation genannt:

1. **Integrität** - Die Daten sind vollständig und unverändert.
2. **Authentizität** - Die Echtheit einer Person oder eines Dienstes muss nachvollziehbar sein.
3. **Anonymität** - Der Urheber einer Handlung darf nicht nachvollziehbar sein.
4. **Vertraulichkeit** - Die Daten dürfen nur von autorisierten Kommunikationsteilnehmern eingesehen werden.

5. **Verfügbarkeit** - Der Zugriff auf die Daten muss zu jedem Zeitpunkt und von jedem Ort aus gewährleistet sein.

6. **Nicht-Abstreitbarkeit** - Der Urheber einer Handlung muss beweisbar sein.

Auf den ersten Blick erscheinen diese allgemein definierten Sicherheitsanforderungen teilweise in direktem Widerspruch zu einander zu stehen. Allerdings verhält es sich nur auf den ersten Blick so. Denn ein hinreichend umfangreiches System besteht meist nicht nur aus einer Transaktion, sondern aus mehreren. Jede dieser Transaktionen besitzt ihre eigenen Sicherheitsanforderungen, die sich durchaus von denen der anderen unterscheiden können.

2.10.2 Hashs

Ein Hash ist das Ergebnis der Anwendung einer Hashfunktion h auf einen Quellwert m also $m \longrightarrow h(m)$. Die Aufgabe einer Hashfunktion besteht darin, weit streuende Quellwerte von unterschiedlicher Länge in einen Zielwert zu überführen, der eine feste, meist erheblich geringere Länge hat als der Ausgangswert. Eine gute Hashfunktion verteilt die Hashs unterschiedlicher Nachrichten möglichst gleichmäßig über den Wertebereich. Man nennt den Hash auch den Fingerabdruck des Wertes, da er zwar üblicherweise vollkommen abstrakt ist und keine verwertbaren Informationen mehr enthält, die Rückschlüsse auf den Ausgangswert zuließen, er aber stellvertretend für den Ausgangswert stehen kann. Denn die Anwendung einer Hashfunktion auf zwei gleiche Ausgangswerte wird immer zum gleichen Hash führen.

Für die Kryptographie spielen Hashfunktionen eine besondere Rolle, da bei der Erzeugung des Hashs so viele Informationen verloren gehen, dass der Schluss vom Hash auf den Ausgangswert unmöglich wird, der Hash aber trotzdem eindeutig einem bekannten Ausgangswert zugeordnet werden kann. Hashfunktionen sind eine einfache Art, Passwörter sicher zu speichern. Dazu speichert man nicht das Passwort selbst, sondern seinen Hash. Somit ist es auch einem Angreifer mit Zugang zum Passwortspeicher nicht mehr möglich, das Passwort zu entziffern. Wird allerdings das richtige Passwort eingegeben und mit der gleichen Hashfunktion kodiert wie beim erstmaligen Speichern des Passwortes, wird der gleiche Hash

entstehen. Die beiden Passwörter stimmen also überein und das System kann die Anmeldung als erfolgreich betrachten. Kryptographische Hashfunktionen müssen die folgenden drei Kriterien erfüllen:

1. Es muss schwer sein, zu gegebenem m ein m' zu finden mit $h(m') = h(m)$.
2. Es muss schwer sein, überhaupt ein Wertepaar m und m' zu finden mit $h(m) = h(m')$.
3. Es muss sogar schwer sein, ein Wertepaar m und m' zu finden, deren Hashs in einem verwertbaren Zusammenhang stehen.

„Schwer“ bedeutet in diesem Fall, dass die notwendigen Berechnungen, die zum Lösen des Problems notwendig sind, in einer äußerst komplexen Aufwandsklasse anzusiedeln sind. Diese sollte möglichst $O(x^n)$ betragen. „Schwer“ ist in diesem Fall also ein Synonym für „möglichst unmöglich“. Hashfunktionen, die diese Forderungen erfüllen sind MD4, SqMOD n und der in dieser Arbeit zum Verbergen der Passwörter verwendete MD5.

2.10.3 RSA

Das RSA[12] Kryptosystem zählt zu den asymmetrischen Verschlüsselungsverfahren. Das bedeutet, dass zum Verschlüsseln eines Klartextes in ein Kryptogramm ein anderer Schlüssel verwendet wird, als bei der Wiederherstellung desselben Klartextes aus dem Kryptogramm. Hier zeigt sich auch schon der Unterschied zu den vorher vorgestellten Hashes: Aus einem Kryptogramm lässt sich der Klartext mit geeigneten Mitteln – in diesem Fall dem Schlüssel und dem richtigen Algorithmus – wieder herstellen, während beim Hashen einer Nachricht zu viele Informationen verloren gehen und nur ein Fingerabdruck des Klartextes übrig bleibt.

In Kapitel 2.10.5 über Public Key Infrastructures wird der Sinn von asymmetrischer Verschlüsselung im Gegensatz zu symmetrischer erklärt, hier soll mit

[12] benannt nach seinen Erfindern Ronald L. Rivest, Adi Shamir und Leonard Adleman

RSA ein einfach zu verstehendes asymmetrisches Verfahren erläutert werden. Es sei darauf hingewiesen, dass bei asymmetrischer Verschlüsselung ein Schlüsselpaar, bestehend aus einem öffentlichen und einem privaten Schlüssel verwendet wird.

RSA arbeitet wie die meisten Verschlüsselungsverfahren mit großen Primzahlen. Eine Schlüssellänge von 2048 Bit, die 2^{2048} verschiedene Schlüssel ermöglicht, gilt laut [GRIM05] momentan als sicher. Um einen Schlüssel zu erzeugen, wählt der Algorithmus zufällig zwei große Primzahlen p und q und errechnet daraus den öffentlichen Schlüssel n:

$$n = p * q$$

Die Sicherheit des gesamten Kryptosystems basiert darauf, dass die Primfaktorzerlegung der 2048 Bit langen Zahl n mit der Rechenleistung heutiger Computer nahezu unmöglich ist. Daraus ergibt sich, dass die Primfaktoren p und q geheim sein müssen. Im Gegensatz dazu ist die vom Algorithmus gewählte Konstante e zur Berechnung des privaten Schlüssels systemweit bekannt. Der private Schlüssel d wird mit Hilfe von e folgendermaßen berechnet:

$$d = 1/e \ (mod \ (p-1)(q-1))$$

Die Kurzform *mod* steht hier für den Modulooperator, also die Restbildung. Zur Restbildung wird das Produkt $(p-1)(q-1)$ verwendet, da es sich dabei laut der Eulerschen ϕ-Funktion um die Anzahl der zu n teilerfremden ganzen Zahlen kleiner n handelt. Diese Eigenschaft ist für die mathematische Abgeschlossenheit des Systems wichtig, wie aus dem letzten Schritt dieses Kapitels hervorgeht.

Nun liegen also der öffentliche Schlüssel n und der private Schlüssel d vor. Während mit dem öffentlichen Schlüssel Daten verschlüsselt, aber nicht wieder entschlüsselt werden können, ist das Entschlüsseln mit dem privaten Schlüssel möglich. Andersherum können mit dem privaten Schlüssel Daten signiert werden. Diese Signatur kann in einem Folgeschritt mit dem öffentlichen Schlüssel überprüft werden.

Um nun einen Klartext m zu verschlüsseln, verwendet der Sender den öffentlichen Schlüssel des Empfängers n_E und berechnet:

$$c = m^e \ (mod \ n_E)$$

Hierbei ist c das entstehende Kryptogramm. Der Empfänger kann dieses mit seinem privaten Schlüssel d_E wiederum entschlüsseln:

$$x = c^{d_E} \ (mod \ n_E)$$

Nach den weiter oben eingeführten Ver- und Entschlüsselungsregeln gelten also die nachfolgend aufgezeigten ersten drei Schritte, während nach dem kleinen Satz von Fermat mit Hilfe der Eulerschen ϕ-Funktion der vierte Schritt gilt:

$$m = c^d = (m^e)^d = m^{ed} = m(mod \ n)$$

Damit im vierten Schritt allerdings der genannte Satz angewendet werden kann, muss der private Schlüssel d auf eben die Methode errechnet werden, die weiter oben skizziert ist, denn $(p-1)(q-1)$ ist genau die Anzahl der zu n teilerfremden Zahlen kleiner als n und damit eben $\phi(n)$ nach Euler. Die hier präsentierten Informationen stammen aus [GRIM05].

2.10.4 Zertifikate

Im letzten Kapitel wurden öffentliche und private Schlüssel eingeführt. Mit diesen ist, wie später noch genauer ausgeführt werden wird sowohl eine verschlüsselte Kommunikation als auch die Schlüsselverteilung selbst über das gleiche Medium möglich. Doch bei der Benutzung eines öffentlichen Schlüssels stellt sich weiterhin die wichtige sicherheitsrelevante Frage, ob der Schlüssel, der zum Erzeugen des Kryptogramms benutzt wurde, tatsächlich dem zum Entschlüsseln berechtigten Empfänger gehört. Denn wenn hier eine Man-In-The-Middle Attacke stattgefunden hat, hat unter Umständen ein Angreifer dem Sender seinen eigenen öffentlichen Schlüssel untergeschoben.

Diese Frage kann mit Zertifikaten beantwortet werden. Ein Zertifikat bindet einen öffentlichen Schlüssel an eine Person, indem die Verbindung zwischen beiden zum Beispiel in einer XML Datei hergestellt wird und diese anschließend von einem vertrauenswürdigen Dritten digital signiert wird. Natürlich ist diese Signatur nur sinnvoll, wenn der unterzeichnende Dritte das Vertrauen des Senders genießt. Mit einem solcherart signierten Zertifikat könnte der Inhaber dann weitere vertrauenswürdige Zertifikate ausstellen, für deren Echtheit er mit seiner Signatur garantiert. Dieser Vorgang kann unendlich oft wiederholt werden. Um solche Vertrauensbeziehungen herzustellen, gibt es zwei Modelle:

Das erste baut die Vertrauenswürdigkeit in einem Netz gleichberechtigter Teilnehmer auf. Indem jeder dieser Teilnehmer den Kommunikationspartnern, die er kennt, sein Vertrauen ausspricht und deren Schlüssel zertifiziert, entstehen viele kleine Zellen, in denen Vertrauen herrscht. Da jeder Teilnehmer üblicherweise Kommunikationsbeziehungen zu verschiedenen Partnern pflegt, bildet sich so ein Netz des Vertrauens (Web of Trust). Soll eine neue Kommunikationssitzung mit einem bisher unbekannten Partner initiiert werden, kann der Sender das Zertifikat des Empfängers überprüfen. Findet er eine Unterschrift von einem von ihm selbst als vertrauenswürdig eingestuften früheren Kommunikationspartner, liegt die Entscheidung nahe, auch dem neuen Partner zu vertrauen.

Dieser Ansatz ist in kleinen Gruppen durchführbar, wenn es genaue Regeln gibt, wie ein neues Zertifikat und dessen Zuordnung zu einem Teilnehmer zu prüfen ist. Denn die Frage, die sich bei diesem Ansatz stellt ist die, ob Vertrauen übertragbar ist. Dieses Problem lässt sich als Vertrauensrelation als mathematische Transitivität ausdrücken:

Folgt aus $A \Rightarrow B$ und $B \Rightarrow C$ auch $A \Rightarrow C$?

Da diese Frage nicht allgemeingültig beantwortet werden kann, gibt es neben diesem Bottom-Up Modell auch noch ein anderes Vertrauensmodell. In diesem gibt es eine hierarchisch strukturierte Kette von Zertifizierern, die bei einer Wurzelzertifizierungsstelle beginnt, die wiederum von jeder Stelle im so entstehenden Zertifizierungsbaum aus direkt durch das Rückverfolgen der Zertifizierungshandlungen erreicht werden kann. Das dabei entstehende System von hierarchisch auf einander aufbauenden Zertifikaten bildet einen Teil einer Public Key Infrastructure.

Listing 2.4 zeigt ein sehr einfaches Beispiel für ein Zertifikat. Es enthält ein Element namens Inhalt, das einen Schlüssel und dessen Besitzer definiert. Über dieses Element wird ein Hash (Hash) gebildet und dieser dann signiert (Value). Die Signatur ist jederzeit überprüfbar, aber nicht imitierbar. Sobald ein Angreifer Veränderungen am Element Inhalt vornimmt, verändert sich der resultierende Hash und die Signatur ist inkorrekt. So lange aber die Signatur korrekt ist, können die Nutzer davon ausgehen, dass der Schlüssel, der im Zertifikat genannt wird, tatsächlich Michael Mustermann gehört.

```
<?xml version="1.0" encoding="UTF-8" ?>
<Zertifikat>
        <Inhalt>
                <Eigentümer>
                        <Vorname>Michael</Vorname>
                        <Nachname>Mustermann</Nachname>
                </Eigentümer>
                <Schlüssel>
                        <ID>01234567890123456789</ID>
                </Schlüssel>
        </Inhalt>
        <Signatur>
                <Signierer>Trend</Signierer>
                <Hash>a38ö8f8ajflgbeqxc9v</Hash>
                <Value>6ett53xcv8th8seqi3</Value>
        </Signatur>
</Zertifikat>
```

Listing 2.4: Beispielhafter Aufbau eines Zertifikats

2.10.5 Public Key Infrastructure

Nachdem in den letzten Kapiteln die Erzeugung und Verwendung eines Schlüsselpaares und dessen Bindung an einen Namen abgehandelt wurden, folgt in diesem Kapitel die Beschreibung der Infrastruktur, über die Schlüsselpaare verteilt werden. Ein großes Problem symmetrischer Verschlüsselungsverfahren ist die Über-

mittlung des Schlüssels. Möchte der Sender einer Nachricht diese verschlüsseln, verwendet er den gleichen Schlüssel, den auch der Empfänger zum Entschlüsseln braucht. Hier zeigt sich das Problem das synchronen Verschlüsselung: Über den gleichen Kanal wie die Nachricht kann der Schlüssel nicht versendet werden, denn ein Angreifer, der die Nachricht abfängt, würde mit hoher Sicherheit auch den Schlüssel abfangen. Er müsste also über einen anderen Kanal gesendet oder persönlich übergeben werden. In Fällen, in denen die räumliche Distanz zwischen Sender und Empfänger zu groß ist, wie im Internet üblich, entfällt diese Möglichkeit. Hier könnte eine Overlayinfrastruktur das Problem lösen. Während die Nachricht über das Internet übermittelt wird, könnte der Schlüssel über das Mobilfunknetz verteilt werden. Doch hier entstehen weitere Probleme. Es entsteht ein Medienbruch, denn der Schlüssel muss fehlerfrei vom Mobilfunkendgerät zum Internetendgerät gelangen, man braucht die Mobilfunknummer jedes Kommunikationspartners und es entstehen weitere Kosten.

Es muss also eine Möglichkeit geben, einen Schlüssel sicher über das gleiche Medium zu übertragen wie die Nachricht. Natürlich könnte man den Schlüssel verschlüsseln, was aber das Problem nur auf die nächst höhere Ebene transponieren würde. Genau dieses Problem der Schlüsselweitergabe löst eine Public Key Infrastrukture in Verbindung mit asymmetrischer Verschlüsselung.

Die PKI basiert auf dem Prinzip der asymmetrischen Verschlüsselung und damit auf der Verwendung von öffentlichen und privaten Schlüsseln. Jeder Kommunikationsteilnehmer stellt seinen öffentlichen Schlüssel den anderen Teilnehmern zur Verfügung, während er seinen privaten Schlüssel geheim hält. Möchte nun ein Sender eine Nachricht verfassen, verschlüsselt er diese mit dem öffentlichen Schlüssel des Empfängers, der frei zugänglich ist. Ab diesem Zeitpunkt kann niemand, der nicht den privaten Schlüssel besitzt, die Nachricht öffnen. Selbst der Sender kann die Nachricht nicht mehr entschlüsseln. Nur der Besitzer des geheimen privaten Schlüssels kann die Nachricht dechiffrieren und ihren Inhalt einsehen. Somit ist es also möglich, Schlüssel ganz offen und für jeden Angreifer zugänglich über unsichere Netze zu verteilen, während die mit diesen Schlüsseln kodierten Kryptogramme absolut sicher sind.

Wie in den vorherigen Kapiteln beschrieben muss nun sicher gestellt werden, dass ein öffentlicher Schlüssel auch tatsächlich dem Empfänger gehört, der

als einziger befugt sein soll, die Nachricht zu entschlüsseln. Hierzu wird jeder Schlüssel über ein digital signiertes Zertifikat an den Besitzer gebunden. Für das Ausstellen der Zertifikate gibt es eine spezielle vertrauenswürdige Certification Authority. Sie besitzt selbst ein von der ihr übergeordneten CA signiertes Zertifikat und ist damit vertrauenswürdig. Mit dem Schlüssel, der zu dem signierten Zertifikat gehört, kann die Certification Authority wiederum die Zertifikate der ihr nachgeordneten Instanzen signieren. So entsteht eine Baumstruktur, deren Wurzel von jedem Knoten aus durch das Zurückverfolgen der Zertifikate erreicht werden kann. Es entstehen also Zertifikatsketten, die so genannten Ketten des Vertrauens (Chains of Trust). Eine PKI stellt weiterhin Einrichtungen zum Beantragen, Prüfen, Finden und Widerrufen von Zertifikaten bereit. Es handelt sich also um ein geschlossenes System unter der administrativen Kontrolle der Wurzel-Certification Authority.

Eine PKI eignet sich hervorragend, um die Kommunikation zwischen Personen, aber auch Maschinen in einem Netzwerk zu sichern, da sie einerseits Datenintegrität durch Verschlüsselung und andererseits durch die feste Zuordnung von Namen oder Adressen zu bestimmten Schlüsseln Originalität gewährleistet.

Kapitel 3

Konzeptionelle Planung

Nachdem im vorherigen Kapitel die Grundlagen für die Entwicklung einer Verortungslösung für die LBS-Architektur aus [STEI10] erläutert wurden, folgt in diesem Kapitel die Planung der Implementierung des Prototypen.

Dazu enthält das Kapitel 3.1 eine Ist-Analyse der zur Zeit auf dem Markt vorherrschenden Technologien. Im Kapitel 3.2 werden dann die Anforderungen an den Prototypen definiert.

3.1 Ist-Analyse

In diesem Kapitel soll untersucht werden, welche Voraussetzungen für die Realisierung des Prototyps und die Bearbeitung der in der Fragestellung definierten Aufgabenschwerpunkte bestehen. Dazu werden sowohl die verfügbaren Systemplattformen betrachtet als auch verwandte Arbeiten. Danach folgt in Kapitel 3.2 die Planung des Prototypen. Das Kapitel gibt einen Überblick über die Designentscheidungen und begründet die für die Implementierung getroffene Technologiewahl.

3.1.1 Mobile Betriebssysteme

Der Markt für mobile Betriebssysteme ist aufgrund der hohen Entwicklungsgeschwindigkeit der mobilen Hardware einem sehr schnellen Wandel unterworfen. Auch der Anspruch der Anwender nach neuen Funktionalitäten zwingt die Be-

triebssystementwickler, in kurzen Zeitabständen neue Versionen ihrer Systeme zu veröffentlichen. Momentan (Stand 3. Quartal 2010) halten die folgenden drei Systeme den größten Marktanteil.

3.1.1.1 Windows Mobile

Windows Mobile liegt in der aktuellen Version 6.5 vor. Programme für dieses System werden in .NET programmiert. Hierzu bietet Microsoft die Entwicklungsumgebung Visual Studio an. Die aus dem Desktopbereich bekannte Klassenbibliothek .NET Framework liegt für Windows Mobile Geräte in einer funktionsreduzierten Version als .NET Compact Framework vor. Um spezielle Hardwarefunktionen wie die Umgebungssensoren ansprechen zu können, sind im Internet freie Treiberimplementierungen erhältlich.

Weiterhin werden im 3. Quartal 2010 die ersten Smartphones mit Windows Mobile 7 erwartet. Programme für dieses Betriebssystem werden wie bei den Vorgängerversionen mit Visual Studio erstellt. Sie verwenden die .NET Programmiersprache C# und zur Darstellung Silverlight.

3.1.1.2 Android

Programme für das von Google entwickelte Smartphonebetriebssystem Android werden in JAVA mithilfe von Googles Android SDK entwickelt. Die Entwicklungsumgebung bietet einen komfortablen Zugriff auf Hardwarefunktionen, so dass keine zusätzlichen Klassenbibliotheken von Drittanbietern notwendig sind. Android liegt aktuell in der Version 2.2 vor.

3.1.1.3 iPhone OS

Das von Apple entwickelte iPhone OS, das aktuell in Version 4.1 auf dem Markt ist, unterscheidet sich von den vorgenannten in drei Punkten:

Zum Einen ist es nicht multitaskingfähig. Das bedeutet, dass zu einem Zeitpunkt immer nur ein Programm auf dem Gerät aktiv sein kann. Zum Anderen ist iPhone OS ausschließlich für Geräte der Marke Apple verfügbar. Das bedeutet für die Betriebssystementwickler eine zum Entwicklungszeitpunkt genau definier-

te Hardwareplattform. Dadurch ist das System besonders gut an die Fähigkeiten der Hardware angepasst.

Der dritte Unterschied zu den vorher genannten Systemen besteht in der Art, wie Programme verteilt werden. Die mithilfe des iPhone SDK entwickelten Programme können nur über den von Apple betriebenen App Store an die Anwender verteilt werden. Um in den App Store aufgenommen zu werden, muss ein Programm jedoch zuerst vom Betreiber freigegeben werden. Dies dauert etwa eine Woche und ist mit zusätzlichen Kosten verbunden. Durch die offizielle Validierung und die umfangreiche Klassenbibliothek haben alle Anwendungen ein einheitliches Look-and-Feel.

3.1.2 verwandte Arbeiten

Aufgrund des hohen Bedarfs an Verortungslösungen in den verschiedensten Bereichen existieren schon einige interessante Lösungen, von denen diese Arbeit in vielen Bereichen profitiert. Im Folgenden werden schon vorhandene Arbeiten vorgestellt, die bei der Realisierung des hier definierten Systems wichtige Fragestellungen ebenfalls aufgreifen oder eventuell sogar schon auf ähnliche Weise beantwortet haben.

3.1.2.1 Placelab

Das Projekt Placelab [PLAB] wurde 2003 publiziert (vergl. [LCYC05]) und bis 2006 fortgeführt. Es beschäftigt sich mit der Verortung mobiler Clients. Zum damaligen Zeitpunkt waren GPS-Empfänger in Laptops und Smartphones aufgrund ihrer Größe, ihres hohen Energieverbrauchs und ihres Preises kaum verbreitet. Um trotzdem ortsbezogene Dienste nutzen zu können, bedienten sich die Entwickler der Signale, die die zu diesem Zeitpunkt verfügbaren Geräte empfangen konnten.

Dies waren die Signale von WLAN-Accesspoints, Bluetoothstationen und GSM-Basisstationen. Wie eine Verortung anhand der Signale dieser Stationen erfolgen kann, beschreibt Kapitel 2. Im Laufe des Placelab-Projekts entstand eine umfangreiche Datenbank mit Positionen der verschiedenen Referenzstationen. Aus der Signalstärke der jeweils empfangenen Signale schätzt Placelab clientseitig eine Position. Das Projekt ist hier aufgeführt, da es für die Erstellung seiner

Datenbank einen Communityansatz verwendete. Das bedeutet, dass jeder Nutzer ständig die ermittelten Stationen mit seiner lokalen Datenbank abgleicht. Werden bisher unbekannte Stationen gefunden, werden diese mit einer Positionsschätzung gespeichert und beim nächsten Synchronisationsvorgang an den zentralen Datenbankserver des Projekts übertragen.

3.1.2.2 GoogleMaps

GoogleMaps ist ein einfaches aber beliebtes Navigationsprogramm für Smartphones und den PC. Es stellt dem Anwender Satellitenbilder und Kartenansichten zur Verfügung, auf denen auf Wunsch die Position des Nutzers angezeigt wird. Zur Ermittlung der Position kann entweder ein GPS-Empfänger verwendet werden oder das Programm liest die Daten des GSM Moduls aus. Durch das Abhören des Kanals 221 im Netz von O_2 oder das Nachschlagen des eindeutigen Bezeichners einer GSM Basisstation kann so eine für die Kartenanwendung ausreichend genaue Positionsschätzung erzeugt werden. Weitere Informationen über die Genauigkeit der Positionsschätzung alleine mit GSM Daten befinden sich in Kapitel 2.5.5. GoogleMaps ist ein Closed Source Programm[1] und kann deshalb in dieser Arbeit nicht zur Positionsbestimmung beitragen.

Um die Verortungsgenauigkeit von GoogleMaps zu steigern, werden Daten von öffentlich empfangbaren WLAN-Accesspoints über den Dienst Skyhook ([SKY2010]) verwendet. Das Unternehmen Google hat dafür die Fahrzeuge, die Bilder für den Kartendienst StreetView aufnehmen, mit WLAN Empfängern ausgerüstet und die Positionen der registrierten Signalquellen in einer Datenbank festgehalten (vergl. [FAZ2010]). Dieses Vorgehen sorgte für große datenschutzrechtliche Bedenken bei den zuständigen Behörden und bei vielen Nutzern.

3.1.2.3 MagicMap

MagicMap ist ein System zur „Kooperativen Positionsbestimmung über WLAN“ [MMAP], das an der Humboldt-Universität in Berlin entwickelt wird. Es verknüpft die eigene auf Lateration (vergl. Kapitel 2.3.2 basierende Positionsschätzung mit denen benachbarter Clients über ein Peer-to-Peer Netzwerk. MagicMap

[1] Im Gegensatz zu Open Source Software, bei der der Quelltext verfügbar ist

verwendet probabilistisches Bewegungsmapping ähnlich des in dieser Arbeit verwendeten Partikelfilters. Das System eignet sich besonders gut für Setups mit vielen Clients ohne Berücksichtigung des Anonymitätsgedankens der einzelnen Nutzer. Anwendungsbereiche sind beispielsweise der Betriebshof eines Verkehrsunternehmens oder Verladehöfe.

3.1.2.4 Studienarbeit: Fusion von WLAN- und kameragestützten Positionsdaten

Das Ziel dieser Arbeit, die in [WICK05] als Studienarbeit veröffentlicht wurde, ist es, ein System zu entwickeln, „*das anhand von WLAN-Informationen (Signalstärken und Position von WLAN–Access–Points) sowie anhand von Bildfolgenanalyse die eigene Position berechnet.*„

Da die Software aus dem Bereich der Bildverarbeitung stammt, misst sie der Verortung durch WLAN Signale einen geringen Stellenwert bei. Anhand der über das WLAN Modul gewonnenen Informationen wird die Positionsschätzung, die per Bildfolgeanalyse gewonnen wird, nur verifiziert. Hierzu ermittelt ein einfacher Filter aus der Veränderung der Signalstärken der empfangenen Accesspoints Bewegungsvektoren, die mit den Ergebnissen der visuellen Beobachtungen verglichen werden.

3.1.2.5 Studienarbeit: Positionsdatenintegration im Indoor- und Outdoorbereich mithilfe eines probabilistischen Filters

Kern dieser Arbeit, die in [HEBE07] als Studienarbeit veröffentlicht wurde, ist die Verortung mobiler Clients in vorher unbekannten Umgebungen. Hierzu sammelt das System Daten von einem GPS Empfänger und einer WLAN–Karte und verfolgt die Positionsschätzungen mithilfe eines Partikelfilters. Im Gegensatz zur vorliegenden Arbeit basierte das System jedoch nicht auf Smartphones, die als Clients mit einem Server kommunizieren, sondern es lief als eigenständige Anwendung auf einem Laptop.

Die Erfahrungen dieser Arbeit flossen in die Entwicklung des nun vorliegenden Systems insofern ein, als dass die Implementierung des Partikelfilters komplett

übernommen werden konnte und nach einer Portierung von JAVA nach C# sofort einsatzfähig war.

3.2 Sollkonzeption

In diesem Kapitel werden nach einander die einzelnen Teilaspekte der Implementierung betrachtet und die notwendigen Designentscheidungen erklärt. Kapitel 3.2.1 betrachtet die verwendete Plattform. Danach wird die Auswahl der Signalquellen und die zur Positionsbestimmung verwendeten Technologien beschrieben. Anschließend wird die angestrebte Architektur des Systems betrachtet.

3.2.1 Plattform

Die gesamte Architektur des Prototypen wird in einer Microsoft Windows Umgebung entstehen. Dies hat mehrere Gründe. Auf der Clientseite bietet das Betriebssystem Windows Mobile die größte Freiheit bei der Implementierung und Veröffentlichung von Programmen und einen einfachen Zugang zu Systemfunktionen, um die Signale der verschiedenen Sensoren auszuwerten. Zum Anderen ist durch die integrierte Entwicklungsumgebung Microsoft Visual Studio eine parallele und von der IDE[2] unterstütze Entwicklung von Client und Server möglich. So können beispielsweise Webserviceschnittstellen direkt aus Bibliotheken extrahiert werden.

Der Server wird als Anwendung auf einer virtuellen Maschine realisiert. Mit Microsoft Windows als Betriebssystem kann das System über eine Remotedesktopverbindung angesprochen und wie ein lokales System durch eine grafische Benutzerschnittstelle administriert werden.

3.2.2 Signalquellen und Verortungstechnik

Zur Bestimmung der Position des mobilen Systems werden folgende Signalquellen und Verortungstechniken herangezogen: Als primäre Quelle für Positionsschätzungen wird das in den meisten Smartphones und PDAs vorhandene GPS

[2]Integrated Development Environment

verwendet. Es arbeitet mit Lateration, die von der Gerätehardware selbst durchgeführt wird. Die Software greift auf die fertig erstellten Positionsschätzungen und weitere Kontextinformationen zu. Da GPS nicht überall nutzbar ist, werden für Gebiete, in denen kein GPS-Empfang verfügbar ist, andere Signalquellen benötigt. Hierzu bietet sich der in den meisten Geräten vorhandene WLAN-Empfänger an. Weil von einer ausreichend hohen Dichte an WLAN Accesspoints ausgegangen werden kann, wird die Positionsschätzung anhand von Lateration durchgeführt. Die Signalstärke nimmt mit der Entfernung ab und ermöglicht so eine annäherungsweise Bestimmung der Entfernung zur Referenzstation. Zur Unterstützung werden weiterhin die ebenfalls in den meisten Geräten vorhandenen Bluetoothempfänger als Signalquelle verwendet. Da Bluetoothsender eine geringe Reichweite von meist unter 10 Metern haben, wird zur Positionsschätzung anhand von Bluetoothsignalen Proximation verwendet.

Weiterhin wird eine Komponente integriert, die anhand der IP-Adresse des Nutzers dessen Position schätzt. Dieses Verfahren eignet sich besonders, da bei der Nutzung von Datendiensten wie im vorliegenden Fall immer eine solche Adresse verfügbar ist, so dass die Verortung anhand der IP-Adresse, als Fallback dienen kann, wenn keine andere Signalquelle verfügbar ist. Zusätzlich wird eine Verortung anhand von Hauskoordinaten implementiert. Diese Technologie ermöglicht es auch Anwendern an stationären Terminals, auf ortsbezogene Dienste zuzugreifen, indem sie ihre Adresse eingeben und über die so ermittelte Hauskoordinate verortet werden. Für beide Verfahren wird ein Proximationsansatz gewählt.

Auf die Implementierung einer GSM-Verortungskomponente wird verzichtet, da ein Zugriff auf die hierzu notwendigen Gerätefunktionen vom Hersteller des Testgerätes nicht unterstützt wird und somit nur mit ungetesteten Programmbibliotheken von Drittanbietern möglich ist. Eine Implementierung in zukünftigen Versionen ist möglich. Die Verortung anhand von visuellen Markierungen wird ebenfalls nicht unterstützt werden. Hierzu wären zum Einen umfangreiche Kartographierungsanstrengungen nötig, um ein Gelände mit den benötigten Markierungen zu versehen. Zum Anderen würden solche Markierungen eine ständige Aktivität des Benutzers erfordern, um die Markierungen mit der Kamera zu erfassen. Odometrie ist mit einem Smartphone oder ähnlichen mobilen Geräten ausgeschlossen, da diese Geräte keinen Bewegungsapparat besitzen, sondern pas-

siv vom Benutzer bewegt werden.

Zur Verbesserung der Verortung werden zwei Konzepte eingesetzt. Zum einen handelt es sich um einfache Aneinanderreihung der Positionsschätzung, die von den unterschiedlichen Signalquellen ermittelt werden, von der erwartungsgemäß ungenauesten zur genauesten. So ist eine wenig rechenintensive Verortung für Anwendungen mit geringen Anforderungen an die Genauigkeit möglich. Als zweite Alternative wird ein Partikelfilter verwendet. Dieser erhält für seinen Gewichtungsschritt die anhand der verschiedenen Signalquellen ermittelten Positionen, die über eine Gewichtungsmatrix unterschiedlich genau in die abschließende Positionsermittlung eingehen.

3.2.3 Datenhaltung

Plan.Soll.Daten Zur Speicherung der anfallenden Positionen von Referenzstationen, Benutzerdaten und Hauskoordinaten wird eine Datenbank verwendet werden. Da PostgreSQL in der Standardinstallation schon Funktionen von ortsbezogene Dienste mitbringt, wird für den Prototypen eine PostgreSQL Datenbank verwendet werden.

Sie speichert die Positionen der Referenzstationen. Bisher unbekannte Stationen werden mit der Positionsschätzung in der Datenbank abgelegt, die der Server für das mobile System, dass die unbekannte Station registriert hat, ermitteln konnte. Ein Problem stellen hier mobile und damit nicht zwingend ortsfeste Sender wie die in Kapitel 2.5.3.1 beschriebenen Bluetoothsender dar. Werden diese automatisiert in die Datenbank der Referenzstationen aufgenommen, führen sie bei einer unbemerkten Positionsänderung zu falschen Verortungen.

Deshalb müssen automatisiert in eine Positionsdatenbank aufgenommene Sender vom Administrator gründlich geprüft werden, bevor sie als verifiziert gelten dürfen. Dies verhindert, dass nicht ortsfeste Sender aufgenommen werden und bei einer späteren Verwendung die Positionsschätzung verfälschen. Schließlich darf die Position des Senders in der Realität nicht von der gespeicherten Position abweichen, wenn sie zur Verortung weiterer Stationen verwendet werden soll.

Dies kann bei einem Laptop, der über Bluetoothhardware verfügt, jedoch nicht sichergestellt werden.

3.2.4 Technik und Architektur

Um den Client zu entlasten und nicht zu viel Rechenleistung auf dem Gerät zu benötigen, wird der Location-Provider als Client-Server System realisiert. Dabei kommt dem Client die Aufgabe zu, über die integrierten oder angeschlossenen Sensoren Signale von Referenzstationen zu sammeln und diese an den Server zu übertragen. Durch die Verlagerung der rechenintensiven Aufgaben auf einen Server wird die Clienthardware entlastet. Das ist notwendig, da die gleiche Hardware, die die Signale, die zur Verortung verwendet werden, sammelt, vorrangig auch zum Konsumieren der mobilen Dienste verwendet werden soll. Der Server nimmt die Daten, die die mobilen Clients senden, entgegen und integriert diese zu einer Positionsschätzung. Weiterhin bietet der Server Konfigurationsmenüs, um Referenzstationen und Benutzer zu verwalten.

Die Kommunikation zwischen Client und Server erfolgt über einen Webservice. Zur Implementierung eines solchen bietet wiederum die Windows Communication Foundation hervorragende Werkzeuge. Die Nachrichten, die zwischen Client und Server ausgetauscht werden, bestehen aus .NET Objekten, die für die Übertragung in XML Dokumente serialisiert werden. Als Verschlüsselungsmethode kommt die in .NET integrierte PKI Implementierung zum Einsatz.

Da die gesamte Metaarchitektur das Gauß-Krüger Format für Positionsangaben verwendet, wird dieses auch im Location-Provider eingesetzt. Positionen, die in anderen Formaten angegeben werden, werden von der Programmlogik so schnell wie möglich umgewandelt. Durch die Verwendung des Gauß-Krüger Formats für Positionsangaben können Entfernungen ohne Umrechnungen direkt durch eine Differenzbildung zweier Datensätze in Metern angegeben werden. Auch die Daten, die von Vermessungsämtern und anderen offiziellen Stellen zur Verfügung gestellt werden, liegen im GK-Format vor, so dass auch hier keine Umrechnung nötig wird.

Sowohl der Aufbau des Clients als auch der des Servers wird modular erfolgen. Das bedeutet, dass jede Funktion in einem Paket oder einer Klasse gekapselt

wird. So ist jede Funktion für sich in anderen Entwicklungen wiederverwendbar. Die einzelnen Module kommunizieren unter einander mit Hilfe von Events, die ein zentraler Dispatcher an die jeweiligen Empfänger weiterleitet. So ist es möglich, bei Bedarf weitere Funktionen nachzurüsten, ohne den Aufbau des Systems grundlegend ändern zu müssen.

Kapitel 4

Implementierung

Während die bisherigen Kapitel der Arbeit die Grundlagen dieser Implementierung sowohl aus theoretischer als auch aus praktischer Sicht betrachtet haben, behandelt dieser Abschnitt die Implementierung des Systems. Es besteht aus einem Client und einem Server, die über eine TCP Verbindung Daten austauschen. Der Client registriert Signale aus seiner Umgebung und sendet sie an den Server. Dieser errechnet anhand der vom Client empfangenen Signale dessen Position und speichert diese in einer Datenbank. Außerdem bietet der Server eine Schnittstelle für die anderen Dienste der Infrastruktur, die Positionsdaten der Clients abfragen dürfen.

Wie in der Sollkonzeption in Kapitel 3.2 beschrieben, ist die Wiederverwendbarkeit der Teilsysteme sehr wichtig. Deshalb sind alle wichtigen Funktionen in Modulen gekapselt und liegen im Programmcode jeweils als eigene Pakete vor. Das hat den Vorteil, dass Benutzer, die später den Code weiterverwenden wollen, nur die Bibliothek in ihren Code importieren müssen und sofort auf die Funktionalität zugreifen können.

Die Aufteilung in Module macht auch die Erweiterung der Funktionalität wesentlich einfacher. Sollen später noch weitere Verortungstechniken implementiert werden, können diese einfach zu den bereits vorhandenen hinzugefügt werden. Es sind im bestehenden Code nur wenige Veränderungen nötig.

Kapitel 4.2 beschäftigt sich mit dem Aufbau und der Funktion des Clients und seiner Module. Gleiches gilt für das folgende Kapitel bezüglich des Servers. Die Kapitel 4.4 und 4.6 befassen sich zum einen mit der Kommunikation inner-

halb der einzelnen Programme und zum anderen mit dem Nachrichtenaustausch zwischen Client und Server, während das Kapitel 4.7 Betrachtungen über die Sicherheitsvorkehrungen enthält, die unautorisierte Logins und Angriffe auf die Sicherheit der übertragenen Daten verhindern sollen.

Die Implementierung eines Systems, das sich mit so vielen unterschiedlichen Bereichen beschäftigt wie es bei diesem der Fall ist, fällt leichter, wenn man auf schon erstellte Bibliotheken für verschiedene Aufgaben zurückgreifen kann. Einen Überblick der verwendeten Fremdsoftware gibt Kapitel 4.9. Und auch für die Weiterverwendbarkeit dieses Systems oder spezieller Teile ist alles vorbereitet. Wie eine Implementierung der notwendigen Bibliotheken aussehen könnte, zeigt das letzte Kapitel dieses Teils der Arbeit.

4.1 Plattform

Die Versuchsplattform für den Prototypen des Servers besteht aus einer virtuellen Maschine mit dem Betriebssystem Windows XP SP3. Um .NET Anwendungen ausführen zu können, enthält das System das Microsoft .NET Framework 3.5. Die ermittelten Daten werden in einer PostgreSQL 8.4 Datenbank gespeichert.

Der Client läuft auf unterschiedlichen mobilen Endgeräten des Herstellers HTC. Als Betriebssystem kommt Microsoft Windows Mobile 6.1 zum Einsatz. Für mobile Endgeräte stellt Microsoft das .NET Framework in einer „Compact“ genannten, funktionsreduzierten Version bereit. Diese ist ebenfalls zum Betrieb des Clients notwendig.

Entwickelt werden sowohl Client als auch Server unter Microsoft Windows7 mit dem Microsoft Visual Studio 2008 Professional Service Pack 1. Zusätzlich ist die folgende Software notwendig:

- Microsoft Windows Mobile 6 SDK: Stellt Programmierschnittstellen, Bibliotheken und Emulatoren für die Programmierung der mobilen Geräte zur Verfügung.
- Microsoft Windows SDK: Enthält Entwicklertools zum automatischen Erzeugen von WCF[1]–Schnittstellen und Implementierungen.

[1] Windows Communication Foundation (siehe Kapitel 4.6)

4.2 Client

Als Client wird in dieser Arbeit der mobile Teil des Systems bezeichnet, der für das Sammeln von Informationen zuständig ist. Die Software ist in C# geschrieben und läuft auf Smartphones und PDAs. Da diese Geräte über vergleichsweise wenig Rechenleistung und Speicher verfügen, existiert eine speziell angepasste Version des Microsoft .NET Frameworks für mobile Geräte namens .NET Compact Framework. Es bietet weniger Möglichkeiten, mit der Hardware zu interagieren und hat nur eingeschränkte Netzwerkfunktionen, was speziell bei der Arbeit mit der Windows Communication Foundation (WCF) auffiel. So ist beispielsweise die spezielle HTTPS-Bindung „wsHTTPBinding“ auf Smartphones nicht verfügbar. Statt dessen muss die funktionsreduzierte „BasicHTTPBinding“ verwendet werden.

Wie in Grafik 4.1 dargestellt ist, besteht der Client aus mehreren Modulen, die über Events miteinander kommunizieren. Wie die Kommunikation innerhalb der Teilsysteme mithilfe des Eventsystem funktioniert, wird in Kapitel 4.4 beschrieben.

Wie aus Grafik 4.1 hervor geht, besteht der Client aus mehreren Collector-Modulen. Diese erfassen über einen ins Gerät integrierten Sensor Signale aus der Umgebung, mit deren Hilfe das System die Position des Clients bestimmen kann. Die Collector-Module leiten alle Ihre Messungen an das Sendermodul weiter. Dieses ist über eine WCF-Verbindung mit dem Server verbunden und überträgt alle Messungen zur Weiterverarbeitung an diesen. Zusätzlich gibt es eine grafische Benutzerschnittstelle, die Benutzereingaben entgegen nimmt und Debugdaten des Senders und der anderen Module darstellt.

Jedes Modul übernimmt eine spezielle Aufgabe, die in den folgenden Abschnitten erklärt wird. Durch die Aufteilung in verschiedene funktionale Teile kann ein Modul auch unabhängig von den anderen Programmteilen ausgeführt werden. So ist es möglich, die eigentliche Programmlogik als Bibliothek in andere Systeme einzubinden, was in Kapitel 4.10 näher beschrieben wird.

Ein weiterer Vorteil der modularen Struktur wird im Zusammenhang mit dem in Kapitel 1.1.2 beschriebenen Szenario einer automatisierten Ausstellungsführung mit Kontextinformationen zu den einzelnen Ausstellungsstücken deutlich.

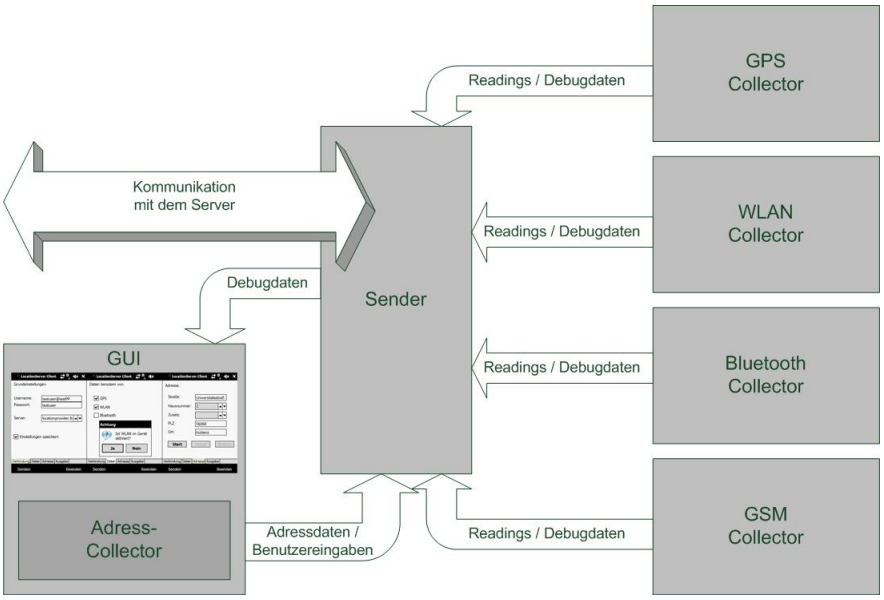

Abbildung 4.1: Aufbau des Location-Provider Clients

In diesem Zusammenhang ist die Implementierung spezieller Verortungsmeschanismen an den Exponaten sinnvoll, um präzisere Koordinaten der Benutzer zu erhalten. Der hier vorgestellte modulare Aufbau des Systems ermöglicht es, dem Client sehr leicht weitere Verortungstechnologien als eigenständige Module hinzuzufügen.

Damit sich die Module nicht gegenseitig blockieren, arbeitet jeder Programmteil in einem eigenen Thread. Die jeweiligen Algorithmen können also parallel arbeiten. Dies ist wichtig, da beispielsweise das Bluetoothmodul lange auf Antworten der Hardware wartet, so dass der gesamte Prozess blockieren würde, wenn die Funktionen nicht in eigene Threads aufgeteilt wären. Durch die einzelnen Verarbeitungsstränge ist es möglich, dass ein Thread in der Zwischenzeit den Prozessor und die übrigen Betriebsmittel belegen kann, während ein anderes Teilsystem auf eine Eingabe oder eine Hardwarereaktion wartet.

Diese Aufteilung hat jedoch den Nachteil, dass es zu Race Conditions zwischen den einzelnen Programmsträngen kommen kann. Eine Race Condition liegt vor, wenn n Programmteile auf eine gemeinsame Ressource zugreifen wollen und der Ausgang einer Berechnung, also der Zustand des Systems nach dem Zugriff davon abhängt, welcher Programmteil die Ressource wann erhalten hat. Da dies jedoch nicht vollständig vorhersagbar ist, entsteht hier ein undefinierter Zustand. Im

vorliegenden Fall äußert sich ein solcher undefinierter Zustand folgendermaßen: Durch die Unsicherheit, ob beispielsweise der *GPSCollector* seine neuen Daten beim Sender deponieren kann, bevor oder nachdem ein neues Paket über das Netzwerk gesendet wurde, könnte eine veraltete GPS Information übertragen werden. Deshalb ist der Empfänger des Pakets, also der Sender direkt an seine wichtigste Informationsquelle, das GPS Subsystem gekoppelt und sendet immer ein neues Paket, wenn vom *GPSCollector* ein neues Event ausgelöst wurde.

Wie schon anklang, ist die Kommunikation zwischen den Modulen unterschiedlicher Threads durch Events realisiert. Mit diesem Sachverhalt wird sich Kapitel 4.4 noch eingehender befassen.

4.2.1 GUI

Das graphische Benutzerinterface des Clients ist die Interaktionsschnittstelle zwischen dem Benutzer und der Programmlogik. Gleichzeitig ist sie das erste Modul, das beim Programmstart gestartet wird. Aus ihrem Konstruktor wird sofort beim Start eine Instanz des Senders erzeugt, der allerdings noch im Ruhezustand verbleibt, bis der Benutzer das Senden der Umgebungsinformationen von Hand startet. Gleichzeitig werden auch die verschiedenen Eventhandler (siehe Kapitel 4.4) verkettet. Zu diesem Zeitpunkt existiert die gesamte Anwendung nur in einem Thread. Zu einer Aufspaltung in mehrere Threads kommt es erst zu dem Zeitpunkt, an dem die Übertragung zum Server gestartet wird.

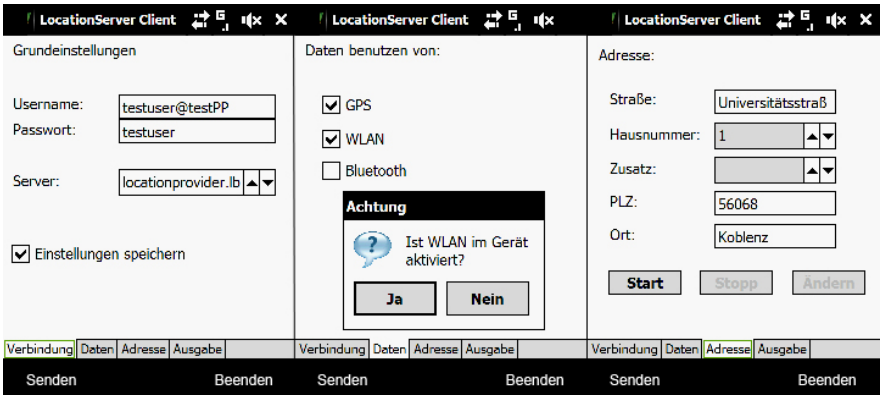

Abbildung 4.2: Graphisches Benutzerinterface des Clients

Die GUI besteht aus mehreren Tabs, die thematisch geordnet verschiedene Konfigurationsmöglichkeiten enthalten. Diese Tabs sind in Abbildung 4.2 nebeneinander dargestellt.

Die Einstellmöglichkeiten auf der ersten Seite steuern die Serververbindung. Hier kann der Benutzer seinen Login und das dazu passende Kennwort eingeben. Weiterhin ist es möglich, zwischen verschiedenen Location Providern zu wählen. Die letzte Einstellmöglichkeit auf dieser Seite beeinflusst die Speicherung der Konfigurationsdaten. Ist die Checkbox angewählt, werden alle Daten dem Konfigurationsmodul übergeben, das die Daten beim Beenden des Programms in eine XML Datei schreibt und beim nächsten Start wieder einliest.

Auf der nächsten Seite kann der Benutzer die bis jetzt implementierten Signalquellen auswählen. Setzt er ein Häkchen in eine der Checkboxen, fragt das Programm mit einer Messagebox nach, ob die jeweilige Funktion im Gerät aktiviert ist. Leider bietet das .NET Compact Framework keine Möglichkeit, aus managed Code[2] auf Hardwarekonfigurationen des Gerätes zuzugreifen. Sollte der Benutzer die Messagebox negativ beantwortet schließen, wird die jeweilige Signalquelle nicht verwendet und das bereits gesetzte Häkchen wieder entfernt.

Die dritte Seite dient zur Eingabe der Adresse, die zur Verortung unter Verwendung der Hauskoordinaten benutzt werden soll. Neben der Adresse kann der Benutzer hier die Verwendung der Adresse zur Verortung aktiv ein- und ausschalten. Dies ist notwendig, da nur der Benutzer selbst weiß, ob er sich zu einem bestimmten Zeitpunkt tatsächlich an der angegebenen Adresse befindet. Diese Information kann nicht über elektronische Hilfsmittel erlangt werden und erfordert somit eine direkte Interaktion des Nutzers.

Das letzte Tab beinhaltet keine weiteren Einstellmöglichkeiten, sondern dient als Debugausgabe. Hier werden sowohl der Loginvorgang als auch die verschiedenen Daten dargestellt, die an den Server übertragen werden.

Die Benutzerschnittstelle ist auf eine direkte Bedienung über den Touchscreen hin optimiert. Allerdings ist ein Start des Systems bei vorkonfigurierten Einstel-

[2]Managed Code bezeichnet Programmteile, die unter der Kontrolle der .NET Laufzeitumgebung stehen. Sie unterscheiden sich damit von hardwarenäherem Programmcode mit Treibercharakter, der ohne Eingriffsmöglichkeiten für die Laufzeitumgebung direkt in Maschinencode kompiliert wird.

lungen auch über die Softkeys möglich, die viele Geräte links und rechts unterhalb des Bildschirms haben.

4.2.2 Sender

Das in diesem System als Sender bezeichnete Modul ist ein Teilsystem jedes Clients. Es stellt einen Wrapper der WCF-Klassen dar, die zur Kommunikation mit dem Server benötigt werden und repräsentiert damit das gesamte Kommunikationssubsystem des Location-Provider Clients. Der Sender wird sofort beim Start der Anwendung erzeugt und erzeugt seinerseits sofort Instanzen aller Collectoren. Zu diesem Zeitpunkt existiert nur ein Thread. Die Aufspaltung in verschiedene Threads wird erst durchgeführt, wenn der Benutzer die Datenübertragung startet. Dies geschieht mit der Methode

```
public void startSending(ConfigItem config )
{
    client = new ClientSideClient(new BasicHttpBinding(), new
        EndpointAddress(...);
    sessionId = login();
    btColl.start();
    wlanColl.start();
    gpsColl.start();
    timer.Change(10, 1000);
}
```

Listing 4.1: Die Methode startSending der Senderkomponente

Listing 3.1 zeigt, wie die Methode zuerst eine neue Instanz des WCF-Clients erzeugt, der die Schnittstelle zum Server darstellt und dieser eine neue Bindung und die Serveradresse übergibt. Dann folgt der Loginvorgang, der in einer *sessionId* resultiert, anhand der sich der Client ab diesem Zeitpunkt dem Server gegenüber authentifiziert. Darauf folgt der Start der noch nicht aktiven Module. Ab jetzt laufen die Timer der Module, deren Ausführung jeweils in einen eigenen Thread springt. Zuletzt startet die Methode auch den eigenen Timer und verschiebt sich damit selbst in einen eigenen Thread. Wie aus dem Code hervorgeht, tickt der

Timer einmal pro Sekunde (1000 Millisekunden) und gibt damit den Anstoß, eine Datenübertragung zu initiieren.

Das Stoppen der Übertragung funktioniert entsprechend mit der Methode, die in Listing 3.2 dargestellt wird.

Das Codelisting zeigt, dass zuerst die Collectoren gestoppt werden, bevor der Sender seinen eigenen Timer durch die Angabe des Tickintervalls Timeout.Infinite abschaltet.

Aus den Listings geht hervor, dass der Sender der eigentliche Kern der Clientanwendung ist, da er als Kommunikationszentrum zwischen der GUI und den Collectoren bezüglich der Konfigurationsdaten dient und als Sammelstelle für die Datenübertragung zum Server. Das Eventsystem für die interne Kommunikation wird in Kapitel 4.4 beschrieben, während Kapitel 4.6 Informationen über die externe Kommunikation zwischen einer Sender-Instanz und dem Server enthält.

```
public void stopSending()
{
    btColl.stop();
    wlanColl.stop();
    gpsColl.stop();
    timer.Change(Timeout.Infinite, Timeout.Infinite);
    start = true;
}
```

Listing 4.2: Die Methode stopSending der Senderkomponente

Zur Übertragung der gesammelten Daten an den Server wird eine Instanz der Klasse InfoContainer erzeugt. Dazu fragt der Sender, wenn sein Timer die entsprechende Methode private void sendData() aufruft seine Felder ab, die durch Events der angeschlossenen Collectoren mit aktuellen Daten gefüllt werden und integriert sie in den InfoContainer. Dann fügt er die aktuelle SessionId hinzu, mit der der Server erkennt, von welchem Benutzer die Daten kommen. Anschließend ruft er die Methode public bool Client.submitData(Infocontainer container) auf. Diese Methode gehört zur automatisch erzeugten Schnittstelle des Servers. Sie serialisiert die gesamten Daten in einen XML-formatierten Datenstrom und sendet sie an den Server. Kommt es serverseitig zu einem Fehler, gibt dieser den boolschen

Wert „false“ zurück und der Sender stoppt das Senden mit einer entsprechenden Fehlermeldung.

Anschließend löscht der Sender alle Felder, die Informationen aus Subsystemen enthalten, damit beim nächsten Übertragungsversuch nicht versehentlich veraltete Informatoinen übertragen werden.

4.2.3 GPS Subsystem

Wie der Name schon erahnen lässt handelt es sich hierbei um die Teile des Clients, die für das Auslesen eines GPS-Empfängers und das Aufbereiten der Daten für das Versenden erforderlich sind. Diese Funktionen sind in den beiden Klassen *GPSReader* und *GPSCollector* untergebracht. Der *GPSReader* liest Datensätze über den Microsoft Intermediate GPS Driver ein. Dieser ist ein von Windows Mobile verwalteter virtueller GPS Empfänger, der als Zwischenschicht zwischen dem realen GPS Empfänger und den Programmen liegt. Er kommuniziert mit Programmen über virtuelle serielle Schnittstellen. Durch die Virtualisierung der Kommunikation können mehrere Programme von der gleichen Schnittstelle lesen. Dies ist unter Windows bei realen seriellen Schnittstellen nicht möglich. Hier könnte nur ein Programm die Daten empfangen.

Sobald ein neuer Satz aus dem Eingangsdatenstrom gelesen werden kann, parst der *GPSReader* ihn in ein Objekt des Typs *NewGPSEventArgs* und löst ein Event aus, über das der *GPSCollector* aktiviert wird. Dieser ist über einen Eventhandler mit dem *GPSReader* verbunden. Er überprüft bei einem neu gelesenen NMEA Satz zuerst die Satzkennung (vergl. Kapitel 2.9.1). Handelt es sich um eine der Kennungen GPGSA (Art des Fixes), GPGGA (Position) oder GPRMC (Richtung, Geschwindigkeit), entnimmt der Algorithmus die benötigten Informationen aus dem *NewGPSEventArgs*-Objekt und fügt sie in ein Objekt vom Typ GpsPosition ein. Da alle NMEA Datensätze einmal pro Sekunde in einer festen Reihenfolge auftreten, liest der Algorithmus zuerst alle drei Sätze, bevor er eine neue Position an die angeschlossenen Handler meldet. Zu diesen gehört momentan nur das Modul, das die Daten sammelt und an den Server überträgt.

4.2.4 WLAN Subsystem

Das Sammeln aller verfügbaren Informationen über die in Reichweite befindlichen Accesspoints ist die Aufgabe des *WlanCollector*. Er verwendet die von InTheHand kostenlos bereitgestellte OpenNETCF Bibliothek ([INTH2007]), um Daten zu erheben. Dazu wird timergesteuert alle zwei Sekunden eine Liste aller WLAN-fähigen Netzwerkkarten erzeugt und von diesen eine Liste der Accesspoints in Reichweite angefordert. Da die resultierenden Daten die benötigten bei Weitem übersteigen, extrahiert der Algorithmus nur die Felder, die sich auf die MAC-Adresse des Accesspoints und die Signalstärke beziehen, mit der seine Signale empfangen werden. Alle so ermittelten Accesspoints werden anschließend in einer Liste zusammengefasst, bevor der Sender durch ein Event über die neuen Daten informiert wird.

Das anfangs erwähnte Intervall von zwei Sekunden, in dem jeweils neue Daten erhoben werden, mag dem Leser etwas lang erscheinen. Allerdings haben Accesspoints eine vergleichsweise hohe Reichweite, sodass man sich im typischen Szenario, das als Anwendungsfall dieses Systems dient, nicht schnell genug bewegen könnte, um einen zur Verortung eventuell wichtigen Accesspoint zu „übersehen“. Schließlich bewegt sich der Durchschnittsbenutzer in einer urbanen Region gesetzlich festgeschrieben nie schneller als mit $50km/h$. Dies entspricht etwa $14m/s$. In zwei Sekunden legt der Benutzer also 28 Meter zurück – eine Reichweite, die ein Accesspoint mit Leichtigkeit erreicht, selbst, wenn er nicht direkt an der Strecke steht, auf der sich der Benutzer bewegt.

4.2.5 Bluetooth Subsystem

Die Programmlogik für das Sammeln von Informationen mit Hilfe des Bluetoothempfängers befindet sich in der Klasse *BluetoothCollector*. Sie basiert ebenfalls auf dem OpenNETCF Framwork zum Auslesen des Bluetoothgerätes. Die Vorgehensweise dabei ist ähnlich wie im *WlanCollector*. Zuerst wird eine Anfrage an die Bluetoothhardware gestellt. Aus dem resultierenden Array von Bluetoothstationen extrahiert der Algorithmus dann die benötigten Daten: Den Namen der Station und ihre Geräteadresse. Auf das Auslesen der Signalstärke wird verzich-

tet, da anhand der Daten aus diesem Modul nur eine Proximation durchgeführt werden soll. Die Signalstärke ist somit überflüssig.

Ein Timer steuert auch hier das Auslesen der Daten. Er ist ebenfalls auf ein Intervall von zwei Sekunden eingestellt. Dies hat jedoch andere Gründe als im WLAN-Modul. Während es dort bei einem so langen Intervall nicht dazu kommen kann, dass ein Accesspoint „übersehen“ wird, könnte dies aufgrund der geringen Reichweite der Bluetoothstationen durchaus vorkommen. Eine höhere Auslesegeschwindigkeit ist jedoch nicht möglich, da das Pollen der einzelnen Stationen offenbar sehr viel Zeit in Anspruch nimmt. So würden sich bei einem kürzeren Leseintervall die Anfragen an den Bluetoothstack überlappen, was schnell zum Abstürzen des Systems führt, da die Prozessorlast schnell ansteigt, wenn mehrere Anfragen aktiv sind.

Sobald eine Anfrage von der Bluetoothhardware beantwortet wurde, erzeugt das Modul eine Liste aller Stationen mit den dazu gehörigen Informationen und löst dann ein Event aus, um den Sender darüber zu informieren, dass neue Daten vorliegen.

4.2.6 Adressübertragungssystem

Sollte der Benutzer auf keine elektronisch erfassbaren Signale zurückgreifen können, gibt ihm das Adressverortungssystem die Möglichkeit, analog die Adresse seines Standortes zu bestimmen und an das System zu übertragen. Anhand der übertragenen Adresse kann das System in der Hauskoordinatendatenbank nach den Koordinaten des Benutzers suchen.

Auf den ersten Blick scheint es müßig, bei einer bekannten Position die Koordinaten zu ermitteln. Allerdings ist zu bedenken, dass das System in einem größeren Zusammenhang weiterverwendet werden soll. Die ortsbezogenen Dienste, die über das Gesamtkonstrukt abgewickelt werden sollen, sind dort mit ihren geographischen Koordinaten verzeichnet. Die Adressen der Dienste sind zwar, soweit benötigt hinterlegt, werden aber in der automatischen Suche nicht berücksichtigt. Deshalb ist eine Verortung auch bei bekannter Adresse sinnvoll.

Zur Eingabe der Adresse steht das dritte Tab der GUI des Clients zur Verfügung. Es enthält Felder für die Straße, Hausnummer, Hausnummernzusätze wie

A, B, C, Postleitzahl und Ort. Informationen wie der Ortsname wirken auf den ersten Blick redundant, sind es allerdings nur bedingt. In ländlichen Regionen, wo oft zwanzig und mehr kleine Dörfer mit einer Postleitzahl zusammengefasst werden, kann es durchaus vorkommen, dass es in unterschiedlichen Dörfern zwei Straßen gleichen Namens gibt. Durch die Angabe des Ortsnamens kann hier differenziert werden.

Über die Buttons am unteren Bildrand kann man die Berücksichtigung der Adresse in der Positionsermittlung ein- und ausschalten oder die Daten aktualisieren.
Obwohl dieser Programmteil als Modul benannt ist, ist er genau genommen keines. Die Eingabemaske ist in die GUI integriert und läuft somit auch im Vordergrundthread. Das Übertragen der Daten geschieht wie bei allen anderen Modulen auch durch den Sender. Trotzdem soll es hier als ein Mittel der Informationsgewinnung gleichbedeutend mit den anderen Subsystemen genannt werden.

4.2.7 Konfiguration

Um den Anforderungen einer mit wenig Leistung ausgestatteten Hardwareplattform gerecht zu werden, konnte zur Implementierung der zentralen Konfiguration des Clients nicht auf die vom .NET Framework bereitgestellten Funktionen zurückgegriffen werden. Diese bieten zwar die Möglichkeit, Einstellungen zu speichern und wieder zu laden, legen diese aber in der Registry, also der zentralen Konfigurationsdatei des Betriebssystems ab. Dieses Vorgehen birgt zum einen für die Zukunft das Problem, dass Einstellungen in der Registry aus sicherheitstechnischen Gründen vom Administrator untersagt werden könnten, zum anderen sind die abgelegten Werte in dieser Datei nur umständlich editierbar. Ein weiterer Grund für eine Entscheidung gegen das plattformeigene Speicherkonzept war die Lese- und Schreibgeschwindigkeit, die in mehreren Versuchen unter der Geschwindigkeit des selbst entwickelten Systems lag.

Aus diesem Grund ist in diese Arbeit ein selbst entwickeltes Speicherkonzept eingegangen. Es besteht aus jeweils einem *ConfigReader* und einem *ConfigWriter*. Der Writer erhält über seinen Konstruktor ein Objekt des Typs ConfigItem,

das die Daten der aktuellen Konfiguration enthält und den Dateinamen, in den gespeichert werden soll. Durch den Aufruf der Methode

```
public void ConfigWriter.writeConfig()
```

wird ein *XMLSerializer* erzeugt, der das *ConfigItem*-Objekt serialisiert und danach über einen *FileStream* ins Dateisystem schreibt. Der *ConfigReader* ist ähnlich aufgebaut. Er erhält im Konstruktor jedoch nur den Dateinamen, aus dem eine Konfiguration wieder hergestellt werden soll. Durch den Aufruf von

```
public void ConfigReader.readConfig()
```

wird die angegebene Datei eingelesen und sofern möglich in ein *ConfigItem* deserialisiert. Wie oben schon erwähnt speichert das *ConfigItem* die gesamte Konfiguration der Clientsoftware. Ein *ConfigItem* besteht aus den in Tabelle 4.1 aufgezählten Properties.

Da auf die Konfiguration häufig zugegriffen wird, sind die Felder nicht durch Properties gekapselt. Dies spart bei jedem Auslesen eines Attributs die Zeit, die ein Methodenaufruf und das Umschreiben der Werte auf dem Stack erfordert. Die durch einen Methodenaufruf zusätzlich notwendige Zeit, um die Parameter auf den Stack zu schreiben ist in diesem Fall zwar denkbar gering, sollte jedoch mit Hinblick auf eine Erweiterung des Systems um Module für weitere Positionierungstechnologien nicht außer Acht gelassen werden.

Variablenname	Datentyp	Erläuterung
username	String	Der Benutzername des Teilnehmers. Er wird vom Administrator des jeweiligen Location- und Privacy Providers vergeben und hat die Form „user@privacyProvider“
pass	String	Das ebenfalls vom Privacy Provider vergebene Passwort
server	String	Die Adresse des Location Providers, den der Benutzer verwenden möchte
save	Bool	Angabe, ob die Daten gespeichert und bei einem Neustart restauriert werden sollen
isUseWinMobileGPS	Bool	Legt fest, ob Daten des GPS Empfängers an den Privacy Provider gesendet werden sollen
isUseWlan	Bool	Legt fest, ob Daten des WLAN Empfängers an den Privacy Provider gesendet werden sollen
isUseBluetooth	Bool	Legt fest, ob Daten des Bluetooth Empfängers an den Privacy Provider gesendet werden sollen
street	String	Name der Straße im Adressdatensatz
number	Integer	Hausnummer im Adressdatensatz
numberAdd	String	Zusatz zur Hausnummer im Adressdatensatz - im Regelfall „a“, „b“, „c“ usw.
zip	String	Postleitzahl im Adressdatensatz
city	String	Name der Stadt im Adressdatensatz
sending	Bool	Zustandsvariable, die beschreibt, ob das System gerade im Sendemodus ist

Tabelle 4.1: Übersicht über die Felder eines ConfigItems

4.3 Server

Der Server, der in der Systemarchitektur nach [STEI08] als Location-Provider bezeichnet wird ist ein über das Internet erreichbarer Dienst, der die Daten empfängt, die die Clients über ihre Sensoren wahrnehmen und an ihn senden. Mit diesen Daten, die aus verschiedenen Quellen stammen können, ermittelt der Location-Provider mit Hilfe verschiedener Positionsbestimmungsmodule und den nachgeschalteten Filtern eine möglichst präzise geschätzte Position jedes Clients und legt sie in der Datenbank ab. Ein Privacy-Provider hat Zugriff auf diese Daten und stellt sie in je nach Anwendungszweck mehr oder weniger anonymisierter Form weiteren Diensten zur Verfügung.

4.3.1 Architektur

Wie der Client ist auch der Server in mehrere funktionale Module aufgeteilt. Denn auch der Server soll möglichst einfach weiterentwickelt und an weitere Technologien angepasst werden können. Deshalb gibt es einen Dispatcher, der auf der .NET Technologie des BackgroundWorker basiert. Er ist für die Verteilung der Arbeit auf mehrere Threads und die Verteilung der Daten zuständig. An ihn ist der Datenbankwrapper angeschlossen, der die Verbindung mit einer PostgreSQL Datenbank herstellt und zur Laufzeit alle Daten verwaltet. Außerdem kommuniziert er mit dem Modul, das die Webservices kapselt und verteilt deren Daten an die verschiedenen Benutzerobjekte oder direkt an die Datenbank. Die GUI erhält ihre Daten ebenfalls vom Backgroundworker und sendet Anfragen über diesen Dispatcher an alle anderen Module. Dieser modulare Aufbau ist in Abbildung 4.3 auch grafisch durch die Trennung der einzelnen Elemente dargestellt. Die Pfeile stellen die Kommunikationsbeziehungen zwischen den Modulen dar.

Im Aufrufebaum ist der Dispatcher das Wurzelelement. Er startet alle anderen Systeme und registriert seine Ereignishandler an ihren Eventgeneratoren, sodass die erstellten Objekte Daten an ihn übertragen können.

Eine weitere Aufgabe des Dispatchers ist die Benutzerauthentifikation. Sobald der Webservice den Empfang neuer Daten meldet, prüft der Dispatcher, ob unter der angegebenen SessionId schon ein Benutzer registriert ist. Nur in diesem

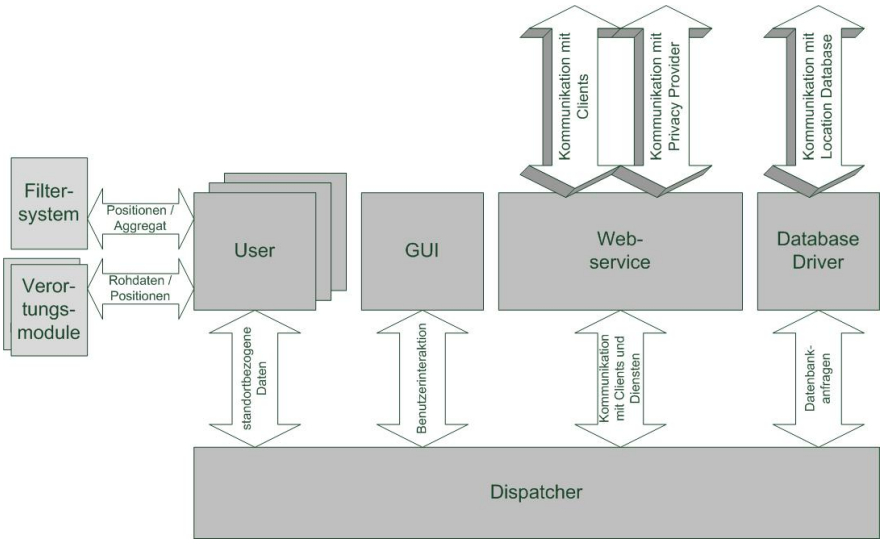

Abbildung 4.3: Aufbau des Location-Provider Servers

Fall leitet er die Daten an das entsprechende Benutzerobjekt weiter. Der nächste Abschnitt wird auf diese Funktion noch weiter eingehen.

4.3.2 Userverwaltung

Die Userverwaltung für das hier vorgestellte System findet, wie bei allen anderen Systemen der Metaarchitektur, in jedem Teilsystem separat statt. So ist gewährleistet, dass die einzelnen Teilsysteme von verschiedenen Anbietern realisiert und betrieben werden können. Dank dieser Herangehensweise ist es möglich, jedes Teilsystem intern auf eine anbieterspezifische Weise zu realisieren, so lange die Schnittstellen zu den anderen Architekturteilen definiert sind. Der Kunde wählt dann den Betreiber aus, der seine funktionalen, aber auch sicherheitstechnischen Bedürfnisse am ehesten bedient.

Die Benutzerverwaltung in diesem System besteht aus mehreren Teilen. Allem zu Grunde liegt die Benutzerdatenbank. Diese speichert momentan den Benutzernamen und ein Passwort für jeden Benutzer. Der Benutzername besteht aus einem selbst gewählten Pseudonym, dem als Postfix ähnlich wie bei einer Emailadresse ein Zeichen und der Name des gewählten Privacy Providers folgt. Ein Benutzername könnte also folgendermaßen aussehen: Benutzer0815@MyPrivacyProvider. Das Passwort wird nicht im Klartext gespeichert, damit ein Angreifer mit Zugriff auf

die Datenbank es nicht korrumpieren kann. Stattdessen erzeugt das System einen MD5 Hash des eingegebenen Passwortes und speichert diesen statt des Klartextes. Ein irgendwann eingegebenes Passwort wird immer zum gleichen Hash führen, aber der Hash kann nicht zurück in das Passwort übersetzt werden. So ist das eigentliche Passwort selbst jemandem mit Zugang zur Datenbank verborgen.

Neue Benutzer werden über die GUI angelegt. Dort kann auch das Passwort von schon angemeldeten Benutzern geändert werden.

Loggt sich ein Benutzer im System ein, erfolgt eine Challenge Response Authentifizierung, wie sie schon in Kapitel 4.2.2 beschrieben wurde. Ist die Identität des Benutzers anhand seines Loginnamens und seines Passworts über die Datenbank bestätigt, erzeugt das System eine zufällige Session ID, über die der Benutzer im weiteren Verlauf der Kommunikation identifiziert wird. Das System erzeugt ein Objekt vom Typ User und reiht es, identifiziert nur durch die Session ID, in die Liste der aktiven Benutzer ein. Außerdem legt es die Session ID in einem Dictionary ab, über das sie intern jederzeit einem Benutzer zugeordnet werden kann.

Die User Objekte bilden die oberste Schicht der Benutzerverwaltung. Sie kapseln alle benutzerrelevanten Funktionen. Sobald ein neuer Datensatz empfangen wird, erzeugt der Dispatcher ein Event, mit dem jeweils das Objekt angesprochen wird, das den Benutzer darstellt, der die Datenübertragung initiiert hat. Das Objekt übernimmt die Daten, stellt Vorabberechnungen wie die Umrechnung von geographischen Koordinaten zu Gauß-Krüger Koordinaten an und übergibt alle Daten einem der benutzerspezifischen Filter. Damit sich die Berechnungsabläufe nicht gegenseitig blockieren können, läuft die Datenberechnung in jedem Benutzerobjekt in einem eigenen Thread.

Jedes Benutzerobjekt verfügt über eine Eigenschaft, die die letzte aktuelle Position des Benutzers enthält und vom System jederzeit abgefragt werden kann. Sobald ein Berechnungsschritt der Filteralgorithmen abgelaufen ist, wird diese Eigenschaft aktualisiert. Sie ist es auch, die abgefragt wird, wenn der Privacy Provider ein Location Update für einen seiner Clients anfordert.

Um die Mobilität der Clients zu unterstützen, ist die Kommunikation zwischen Client und Server nur sehr schwach verbindungsorientiert. Schließlich kann es für den Client durch einen Handover des Mobilfunknetzbetreibers oder wenn

die gerade verfügbaren WLAN Accesspoints zum Verbindungsaufbau verwendet werden ständig zu Timeouts oder sogar Verbindungsabbrüchen kommen. Aus diesem Grund überwacht der Server mit Timern die Kommunikation zu jedem Client. Sobald ein neues Datenpaket eingeht, wird der Timer, der zum entsprechenden Client gehört, neu gestartet. Geht 15 Sekunden lang kein weiteres Datenpaket ein, läuft der Timer ab und es kommt zu einem Timeout.

Clients, bei denen es zu einem Timeout kam, werden in regelmäßigen Abständen vom Server entfernt. Dieser Vorgang wird auch im Debuggingfenster der GUI angezeigt, die im folgenden Kapitel beschrieben wird.

4.3.3 GUI

Die GUI des Servers ist das Hauptsteuerelement des Systems. Sie erlaubt das Hinzufügen und Editieren von Accesspoints, Bluetoothstationen und Benutzern ebenso wie das Starten und Stoppen der beiden Kommunikationsdienste. Wie in Abbildung 4.4 zu sehen ist, ermöglicht ein Treeview in der linken Hälfte der Oberfläche den Zugriff auf vier verschiedene Oberkategorien: Dienste, Benutzer, Accesspoints und Bluetoothstationen (in der Abbildung mit den Zahlen 1-4 markiert). Je nach der Auswahl im linken Bereich zeigt die Detailansicht rechts (5) weitere Informationen, wie in der Abbildung die Koordinaten eines Accesspoints in Gauß-Krüger Notation. Im rechten unteren Bereich der Benutzerschnittstelle befindet sich das Debuggingfenster (6). Hier werden Statusinformationen des Servers angezeigt.

Über einen Dialog (7) kann der Benutzer Daten zu einzelnen Referenzstationen hinterlegen oder ändern. Ebenfalls über einen Dialog werden neue Benutzer angelegt (8) oder bestehende Benutzer editiert.

Klickt man mit der rechten Maustaste in die GUI, öffnet sich ein Kontextmenü mit drei Einträgen. Hier können Bluetoothstationen, Accesspoints und Benutzer hinzugefügt, editiert und gelöscht werden. Im Fall der Benutzer öffnet sich zum Hinzufügen ein Fenster, das Eingabefelder für einen Benutzernamen und ein Passwort samt Passwortwiederholung enthält. Trägt man hier einen neuen Benutzernamen ein, wird der Benutzer angelegt. Trägt man einen Namen ein, der schon existiert, wird das Passwort des Benutzers aktualisiert.

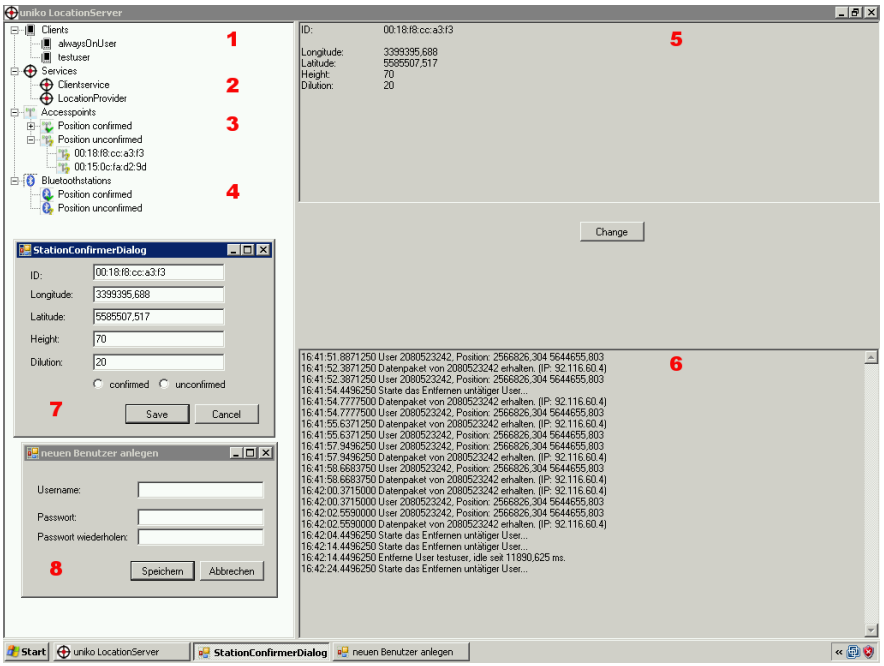

Abbildung 4.4: GUI des Servers mit Dialogen

Zum Hinzufügen von Bluetoothstationen und Accesspoints wird der gleiche Dialog angezeigt. Er heißt *StationConfirmerDialog* und unterscheidet sich für die jeweilige Art der Referenzstation nur in einem internen Flag, das beim Eintragen in die Datenbank festlegt, um welche Art von Station es sich handelt. Eingegeben werden können die ID der Station, die bei Accesspoints die MAC Adresse ist und bei Bluetoothsendern die Stationskennung, die dreidimensionale Position der Station und eine Genauigkeitsabweichung. Weiterhin kann festgelegt werden, ob es sich um eine verifizierte („confirmed“) Position handelt oder nicht. Verifiziert bedeutet, dass die Angaben zur Position der Station einer administrativen Kontrolle unterliegen. Dies wird im Kapitel 4.5 zu Positionen genauer erklärt werden.

Weiterhin ermöglicht die GUI es, die Positionen von schon in der Datenbank vorhandenen Stationen zu korrigieren. Wie in Kapitel 4.8 beschrieben, registriert das System unbekannte Accesspoints und Bluetoothsender, deren Signal von einem Client übertragen wurde mit der aktuell ermittelten Position des Clients in der Datenbank. Die hierbei abgelegte Position ist allerdings mit einer hohen Ungenauigkeit belastet und wird deshalb speziell markiert, sodass ein Administrator

sie in einem späteren Schritt noch einmal kontrollieren kann. Dies wird von der GUI unterstützt, indem jedes Wurzelelement, das eine Art von Referenzstation darstellt, zwei Unterelemente namens „Confirmed“ und „Unconfirmed“ besitzt. Auf diese Elemente werden entsprechend ihrer Markierung in der Datenbank die Stationen aufgeteilt, sodass sofort ersichtlich ist, welche Station einer Überprüfung bedarf und welche nicht.

Wird eine Station angewählt, zeigt die GUI im rechten oberen Bereich die Daten des jeweiligen Referenzpunktes an. Ein Klick auf den Button Change öffnet einen *StationConfirmerDialog*, dessen Felder schon mit den Werten der gewählten Station vorbelegt sind, sodass hier eine komfortable Bedienung möglich ist.

4.3.4 WLAN Verortungssystem

Die in der Klasse *WLAN-Localizer* zu findende Routine zur Verortung anhand von WLAN Signalen stammt, was die Programmlogik betrifft, größtenteils aus [WICK05]. Beim Aufruf der Methode

```
public Position calculatePosition(List<StationPosition> aplist)
```

wird eine Liste von StationPositions übergeben. Diese Liste enthält die Positionen aller Referenzpunkte, von denen der Client Signale registriert hat und deren Standort schon in der Datenbank enthalten war. Zusätzlich enthalten die Objekte die Signalstärke, die der Client gemessen hat.

Da WLAN Signale eine Reichweite von über 100 Metern haben können, ist es sehr wahrscheinlich, im öffentlichen Raum die Signale von mehreren WLAN Accesspoints gleichzeitig empfangen zu können. Deshalb bietet es sich an, als Verortungstechnik die Lateration zu wählen. Hierfür sind jedoch die Entfernungen zu den jeweiligen Referenzpunkten zu bestimmen. Um eine Positionsschätzung erzeugen zu können, muss das System also von den Signalstärken, die üblicherweise in Dezibel angegeben werden, auf eine Entfernung zwischen Sender und Empfänger schließen.

Nach dem Vorschlag von [WICK05] ist eine gute Näherung bei der Umrechnung von der Signalqualität sq in Dezibel zur Entfernung d in Metern, wenn a der Verstärkungsfaktor der Antenne des Accesspoints in Prozent ist:

$$d = (500 * a/sq) - 5$$

Diese Formel basiert auf empirisch ermittelten Werten. Die Tests mit verschiedenen mobilen Endgeräten haben jedoch gezeigt, dass es nicht bei jedem WLAN Chipsatz möglich ist, die Signalqualität in Dezibel auszulesen. Oft war nur die auch vom Desktopsystem bekannte Visualisierung der Signalqualität in „Balken“ auslesbar. Diese Visualisierung wird vom Chipsatz als eine Enumeration, bestehend aus 5 diskreten Werten veröffentlicht, die im Fall des OpenNETCF Frameworks mit den Lexemen in Tabelle 5.1 wiedergegeben werden können. Um auch aus diesen unpräzisen Angaben eine Entfernung ableiten zu können, fanden Tests statt, die in Kapitel 5.1 wiedergegeben werden. Die in der Tabelle zitierten Werte sind Ergebnisse dieser Versuche.

Nach der Ermittlung der Entfernungen kann die eigentliche Positionsbestimmung beginnen. Dazu wird eine neue Position angelegt, deren Abweichung auf 1000 gesetzt wird. Dann bildet der Algorithmus einen gewichteten Mittelwert aller Positionen, in dem die Referenzstationen höher gewichtet sind, je geringer ihre Entfernung zum Client ist. Nach jedem Schritt wird die Unsicherheit mit der Entfernung zum aktuellen Referenzpunkt verglichen. Ist sie geringer, als die momentan gespeicherte, wird die berechnete Entfernung des Accesspoints als neue Unsicherheit übernommen.

4.3.5 Bluetooth Verortungssystem

Für die Verortung der Bluetoothsignale, die von den Clients übertragen werden, wird ein sehr einfaches System verwendet, das auf Proximation basiert. Die Programmlogik liegt in der Klasse *Bluetoothlocalizer*. Über die Methode

```
public Position calculatePosition(List<StationPosition>
    btStations)
```

wird eine Liste, bestehend aus einzelnen StationPositions übertragen. Dabei handelt es sich um die Liste von Positionen derjenigen bekannten und in Reichweite des Clients befindlichen Bluetoothsender, die in einem vorhergehenden Schritt vom Datenbankmodul ermittelt wurden. Anhand dieser Positionen ermittelt der Localizer eine Positionsschätzung, indem er folgendermaßen vorgeht. In den meisten Fällen wird nur ein Bluetoothsender in Reichweite sein. Dann wird, wie bei der Proximation üblich dessen Position als Schätzwert zurückgegeben. Die Unsicherheit wird auf die zu erwartende Reichweite des Senders von 10 Metern eingestellt. Es wird also angenommen, dass der Client sich in einem Kreis mit dem Radius 10 Meter befindet, dessen Mittelpunkt der Bluetoothsender bildet.

In dem unwahrscheinlichen Fall, dass sich mehrere Bluetoothsender in Reichweite befinden, ermittelt der Localizer den Durchschnittswert der einzelnen Positionen und gibt diesen zurück. Auch hier beträgt die Abweichung das theoretische Maximum der Reichweite von 10 Metern.

4.3.6 Datenbank Subsystem

Dieses Modul kapselt alle Zugriffe auf Datenquellen. Entsprechend der in Kapitel 1.3 diskutierten Anforderungen handelt es sich um eine PostGreSQL Datenbank. Diese SQL Variante hat den Vorteil, dass sie besondere Funktionen, wie zum Beispiel eine Umkreissuche oder Entfernungsberechnungen für Geoinformationssysteme bietet. Das Modul ist wie in der .NET Entwicklung üblich mit ADO.NET[3] realisiert und bildet damit die Schnittstelle für alle Datenbankzugriffe. Da mit Hilfe von ADO.NET automatisch ein MVC[4]-Konzept angewendet wird, besteht das Datenbankmodul aus mehreren Schichten. Die unterste Schicht bildet der Datenbankzugriff mit SQL-Statements. Hier werden die einzelnen Tabellen über Selectkommandos wie zum Beispiel SELECT $*$ FROM Users; geladen. Jeweils ein *DataAdapter* erzeugt automatisch zu den Selects passende Insert- und Update-

[3] ActiveX Data Objects for .NET
[4] Model-View-Controller

kommandos und bildet außerdem die Schnittstelle zur nächst höheren Schicht. Die bis jetzt besprochenen Klassen, die für den Verbindungsaufbau, direkte Datenbankzugriffe und die Konfiguration der Kommandos zuständig sind, sind speziell auf die verwendete PostGreSQL Datenbank abgestimmt. Alle weiteren Klassen, die im Folgenden verwendet werden sind in ihrer Verwendbarkeit nicht auf eine spezielle Datenbanktechnologie beschränkt. Das DataSet, das durch die *DataAdapter* befüllt wird, könnte ebenso gut aus einer MySQL Datenbank, einer XML Quelle oder einer kommaseparierten Liste befüllt werden und würde für die darauf zugreifenden Verbraucher die gleichen Funktionen bereit stellen. Sollte also ein Austausch der zu Grunde liegenden Datenbank notwendig werden, wären dazu nur Änderungen auf der Zugriffsebene nötig, nicht aber bei der Präsentation der Daten gegenüber den anderen Modulen.

Beim Programmstart öffnen die *DataAdapter* eine Verbindung zur Datenbank und befüllen mit den gelesenen Daten das DataSet. Dieses enthält von diesem Zeipunkt an die drei Tabellen Users, Accesspoints und Bluetoothstationen, deren Inhalte in den folgenden Tabellen dargestellt sind.

Spaltenname	Datentyp	Beschreibung
username	varchar	Benutzername, der beim Login verwendet wird.
passwd_hash	varchar	Hash des Loginpasswortes

Tabelle 4.2: Inhalt der Datenbanktabelle userDB des Location-Providers

Spaltenname	Datentyp	Beschreibung
longitude	double	Rechtswert der Position der Station in Gauß-Krüger Notation
latitude	double	Hochwert der Position der Station in Gauß-Krüger Notation
height	double	Höhe der Position der Station in Meter
dilution	double	Ungenauigkeit in der Position der Station
checked	boolean	Gibt an, ob die Position von einem Administrator überprüft wurde
stationid	varchar	eindeutige Hardwareadresse der Station

Tabelle 4.3: Inhalt der Datenbanktabelle bluetoothstations des Location-Providers

Spaltenname	Datentyp	Beschreibung
longitude	double	Rechtswert der Position der Station in Gauß-Krüger Notation
latitude	double	Hochwert der Position der Station in Gauß-Krüger Notation
height	double	Höhe der Position der Station in Meter
antenna_multiplikator	double	Faktor, um den die Sendeleistung des Systems verstärkt ist
dilution	double	Ungenauigkeit in der Position der Station
checked	boolean	Gibt an, ob die Position von einem Administrator überprüft wurde
mac_address	varchar	eindeutige Hardwareadresse der Station

Tabelle 4.4: Inhalt der Datenbanktabelle accesspoints des Location-Providers

Während des gesamten Programmablaufs werden Daten nur in diese Tabellen geschrieben oder aus ihnen gelesen, um Wartezeiten während des SQL Zugriffs zu umgehen. Jedoch löst ein Timer alle 30 Sekunden ein Updatekommando aus,

das den *DataAdapter* veranlasst, selbstständig die als geändert markierten Tabellenzeilen zu suchen und diese in der PostGreSQL Datenbank zu aktualisieren.

Aus Performancegründen werden die Daten aus der Hauskoordinatentabelle nicht in den Speicher geladen, da diese alleine für Rheinland-Pfalz schon 1,2 Millionen Einträge umfasst. Um Hauskoordinaten abzufragen, wird innerhalb der entsprechenden Methode ein eigenes SQL Statement generiert und dann an die Datenbank abgesetzt.

Für den Lookupservice, also die Komponente, die für die Zuordnung von Koordinaten zu einer IP Adresse zuständig ist und die ebenfalls vom Datenbankmodul gekapselt wird, wird kein SQL verwendet. Die von MaxMind angebotene Bibliothek besteht aus einer Datenbankdatei und einer Klasse, die Zugriffe auf diese Datei kapselt. Für eine Abfrage wird nur eine Instanz dieser Wrapperklasse benötigt, die im Konstruktor das Datenbanktreibers erstellt wird.

Das Datenbankmodul bietet die folgenden Schnittstellen nach außen:

```
public void connect();
public void updateDb(Object o);
public void close();
```

Listing 4.3: Allgemeine Methoden der Klasse dbDriver

Ein Aufruf von connect() bewirkt das oben schon beschriebene Öffnen der Datenbankverbindung und Befüllen des DataSets mit den drei Tabellen. Weiterhin wird das Tickintervall des Timers auf 30 Sekunden eingestellt.

Die Methode updateDb(), die für gewöhnlich nur durch einen Timer gestartet wird, führt zum Start der update() Methoden der jeweiligen *DataAdapter*. Diese überprüfen jede Reihe der zugeordneten Tabellen darauf, ob das RowContentChanged Flag gesetzt ist. Ist dies der Fall, wird die Reihe zurück in die Datenbank geschrieben.

Die letzte Methode im obigen Listing schließt alle Datenbankverbindungen und den Lookupserve, nachdem noch einmal ein Updateevent ausgelöst wurde.

```
public bool checkUser(String username, String password)
public bool newUser(String username, String password)
public bool removeUser(String username)
```

Listing 4.4: Methoden der Klasse dbDriver um Benutzer zu manipulieren

Hier handelt es sich um Methoden, die für die Benutzerverwaltung wichtig sind. Während checkUser in der Tabelle Users nach einem bestehenden Eintrag sucht, um den Benutzer zu autorisieren, legt newUser einen neuen Benutzer an. Beide Methoden kodieren erst das Passwort zu seinem eigenen MD5 Hash, da alle Passwörter in der Datenbank nur so abgelegt sind. Dies erhöht die Sicherheit der Anwendung, da ein potentieller Angreifer nicht einmal durch das Auslesen der Datenbank das wahre Passwort bekommt. Wird ein Benutzer mit dem passenden Passwort gefunden, gibt die Methode checkUser() den boolschen Wert „true“ zurück, um zu signalisieren, dass es sich um einen autorisierten Nutzer handelt.

Zum Entfernen eines Nutzers ist nur die Angabe des Benutzernamens nötig. Die Methode prüft, ob der Benutzer existiert und entfernt ihn gegebenenfalls aus der Datenbank.

```
public StationPosition getApPosition(String apMac)
public StationPosition getBluetoothPosition(String btMac)
public Position getPositionByAddress(Address addi)
public Position getIpPosition(String ipAddress)
```

Listing 4.5: Methoden der Klasse dbDriver um Positionen abzufragen

All diese Methoden lesen Positionen aus der Datenbank aus. Dazu wird ihnen das eindeutig identifizierende Merkmal der jeweiligen Signalquelle als Parameter übergeben. Bei Accesspoints und Bluetoothsendern sind dies die jeweiligen Gerätehardwareadressen. Eine Adresse und eine IP Adresse identifizieren sich natürlich selbst eindeutig. Der Rückgabewert ist jeweils eine Position. Im Fall von Accesspoints und Bluetoothsendern handelt es sich um ein von Position abgeleitetes Objekt namens Stationposition. Es enthält weitere Angaben, die bei der Verortung mit Hilfe dieser beiden Technologien hilfreich sind.

```
public void insertApPosition(String apMac, Position pos)
public void insertBtPosition(String btMac, Position pos)
```

Listing 4.6: Methoden der Klasse dbDriver um Referenzpunktpositionen hinzuzufügen

Meldet der Datenbanktreiber, dass eine angefragte Position nicht bekannt ist, werden diese beiden Methoden wichtig. Sie fügen neu registrierte Stationen in die Datenbank ein, nachdem ein Benutzer deren Position gemeldet hat. Dazu wird angenommen, dass sich die bisher unbekannten Stationen in der Nähe des Benutzers befinden, der ihre Daten übermittelt hat. Sobald das System anhand anderer Daten aus der gleichen Datenübermittlung die Position des Nutzers bestimmt hat, werden die neuen Stationen mit dieser Position und dem Status „unconfirmed“ in die Datenbank aufgenommen. Es ist dann die Aufgabe eines Administrators, in bestimmten Abständen diese automatisiert aufgenommenen Stationen zu überprüfen und ihre Position zu bestätigen. Eine genaue Beschreibung der Verarbeitung bisher unbekannter Referenzstationen bietet Kapiel 4.8.

4.3.7 Filter

Unter diesem Oberbegriff sind in der Implementierung die beiden verwendeten Arten der Sensordatenintegration zusammengefasst. Neben der schon in [HEBE07] eingeführten Technologie des Partikelfilters kommt in der aktuellen Implementierung eine weitere Art von Gewichtungssystem zum Einsatz. Im Folgenden wird zuerst das neue Filtersystem beschrieben, bevor danach die technische Funktionsweise des Partikelfilters erklärt wird.

Wichtig ist zu bemerken, dass hier keine Techniken zur Ermittlung einer Position aus rohen Sensordaten vorgestellt wird. Die hier besprochenen Techniken arbeiten mit bereits in vorgeschalteten Modulen errechneten Einzelpositionen aus verschiedenen Datenquellen und integrieren diese zu einer globalen Positionsschätzung. Das Ziel ist eine Verbesserung der Genauigkeit der einzelnen Messwerte und eine Informationsanreicherung, falls eine Technologie nicht alle benötigten Daten wie Blickrichtung oder Geschwindigkeit liefern kann.

Da das System sowohl innerhalb (indoor) als auch außerhalb (outdoor) von Gebäuden verwendet werden kann, sind grundsätzlich zwei Betriebsmodi zu unterscheiden, die in der Verfügbarkeit der einzelnen Verortungstechnologien differieren. So ist in Gebäuden der GPS Empfang nur eingeschränkt möglich, während im Außenbereich Nahbereichsfunk zur Verortung größtenteils nicht verfügbar ist. Dadurch ergeben sich für die Datenintegration je nach Betriebsmodus veränderte Bedingungen, denen über die weiter unten eingeführte Gewichtungsmatrix implizit Rechnung getragen wird. Der erste verwendete Filter heißt in der Implementierung bezeichnenderweise SimpleFilter. Seine Funktionsweise basiert auf der Idee, dass verschiedene Verortungstechnologien Positionen mit verschiedenen Genauigkeiten und damit einhergehend auch verschiedenen Zuverlässigkeiten liefern. In Tabelle 4.5 ist eine Auflistung der einzelnen Technologien mit der zu erwartenden Genauigkeit eines Verortungsergebnisses zu sehen.

Verortungslösung	erwartete Genauigkeit
IP Adresse	100 Kilometer
WLAN	100 Meter
GPS	10 Meter
Bluetooth	10 Meter
Hauskoordinaten	20 Meter

Tabelle 4.5: zu erwartenden Zuverlässigkeit der unterschiedlichen Verortungstechnologien

Wie aus der Tabelle zu entnehmen ist, unterscheiden sich die verschiedenen Techniken grundlegend in ihrer Genauigkeit. Außerdem ist die Verfügbarkeit der Verortungslösung gerade bei GPS nicht überall gegeben. Deshalb sind die Datenquellen in einer bestimmten Abarbeitungsreihenfolge in der Software angelegt. Die Idee besteht also darin, alle zur Verfügung stehenden Technologien nach ihrer Relevanz zu ordnen und die Daten von den ungenauesten zu den genauesten zu verbessern, wie in Abbildung 4.5 dargestellt. Bei diesen Überlegungen spielt allerdings nicht nur die Genauigkeit eine Rolle. Während in der obigen Tabelle die Genauigkeit und vor allem die Zuverlässigkeit von oben nach unten zunimmt, nimmt die Verfügbarkeit der Angaben ab. GPS Empfang ist nicht immer möglich,

eine Bluetoothstation selten in Reichweite und die Chance, dass ein Benutzer die Adresse, an der er sich gerade aufhält, immer aktuell hält ist recht unwahrscheinlich.

Da die Abweichung bei einer Verortung anhand der IP Adresse recht hoch ist, die Technologie jedoch in den meisten Fällen ein Ergebnis liefert, wird sie zuerst durchgeführt. Eine Ausnahme stellt hier die Verwendung von IP Adresse aus privaten Bereichen wie 192.168.0.0/24 dar, da diese nicht registriert sind und somit auch nicht verortet werden können. Sollten im gleichen übertragenen Datensatz ausreichend Daten einer genaueren Technologie vorhanden sein, werden die schon berechneten Koordinaten von den später ermittelten überschrieben.
Als nächst zuverlässigere Methode der Positionsbestimmung wird WLAN angenommen. Sollten keine besseren Daten vorhanden sein, wird WLAN die Verortungsgenauigkeit im Gegensatz zu der IP Adresse im ungünstigsten Fall um den Faktor 10^2 verbessern. Ist jedoch GPS verfügbar, wird auch die Verortung mit WLAN verworfen. Denn GPS bietet eine durchschnittliche Genauigkeit von etwa 10 Metern bei einer wesentlich höheren Zuverlässigkeit als WLAN. Außerdem liefert GPS viele weitere Informationen wie die Geschwindigkeit und die Bewegungsrichtung mit, sodass der Server diese nicht umständlich selbst anhand von älteren Daten berechnen muss.

Die letzte verwendete Technologie ist die Hauskoordinate, die auf der vom Benutzer selbst eingegebenen Adresse beruht. Dadurch, dass der Nutzer die Adresse selbst einträgt, sollte eine verantwortungsbewusste Nutzung dieser Technik gewährleistet sein und Fehler weitestgehend ausgeschlossen werden. Obwohl die Genauigkeit mit 20 Metern angegeben ist – wo im Gebäude sich der Client befindet ist natürlich nicht zu sagen – ist die Zuverlässigkeit dieser Angabe doch sehr hoch, dadurch ist sie höher eingestuft als GPS. So bietet die Hauskoordinate eine komfortable Möglichkeit, die eigene Position auch ohne GPS Empfang sicher bestimmen zu können.

Da Bluetooth über eine äußerst geringe Reichweite von unter 10 Metern verfügt, ist bei einem Signalkontakt mit absoluter Sicherheit davon auszugehen, dass der Client sich in der direkten Umgebung des Senders befindet. Allerdings gibt es viele mobile Bluetoothsender, sodass es hier zu einem Sonderfall kommt. In der Datenbank gefundene Bluetoothstationen werden nur zur Verortung verwendet,

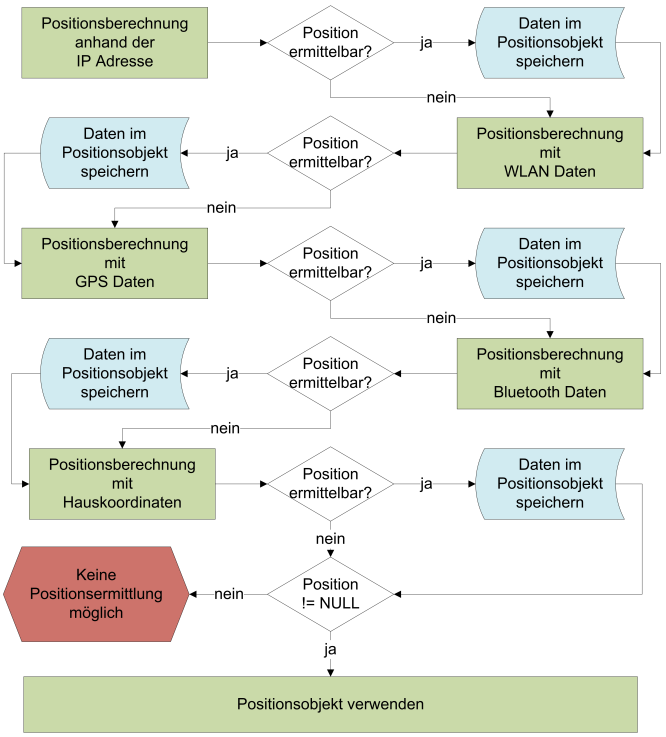

Abbildung 4.5: Ablauf einer Positionsbestimmung mit dem SimpleFilter

wenn sie als Confirmed gelten (siehe Kapitel 4.5). Damit sollte sicher gestellt sein, dass nur unbewegliche Sender, deren Positionen überprüft wurden, zur Verortung verwendet werden.

Ist diese geordnete Liste von Verortungsmöglichkeiten durchlaufen, verfügt das Filtermodul über eine Clientposition. Anhand dieser und der vorhergehenden, die in einem speziellen Datensatz hinterlegt ist, wird nun die Geschwindigkeit, die Bewegungsrichtung und die Blickrichtung des Clients berechnet.

Neben dieser einfachen Datenintegration existiert noch die weitere Möglichkeit, eine Datenintegration über den integrierten Partikelfilter vorzunehmen. Da es sich hier um den in [HEBE07] entwickelten Filter handelt, dessen theoretische Funktionsweise auch in Kapitel 2.7.4 vorgestellt wurde, beschränkt sich diese Beschreibung nur auf die für diese Implementierung wichtigen Veränderungen.

Die erste Veränderung ist die Ersetzung der Programmbibliothek, mit der die mathematischen Wahrscheinlichkeitsverteilungen realisiert wurden, die das Rauschen des Systems simulieren. Da die erste Implementierung der Software in der Programmiersprache JAVA erfolgte, war zur Realisierung Colt [COLT] verwendet

worden. Für C# bot sich die mathematische Bibliothek MathNET Iridium [IRID] an. Sie bot das gleiche Verhalten, sodass nur einige Methodenaufrufe verändert werden mussten. Normalverteilungen werden im Partikelfilter zur Gewichtung der einzelnen Partikel und zur Simulation von Messrauschen verwendet. Sie liefern mit den einstellbaren Parametern Mittelwert und Standardabweichung um einen Mittelwert normalverteilt streuende Zufallswerte. Der Mittelwert ist dabei die reale Kontrollposition während die Standardabweichung sich proportional aus der Ungenauigkeit der Kontrollposition ergibt. Ist diese groß, muss auch die Standardabweichung hoch sein, um auch in einem großen Abstand zum Mittelwert noch ausreichend hoch gewichtete Partikelwertigkeiten zuordnen zu können.

In [HEBE07] wurde jeder nächste Schritt des Filters durch die Übergabe einer unverarbeiteten Positionsschätzung entweder aus dem GPS Subsystem oder, wenn keine GPS Daten verfügbar waren, aus dem WLAN Subsystem eingeleitet. Dieses Verhalten ist für ein System, das Informationen aus mehr als zwei Quellen bezieht, nicht mehr verwendbar. Stattdessen gehen alle ermittelten Positionen der einzelnen Subsysteme in die Positionsermittlung ein und werden entsprechend einer Gewichtung in die Neugewichtung der Partikel integriert. Es leuchtet ein, dass bei unterschiedlichen Kombinationen von Eingangsdaten die einzelnen Komponenten mit einer unterschiedlichen Wertigkeit an der Gewichtung teilnehmen sollten. Sind beispielsweise nur Daten aus dem GPS System vorhanden, wird zur Neugewichtung der Partikel die GPS Position als Mittelwert und die Ungenauigkeit der ermittelten Position als Standardabweichung verwendet. Entsprechend werden die Partikel, die nahe bei der ermittelten GPS Position liegen hoch gewichtet, während weiter entfernte Partikel eine geringe Gewichtung erfahren und bei der nächsten Stichprobenentnahme höchstwahrscheinlich verloren gehen.

Liegen nun mehrere Positionen vor, wäre es das naheliegendste, zur Gewichtung der Partikel mehrere Normalverteilungen mit den entsprechenden Positionen, die die Subsysteme ermittelt haben, als Mittelwert und deren Ungenauigkeiten als Standardabweichungen zu überlagern. So wird die Stärke des Partikelfilters, mehrere wahrscheinliche Positionen gleichzeitig verfolgen zu können ausgenutzt. Allerdings fehlt dann die Beachtung der Zuverlässigkeit der jeweiligen Verortungslösung. Deshalb wird hier das weitere Mittel der Gewichtungsmatrix eingeführt. Es kommt zwar weiterhin zu einer Überlagerung mehrerer Normal-

verteilungen, nämlich für jede Datenquelle eine, aber die Wertigkeiten, die die Partikel bei der Neugewichtung durch die einzelnen Verteilungen erhalten, unterscheiden sich. So können spezielle Datenquellen im Zusammenspiel mit bestimmten anderen in ihrer Relevanz für das Ergebnis gedämpft werden, während andere aufgewertet werden. Tabelle 4.6 zeigt einen Auszug aus der Gewichtungsmatrix.

Schlüssel	G	W	B	I	A
„G“	1				
„W“		0,8			
„B“			1		
„I“				0,01	
„A“					1
„GW“	1	0,6			
„WB“		0,4	1		
„GWB“	0,8	0,4	0,8		
„GWBIA“	0,8	0,1	0,8	0,01	0,8
⋮	⋮	⋮	⋮	⋮	⋮

Tabelle 4.6: Auszug aus der Gewichtungsmatrix, die den Einfluss der verschiedenen Verortungslösungen im jeweiligen Zusammenhang darstellt. G = GPS, W = WLAN, B = Bluetooth, I = IP Adresse, A = Adresse (Hauskoordinate)

Die genannten Werte wurden anhand der bekannten Zuverlässigkeitsdaten der jeweiligen Verortungstechnologie empirisch geschätzt. Die Gewichtungsmatrix ist als typisiertes Dictionary (Dictionary<String, Double[]>) realisiert. Es besteht also aus einem eindeutigen Schlüssel, der in diesem Fall ein String ist und einem Wert, der als Array aus Gleitkommazahlen realisiert ist. Der Schlüssel gibt die zur Verfügung stehenden Datenquellen an. „G“ bedeutet, dass nur Daten aus dem GPS Subsystem vorhanden sind, während „GWBIA“ bedeutet, dass alle Subsysteme valide Daten geliefert haben und in die Berechnung einbezogen werden müssen. Das durch den Schlüssel indizierte Array aus Gleitkommazahlen gibt für jede Datenquelle den Faktor an, um den die Standardabweichung der zugehörigen Normalverteilung variiert wird. Ein niedriger Wert bedeutet eine flache Glockenkurve, die auch weit entfernte Partikel noch gewichtet, während ein hoher Wert

zu einer steilen Kurve führt. Die entsprechende Normalverteilung gewichtet dann nur nahe am Mittelwert liegende Partikel hoch. So können Technologien mit einer hohen Verortungsungenauigkeit neben sehr genauen Technologien verwendet werden, ohne dass Erstere die Verortung durch ihre Abweichungen negativ beeinflussen. Bei der gleichzeitigen Verwendung von GPS und der Verortung anhand der IP Adresse, das heißt bei Technologien, die in der Genauigkeit stark differenzieren, sorgt der entsprechende Eintrag in der Gewichtungsmatrix also dafür, dass die Gewichtungsfunktion, die die Daten des GPS Moduls verwendet, eine Normalverteilung mit genau definiertem Maximum und steilen Flanken erzeugt, während die Daten aus der IP-Verortung in diesem Fall zu einer äußerst flachen Normalverteilung führen. Die Überlagerung der beiden kann also als Summe aus einer Konstanten und einer Normalverteilung $k_{IP} + N_{GPS}(\mu, \sigma)$ betrachtet werden. Das durch N_{GPS} definierte Maximum ändert sich dadurch nicht.

Diese Form der Realisierung ermöglicht es, leicht weitere Datenquellen hinzuzufügen. Dazu erhalten die Arrays ein weiteres Feld und das Dictionary einige weitere Einträge. Die Anzahl der Einträge des Dictionarys entspricht bei n Datenquellen $2^n - 1$ Zeilen, wenn man davon ausgeht, dass für den Fall, dass keine Quelle Daten liefert, kein Eintrag in der Tabelle notwendig ist. Da ein Dictionary in C# die Implementierung einer Hashtabelle ist, wird der Zugriff auch bei großen Datenmengen noch performant funktionieren. In diesem Fall wird die Kombination der passenden Faktoren und ihre Abstimmung, die auch auf Erfahrungswerten beruht, früher zu Problemen führen.

Aus Tabelle 4.6 geht hervor, dass die Summe der Wertigkeiten nicht 1 betragen muss. Im Gegenteil könnten sogar mehrere Datenquellen gleichzeitig einen hohen Wert haben. Dies würde dazu führen, dass bei der Neugewichtung der Partikel in mehreren eventuell verschiedenen Regionen hoch gewichtete Partikel entstehen. Diese werden sehr wahrscheinlich die Stichprobenentnahme des nächsten Schrittes überstehen und der Filter würde in diesem Fall mehrere Maxima verfolgen. Weiterhin hypothetisch könnte sich dann erst in einem viel späteren Schritt eines der Maxima als das echte herausstellen und weiter verfolgt werden.

Treten mehrere Maxima auf, ermittelt der Partikelfilter zwischen diesen einen gewichteten Mittelwert. Dabei fließen wiederum die Koeffizienten aus der Gewichtungsmatrix ein. Die Maxima, die auf höher bewerteten Datenquellen beruhen,

fließen stärker in die endgültige Position ein, während niedriger bewertete Quellen weniger Einfluss bei der Ermittlung des Mittelwertes haben.

Das übrige Verhalten des Partikelfilters entspricht der Implementierung, die in [HEBE07] genauer beschrieben ist.

4.4 Eventsystem

Bei der Aufteilung eines umfangreichen Systems in mehrere Module ist die Kommunikation innerhalb de Programmteile ein entscheidender Faktor. Soll ein Modul in ein Paket eingeschlossen werden, sodass daraus eine eigenständige Bibliothek erzeugt werden kann, hat dies den Vorteil, dass das Modul eine noch höhere Abstraktionsstufe erreicht, als es bei der Kapselung in einer Klasse der Fall wäre. Eine Klasse kapselt verschiedene, thematisch zusammengehörende Funktionen. Zum Beispiel enthält die Klasse System.Math des .NET Frameworks Methoden zur Berechnung mathematischer Funktionen.

Im Fall dieser Anwendung sollen allerdings ganze, nach dem Starten selbständig tätige Einheiten gekapselt werden, die häufig aus einer Ansammlung von Klassen, Aufzählungen, Strukturen und Delegaten bestehen. Von diesen Elementen sollen aber nur wenige für den Benutzer sichtbar sein. Viele Klassen dienen nur internen Abläufen und eine Instanziierung von außerhalb würde zu inkonsistenten Systemzuständen führen . Deshalb bietet sich hier die Verwendung eines Pakets an, in dem man diese internen Klassen verdecken kann, sodass der Benutzer nur sieht, was er sehen darf.

Erzeugt nun ein Programmteil eine Instanz einer Klasse aus einem solchen Paket, kann das aufrufende Programm alle öffentlichen Methoden der aufgerufenen Klasse starten und darüber Parameter an die Instanz übergeben. Der entgegengesetzte Weg ist allerdings nicht möglich. Ein Objekt besitzt keine Informationen über die aufrufende Instanz und kann dieser deshalb keine Daten übertragen. Das ist aus programmiertechnischer Sicht wichtig, denn damit das aufgerufene Objekt aktiv Daten an den Aufrufer übertragen kann, müsste es ja die dafür nötige Methode des Aufrufers schon zur Kompilierzeit kennen. Damit würde aber die Flexibilität der Kapselung verloren gehen, denn es würde ja nur einer bestimmten, vorher festgelegten Art von Objekten möglich sein, Instanzen anderer Objekte aus

fremden Paketen aufzurufen. Dieses Problem der einseitigen Datenübertragung ist jedoch für unser System nicht tragbar. Es muss einerseits möglich sein, dass der Sender Einstellungen der Collectoren korrigiert und andererseits müssen die Collectoren natürlich auch ihre Daten an den Server übertragen können. Außerdem muss das verwendete Konzept auch threadsicher sein, da ja beinahe jedes Modul in einem eigenen Thread arbeitet. In JAVA gibt es hierfür PipedQueues, die man einem Modul im Konstruktor übergeben kann und die dann eine unidirektionale Verbindung zwischen den beiden in unterschiedlichen Threads arbeitenden Programmteilen bilden. Leider kennt .NET dieses Konzept nicht. Deshalb kommt in dieser Implementierung ein anderes Kommunikationsmodell zum Einsatz. Es ist ebenfalls sehr mächtig und erlaubt es, Daten zwischen Threads zu verschieben. Werden statt PipedQueus Events verwendet, können Threads, für die gerade keine Daten vorhanden sind, sogar ohne busy waiting, also das ständige Abfragen der Datenquelle, so lange den Prozessor abgeben, bis neue Daten für sie bereit stehen.

Events sind Ereignisse, die bestimmte Programmkomponenten auslösen, wenn sie einen vorher definierten Zustand erreichen. Ein Button in einer GUI löst zum Beispiel, wenn er geklickt wird das Ereignis ButtonClicked aus. Ein Programm, das eine Aktion starten will, sobald der Button geklickt wurde kann nun dieses Ereignis „abonnieren“. Dazu erstellt es einen Handler für dieses Ereignis. Das ist eine Methode, die mit speziellen Parametern aufgerufen wird, sobald das Ereignis eintritt. Aufrufer ist in diesem Fall das Objekt, das das Ereignis ausgelöst hat. So kann es Daten an ein anderes Objekt übertragen, ohne dass es mehr über das andere Objekt weiß als dass es dort eine Methode gibt, die spezielle Parameter akzeptiert.

Eben ein solches System verwenden sowohl der Client als auch der Server in dieser Implementierung.

Der Aufbau des Clients ist noch einmal in der Grafik 4.6 dargestellt. Diese zeigt in den weißen Pfeilen den Weg, den Informationen mithilfe von Events im System nehmen können. Während der Sender direkt Methoden der Collectoren aufrufen kann, senden diese alle Readings über Events an den Sender. Da sowohl die Collectoren als auch der Sender in einem eigenen Thread arbeiten und die Collectoren keine Informationen über den Sender besitzen, ist eine Kommunika-

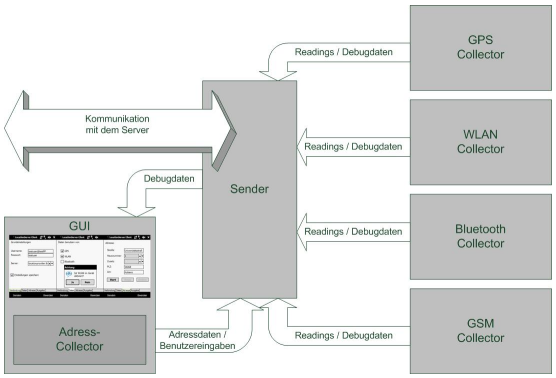

Abbildung 4.6: Aufbau des Location-Provider Clients

tion mit Methoden nicht möglich. Schließlich weiß kein Programmteil, was der andere gerade macht und ob eine Unterbrechung dieser Handlung nicht zu einer Inkonsistenz des Programms führen könnte. Deshalb löst jeder Collector ein Event aus, wenn er neue Daten gesammelt hat. Der Sender registriert schon beim Initialisieren der Collectoren für jeden einen Eventhandler, der ihn über neue Daten informiert. Wird ein Event ausgelöst, holt der Sender die Daten, sobald er nicht mehr ausgelastet ist und speichert sie für die nächste Verwendung, in diesem Fall zum Versenden, zwischen. Genau so werden Debugginginformationen in der Aufruferkette rückwärts bis zur GUI transportiert, wo sie dann dargestellt werden. Auch hierfür gibt es ein Event, das jedes Modul auslösen kann. Der Sender dient dabei für die Collectoren als Aggregator. Er handhabt deren Events und leitet alle Informationen an die GUI weiter, indem er ebenfalls ein Event auslöst.

Im Server ist nicht die Kommunikationsschnittstelle das Zentrum der Intrasystemkommunikation, sondern ein eigens dafür geschriebener Dispatcher. Er ist die Wurzel des Aufrufebaums, denn er startet und initialisiert alle anderen Systembestandteile.

Ein üblicher Kommunikationsablauf sieht dabei wie folgt aus: Die Kommunikationsschnittstelle ServiceStarter erhält eine neue Datenübertragung von einem Client. Sie signalisiert dies durch ein Event. Der Dispatcher registriert das Event und startet einen neuen Thread, in dem er als erstes die SessionId mit den vorhandenen Benutzern vergleicht. Findet er den passenden Nutzer, übergibt er diesem die Daten. Das Lokalisierungsmodul des Nutzers erzeugt nun für jede Datenquelle eine Positionsschätzung. In vielen Fällen sind hierfür Datenbankabfragen notwen-

dig. Diese werden vom Datenbankmodul durchgeführt, das die Zugriffe auf die SQL Datenbank kapselt. Da das Lokalisierungsmodul aber keine Informationen über das Datenbanksystem besitzt, löst es ein Event aus, dass die Anfrage enthält. Der Dispatcher erkennt die Anfrage und das Ziel und gibt sie deshalb an das Datenbankmodul weiter. Der Weg der Antwort führt entsprechend auf dem gleichen Weg rückwärts über den Dispatcher wieder zum Lokalisierungsmodul. Nachdem dieses eine Position ermittelt hat, löst es wieder ein Event aus, das den Dispatcher veranlasst, dem Datenbanksystem die neue Position des Nutzers zum Speichern zu übergeben.

Anfragen eines PrivacyProviders werden ähnlich gehandhabt. Sie erreichen das System ebenfalls über den ServiceStarter. Der Dispatcher erkennt das entsprechende Event und veranlasst das Datenbankmodul, die Position des angefragten Benutzers aus der Datenbank zu lesen. War dies erfolgreich, gibt er die Position über den ServiceStarter als Antwort an den Privacy Provider zurück.

4.5 Positionen

Der einzige Zweck des Location Providers ist der Umgang mit Positionen. Allerdings treten in einem so komplexen System verschiedene Arten von Positionen auf. Da gibt es die letztendlich ermittelten Clientpositionen, die sicher die nach außen hin wichtigste Klasse darstellen, da sie als Schnittstelle mit den nachgeschalteten Programmen des Metasystems wie dem Privacy Provider genau definiert sein müssen.

Variablenname	Datentyp	Erläuterung
longitude	Double	geographische Länge beziehungsweise GK-Rechtswert (abhängig von isGkEncoded)
latitude	Double	geographische Breite beziehungsweise GK-Hochwert (abhängig von isGkEncoded)
isGkEncoded	Bool	Gibt an, ob die Daten in longitude und latitude geographische Längen- und Breitengrade sind oder ob sie im Gauß-Krüger Format vorliegen
gpsQualli	Integer	Die Qualität der GPS-Verortung. Das Feld kann die Werte 0=ungültig, 1=GPS, 2=DGPS und 6=geschätzt annehmen.
longDirection	String	Im Fall von geographischen Positionsangaben entweder „W“ oder „E“ für Westen oder Osten
latDirection	String	Im Fall von geographischen Positionsangaben entweder „N“ oder „S“ für Norden oder Süden
height	Double	Die Höhe über dem Meeresspiegel
vDop	Double	vertikale Genauigkeitsabweichung
hDop	Double	horizontale Genauigkeitsabweichung
dilution	Double	Genauigkeitsabweichung: $\sqrt{vDop^2 + hDop^2}$
bearing	Integer	Blickrichtung im Bereich 0° - 360°
heading	Integer	Bewegungsrichtung im Bereich 0° - 360°
speed	Double	Geschwindigkeit über Grund
status	Double	Eine Aufzählung verschiedener Positionsarten. Siehe eigene Tabelle

Tabelle 4.7: Felder eines GpsPosition Objekts

Doch neben dieser Klasse gibt es noch einige weitere: Die Referenzpunkte, die bei der Positionsschätzung im WLAN- und Bluetoothverorter verwendet werden, bilden eine weitere Klasse. Sie zeichnen sich dadurch aus, dass für sie einige An-

gaben, die die umfangreiche GpsPosition enthält, nicht wichtig sind. Ein Beispiel dafür ist die Bewegungsrichtung, da die Referenzpunkte natürlich ortsfest sein müssen. Andererseits müssen Sie aber Eigenschaften haben, die festlegen, ob es sich bei der Position des Referenzpunktes um eine Schätzung aus einem früheren Schritt handelt oder um eine vom Administrator bestätigte Position. Grundlegende Angaben wie die geographische Länge und Breite oder die Höhe müssen natürlich in jeder Klasse vorhanden sein.

Tabelle 4.7 gibt einen Überblick über alle Felder der Klasse Position. Je nach Verwendungszweck sind einige dieser Felder nicht mit Werten belegt.

Bezeichnung	Erläuterung
Confirmed	Die Position einer Referenzstation hat diesen Status, wenn sie von einem Administrator bestätigt wurde.
Unconfirmed	Positionen von Referenzstationen, die automatisiert in die Datenbank übernommen werden, bekommen diesen Status, bis sie von einem Administrator bestätigt werden.
FormerlyUnknown	Während des Sortiervorgangs, wenn eine Station als neu erkannt wird und ihre Position somit nicht in die Verortung des Clients einbezogen werden soll, wird die Position mit dieser Bezeichnung markiert.
forConsumer	Nachdem der gesamte Verortungsvorgang abgeschlossen ist, wird die Position, die an Verbraucher weitergegeben werden kann, mit dieser Bezeichnung markiert.
notValid	Eine Position, die ansonsten zu einem Fehler im Verortungsvorgang führen würde, kann mit dieser Bezeichnung markiert werden. Das System weiß dann, dass sie verworfen werden kann.

Tabelle 4.8: Positionsstatus

Tabelle 3.8 gibt eine Übersicht über die Enumeration, deren Werte das Feld Status des Objekts Position annehmen kann. Es dient dazu, die Art der Position näher zu beschreiben. Bei Positionsobjekten, die einen Referenzpunkt beschreiben, kann mithilfe dieser Aufzählung dargestellt werden, ob es sich um eine vom Administrator bestätigte Position, eine automatisch ermittelte oder sogar um eine bis zu diesem Zeitpunkt unbekannte Position handelt. Außerdem kann die Position mit dem Wert forConsumer zur Weitergabe an Konsumenten freigegeben und mit dem Wert notValid als ungültig markiert werden. Der Positionsstatus ist eine Aufzählung, die die Art der Position und deren Verwendung angibt.

4.6 Kommunikation

Bei dem hier vorgestellten System handelt es sich entsprechend den in Kapitel 1.3 definierten Fragestellungen um eine Client-Server-Infrastruktur. Genauer gesagt sollen viele Clients gleichzeitig Daten an einen Server übermitteln können. Da sowohl zwischen Client und Server als auch bei der Kommunikation mit anderen Instanzen der Metaarchitektur offene Netzwerke wie das Internet als Übertragungsmedium verwendet werden, ist es nötig, die Daten vor unberechtigtem Zugriff und vor Veränderung zu schützen. Während sich dieses Kapitel mit der Realisierung der Kommunikation befasst, enthält das nächste Kapitel alle Informationen, die sich auf die Sicherung der Übertragung beziehen.

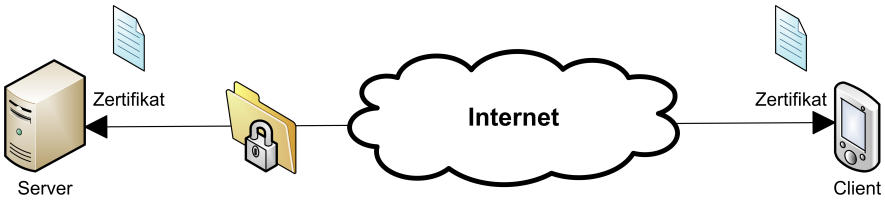

Abbildung 4.7: Modell der verschlüsselten Kommunikation zwischen Client und Server

Die .NET Bibliothek beinhaltet zur Realisierung einer Kommunikation innerhalb räumlich von einander getrennter Systeme über offene Netzwerke die Windows Communication Foundation (WCF). Sie umfasst Funktionen zum einfachen Erstellen, Anbieten und Nutzen von Webservices. Hierzu stehen verschiedene IP basierte Übertragungstechniken zur Verfügung. Während die Verwendung eines

Bytestroms Übertragungsbandbreite spart, eignet sich ein HTTP Endpunkt besser zum Debuggen der Anwendung. Deshalb werden für diese Anwendung HTTP Endpunkte verwendet, die die angeforderten Daten in ein XML Objekt serialisieren und als solches im Klartext an den Kommunikationspartner übertragen. Eine erschöpfende Erklärung zur WCF enthält beispielsweise [HÖKO08]. An dieser Stelle soll nur auf die für die gegenwärtige Implementierung wichtigen Punkte eingegangen werden. Um einen WCF Dienst anzubieten, gibt es zwei Möglichkeiten. Zum einen kann der Dienst in den Internet Information Services (IIS), dem von Microsoft entwickelten Webserver, gehostet werden. Dies hat den Vorteil, dass sich der Entwickler nicht um das Hosting selbst kümmern muss. Es ist nur der IIS zu konfigurieren, sodass er an einem bestimmten Endpunkt den vorher definierten Dienst zu Verfügung stellt. Dieser Ansatz erschien für das vorliegende Projekt allerdings zu unflexibel. Durch einen im Programm selbst gehosteten Service gewinnt das System einen höheren Grad an Flexibilität, da keine Zusatzsoftware nötig ist. Weiterhin ist laut Microsoft [MICR2002] die Performance eines nicht im IIS gehosteten Webservice gerade bei einer höheren Benutzerzahl besser.

Die WCF bietet jedoch auch die Möglichkeit, Dienste direkt in Anwendungen zu hosten. Hierzu erstellt man eine Schnittstellendefinition des Dienstes und die dazu gehörende Implementierung. Diese wird dann mit der URI, unter der der Dienst erreichbar sein soll, an einen ServiceHost übergeben. Der ServiceHost stellt die gesamte Logik zum Empfangen von Nachrichten und zum Senden von Antworten zur Verfügung. Sobald er an einen Endpunkt gebunden wird, öffnet er einen Port auf der lokalen Maschine und wartet auf Anfragen. Als Endpunkte stehen HTTP, Bytestrom und viele weitere Verbindungstypen zur Verfügung. Da die mobilen Geräte unter dem .NET Compact Framework nur eine funktionsreduzierte Version der WCF besitzen und deshalb nur HTTP-Bindungen kennen, müssen in diesem Fall solche Bindungen verwendet werden.

Die vorliegende Implementierung verwendet zwei ServiceHosts. Einer ist für die Verbindung zu den Clients zuständig, während der andere die Kommunikationsschnittstelle zu den nachgeschalteten Diensten wie dem Privacy Provider bildet. Auf diese Weise können die Verbindungseinstellungen wie die Übertragungsmethode oder die Verschlüsselungsmethode für die jeweiligen Arten von Dienstnutzern getrennt von einander konfiguriert werden. Beispielsweise wäre so

eine unterschiedliche Authentifizierungsmethode möglich. Während sich Clients mit Benutzername und Passwort am System anmelden, könnten die Dienste sich untereinander mit Zertifikaten authentifizieren. Das folgende Listing zeigt beispielhaft den Start eines der beiden Dienste. Er heißt ClientSideService und realisiert die Kommunikation mit den Clients.

```
/* Dienstimplementierung erstellen */
ClientSideService clientSideService = new ClientSideService();
/* Dienstadresse festlegen */
Uri baseAddressClients = new
    Uri(http://[Serveradresse]/Locationprovider/...);
/* ServiceHost erstellen */
ServiceHost clientService;
clientService = new ServiceHost(clientSideService,
    baseAddressClients);
/* ServiceEndpunkt hinzufügen */
clientService.AddServiceEndpoint(typeof(IClientSide),
                                 new BasicHttpBinding(),
                                     "ClientSideService");
```

Listing 4.7: Ablauf des Startens eines WCF Webservices

Soll an diesem Endpunkt eine Verschlüsselung der Kommunikation stattfinden, zum Beispiel anhand von Zertifikaten, könnte dies ebenfalls am ServiceHost Objekt konfiguriert werden. Dazu müssten die Credentials Objekte für Client und Server entsprechend geändert werden, sodass .NET im richtigen Zertifikatsspeicher nach den Zertifikaten der Clients und Dienste sucht.

Wie ein solcher in der Anwendung gehosteter Dienst konsumiert werden kann, wird in Kapitel 4.10 beschrieben. Um die Erstellung weiterer Komponenten nicht zu erschweren, sind in der momentan auf dem Testsystem laufenden Implementierung alle Verschlüsselungsmechanismen deaktiviert, damit die zur Verfügung gestellten Dienste von Dritten leicht konsumiert und ausgelesen werden können.

Eine Kommunikationssession zwischen einem Client und dem Server läuft in zwei Phasen ab. Die erste Phase ist das Login. Dieser findet, wie im nächsten Kapitel näher beschrieben in einem Challenge-Response Verfahren statt.

Ist das Login erfolgreich abgeschlossen, kann der Client mit der Datenübertragung beginnen. Dazu ruft er die Methode

```
public bool submitData(InfoContainer info)
```

auf, die als Parameter ein Objekt des Typs InfoContainer erhält. Dieses Objekt enthält alle vom Client gesammelten Daten seiner Umgebung. Außerdem beinhaltet es einen Zeitstempel und die SessionID des Clients. Der Rückgabewert der Methode ist ein boolescher Wert, der angibt, ob der Server die Übertragung akzeptiert hat. Zur Übertragung wird das InfoContainer Objekt in einen XML Datenstrom serialisiert und am Endpunkt des Servers wieder deserialisiert. Eine Nachricht könnte folgendermaßen aussehen:

```
<s:Envelope
    xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
<s:Body>
<submitData>
  <info>
    <address>
      <street>Gymnasialstrasse</street>
      <number>2</number>
      <numberAdd></numberAdd>
      <city>Koblenz</city>
      <zip>56068</zip>
    </address>
    <apData>
      <AP>
        <mac>00:18:f8:cc:a3:f3</mac>
        <signalStrengthInDb>-56</signalStrengthInDb>
        <signalStrengthInWords>Excellent</signalStrengthInWords>
        <ssid>Tropisch</ssid>
      </AP>
      <AP>
        <mac>00:1c:4a:07:55:5b</mac>
        <signalStrengthInDb>-62</signalStrengthInDb>
        <signalStrengthInWords>VeryGood</signalStrengthInWords>
```

```
        <ssid>jojo</ssid>
      </AP>
    </apData>
    <bluetoothData>
      <BluetoothStation>
        <mac>0009DD105B95</mac>
        <name>TROJA</name>
      </BluetoothStation>
    </bluetoothData>
    <gpsData>
      <bearing>0</bearing>
      <gpsQualli>normalGPS</gpsQualli>
      <height>75.8</height>
      <isGkEncoded>false</isGkEncoded>
      <latDirection>N</latDirection>
      <latitude>5024.3439266</latitude>
      <longDirection>E</longDirection>
      <longitude>735.064175</longitude>
      <hDop>2.3</hDop>
      <vDop>1.4</vDop>
    </gpsData>
    <ipAddress xsi:nil="true"/>
    <sessionId>1620745768</sessionId>
    <timestamp>2010-04-10T14:47:55+02:00</timestamp>
  </info>
</submitData>
</s:Body>
</s:Envelope>
```

Listing 4.8: eine SOAP Nachricht, die alle gesammelten Daten eines Clients enthält. Die Namespacebezeichner wurden entfernt, um die Übersichtlichkeit zu verbessern.

Es ist zu beachten, dass die Adresse nur übertragen wird, wenn der Benutzer dies explizit wünscht. In dem oben zitierten beispielhaften Datensatz hätte der Benutzer das Senden der Adresse also aktiviert. Anhand der durch die Adresse

gewonnenen Position könnte der Server dann die beiden ebenfalls gefundenen Accesspoints und den Bluetoothsender in die Datenbank aufnehmen, sofern diese noch unbekannt wären.

Offensichtlich werden die Nachrichten mit zunehmender Menge an registrierbaren Accesspoints und Bluetoothstationen schnell sehr umfangreich. Eine Betrachtung der übertragenen Datenmenge liefern die in Kapitel 5.2 vorgestellten Versuche.

Damit die automatische Serialisierung allerdings erfolgreich abläuft, müssen einige Dinge beachtet werden. Zum einen muss man von Hand einen DataContract festlegen. Dabei handelt es sich in C# um ein Attribut einer Klasse, das anzeigt, dass ein Objekt serialisiert werden kann. Weiterhin kann ein DataContract die Serialisierung auch selbst implementieren. Die gleiche Funktion wie der DataContract für Objekte hat der MemberContract für Felder von Objekten. Im einfachsten Fall zeigt man mit dem Setzen des Attributs die Serialisierbarkeit wie folgt an:

```
using System.Runtime.Serialization;

[DataContract]
public class SollInWCFSerialisiertWerden{
        [MemberContract]
        public int irgendeineZahl;
}
```

Listing 4.9: Die DataContract- und MemberContractattribute

Ein weiterer gängiger Fehler bei der Serialisierung in WCF ist das Feld public bool <FieldName>Specified. Dabei handelt es sich um ein Feld, das einen booleschen Wert enthält und den gleichen Namen wie ein vom Benutzer angelegtes Feld mit dem Suffix -Specified hat. Es wird beim Abfragen des Dienstes über die WSDL von der Laufzeitumgebung automatisch angelegt, um zwischen einem nicht angegebenen und einem nicht Standardwert im zugehörigen Feld unterscheiden zu können. Zum Beispiel hat ein Integer, wenn es deklariert wird, automatisch den Wert 0, ohne dass ihm jemals explizit dieser Wert zugewiesen wurde. Genau so könnte das Feld aber auch mit voller Absicht des Benutzers den Wert 0 enthal-

ten. Setzt man den Wert des Specified-Feldes nicht von Hand auf „true“, nimmt die Laufzeitumgebung an, dass der Inhalt des korrespondierenden Feldes nicht spezifiziert ist und überträgt es deshalb auch nicht. Betrachtet man das weiter oben entwickelte Beispiel, würde das Objekt, das über den Webservice übertragen wird, nicht mehr nur ein Feld namens irgendeineZahl enthalten, sondern noch ein weiteres namens irgendeineZahlSpecified. Ist dieses nicht „true“, wird irgendeineZahl nicht serialisiert und übertragen werden.

Einen Verbindungsabbau gibt es nicht. Jeder Client kann seine Daten in unregelmäßigen Abständen übertragen, deshalb ist eine Verfolgung der Aktivitäten des Clients schwierig. Außerdem kann es zu Netzwerkausfällen jeder Art kommen bis hin zu einem kompletten Wechsel der Übertragungstechnik beim Client, zum Beispiel von UMTS zu WLAN. Es hat sich jedoch gezeigt, dass nach 15 Sekunden, ohne dass eine weitere Nachricht empfangen wurde, die Wahrscheinlichkeit einer erneuten Verbindungsaufnahme äußerst gering ist. Deshalb entfernt der Location Provider alle Clients aus seiner Datenbank, die länger als 15 Sekunden inaktiv waren.

Da sich beim Testen gezeigt hat, dass es häufig zu Verbindungsproblemen aufgrund der niedrigen Bandbreite kam, die im Netzwerk verfügbar war, werden Timeout-Fehler des Kommunikationsstacks vom Programm abgefangen. Tritt ein Timeout auf, wird die Frequenz der übertragenen Datenpakete gesenkt. Dies geschieht, indem das Intervall des Timers, der das Absenden der Datenpakete auslöst, pro Timeout um eine Sekunde erhöht wird.

4.7 Sicherheit

Im Folgenden wird der Prototyp anhand der in Kapitel 2.10.1 eingeführten Sicherheitsanforderungen untersucht. Dabei wird basierend auf der Definition der jeweiligen Anforderung das implementierte Verfahren beschrieben und bewertet.

Nicht-Abstreitbarkeit spielt im Rahmen des Metasystems, zu dem der Location Provider als ein Teil gehört, eine wichtige Rolle bei der Inanspruchnahme von Mehrwertdiensten. Denn es muss zu jedem Zeitpunkt nachvollziehbar sein, welcher Kunde welchen Dienst in Anspruch genommen hat, so dass eine korrekte Abrechnung durch den Payment-Provider durchgeführt werden kann. Im Fall des

Location Providers selbst ist dieses Kriterium jedoch zweitrangig. Eine Datenübertragung eines Clients muss diesem zwar zugewiesen werden, da darauf die Positionsschätzung basiert, es ist jedoch nicht unbedingt nötig, im Nachhinein noch eine zweifelsfreie und beweisbare Zuordnung treffen zu können. Im Rahmen der Anforderungen dieser Arbeit deckt die Verwendung von Zertifikaten den Bedarf an Nicht-Abstreitbarkeit vollkommen ab.

Um die Verfügbarkeit des Systems beurteilen zu können, muss sowohl die Hardware- als auch die Softwareverfügbarkeit betrachtet werden. Also ist sowohl die Stabilität und Langlebigkeit der eingesetzten Hardware als auch die Ausfallsicherheit des Betriebssystems zu betrachten. Die Hardware sollte aus Komponenten bestehen, die für den Serverbetrieb zertifiziert sind. Damit unterscheiden sie sich in ihrer Zuverlässigkeit, aber auch im Preis erheblich von Endkundenkomponenten. Als Betriebssystem kommt das laut [WEB2010] mit einem Marktanteil von etwa 25,27% verbreitete Windows Server 2008 R2 in Frage, da die Verfügbarkeit von Security Patches für dieses System noch lange vom Hersteller Microsoft garantiert wird.

In der Vergangenheit gab es häufiger den Fall, dass das Windowssystem zur Installation von automatisch heruntergeladenen Updates eigenmächtig neu gestartet hat. Da die Installation von Updates grundsätzlich die Sicherheit des Systems erhöht, soll dies beibehalten werden. Deshalb ist inzwischen ein Neustart des Location Providers nach einem Reboot sichergestellt.

Da es sich bei der vorliegenden Implementierung um ein Multiusersystem handelt, muss auch eine Lastabschätzung zu einem Zeitpunkt durchgeführt werden, an dem mehrere Benutzer angemeldet sind, um die Verfügbarkeit bei starken Beanspruchungen zu testen. Dies erfolgte in einem Versuch, der in Kapitel 5.4 dokumentiert ist.

Anonymität ist eine Sicherheitsanforderung, die im vorliegenden System selbst nicht auftaucht. Wie im Kapitel 4.7.1 beschrieben, bekommen nur Benutzer Zugriff auf das System, die dazu berechtigt sind und auch während der gesamten Dauer einer Verbindung ist ein Benutzer eindeutig identifizierbar.

Das Metasystem, zu dem der Location Provider gehört, ist jedoch so aufgebaut, dass ein Benutzer gegenüber einem Dienst anonym sein kann, wenn er möchte. Dies stellt der Privacy Provider sicher, der als eigene Arbeit realisiert

wurde, später jedoch eng mit dem Location Provider vernetzt sein wird. Weitere Informationen zum Privacy Provider bieten [STEI08] und die auf dem Entwurf basierenden Arbeiten.

Im Folgenden werden die für diese Arbeit besonders wichtigen Sicherheitskriterien und die Maßnahmen, die zu deren Umsetzung ergriffen wurden, noch einmal einzeln vorgestellt.

4.7.1 Authentizität

Wie schon der Kurzbeschreibung zu entnehmen ist, muss die Echtheit einer Entität nachvollziehbar sein. Entität deshalb, weil es sich hier sowohl um eine Person als auch um einen Dienst oder ein Gerät handeln kann. Diese drei Fälle, die zum Teil nicht ganz trennscharf sind, sollen hier nacheinander betrachtet werden.

Die Echtheit einer Maschine kann über ein qualifiziertes Zertifikat bestätigt werden. Dabei handelt es sich nicht nur um eine Verbindung von Schlüssel und Identität. Zusätzlich ist das Zertifikat noch von einem vertrauenswürdigen Aussteller unterzeichnet. Kapitel 4.7.2 beschäftigt sich mit der Verwendung von Zertifikaten.

Um die Echtheit einer Person zu garantieren, werden in der Literatur zwei Möglichkeiten genannt. Beiden liegt die Idee zu Grunde, dass die Person über etwas Einzigartiges verfügt. Dies kann ein materieller Besitz wie eine Keykarte oder ein USB Stick sein. Allerdings würde sich eine solche Hardware nur schlecht über die designierte Clienthardware auslesen lassen. Schließlich sind hier Smartphones geplant und diese haben in den seltensten Fällen einen USB Rootport oder einen von außen zugänglichen Kartenleser.

Die zweite Möglichkeit ist ein geheimes Wissen, das eine Person eindeutig identifiziert. Im Umfeld von Computernetzwerken wird das geheime Wissen meist in Form eines Passwortes abgefragt. Auch das vorliegende System verwendet zur Identifikation der Nutzer ein selbst geschriebenes Authentifikationsverfahren, das auf einem geheimen Wissen basiert. Es verwendet Benutzernamen und Passworte, mit denen sich die Benutzer authentifizieren. Wie in Kapitel 4.6 beschrieben, kommt es zu Beginn jeder Verbindung zu einem Login. Hierbei wird der Loginname und das Passwort des Benutzers abgefragt. Damit das Passwort auch vom

Serveradministrator nicht demaskiert werden kann, ist es in der Datenbank nicht direkt gespeichert. Stattdessen ist nur sein MD5-Hash hinterlegt. Was ein Hash ist, ist in Kapitel 2.10.2 erklärt. Beim Login überträgt der Benutzer ebenfalls nicht sein Passwort, sondern die Clientsoftware errechnet daraus den Hash und schickt diesen an Stelle des eigentlichen Passwortes an den Server. Dieser vergleicht den empfangenen und den gespeicherten Wert und gibt bei Gleichheit eine SessionID zurück.

Durch dieses Vorgehen ist die Vertraulichkeit des Passwortes selbst geschützt, der Loginvorgang allerdings noch nicht. Denn ein potentieller Angreifer könnte den Hash des Passwortes unterwegs abfangen und eine Replayattacke starten. Dabei sendet er einfach den vorher mitgelesenen Hash erneut. Der Server kann nicht unterscheiden, ob das Passwort neu kodiert wurde oder ein Mitschnitt gesendet wird. Eine Lösung hierfür schlägt [GRIM05] mit dem Challenge-Response Verfahren vor. Die Idee dabei ist, dass der Server dem Client eine Zufallszahl mitteilt, die in die Generierung des Hashs einbezogen wird. Dies führt dazu, dass der Passworthash von einem Loginvorgang zum nächsten trotz gleichen Passworts vollkommen unterschiedlich aussieht. Eine Replayattacke ist nicht mehr möglich.

In der vorliegenden Software liefert der Client als Antwort auf die Zufallszahl des Servers eine Art keyed Hash zurück. Zuerst erzeugt er einen Hash des Passwortes, hängt dann die Zufallszahl an und erzeugt noch einen Hash. Ein zweistufiges Vorgehen ist hierbei nötig, da der Server nicht das Passwort, wohl aber dessen Hash kennt. Er braucht also einfach nur das maskierte Passwort aus der Datenbank und die Seriennummer zu verketten. Diese Zeichenkette wird gehasht und mit der des Clients verglichen. So hält sich auch der Rechenaufwand des Servers in Grenzen.

Die Zufallszahl wird im weiteren Ablauf der Kommunikation als SessionID verwendet, über die der Client ausschließlich identifiziert wird.

4.7.2 Integrität und Vertraulichkeit

Hierbei handelt es sich um nahe verwandte Themen. Integrität der Daten bedeutet, dass genau die Daten und nur die Daten, die der Sender abgeschickt hat, auch tatsächlich beim Empfänger ankommen. Es darf während der Übertragung, die

laut Vorgaben über potentiell unsichere Netze erfolgen können muss, einem Angreifer nicht möglich sein, die Daten zu ändern. Die Möglichkeit eines unbefugten Dritten, die übertragenen Daten mitzulesen, beeinflusst diese Sicherheitsanforderung nicht. Hier greift die Anforderung der Vertraulichkeit. Vertrauliche Daten sollen nur von berechtigten Personen eingesehen werden dürfen.

Die Umsetzung der Anforderungen von Integrität und Vertraulichkeit zusammen genommen bewirken also, dass ein unbefugter Dritter die übertragenen Daten weder einsehen noch in irgendeiner Form verändern kann.

Der einzige Weg, während der Übertragung der Daten über Netze, die unter fremder administrativer Kontrolle stehen, Vertraulichkeit zu erreichen ist, die Übertragung von einen Ende zum anderen zu verschlüsseln. So lange, wie der verwendete Schlüssel ausreichend sicher und nur den beiden kommunizierenden Parteien bekannt ist, kann kein Angreifer die Daten einsehen. Selbstverständlich kann er sie weiterhin verändern, indem er zum Beispiel Datenpakete aus dem Datenstrom entnimmt. Dies wird zwar durch die TCP Sequenznummern, die jedes Datenpaket und deren Reihenfolge identifizieren, schon auf Netzwerkebene erschwert, ist aber nicht unmöglich. Deshalb ist neben der Verschlüsselung der Daten noch ein weiteres Instrument nötig, um Manipulationen erkennen zu können - das Signieren der Daten. Es ist also ein asynchrones Verschlüsselungsverfahren notwendig wie in Kapitel 2.10.3 beschrieben.

Einen günstigen Weg, solche Schlüssel zu verteilen, stellt eine Public Key Infrastructure (PKI) dar. Einen Überblick bietet Kapitel 2.10.5.

Die Windows Communication Foundation unterstützt die Verwendung von Zertifikaten zur verschlüsselten Übertragung und Integritätsprüfung. Um die integrierten Methoden verwenden zu können, muss zuerst ein gültiges Zertifikat mit öffentlichen und privaten Schlüsseln erzeugt werden. Hierfür gibt es Open Source Programme, die Zertifikatshierarchien mit Wurzelzertifikaten und deren Kindern erstellen können. Ein Beispiel eines solchen Programms wäre OpenSSL (siehe [OSSL]). Nachdem die Zertifikate erzeugt sind, werden sie unter Windows in den zentralen Zertifikatsspeicher kopiert. Dieser befindet sich in der Managementkonsole (mmc.exe), Datei -> SnapIn Hinzufügen -> Hinzufügen -> Zertifikate. In die Konsole wird dann eine Baumstruktur mit mehreren Zertifikatsspeichern aufgenommen. Das Wurzelzertifikat sollte man nach „vertrauenswürdige Herausgeber“

und das Clientzertifikat nach „Eigene Zertifikate“ importieren. Analog, jedoch weniger sicher, können die Zertifikate auch als Dateien im Verzeichnisbaum abgelegt werden.

Anschließend kann in der Entwicklungsumgebung die Verwendung von Zertifikaten aktiviert werden:

```
ServiceHost.Credentials.ServiceCertificate.Certificate =
        new X509Certificate2(String filename, String password)
```

Listing 4.10: Zertifikat zur Verschlüsselung zu einem WCF Service hinzufügen

Beim Location Provider Client gestaltet sich das Einfügen von Zertifikaten in den lokalen Zertifikatsspeicher etwas schwieriger, da die Bordmittel von Windows Mobile nur lesenden Zugriff erlauben. Um Zertifikate in den Speicher importieren zu können, ist ein externes Programm notwendig. Für diese Implementierung wurde das Tool P12Import (zu finden unter [P12I]) verwendet. Am Client registriert man das Zertifikat analog zum Server:

```
ServiceClient.ClientCredentials.ClientCertificate.Certificate =
        new X509Certificate2(String filename, String password)
```

Listing 4.11: Zertifikat zur Verschlüsselung zu einem WCF Client hinzufügen

Diese Einstellungen können zusätzlich in der XML Konfigurationsdatei namens App.config vorgenommen werden, die .NET zu jeder Anwendung anlegt. In ihr werden auch viele weitere Details der Verbindung festgelegt. Visual Studio 2008 bietet ein eigenes Tool, um WCF Einstellungen in der Anwendungskonfigurationsdatei vorzunehmen. Es heißt „WCF-Dienstkonfigurationseditor“ und befindet sich unter dem Reiter „Extras“ in der Menüleiste der IDE.

4.8 Umgang mit unbekannten Referenzstationen

Wie jede Computerhardware unterliegen auch drahtlose Netzzugangstechnologien einer ständigen Weiterentwicklung. Wo vor wenigen Jahren noch WLAN Accesspoints nach IEEE 802.11b, die eine Geschwindigkeit von 11 MBit/Sekunde boten, verwendet wurden, hängen heute mindestens 54 Mbit/Sekunde schnelle

Geräte nach IEEE 802.11g und der nächstschnellere Standard 802.11n ist schon entwickelt.

Diese Arbeit verwendet intensiv Signale von privaten Accesspoints, die in den öffentlichen Raum hinein strahlen, als Referenzstationen für die Verortung der mobilen Clients. Gerade hier ist jedoch die Gerätefluktuation vergleichsweise hoch, da ein Gerätewechsel durch die hohen Subventionen begünstigt wird, die Netzbetreiber beim Abschluss eines neuen Vertrags für neue Hardware gewähren. So kommt es in regelmäßigen Abständen zur Ersetzung schon registrierter Stationen, die dann als Altlast ohne praktischen Nutzen in der Datenbank zurück bleiben. Gleichzeitig werden an bereits kartographierten Stellen neue Geräte aufgestellt, die in der Datenbank noch nicht enthalten sind.

Glücklicherweise werden nicht alle Geräte an einem Ort zur gleichen Zeit ersetzt. Dadurch sollte es in den meisten Fällen weiterhin möglich sein, an Orten, an denen eines der vorhandenen Geräte ausgetauscht wurde, eine Positionsschätzung zu ermitteln. Doch wenn nach und nach alle Geräte ausgetauscht werden, müsste die vorher gut kartographierte Gegend plötzlich wieder als unbekannt gelten. Um dies zu vermeiden, werden neue Referenzstationen, deren bisher unbekanntes Signal in einem Datensatz enthalten ist, den ein Client übermittelt hat, automatisch in die Datenbank aufgenommen. Dazu werden zuerst die bekannten von den unbekannten Referenzstationen getrennt. Anhand der bekannten Stationen und den Daten des GPS Empfängers, sofern diese vorhanden sind, wird dann eine Positionsschätzung ermittelt. Liegt deren Standardabweichung unter einer Höchstgrenze von 100 Metern, trägt der Server die unbekannten Referenzstationen automatisch in seiner Datenbank ein. Dabei verwendet er die gerade ermittelte Position des Clients, der die Signale der Stationen registriert und übertragen hat, als Standort der neu einzutragenden Stationen. Durch die Festlegung der Höchstgrenze von 100 Metern wird verhindert, dass bei einer zu schlechten Signallage absolut verfälschte Werte in die Datenbank aufgenommen werden. Dies gilt insbesondere, wenn nur die IP-Adresse als Informationsquelle für eine Positionsschätzung vorhanden ist. Wie Kapitel 5.5 zeigt, ist diese Art der Verortung in höchstem Maße ungenau. Die Eintragung einer neuen Referenzstation würde hier einen Fehler in die Datenbank einbringen, der im schlimmsten Fall andere Verortungsvorgänge beeinträchtigen könnte.

Dieser maschinengestützte erste Kontrollmechanismus wird von einem weiteren unterstützt, der die Kontrollfähigkeiten eines Menschen erzwingt. Bei der automatischen Eintragung gerade registrierter Referenzstationen werden diese mit einem speziellen Flag versehen. Diese Markierung sagt aus, ob die Position als von einem Administrator bestätigt gelten kann. Ist es gesetzt, erhält die entsprechende Referenzstation dadurch bei den weiteren Verortungsvorgängen einen höheren Stellenwert, denn nach der Kontrolle der Station durch den Administrator gilt die Position als bestätigt, sofern dies explizit vermerkt wird.

4.9 Software Dritter

Ein Grundsatz der objektorientierten Programmierung ist die Wiederverwendbarkeit von Code (vergl. [KRUE00]). Durch die Kapselung von Eigenschaften in Klassen und von Funktionen in Paketen wird diese in einem hohen Grade erreicht. Das führt dazu, dass viele programmiertechnische Aufgaben schon mehrfach gelöst worden sind und man als Entwickler Zugriff auf die so geschaffenen Funktionalitäten hat. Man kann sich also auf die eigentliche Aufgabe konzentrieren, ohne sich vorher um Nebensächlichkeiten kümmern zu müssen.

Ein Beispiel für eine solche Sammlung von vorgefertigten Funktionen ist das Entwicklungsframework, das heutzutage jedem Compiler beiliegt. Im Fall von .NET ist das das .NET Framework, bei JAVA heißt es JAVA Software Development Kit.

4.9.1 Gauß-Krüger Umrechnung

Die Software zur Umrechnung von sphärischen WGS84 Koordinaten in kartesische Gauß-Krüger Koordinaten stammt von Kay Zobel und lag ursprünglich als PERL Skript vor. Es wurde für diese und weitere Arbeiten des Instituts nach C# übersetzt. Das Script akzeptiert normalisierte sphärische Koordinaten, die auf dem Referenzellipsoid WGS84 basieren und transformiert diese in mehreren Schritten in kartesische Koordinaten, die auf dem Bessel-Referenzellipsoid basieren.

4.9.2 MatNET Iridium

Hier handelt es sich um eine einfach C# Bibliothek, die mathematische Funktionen in einer wesentlich umfangreicheren Menge zur Verfügung stellt, als das .NET-eigene Paket System.Math dies tut. Die Funktionen der Bibliothek finden im Partikelfilter Verwendung. Dort realisieren sie die Normalverteilung, die die Gewichtung der einzelnen Partikel steuert.

4.9.3 WLAN Positionsbestimmung

Die Positionsbestimmungsengine für die WLAN Lokalisierung, die in dieser Arbeit verwendet wird und die aus der Anwesenheit eines oder mehrerer WLAN Accesspoints im Empfangsbereich des Clients auf dessen Position schließt, stammt aus der Studienarbeit von Dean Wickert [WICK05]. Sie fand auch schon in [HEBE07] Verwendung und zeichnet sich durch ihre Robustheit bei einer annehmbaren Performance aus. Die Engine arbeitet mit einem vom Autor selbst entwickelten Laterationsverfahren, das die Accesspoints als Referenzpunkte verwendet. Es berücksichtigt sogar, ob ein Accesspoint eine nicht gerätespezifische Antenne besitzt, die die Sendeleistung beeinflussen könnte.

4.9.4 OpenNETCF Smart Device Framework

Das Smart Device Framwork des Herstellers OpenNetCF ist eine Sammlung von Zusatzbibliotheken für das .NET Compact Framwork. Es beinhaltet Bibliotheken zum vereinfachten Zugriff auf die Funktionen eines Smartphones. So ermöglicht es die Konfiguration der WLAN– und Bluetoothhardware und das Auslesen von Daten aus diesen mit Managed Code. Das soll es dem Entwickler ermöglichen, mit einfachen Mitteln Aufgaben zu erfüllen, die mit den Fähigkeiten des .NET Compact Frameworks nur mit großem programmiererischem Aufwand zu erfüllen wären. So soll die Lücke zwischen dem für vollwertige Rechner existierenden .NET Framework und dem .NET Compakt Framework für Smartphones und PDAs geschlossen werden.

4.9.5 MaxMind GeoLight City

Bei dieser Software handelt es sich um eine Lösung zur Verortung von IP Adressen [MAXM]. Die Übermittlung der zu verortenden Adresse kann entweder online über einen Webservice erfolgen oder lokal unter Verwendung einer Datenbank und einer API mit Managed Code, die beide online bezogen werden können. Ein Verortungsvorgang liefert die folgenden für uns relevanten Angaben: Ursprungsland, Bundesland, Stadt, geografische Länge und Breite und einige Daten des Internetserviceproviders. Für Kunden aus den USA stehen weiterhin die Postleitzahl und genauere Regionalcodes zur Verfügung.

4.10 Weiterverwendung in anderen Projekten

Eines der Designziele aus Kapitel 3 ist die Wiederverwendbarkeit des Systems und seiner Komponenten in weiteren Arbeiten wie dem mGeoWiki-Projekt (vergl. [MGWI08]). Dieses Ziel wurde durch den modularen Aufbau der Clientsoftware erreicht, die nun mehrere Integrationsstufen zulässt. Bei der einfachsten Art der Integration läuft der Client eigenständig auf dem mobilen System und die Komponente, die die Positionsschätzungen konsumiert ruft in regelmäßigen Abständen eine Position vom Server ab. Hierzu muss der Client auf dem mobilen System installiert und wie gewöhnlich mit dem Server verbunden werden.

Weiterhin ist es möglich, die relevanten Teile des Clients in eigene Anwendungen zu integrieren. Die nötigen Parameter müssen dann über eine Konfigurationsdatei übergeben werden, da die GUI des Clients ein eigenes Modul bildet und als Teil eines anderen Programms schlecht in dieses eingebunden werden kann. Hierzu bindet der Entwickler die Bibliotheken

```
uniko.thebel.DataCollectors
uniko.thebel.Configuration
```

in seiner Entwicklungsumgebung ein. Dadurch werden im aktuellen Entwicklungskontext die Bibliotheken für das Datensammeln und Versenden (DataCollectors) und die Erstellung einer Konfigurationsdatei (Configuration) bekannt. Durch das Instanzieren eines neuen Objekts vom Typ Sender (ConfigItem config) werden alle Kollektoren und die Verbindung zum Server initialisiert. Die Methode

```
public void startSending(ConfigItem config)
```

bewirkt einen Verbindungsaufbau zum Server. Sobald die Verbindung besteht, sammelt der Treiber Daten und schickt diese einmal pro Sekunde an den Server. Durch den Aufruf von

```
public void stopSending()
```

wird das Senden beendet und die Verbindung geschlossen.

Die Verwendung der Bibliothek in der eigenen Anwendung ermöglicht es auch, direkt über die gleiche Verbindung Positionsdaten zu beziehen. Hierzu wird die Methode

```
public Position getMyPosition()
```

aufgerufen. Sie liefert ein Objekt vom Typ Position vom Server, das die bekannten Felder enthält. Als Zugangsdaten werden die Daten verwendet, die in der Konfiguration angegeben sind.

Anwendungen und Dienste, die nicht die Kollektorenbibliothek implementieren, sondern den eigenständigen Client nutzen, müssen selbst eine Verbindung zum Server aufbauen, um Positionsdaten herunterladen zu können. Dazu wird mit dem Programm netcfsvcutil.exe aus dem Microsoft Windows Mobile Development Kit oder svcutil.exe aus dem Microsoft Windows SDK mit einem der folgenden Befehle eine Schnittstellendefinition des Servers erzeugt:

```
NetCFSvcUtil.exe
        http://locationprovider.lbs.iwvi.uni-koblenz.de/
                                LocationServer/Services


svcutil.exe
        http://locationprovider.lbs.iwvi.uni-koblenz.de/
                                LocationServer/Services
```

Listing 4.12: Webschnittstelle für Services erzeugen

Das Programm NetCFSvcUtil.exe erzeugt eine Serviceschnittstelle, die in einem Programm für das .NET Compact Framework, also auf mobilen Endgeräten lauf-

fähig ist, während svcutil.exe eine Schnittstelle für das normale .NET Framwork erzeugt.

Die generierten zwei Dateien können nun in das eigene Projekt eingebunden werden. Indem man eine Instanz des ServiceSideClient anlegt, erhält man eine einfach zu benutzende Schnittstelle zum Location-Provider. Wie weiter oben schon erklärt, erhält man mit der Membermethode

```
public Position getMyPosition(String user, String pass)
```

direkt vom Server ein Objekt des Typs Position, das alle wichtigen Angaben enthält. Dieser Vorgang erfolgt vollkommen transparent. Der Entwickler braucht sich nicht um Netzwerkinteraktion zu kümmern.

Es empfiehlt sich, einen Timer aus der Threadingbibliothek einzusetzen, um etwa jede Sekunde eine aktuelle Position abzurufen.

Da die Kommunikation zwischen dem Server und den diversen Clients generell verschlüsselt erfolgen sollte, sind zum ordnungsgemäßen Betrieb Zertifikate nötig. Um ein Testen zu ermöglichen, werden diese ebenso wie die Verschlüsselungsparameter nur optional abgefragt beziehungsweise sind deaktiviert.

Kapitel 5

Versuche

5.1 WLAN Positionsschätzung

Der WLAN Chipsatz des Testgerätes HTC Touch Diamond 2 ermöglicht es nicht, die Signalqualität in Dezibel über die Software auszulesen. Stattdessen erhält man als Indikator für die Stärke des empfangenen Signals einen der folgenden Werte: „Excellent“, „VeryGood“, „Good“, „Low“ und „VeryLow“. Um auch diese Lexeme in Entfernungen umrechnen zu können, wurde folgender Versuch durchgeführt:

5.1.1 Versuch

In einer üblichen Büroumgebung wird in der Gebäudeecke ein WLAN Accesspoint aufgestellt. Die Versuchsumgebung ist eine U-förmige Büroetage mit einer Seitenlänge von 23 Metern auf 18 Meter. Die einzelnen Büros und Versorgungsräume sind durch dünne, nicht tragende Wände unterteilt und mit Metallschränken, Schreibtischen, Bürostühlen und EDV-Geräten möbliert. Bei dem Accesspoint handelt es sich um einen Linksys WRT54G mit einer Sendeleistung von 170mW und den ab Werk montierten Antennen mit einem Gewinn von +3dB. Anschließend wird die Signalqualität in verschiedenen Entfernungen mit dem Testgerät gemessen und das angezeigte Lexem notiert. Die höchste Entfernung, aus der die Signalqualität gemessen wurde, betrug 40 Meter. Dies war allerdings nicht mehr innerhalb des Gebäudes möglich. Um trotzdem realistische Ergebnisse zu erhalten, wurde die Messung hinter 2 Wänden durchgeführt. Dieser Versuch wurde

an zwei unterschiedlichen Orten zu jeweils zwei unterschiedlichen Zeiten durchgeführt. Die Entfernungen wurden anhand einer Bauskizze gemessen.

5.1.2 Ergebnis

Bei den in Tabelle 5.1 angegebenen Werten handelt es sich um Durchschnittswerte der vier Messungen. Insgesamt lag die Abweichung der Messwerte vom Durchschnitt bei höchstens 0,5 Metern für die Werte „Excellent“ und „VeryGood“, bei etwa 0,8 Metern für den Wert „Good“ und bei den Werten „Low“ und „VeryLow“ betrug sie etwa 1,5 Meter.

Lexem	Entfernung, bis zu der das angegebene Lexem angezeigt wurde
Excellent	1m
VeryGood	5m
Good	10m
Low	25m
VeryLow	40m

Tabelle 5.1: Übersicht über die Lexeme für Signalqualitätsstufen und deren Interpretation als Entfernung zum Sender

Eventuell können die hier genannten Werte bei der Verwendung eines anderen WLAN Chipsatzes ungenauer ausfallen, da sich die Produkte in ihrer Empfangsqualität unterscheiden können und jeder Hersteller die Zuordnung der Qualitätsstufen frei wählen kann.

5.2 Datenmenge

Wie im Kapitel 4.6 über die Kommunikation zwischen Client und Server beschrieben, sendet der Client einmal pro Sekunde ein Datenpaket an den Server. Das im oben genannten Kapitel im Listing dargestellte Datenpaket enthält einen GPS-Datensatz, einen Adressdatensatz, zwei detektierte WLAN-Accesspoints und einen Bluetoothsender als Payload. All diese Daten sind in ein XML Dokument eingebettet, dessen Schemainformationen für zusätzlichen Overhead sorgen.

Der Versuch soll zeigen, mit welchem Datenaufkommen während des Betriebs des Location-Provider Clients zu rechnen ist.

5.2.1 Versuch

Ein Datenpaket mit jeweils mindestens einem Reading aus jeder Datenquelle wird vom Client an den Server übertragen. Im speziellen Fall handelt es sich um den Versuchsaufbau, der zu dem oben genannten und in Kapitel 4.6 aufgeführten Datenpaket bestehend aus einem GPS-Datensatz, einem Adressdatensatz, zwei WLAN-Accesspoints und einem Bletoothsender führt.

Auf dem Server ist ein Paketsniffer (Wireshark: [WIRE2010]) installiert, der alle eingehenden Datenpakete aufzeichnet. Nach der Übertragung werden die vom Client gesendeten TCP/IP Pakete separiert und zu einer Klartextdatei zusammengesetzt, deren Größe darauf hin bestimmt wird.

5.2.2 Ergebnis

Die Größe des gesamten übertragenen Datenpakets beträgt 2,9 Kilobyte. Jeder detektierte Accesspoint vergrößert das Paket um ca. 450 Bytes, abhängig von der Länge der SSID des WLANs. Da für Bluetoothstationen keine Signalstärke übertragen wird, beläuft sich hier, ebenfalls abhängig vom Namen des Senders die Datenmenge auf etwa 250 Byte pro Station.

5.3 Bandbreite

Der vorhergehende Versuch hat Anhaltspunkte für die zu übertragende Datenmenge geliefert. Mit Rücksicht auf die Limitierungen der Mobilfunknetze und Endgeräte, was Übertragungsgeschwindigkeiten betrifft, ist zu prüfen, ob die ermittelten Datenmengen über Mobilfunkverbindungen zuverlässig übertragen werden können. Alleine das genannte Paket ist 2,9 Kilobyte groß und kann in Gegenden mit einer hohen Dichte von empfangbaren Accesspoints pro weiterer detektierter Referenzstationen um etwa 450 Bytes anwachsen. Sollten weitere Signalquellen zusätzlich implementiert werden oder die Nachrichtenverschlüsselung aktiviert werden, bei der für jede Nachricht das Zertifikat mitgesendet wird,

übersteigt die Paketgröße schnell sogar die im .NET Framework standardmäßig definierte maximale Nachrichtengröße von 8 Kilobyte. Während sich dieser Grenzwert beliebig verschieben lässt, ist die zur Übertragung verfügbare Bandbreite der Netzverbindung real limitiert. Um eine Bandbreitenmindestanforderung definieren zu können, wurde der folgende Versuch durchgeführt:

5.3.1 Versuch

Während der mobile Client in verschiedenen drahtlosen Netzwerken eingeloggt ist, wird eine normale Serververbindung ohne Verschlüsselung zur Datenübertragung hergestellt. Zur Übertragung ausgewählt sind alle Signalquellen. Übertragen wird ein Datensatz mit drei detektierten WLAN Accesspoints, einer Bluetoothstation und einem GPS Datensatz.

Als drahtlose Netzwerke wurden getestet: Im Mobilfunkbereich GSM, EDGE, HSPA sowohl von Eplus als auch von O_2. Außerdem WLAN nach 802.11g.

5.3.2 Ergebnis

Tabelle 5.2 zeigt die Ergebnisse der Übertragungsversuche.

Übertragungstechnik	Eplus	O_2
GSM	Verbindungsabbruch	Verbindungsabbruch
EDGE	Verb. möglich	Verb. möglich
UMTS	Verb. möglich	Verb. möglich
HSPA	Verb. möglich	Verb. möglich

Tabelle 5.2: Datenübertragung im mobilen Telefonnetz Netz

Während es bei der Verwendung einer reinen GSM basierten Datenübertragung schon nach dem ersten Datenpaket zu einem Verbindungsabbruch kommt, funktioniert die Übertragung bei allen Übertragungsverfahren mit höheren Bandbreiten reibungslos. Daraus lässt sich schließen, dass die bei GSM maximal verfügbaren 9,6 KB pro Kanal für eine Echtzeitdatenübertragung nicht ausreichend sind. In diesen Fällen erhöht der Client automatisch das Sendeintervall von einer Sekunde auf 5 Sekunden.

Die Datenübertragung über eine WLAN Verbindung stellte dagegen keine Schwierigkeit dar.

5.4 Lastabschätzung

Da es sich hier um ein Client Server System handelt, auf das mehrere Clients parallel zugreifen können sollen, ist die Belastbarkeit des Systems ein wichtiger Evaluationsgegenstand. Bedacht werden soll hier nicht die physische Verfügbarkeit des Dienstes. Diese kann durch geeignete Hardware wie Failoversysteme erreicht werden.

Zu überprüfen ist, ob der Dienst in dieser prototypischen Implementierung die Kernanforderung erfüllt, mehrere Clients zuverlässig zu bedienen.

5.4.1 Versuch

Zum Testen der Verfügbarkeit greifen 10 auf das Sendemodul reduzierte Clientimplementierungen, die auf verschiedenen Systemen im lokalen Netzwerk arbeiten, auf den Server zu. Sie übertragen hierzu einen vom Tester als durchschnittlich groß erachteten Datensatz, der eine Adresse, eine Accesspointliste mit fünf Einträgen, eine Bluetoothliste mit zwei Einträgen und eine GPS Datenstruktur enthält im bekannten Intervall von einer Sekunde an den Server.

5.4.2 Ergebnis

Es kommt zu keinen Geschwindigkeitsproblemen oder Timeouts. Die parallele Annahme von Verbindungsanforderungen mehrerer Clients ist ebenso möglich wie die gleichzeitige Verarbeitung der übertragenen Daten. Die CPU Last der virtuellen Maschine steigt auf 50% an. Es zeigt sich, dass das System für einen prototypischen Betrieb ausreichend leistungsfähig ist und die Implementierung ressourcenschonend 10 Clients bedienen kann. Der Test bestätigt die generelle Eignung des Systems für den Multiuserbetrieb. Ein geeignet dimensioniertes Serversystem wird im Produktiveinsatz eine weit höhere Anzahl an Clients als das Testsystem bedienen können.

5.5 MaxMind IP-Verortung

Die in Tabelle 5.3 zitierten Daten haben den Zweck, die Genauigkeit der Verortung unter alleiniger Berücksichtigung der IP Adresse des Clients zu ermitteln. Selbstverständlich lassen vier Versuche nur eingeschränkt Schlüsse auf die mittlere Genauigkeit eines Systems zu, das annähernd 256^4 unterschiedliche Adressen verorten können soll. Allerdings ist schon bei diesen für die Telekommunikationsinfrastruktur Deutschlands durchaus repräsentativen Versuchen eine Tendenz erkennbar, die Rückschlüsse auf die zu erwartende Genauigkeit im Allgemeinen zulässt.

Dem Test mussten sich IP Adressen der beiden größten deutschen Provider Arcor und T-Online stellen, die beide ein eigenes Netz betreiben und deshalb volle Kontrolle über die ortsbezogene Zuordnung ihrer Adressbereiche haben. Weiterhin sollte mit 1&1 getestet werden, ob es sich bei einem Reseller von T-Onlinezugängen mit der Zuordnung der IP Adresse zu einer Koordinate anders verhält. Zum Schluss folgt noch eine andere Art von Netzwerk. Bei den ersten Dreien handelte es sich um die weit gespannten Netzwerke öffentlicher Internetzugangsprovider, die oft das gesamte Bundesgebiet abdecken. Im Gegensatz dazu repräsentiert der vierte Testkandidat eine Adresse aus einem Klasse B Netz, das einer Organisation, in diesem Fall der Universität Koblenz-Landau, Abteilung Koblenz gehört. Das Netz, aus dem diese Adresse stammt ist ein Campusnetzwerk mit einer räumlichen Ausdehnung von wenigen 100 Metern Seitenlänge. Abgesehen von wenigen Ausnahmen, beispielsweise VPN Einwahlen, befinden sich Nutzer einer Adresse aus diesem Bereich höchstwahrscheinlich tatsächlich auf dem Campus und damit in großer Nähe der Marke, die der Verortungsdienstleister mangels genauerer Informationen möglichst in die Mitte des Gebiets setzt, das von dem entsprechenden Netzwerk abgedeckt wird. Aus der Tabelle geht her-

Adresse	Provider	Standort	Ergebnis	Abweichung
217.225.96.176	T-Online	Koblenz	Neuwied	11,2 km
84.175.143.114	1&1	Rengsdorf	Sinzig	18,0 km
84.63.107.26	Arcor	Koblenz	Cologne	78,6 km
141.26.111.111	Uni-Koblenz	Koblenz	Koblenz	3,2 km

Tabelle 5.3: Ergebnisse der Tests der IP-Verortung von MixMind

vor, dass die Verortung nur bei dem Sonderfall des Campusnetzwerks stadtgenau ist. Im Fall von T-Online und 1&1 liegt die Ungenauigkeit noch im Bereich von unter 30 Kilometern, was für den ein oder anderen Dienstleister noch erträglich sein mag. Die Verortung der IP Adresse aus dem Adressbereich von Arcor fällt jedoch komplett aus dem Rahmen. Mit einer Abweichung von bis zu 100 Kilometern und sogar einem unterschiedlichen Bundesland im Verortungsergebnis (aus der Tabelle nicht zu entnehmen) eignet sich dieses Ergebnis höchstens noch zur Anzeige der richtigen Landesvorwahl.

Erwähnt werden muss, dass die Messungen auf der freien Version der Verortungsbibliothek MaxMind Geocity light basieren. Versuche mit der kostenpflichtigen Version konnten nicht durchgeführt werden.

5.6 Evaluation

Während sich die in den letzten Kapiteln vorgestellten Versuche mit der Fragen aus den Randbereichen dieser Arbeit beschäftigt haben wie der Lastabschätzung des Systems oder der Betrachtung der für eine fehlerfreie Übertragung benötigten Bandbreite, soll in diesem Abschnitt die Erledigung der Kernaufgabe des entwickelten Softwaresystems genauer untersucht werden. Die nächsten Abschnitte stellen die Versuche dar, die durchgeführt wurden, um die Genauigkeit der Verortung, die das System zu erreichen in der Lage ist, zu ermitteln. Weiterhin sollen Hilfsmittel vorgestellt werden, die bei der Evaluation geholfen haben.

5.6.1 Location-Consumer

An die Webserviceschnittstelle, die eigentlich für den Privacy-Provider vorgesehen ist, kann auch jedes andere Programm andocken, sofern es die Schnittstellendefinition des Location-Providers für nachgeschaltete Konsumenten von Clientpositionen implementiert. Eine Autorisation erfolgt hier nicht mit einer Benutzername-Passwort Kombination, sondern anhand eines vorher ausgestellten und im Location-Provider als vertrauenswürdig registrierten Zertifikats. Damit ist sowohl eine integere wie auch abhörsichere Verbindung sichergestellt. Aus Gründen der Einfachheit ist dieses Sicherheitsmerkmal jedoch zu Testzwecken

abgeschaltet. So können die Daten, die via HTTP im Klartext verschickt werden, auch über ein Netzwerkanalysewerkzeug mitgeschnitten werden.

Wie in den Kapiteln 4.6 über die Kommunikationsmechanismen des Systems und 4.10 über das Konsumieren der bereitgestellten WCF Schnittstellen bereits beschrieben, verfügt das System über eine Webserviceschnittstelle für nachgeschaltete Dienste wie den Privacy-Provider.

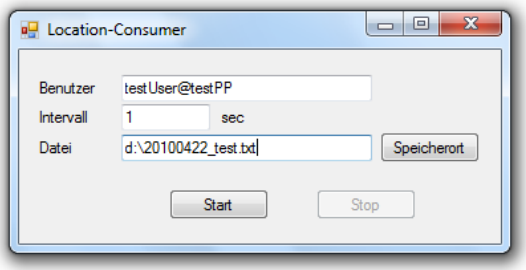

Abbildung 5.1: Screenshot des Location-Consumers. Das Programm fragt periodisch die Position eines Clients ab und speichert diese in der angegebenen Datei.

Diese Schnittstelle implementiert auch der hier verwendete Location-Consumer, dessen Benutzeroberfläche in Abbildung 5.1 zu sehen ist. Seine Aufgabe ist das periodische Abfragen und Protokollieren der vom Location-Provider ermittelten Position des Testclients. Durch die Verwendung des Location-Consumers ist es möglich, über einen längeren Zeitraum Protokollierungen vorzunehmen, ohne die Speicherkapazität eines Smartphones oder dessen Rechenleistung und Hauptspeicher zu überfordern.

Der Location-Consumer liest einmal pro Sekunde die Position des konfigurierten Clients aus und speichert diese mit Zeitstempel in eine kommaseparierte Textdatei. Diese kann später mit einem geeigneten Tabellenkalkulationsprogramm geladen und ausgewertet werden.

5.6.2 Erkennen bestimmter Räume

In der nun folgenden Versuchsreihe soll das Verhalten des Systems hauptsächlich in Bereichen erprobt werden, in denen kein GPS Empfang möglich ist. Dazu zählt insbesondere das Innere von Gebäuden. Speziell wichtig ist die Frage, ob

es möglich ist, die Position eines mobilen Nutzers auf einen Raum, beispielsweise einen Vorlesungssaal einzuschränken und wenn ja, mit welcher Gewissheit.

5.6.2.1 Versuch

Um die Genauigkeit des Systems in Innenräumen zu testen und um zu überprüfen, ob es möglich ist, die Position eines Nutzers auf einen Raum in einem Gebäude genau zu bestimmen, werden zu zwei unterschiedlichen Zeiten Messungen in verschiedenen Vorlesungssälen des Campus durchgeführt. Zum Einen soll die Performance des Systems bei starker Frequentierung und Belastung des WLAN Netzwerkes getestet werden. Hierzu eignen sich Zeiten, in denen der jeweilige Vorlesungssaal durch eine Veranstaltung belegt ist. Da viele Studierende über einen Laptop verfügen und diesen während der Veranstaltung nutzen, sind die Anforderungen an die Belastung des Netzwerks erfüllt. Zum Anderen wird als Vergleich eine Messung am Abend durchgeführt. Diese Vergleichsmessung soll zeigen, ob das System bei hoher Netzwerklast anders reagiert als in einer Ruhephase.

Die Messungen werden jeweils über einen Zeitraum von 30 min durchgeführt. Später wird anhand der ermittelten Werte die Standardabweichung der ermittelten Positionen von der realen Position und damit die Ungenauigkeit des Systems berechnet. Die Messungen werden in folgenden Räumen jeweils einmal bei hohem und einmal bei niedrigem Besucheraufkommen wiederholt:

- Vorlesungssaal D028
- Vorlesungssaal E011
- Labor der Arbeitsgruppe A104
- Universitätsbibliothek

Der Versuchsaufbau besteht aus einem mobilen Endgerät, das Sensordaten vom WLAN und Bluetoothsystem für eine Positionsschätzung an den Server überträgt. Auf einem Laptop wird mit Hilfe des Tools Location-Consumer eine Protokollierung der ermittelten Positionsschätzungen vorgenommen.

5.6.2.2 Ergebnis

Bei allen Messungen war kein GPS-Empfang möglich. Die Verortung basierte also alleine auf den Daten der Kommunikationssensoren des mobilen Endgerätes. Da eine Bluetoothstation nicht in Reichweite war, basieren die Werte ausschließlich auf den Daten des WLAN Systems.

Ausnahmslos alle Messungen führten zu Abweichungen von der realen Position von 30 Metern und mehr. Abbildung 5.2 zeigt ein Luftbild des Gebäudes, in dem sich der Audimax D028 befindet. Der rote Kasten verdeutlicht dessen Ausmaße im Gebäude. Der blaue Kreis hat einen Durchmesser von 30 Metern. Die Verortung eines mobilen Systems mit Erkennung des Raumes, in dem sich das System befindet, muss mit der verwendeten Technik als nicht möglich betrachtet werden, da bei einer ermittelten Ungenauigkeit von 30 Metern bei einer Gleichverteilung der Werte im Wertebereich selbst bei den großen Hörsälen über 50% der ermittelten Positionen außerhalb des Raumes liegen.

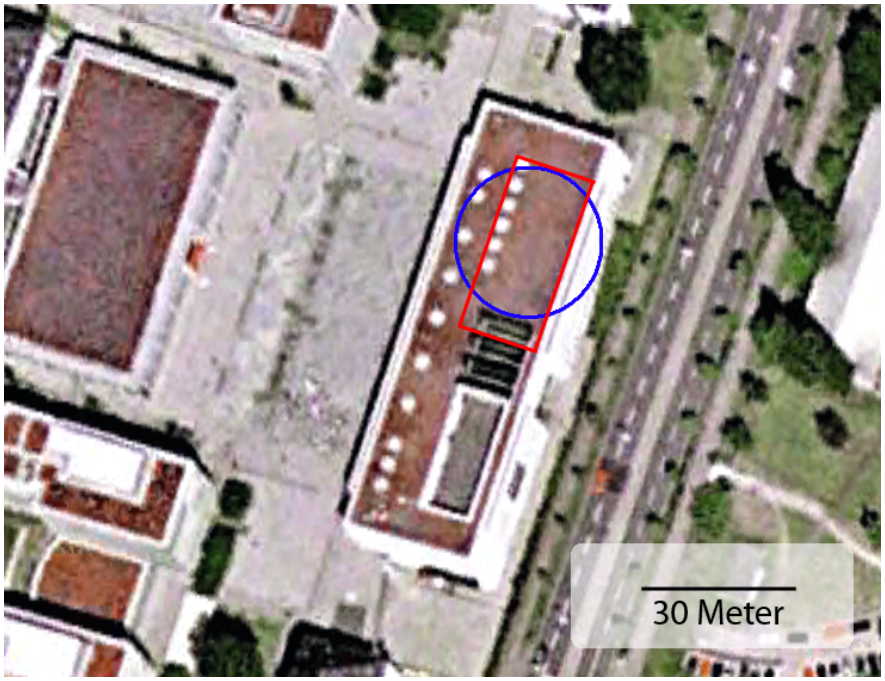

Abbildung 5.2: Luftbild der Universität Koblenz Landau, Hörsaal D028 (rot), Verortungsungenauigkeit von 30 Metern (blau), (Quelle Luftbild: Google Earth)

5.6.3 Verortungsgenauigkeit

Die hier vorgestellten Versuche wurden durchgeführt, um die Qualität der Verortung mit Hilfe des entwickelten Softwaresystems zu testen. Dazu fanden mehrere Tests statt, bei denen das Verortungssystem jeweils mit beiden Filtern eine Positionsschätzung ermitteln musste. Die vorliegenden Versuche wurden alle auf dem Campus der Universität Koblenz-Landau in Koblenz-Metternich durchgeführt. Hier ist die Dichte der WLAN Accesspoints hoch und ihre Positionen sind kartographiert. Die Tests wurden vor der Umstellung der Infrastruktur auf die neue Hardware durchgeführt.

5.6.3.1 Versuch

An einer ausreichend freien Stelle auf dem Campus werden mit jeweils einer Technologie als einziger Signalquelle über einen Zeitraum von 5 Minuten Positionsmessungen durchgeführt.

5.6.3.2 Ergebnis

Die Ergebnisse der Messungen sind in Tabelle 5.4 zusammengefasst. Die Spalten SimpleFilter und Partikelfilter zeigen die Varianz der Messwerte von der eigentlichen Position in Metern an.

Signalquelle	SimpleFilter	Partikelfilter	Verbesserung (%)
GPS	9,21	9,18	0,3
WLAN	28,31	25,76	9

Tabelle 5.4: Verortungsgenauigkeit mit jeweils einer Technologie als Signalquelle

Während der Partikelfilter nicht in der Lage war, die GPS Position merklich zu verbessern, ist dies bei der Verortung mit WLAN Signalen gut gelungen. Dies liegt an der kurzen Beobachtungszeit und der Funktionsweise des Filters. Er bezieht zwar die Ergebnisse mehrere Vorgängerzustände mit in die Vorhersage des aktuellen Zustands ein, dies reicht bei der zeitlich langsamen Veränderung des GPS Signals jedoch nicht. Aufgrund der niedrigen zeitlichen Varianz der Messdaten kann der Filter hier nicht glättend eingreifen. Das normalverteilte Rauschen,

mit dem eine GPS Positionsbestimmung überlagert ist, kommt nur bei sehr langen Messungen zum Tragen. Anders verhält es sich mit der Verortung anhand von WLAN Signalen. Da diese von einem stetigen normalverteilten Rauschen überlagert sind, ist eine Mittelwertbildung oder Glättung der Messergebnisse hier einfacher und führt zu einem verbesserten Verortungsergebnis.

5.6.3.3 Versuch

Um den Übergang von einer GPS basierten zu einer WLAN basierten Verortung zu testen, wird das mobile Endgerät mehrmals von einer Außenfläche in ein Gebäude und wieder heraus bewegt. Zusätzlich wird das Gerät an einem Bluetoothsender vorbei bewegt, um die Reaktion auf die neue Signalquelle zu überprüfen. Die Position des Bluetoothsenders ist in der Datenbank als „confirmed“ erfasst.

5.6.3.4 Ergebnis

Unter freiem Himmel liefert der GPS Empfänger hinreichend gute Daten, sodass die Position mit einer für GPS typischen Abweichung von etwa 8 Metern bestimmt werden konnte. Sobald beim Betreten eines Gebäudes der GPS Empfang abriss, erfolgte die weitere Verortung mit den dann höher priorisierten WLAN Signalen. Hier erhöhte sich die Abweichung auf durchschnittlich 27 Meter. Beim Passieren eines Bluetoothsenders verringerte sich die Abweichung auf 10 Meter (der definierte Wert bei der Verortung mit Bluetooth) und die Positionsschätzung wurde entsprechend präziser. Sobald die Signale des Bluetoothgeräts nicht mehr in Reichweite waren und das System wieder auf WLAN basierte Verortung umschaltete, erhöhte sich auch die Positionsabweichung wieder. Die großen Abweichungen der WLAN Verortung resultieren aus der Ungenauigkeit , mit der die zur Lateration benötigte Entfernung zu den Referenzstationen bestimmt wird. Da nicht bekannt ist, aus welcher Richtung die Signale der empfangenen Accesspoints kommen, muss das System große Ungenauigkeiten einkalkulieren, um nicht gänzlich falsche Positionsangaben zu liefern.

Der Übergang von WLAN- zu GPS basierter Verortung beim Verlassen eines Gebäudes erforderte einige Sekunden, in denen der GPS Empfänger einen gül-

tigen Fix herstellte. Sobald eine gültige GPS Position verfügbar war, sank die Ungenauigkeit wieder auf GPS typische Werte ab.

5.7 Betriebsdauer

Dieser Versuch sollte die Betriebsdauer eines mobilen Endgeräts bei der Verwendung von allen Signalquellen zeigen.

5.7.0.5 Versuch

Das Testgerät, ein HTC Touch Diamond 2, mit einem ein Jahr alten voll aufgeladenen 1100mAh Lithium-Ionen Akku wird mit aktiviertem Bluetooth, WLAN, GPS und GSM und eingeschaltetem Display auf mittlerer Beleuchtungsstufe an einer festen Position als Location-Provider Client betrieben. Die Kommunikation findet über WLAN statt.

5.7.0.6 Ergebnis

Das Gerät meldet nach 1 Stunde und 27 Minuten niedrigen Akkustand und schaltet sich nach weiteren 4 Minuten und 20 Sekunden ab.

Kapitel 6

Abschluss

6.1 Fazit

In dieser Arbeit wurde gemäß den in Kapitel 1.3 definierten Anforderungen ein System entwickelt, um mobile und in der Rechenleistung beschränkte Geräte wie Mobiltelefone und PDAs sowohl im Freien als auch in geschlossenen Räumen verorten zu können. Der eingeschränkten Leistungsfähigkeit der Geräte wird durch die Umsetzung einer Client-Server Lösung Rechnung getragen, bei der der Client als Sensorträger dient, der Daten aus seiner Umgebung registriert und über ein Kommunikationsnetz zur weiteren Berechnung an einen Server überträgt. Die Datenübertragung ist als Webservice realisiert, der auf offenen Standards wie XML und SOAP basiert und leicht um weitere Funktionen erweitert werden kann.

Zur Verortung werden neben dem GPS-System die Signale verschiedener anderer Referenzstationen verwendet, die hierzu zweckentfremdet werden. Bei den Referenzstationen handelt es sich um Sender, die im öffentlichen Raum zur Genüge vorkommen und ortsfest sind. Zur Verortung herangezogen werden die Signale von Bluetoothsendern, WLAN-Accesspoints, die IP Adresse des Senders und die vom Benutzer eingegebene Adresse. Um alle diese Daten zu einer einzigen Positionsschätzung zu kombinieren, die genauer ist als jede Einzelschätzung, wird ein Filtersystem verwendet. Dieses ist in der Lage, einzelne Signalquellen je nach ihrem Auftreten in einer Kombination mit den anderen implementierten Quellen unterschiedlich zu gewichten. Dies ermöglicht eine genauere Kombination der

vorhandenen Informationen, ohne dass ungenauere Verortungstechnologien die Ergebnisse von genaueren Technologien überlagern und verfälschen.

Eine zentrale Fragestellung war der Umgang mit bisher unbekannten potentiellen Referenzpunkten. Diese Arbeit löst das Problem der Kartographierung unbekannter Gebiete durch den selbständigen Aufbau einer Location Database genannten Datenbank, in der alle unbekannten Referenzpunkte mit einer anhand von anderen Datenquellen ermittelten Position gespeichert werden, so dass sie bei einer nachfolgenden Verortung ebenfalls verwendet werden können. So ist es einem Schwarm von Benutzern möglich, schnell neue Gebiete automatisiert zu kartographieren. Um schwerwiegende Fehler in der so erstellten Datenbank zu vermeiden, werden automatisch aufgenommene Referenzstationen als solche markiert. Ein Administrator muss in regelmäßigen Abständen die Positionen der neu hinzu gekommenen Stationen verifizieren. Erst dann gelten die Stationen als sicher zu verwenden und erhalten eine höhere Priorisierung bei der Verortung.

Die interne Architektur von Client und Server erlaubt die geforderte Flexibilität bei der Implementierung neuer Module, da beide Teilsysteme so modular gehalten sind, dass neue Verortungstechnologien als eigenständige Subsysteme integriert werden können. Über ein flexibles Eventsystem werden sie an die Kommunikationsinfrastruktur gekoppelt, die aus einem zentralen Nachrichtendispatcher mit Verbindungen zu jedem Modul besteht. Über diese Verbindungen kann jedes Subsystem sowohl Daten versenden als auch empfangen.

Das Testsystem reagiert ohne nennenswerte Reduzierung der Antwortgeschwindigkeit auf 10 Clients. Dies lässt im Zusammenhang mit der Tatsache, dass eine eher ressourcenschwache virtuelle Maschine die Grundlage des Testsystems bildet, darauf schließen, dass ein entsprechend ausgestattetes System mit einer produktionstypischen Zahl von Clients nicht überfordert sein wird.

Die Bandbreite, die ein Client zur Übertragung seiner Daten an den Server braucht, stellt hier viel eher einen Flaschenhals dar. Eine reine GSM Verbindung ist nicht ausreichend, um alle Daten zu übertragen und es kommt zu Ausfällen. Mindestanforderung an die Übertragungsrate ist also EDGE.

Die Lokalisation anhand einer IP Adresse funktioniert nicht zuverlässig. Während einige Adressen wenigstens im Umkreis von 20 Kilometern zugeordnet wurden, sind andere Adressen, speziell die von Arcor offenbar nur sehr grobgranular

in der Datenbank mit Ortsinformationen hinterlegt. Hier kam es zu unbrauchbaren Abweichungen von bis zu 100 Kilometern. Von der Verwendung der IP Adresse ist abzuraten, wenn es sich nicht um die Adresse einer Organisation wie der Universität handelt, deren Verbreitungsgebiet lokal begrenzt ist.

Die Verortung an sich funktioniert entsprechend der verwendeten Signalquellen unterschiedlich gut. Während eine Verortung mit GPS unter freiem Himmel problemlos mit einer Genauigkeit von etwa 10 Metern funktioniert, kommt es in Gebäuden schnell zu Verbindungsabrissen. Die Verortung mit WLAN Signalen funktioniert bei einer ausreichend guten Signallage von drei oder mehr Stationen, die sich in günstigen Winkeln befinden mit einer Genauigkeit von etwa 30 Metern, sowohl im Außen- als auch im Innenbereich. Die Verortung wird kurzzeitig genauer, wenn eine Bluetoothstation in Sicht kommt. Aufgrund von deren begrenzter Reichweite steigt die Ungenauigkeit sofort wieder, wenn das Signal des Bluetoothsenders verloren geht.

Bei der Verwendung von GPS nützen die Filter fast nichts, da sie keine bemerkbare Verbesserung bringen. Dies liegt daran, dass der GPS Empfänger intern schon eine sehr gute Glättung der Signale vornimmt. Ein weiteres Problem ist, dass das Ergebnis der GPS Verortung nicht in kurzen Zeiträumen normalverteilt springt, sondern sich „fließend“ bewegt. Von einer Normalverteilung der ermittelten Werte um die echte Position ist nur bei sehr großen Beobachtungszeiträumen von über einer Stunde auszugehen. Auch durch die anderen Signalquellen kann hier keine Verbesserung der Positionsschätzung erzielt werden, da alle anderen Lösungen eine höhere Ungenauigkeit aufweisen und deshalb keine präziseren Informationen zum Verortungsprozess beitragen können.

Bei der Glättung der Ergebnisse, die bei der Verortung anhand der übrigen Signalquellen auftreten, ist die Einwirkung des Filters deutlich registrierbar. Die Ungenauigkeit und damit die Abweichung von der realen Position hat sich um 9% verringert. Hier kommt die normalisierende Wirkung des Filters zum Tragen, denn die Positionen, die das WLAN Subsystem ermittelt, springen von Verortungsvorgang zu Verortungsvorgang erheblich um ein gemeinsames Zentrum.

Um das System effektiv zur Verortung verwenden zu können, müssen die mobilen Endgeräte mit leistungsstarken Akkus ausgestattet sein. Ein Standardakku mit altersbedingten Ladungsverlusten hält die Belastung durch die vielen aktiven

Sensoren und die Displaybeleuchtung ca. 1,5 Stunden durch. Bei einer intensiveren Beleuchtung des Displays, die bei Sonneneinstrahlung zum problemlosen Ablesen des Bildschirms nötig ist, verringert sich dieser Wert signifikant. Bedenkt man, dass die Planung für das Metasystem Multimediainhalte, zum Beispiel bei Stadtführungen einschließt, könnte die Betriebsdauer noch erheblich unter dem ermittelten Wert liegen. Hier ist zu überdenken, ob tatsächlich alle Signalquellen zu jeder Zeit nötig sind.

6.2 Ausblick

Nachdem in den vorhergehenden Kapiteln die technischen Grundlagen dieser Arbeit von Verortungslösungen über Filtermechanismen und verschiedene Daten- und Koordinatenformate sowie die Implementierung der Software dokumentiert worden sind, wird dieses Kapitel einen Ausblick auf Möglichkeiten der künftigen Weiterentwicklungen des Systems versuchen.

Zuerst soll dabei die Verbesserung der Implementierung selbst betrachtet werden. Hier könnten einige Überarbeitungen und Ergänzungen die Benutzbarkeit des Systems vereinfachen oder erweitern.

Danach folgt eine Übersicht über mögliche Ergänzungen des Systems, die es einer größeren Nutzergemeinde zugänglich machen und damit den wirtschaftlichen Nutzen der Metaarchitektur, zu der das System gehört, erhöhen könnten.

6.2.1 Weiterentwicklung

In diesem Kapitel liegt der Fokus nicht auf funktionalen Ergänzungen des vorliegenden Systems, sondern auf Verbesserungen der vorhandenen Funktionen. Zu diesen Verbesserungen wäre als erstes die Userverwaltung zu nennen. Im vorliegenden System wurde das Userkonzept sehr grob implementiert und besteht nur aus einem Benutzernamen und einem Kennwort. Verfügt der Benutzer über beides, gilt er als authentifiziert und darf den Dienst benutzen. Die Entscheidung, das Benutzermanagement einfach zu gestalten fiel vor dem Hintergrund, dass in der Metaarchitektur, zu der das vorliegende System gehört, der Location-Provider und der Privacy-Provider zusammen eine Komponente bilden. Für die Integration

ist es also ohnehin nötig, ein gemeinsames Usermanagement zu implementieren. Dieses könnte dann auch weitere benutzerspezifische Einstellmöglichkeiten wie voreingestellte Adressen oder Bewegungsprofile, die als Vergleichsobjekte dienen können, enthalten. Hierzu bietet sich das im .NET Framework vorgesehene Benutzermanagement an. Es bietet Rollen, über die eine feinere Unterscheidung zwischen Nutzern vorgenommen werden kann als es über die Unterscheidung des Loginstatus möglich wäre. Weiterhin bietet es vorgefertigte Routinen und sogar die dazu passenden Usercontrols, also grafische Steuerelemente, zum Anlegen und Freischalten neuer Nutzer, zum Zurücksetzen des Passwortes und vieles mehr.

Im Zusammenhang mit .NET Technologien, die das hier vorgestellte Softwaresystem ergänzen könnten, ist auch ASP[1].NET 2.0 zu nennen. Mit ASP.NET werden Internetseiten erstellt, die über die Internet Information Services, den Webserver von Microsoft, gehostet werden. ASP.NET ist dabei keine Programmiersprache, sondern eine Sammlung von Techniken, um mit allen Sprachen des .NET Frameworks Webanwendungen programmieren zu können. Der Vorteil von ASP.NET liegt dabei in der Verfügbarkeit vieler vorgefertigter Funktionen. So können sowohl das aus .NET bekannte Benutzermanagement als auch die Datenbankzugriffstechniken über ADO.NET und viele weitere Programmteile verwendet werden.

Diese Arbeit könnte mit ASP.NET sehr einfach um eine Webschnittstelle erweitert werden. Über diese wäre ein Remotezugriff auf den Server und seine Funktionen möglich, ohne dass ein Administrator physischen Zugang zu dem Gerät haben müsste. Die Verwaltung des Systems wäre dadurch flexibler und eventuell mit weniger Zeitaufwand möglich. In Kombination mit dem Privacy-Provider wäre auch ein Webinterface für die Benutzer denkbar, über das persönliche Einstellungen vorgenommen werden könnten.

Um die Betriebsdauer der mobilen Endgeräte im Akkubetrieb zu verlängern, könnten die diversen Sensoren nur bei Bedarf zugeschaltet werden. Wie die Versuche gezeigt haben, verbessert die Verwendung keiner Verortungstechnik die Positionsschätzung mit GPS. Also könnte das Endgerät selbständig alle anderen Sensoren abschalten, so lange ein gültiges GPS Signal vorliegt. Fällt dieses weg oder wird ungültig, müsste die Anwendung wiederum selbständig die Sensoren

[1] Active Server Pages

für WLAN und Bluetooth oder je nach Konfiguration eventuell nur jeweils einen aktivieren.

6.2.2 Erweiterung

In Kapitel 2.5 wurden verschiedene Signalquellen vorgestellt, deren Signale im öffentlichen Raum empfangbar und als Referenzdatenquelle für eine Verortung nutzbar sind. Von den dort vorgestellten Quellen sind jedoch in dieser Arbeit nicht alle verwendet worden. Im Fall von Odometrie ist der Grund, dass sich die Technologie mit der vorliegenden Hardware nicht nutzen lässt. Bei einigen anderen Techniken wäre eine Umsetzung mit den Mitteln der vorgegebenen Hardware allerdings möglich.

So könnten visuelle Markierungen, die in Form eines QR Codes an wichtigen Stellen angebracht sind ähnlich wie die Hauskoordinaten verwendet werden. In den Codes könnte entweder eine Seriennummer einkodiert sein, deren Index in der Datenbank mit einer Position verbunden ist oder sogar die Position selbst in einem geeigneten Format. Diese Markierungen werden vom Benutzer aktiv mit der Kamera des mobilen Endgeräts aufgenommen und dekodiert. Um Bandbreite zu sparen sollte diese Dekodierung clientseitig stattfinden. Dies belastet zwar die Hardware des Clients, erspart es dem System aber, ein Bild übertragen zu müssen. Statt dessen wird nur eine dekodierte Seriennummer übertragen. Eine solche Verwendung von visuellen Markierungen zur Verortung ist nur an Stellen sinnvoll, an denen der Benutzer einen Ansporn hat, aktiv in die Verortung einzugreifen, da es die Interaktion des Abfotografierens der Markierung erfordert. Dies ist beim passiven Tracking unkomfortabel und würde von den Nutzern nicht angenommen werden. An Punkten, an denen es dem Nutzer selbst darauf ankommt, eine möglichst präzise Positionsschätzung zu erhalten, wird er bereit sein, die Kamera auf eine Markierung zu richten. Eine solche Situation könnte in einer virtuellen Stadtführung an besonderen Sehenswürdigkeiten bestehen oder in einem lokal enger begrenzten Raum in einer Ausstellung wie der Bundesgartenschau sogar an einzelnen Ausstellungsstücken.

Wie im Kapitel über das GSM Netz beschrieben, könnten auch dessen Signale zur Verortung verwendet werden. Hierzu müsste eine weitere Collector-

komponente in die Clientsoftware integriert werden. Diese könnte jedoch nicht ausschließlich in managed Code ablaufen, sondern müsste Hardwarezugriffe in vollkompiliertem, also nicht mehr von .NET verwaltetem Code, implementieren, um die entscheidenden Parameter aus der Hardware auslesen zu können. Entsprechend des Umfangs der ausgelesenen Daten ließe sich mit Hilfe der verfügbaren GSM Signale eine Positionsschätzung ohne Zuhilfenahme weiterer Sensoren erreichen. Da gerade die Sensoren für GPS, WLAN und Bluetooth, wenn sie in Betrieb sind, einen hohen Energieverbrauch haben, das GSM Modul aber in den meisten Fällen ohnehin in Betrieb ist, ließe sich auf diese Weise fast ohne zusätzlichen Energieverbrauch der Funktionsumfang des mobilen Endgerätes um eine Verortungslösung erweitern.

Ein wichtiger Aspekt für die Verbreitung einer Lösung ist ihre Verfügbarkeit für verschiedene Softwaresysteme. Auf dem Smartphonemarkt sind momentan (Stand 1. Quartal 2010) als große Entwicklungsumgebungen das von Apple eingesetzte iPhone OS, Googles Android und Windows Mobile, das von mehreren Hardwareherstellern eingesetzt wird, vertreten. Um den Kreis der potentiellen Benutzer der Anwendung möglichst groß zu gestalten, könnte es ratsam sein, für die genannten Systeme Clientkomponenten zur Verfügung zu stellen. Kommt es zu einem Wettbewerb zwischen Location-Providern, wie es in [STEI08] angedacht ist, würde die Verfügbarkeit auf mehreren Plattformen ein starkes Alleinstellungsmerkmal darstellen.

In diesem Zusammenhang sollten neben den drei weitest verbreiteten Betriebssystemen auch die neu auf dem Markt angetretenen Systeme nicht außer Acht gelassen werden. Hier wären das auf dem Palm Pre eingesetzte, linuxbasierte webOS und das auf höherwertigen Nokiageräten vorzufindende Symbian OS zu erwähnen.

Literaturverzeichnis

[NEBE97] Nebe, Wolf: Praxis der Astronavigation, 1997, Delius Klasing Verlag

[GROS76] Großmann, Walter: Geodätische Rechnungen und Abbildungen in der Landesvermessung, 1976, nicht mehr verlegt

[TANN03] Tannenbaum, Andrew S.: Computernetzwerke 4. überarbeitete Auflage, 2003, Pearson Education Deutschland GmbH

[PETE08] Peterson, Larry L.; Davie, Bruce S.: Computernetze, Eine systemorienierte Einführung, 2008, d.punkt Verlag GmbH

[KOHJ2006] Kolodziej, Krzysztof W.; Hjelm, Johan: Local Positioning Systems - LBS Applications and Services, 2006, Taylor & Francis Group

[ROTH2005] Roth, Jörg: Mobile Computing - Grundlagen, Technik, Konzepte, 2. aktualisierte Auflage, 2005, d.punkt Verlag GmbH

[KRUE00] Krüger, Guido: Go To Java 2 - Handbuch der Java-Programmierung, 2000, Addison-Wesley

[VONH09] Vonhoegen, Helmut: Einstieg in XML - Aktuelle Standards - XML Schema XSL XLink, 2009, Galileo Press

[LANG06] de Lange, Norbert: Geoinformatik in Theorie und Praxis, 2006, Springer

[GRIM05] Grimm, Rüdiger: Digitale Kommunikation, 2005, Oldenburg Verlag

[HÖKO08] Hölzl, Stephanie; Kotz, Jürgen: WCF - Die Windows Communication Foundation: Verteilte Anwendungsentwicklung mit der Microsoft Kommunikationsplattform, 2008, Addison Wesley

[VEKO08] Vermessungs- und Katasterverwaltung Rheinland-Pfalz: Richtlinien zur Bildung und Abbildung von Objekten der Automatisierten Liegenschaftskarte, 2008, Ministerium des Innern und für Sport Rheinland-Pfalz

[TRGK] Landesvermessungsamt Nordrhein-Westfalen: Transformation von Koordinaten und Höhen in der Landesvermessung, online verfügbar: http://www.bezreg-koeln.nrw.de/brk_internet/organisation/abteilung07_produkte/raumbezug/bezugssystem_uebergaenge/trafo_1.pdf

[LCYC05] LaMarca, Anthony; Chawathe, Yatin; Consolvo, Sunny: Place Lab - Device Positioning Using Radio Beacons in the Wild, 2005, online verfügbar: http://www.placelab.org/publications/pubs/pervasive-placelab-2005-final.pdf

[NTRI05] unbekannt: Networked Transport of RTCM via Internet Protocol (Ntrip), 2005, online verfügbar: http://igs.bkg.bund.de/root_ftp/NTRIP/documentation/NtripDocumentation.pdf

[MGWI08] Stöbel, Daniel; Zeuch, Daniel; Klebe, Fabian; Rödder, Marc; Köhler, Marcus; Vetter, Steffen; Allgood, Tanja; Zho, Xin: mGeoWiki 2, Mobiles Ortsbezogenes Wiki 2, 2008, Arbeitsbericht

[PELL05] Pellenz, Johannes; Bauer, Sabine; Hebel, Tobias; Spiekermann, Sebastian; Tillmann, Gerd: Verbesserte GPS-Positionsschätzung mit IP-transportierten Korrekturdaten für autonome Systeme im Outdoor-Bereich, 2005, Journal: Autonome Mobile Systeme, Seiten 165-170

[IGPS] Eissfeller, Bernd;Teuber, Andreas; Zucker, Peter: Indoor-GPS - Ist der Satellitenempfang in Gebäuden möglich?, 2005, Journal: zfv Zeitschrift für Geodäsie, Geoinformation und Landmanagemt, Seiten: 226-234

[DENZ03] Denzler, Joachim: Probabilistische Zustandsschätzung und Aktionsauswahl im Rechnersehen, 2003, Habilitationsschrift

[STEI08] Stein, Stefan: Entwicklung einer Architektur für komplexe kontextbezogene Dienste im mobilen Umfeld, 2008, Arbeitsbericht

[STEI10] Stein, Stefan: Entwicklung einer Architektur zum Schutz der Privatsphäre bei der Nutzung von kontextbezogenen Diensten im mobilen Umfeld, 2010, Dissertation, Universität Koblenz-Landau

[HAMP06] Hampe, J. Felix: Positionierung, 2006, aus Foliensatz: Mobile Application Systems

[WICK05] Wickert, Dean: Fusion von WLAN- und kameragestützten Positionsdaten, 2005, Studienarbeit

[HEBE07] Hebel, Tobias: Positionsdatenintegration im Indoor- und Outdoorbereich mithilfe eines probabilistischen Filters, 2007, Studienarbeit

[MAXM] Website: MaxMind GeoCity Light, http://www.maxmind.com/, abgerufen am 18.09.2009

[P12I] Website: P12Import, http://sourceforge.net/projects/p12import/, abgerufen am 12.12.2008

[OSSL] Website: OpenSSL, http://www.openssl.org/, abgerufen am 10.04.2009

[MMAP] Website: MagigMap, http://www2.informatik.hu-berlin.de/rok/MagicMap/, abgerufen am 10.04.2009

[GSM] Website: Die GSM Architektur, http://www.umtslink.at/index.php?pageid=GSM_GSM_Architektur, abgerufen am 10.11.2009

[RGPS] Website: Relativistische Korrekturen für GPS, http://homepage.univie.ac.at/Franz.Embacher/rel.html, abgerufen am 18.12.2006

[BLTH] Website: offizielle Website zur Bluetooth Technologie, http://german.bluetooth.com/bluetooth/, abgerufen am 10.02.2007

[PLAB] Website: Placelab, http://www.placelab.org/, abgerufen am 10.04.2006

[NTRIP] Website: GNSS Data Center - about Ntrip, http://igs.bkg.bund.de/ntrip/ntriphomepage, abgerufen am 28.04.2010

[GPS] Website: Wie funktioniert GPS, Alles Wissenswerte, http://www.kowoma.de/gps/, abgerufen am 29.04.2010

[NMEA1] Website: NMEA, die Schnittstelle an Bord, http://www.nmea.de/, abgerufen am 10.06.2006

[NMEA2] Website: Das NMEA-0183 Datenformat, http://www.kowoma.de/gps/zusatzerklaerungen/NMEA.htm, abgerufen am 10.06.2006

[COLT] Website: The Colt Project, http://dsd.lbl.gov/ hoschek/colt/, abgerufen am 10.04.2006

[IRID] Website: MathNET Iridium, http://www.mathdotnet.com/, abgerufen am 13.11.2009

[HTC10] Website: HTC, http://www.htc.com/de/, abgerufen am 14.09.2010

[ALOQ] Website: Aloqa - Always Be A Local, http://www.aloqa.com/, abgerufen am 14.09.2010

[INTH2007] Webseite: 32feet.NET, In The Hand, http://inthehand.com/content/32feet.aspx, abgerufen am 18.04.2010

[MICR2002] Webseite: Microsoft, Performance Comparison: .NET Remoting vs. ASP.NET Web Services, http://msdn.microsoft.com/en-us/library/ms978411.aspx, abgerufen am 17.05.2009

[WEB2010] Webseite: NetCraft, July 2010 Web Server Survey, http://news.netcraft.com/archives/category/web-server-survey/, abgerufen am 13.07.2010

[FAZ2010] Webseite: FAZ.NET, Google stoppt WLAN Datensammlung, http://www.faz.net/-00mwkn, abgerufen am 30.05.2010

[WIRE2010] Website: Wireshark, Go Deep, http://www.wireshark.org/, abgerufen am 02.04.2010

[SKY2010] Website: Skyhook, http://www.skyhookwireless.com/ abgerufen am 20.08.2010